Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques

APPROX/RANDOM 2023, September 11–13, 2023, Atlanta, Georgia, USA

Edited by

Nicole Megow
Adam Smith

LIPIcs

Editors

Nicole Megow
University of Bremen, Germany
nicole.megow@uni-bremen.de

Adam Smith
Boston University, MA, USA
ads22@bu.edu

ACM Classification 2012
Theory of computation

ISBN 978-3-95977-296-9

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/978-3-95977-296-9.

Publication date
September, 2023

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.

Digital Object Identifier: 10.4230/LIPIcs.APPROX/RANDOM.2023.0

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.

ISSN 1868-8969

https://www.dagstuhl.de/lipics

Contents

APPROX

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith
LIPICS Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

RANDOM

Preface

This volume contains the papers presented at the 26th International Conference on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2023) and the 27th International Conference on Randomization and Computation (RANDOM 2023), which were organized at Georgia Institute of Technology, Atlanta, GA, USA, September 11-13, 2023. APPROX focuses on algorithmic and complexity issues surrounding the development of efficient approximate solutions to computationally-difficult problems, and the 2023 edition was the 26th in the series. RANDOM is concerned with applications of randomness to computational and combinatorial problems, and the 2023 edition was the 27th in the series. Prior to 2003, APPROX took place in Aalborg (1998), Berkeley (1999), Saarbruücken (2000), Berkeley (2001), and Rome (2002), while RANDOM took place in Bologna (1997), Barcelona (1998), Berkeley (1999), Geneva (2000), Berkeley (2001), and Harvard (2002). Since 2003, APPROX and RANDOM have been co-located, taking place in Princeton (2003), Cambridge (2004), Berkeley (2005), Barcelona (2006), Princeton (2007), Boston (2008), Berkeley (2009), Barcelona (2010), Princeton (2011), Boston (2012), Berkeley (2013), Barcelona (2014), Princeton (2015), Paris (2016), Berkeley (2017), Princeton (2018), and Boston (2019). In 2020, 2021, and 2022, the conferences were held online. We were delighted to return to an in-person event in 2023!

Topics of interest for APPROX include approximation algorithms, hardness of approximation, small space, sub-linear time and streaming algorithms, online algorithms, approaches that go beyond worst case analysis, distributed and parallel approximation, embeddings and metric-space methods, mathematical-programming methods, spectral methods, combinatorial optimization, algorithmic game theory, mechanism design and economics, computational-geometry problems, approximate learning. Those at RANDOM include the design and analysis of randomized algorithms, randomized complexity theory, pseudorandomness and derandomization, random combinatorial structures, random walks/Markov chains, expander graphs and randomness extractors, probabilistic proof systems, random projections and embeddings, error-correcting codes, average-case analysis, smoothed analysis, property testing, computational learning theory, and the role of (pseudo)randomness in other areas of computer science such as cryptography, data privacy, and quantum information.

The volume contains 28 contributed papers selected by the APPROX Program Committee out of 62 submissions, and 38 contributed papers selected by the RANDOM Program Committee out of 67 submissions. We would like to thank all the authors who submitted papers, the members of the program committees, and the external reviewers. We are grateful for the guidance of the steering committees: Jarosław Byrka, Samir Khuller, Monaldo Mastrolili, Laura Sanità, Chaitanya Swamy, László Végh, Virginia Vassilevska Williams, and David P. Williamson for APPROX, and Oded Goldreich, Raghu Meka, Cris Moore, Anup Rao, Omer Reingold, Dana Ron, Ronitt Rubinfeld, Amit Sahai, Ronen Shaltiel, Alistair Sinclair, and Paul Spirakis for RANDOM.

Program Committees

APPROX

Marcin Bieńkowski	University of Wrocław
Yuri Faenza	Columbia University
Chien-Chung Huang	École Normale Supérieure
Sami Davies	Northwestern University
Naveen Garg	Indian Institute of Technology Delhi
Anupam Gupta	Carnegie Mellon University
Arindam Khan	Indian Institute of Science
Nicole Megow	University of Bremen
Seffi Naor	Technion
Britta Peis	RWTH Aachen University
Lars Rohwedder	Maastricht University
Melanie Schmidt	Heinrich Heine University Düsseldorf
David Shmoys	Cornell University
Mohit Singh	Georgia Institute of Technology
Tami Tamir	Reichman University

RANDOM

Antonio Blanca	Pennsylvania State University
Clément Canonne	University of Sydney
Sitan Chen	Harvard University
Mahdi Cheraghchi	University of Michigan
Dean Doron	Ben-Gurion University
Andreas Galanis	Oxford University
Prahladh Harsha	Tata Institute of Fundamental Research
Hamed Hatami	McGill University
Tali Kaufman	Bar-Ilan University
Esty Kelman	Boston University and MIT
Swastik Kopparty	University of Toronto
Andrew McGregor	University of Massachusetts
Moti Medina	Bar-Ilan University
Will Perkins	Georgia Institute of Technology
Daniel Reichman	Worcester Polytechnic Institute
Noga Ron-Zewi	University of Haifa
Cynthia Rush	Columbia University
Jad Silbak	Tel Aviv University
Adam Smith	Boston University
Madhur Tulsiani	Toyota Technological Institute at Chicago
Chris Umans	California Institute of Technology
Nithin Varma	Chennai Mathematical Institute

Subreviewers

APPROX

Andreas Abels
Robert Andrews
Sepehr Assadi
Nikhil Ayyadevara
Yossi Azar
Mourad Baiou
Erik Balkanski
Etienne Bamas
Sandip Banerjee
Nikhil Bansal
Mahdi Belbasi
Leyla Biabani
Martin Böhm
Sander Borst
Alexander Braun
Adam Brown
Niv Buchbinder
Jaroslaw Byrka
Alejandro Cassis
Diptarka Chakraborty
Lijie Chen
Anshuman Chhabra
Emilio Cruciani
Mónika Csikós
Syamantak Das
Max Deppert
Michael Dinitz
Ilan Doron
Lukas Drexler
Marina Drygala
Talya Eden
Tomer Ezra
Moran Feldman
Vincent Froese
Mohit Garg
Leszek Gasieniec
Elahe Ghasemi
Gramoz Goranci
Rohan Guhge
Annika Hennes
Klaus Jansen
Shaofeng Jiang
Haotian Jiang
Karthik C. S.
Dor Katzelnick
Thomas Kesselheim
Samir Khuller
Philip Klein
Robert Kleinberg
Zhuan Khye Koh
Anand Krishna
Ravishankar Krishnaswamy
Pankaj Kumar
John Kuszmaul
Aditi Laddha
Bundit Laekhanukit
Abhiruk Lahiri
James Lee
Roie Levin
Jian Li
Anand Louis
Raghunath Reddy Madireddy
Pasin Manurangsi
Jai Moondra
Anish Mukherjee
Joydeep Mukherjee
Sagnik Mukhopadhyay
Alexander Munteanu
Viswanath Nagarajan
Giacomo Nannicini
Heather Newman
Ashkan Norouzi Fard
Lehilton L. C. Pedrosa
Noemie Perivier
Ulrich Pferschy
Adam Polak
Sharath Raghvendra
Saladi Rahul
Dror Rawitz
Victor Oliveira Reis
Kevin Schewior
Marc Schroder
François Sellier
Hadas Shachnai
Richard Shapley
Pattara Sukprasert
Warut Suksompong
Zoya Svitkina
Théophile Thiery

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith
LIPICS Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Noam Touitou
Artem Tsikiridis
Seeun William Umboh
Daniel Vaz
Ameya Velingker
Ruben F.A. Verhaegh
José Verschae
Pavel Veselý
Tjark Vredeveld
Hoa Vu
Julian Wargalla
Andreas Wiese
Jie Xue
Yuhao Zhang
Da Wei Zheng

RANDOM

Raghav Addanki
Divesh Aggarwal
Vedat Alev
Eric Allender
Vipul Arora
Per Austrin
Omri Ben-Eliezer
Aaron Berger
Amey Bhangale
Guy Blanc
Jaroslaw Blasiok
Greg Bodwin
Marco Bressan
Mark Bun
Ramya C
Eshan Chattopadhyay
Archit Chauhan
Lijie Chen
Holger Dell
Mason DiCicco
Yotam Dikstein
Irit Dinur
Pranjal Dutta
Talya Eden
Klim Efremenko
Weiming Feng
Yuval Filmus
Noah Fleming
Uma Girish
Themis Gouleakis
Joshua Grochow
Heng Guo
Tom Gur
Nathaniel Harms
Pooya Hatami
Shuichi Hirahara
Kaave Hosseini
William Hoza
Vishesh Jain
Matthew Jenssen
Fernando Granha Jeronimo
Ce Jin
Eliran Kachlon
Iden Kalemaj
Caleb Koch
Simon Korman
Manish Kumar
Troy Lee
Amit Levi
Honghao Lin
Andrea Lincoln
Ephraim Linder
Jingcheng Liu
Nitya Mani
Noam Mazor
Marcus Michelen
Dor Minzer
Tushant Mittal
Jonathan Mosheiff
Shivam Nadimpalli
Shyam Narayanan
Diptaksho Palit
Konstantinos Panagiotou
Bucky Park
Eric Price
Edward Pyne
Lekshmi Ramesh
Christoforos Raptopoulos
Hanlin Ren
Oren Renard
Nicolas Resch
Itamar Rot
Ron Rothblum
Sourya Roy
Swagato Sanyal
Ramprasad Saptharishi
Nitin Saurabh
Raghuvansh Saxena
Jonathan Scarlett

Rik Sengupta
Omri Shmueli
Sandeep Silwal
Makrand Sinha
Shashank Srivastava
Timothy Sun
Roei Tell
Om Thakkar
Justin Thaler
Eliad Tsfadia
Arsen Vasilyan
Prashant Vasudevan
Emanuele Viola
Erik Waingarten
Lun Wang
Takashi Yamakawa
Tal Yankovitz
Justin Yirka
Yuichi Yoshida
Xusheng Zhang

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith
LIPICS Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

List of Authors

Amir Abboud (1)
Weizmann Institute of Science, Rehovot, Israel

Dorna Abdolazimi (40)
University of Washington, Seattle, WA, USA

Eric Allender (56)
Rutgers University, Piscataway, NJ, USA

Prashanth Amireddy (41)
School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA

Konrad Anand (38)
Queen Mary, University of London, UK

Rubi Arviv (42)
Efi Arazi School of Computer Science, Reichman University, Herzliya, Israel

Vikrant Ashvinkumar (58)
Department of Computer Science, Rutgers University, New Brunswick, NJ, USA

Sepehr Assadi (57, 58)
Department of Computer Science, Rutgers University, New Brunswick, NJ, USA; Cheriton School of Computer Science, University of Waterloo, Canada

Nikhil Ayyadevara (21)
University of Michigan, Ann Arbor, MI, USA

Tanvi Bajpai (7)
Department of Computer Science, University of Illinois, Urbana-Champaign, Urbana, IL, USA

Ishan Bansal (14)
Operations Research and Information Engineering, Cornell University, Ithaca, NY, USA

Nikhil Bansal (21)
University of Michigan, Ann Arbor, MI, USA

MohammadHossein Bateni (1)
Google Research, Mountain View, CA, USA

Ruben Becker (29)
Ca' Foscari University of Venice, Italy

Huck Bennett (37)
Oregon State University, Corvallis, OR, USA

Piotr Berman (66)
Unaffiliated Researcher

Arijit Bishnu (48)
Indian Statistical Institute, Kolkata, India

Antonio Blanca (53)
Pennsylvania State University, University Park, PA, USA

Jeremiah Blocki (59)
Purdue University, West Lafayette, IN, USA

Andrej Bogdanov (43)
School of EECS, University of Ottawa, Canada

Vladimir Braverman (45)
Rice University, Houston, TX, USA

Nader H. Bshouty (34)
Department of Computer Science, Technion, Haifa, Israel

Arnaud Casteigts (29)
University of Geneva, Switzerland

Sourav Chakraborty (63)
Indian Statistical Institute, Kolkata, India

Karthekeyan Chandrasekaran (3)
University of Illinois, Urbana-Champaign, IL, USA

Shuchi Chawla (26)
University of Texas - Austin, TX, USA

Chandra Chekuri (7, 24)
Department of Computer Science, University of Illinois, Urbana-Champaign, Urbana, IL, USA

Xi Chen (39)
Columbia University, New York, NY, USA

Joe Cheriyan (14)
Department of Combinatorics and Optimization, University of Waterloo, Canada

Tsun Ming Cheung (43)
School of Computer Science, McGill University, Montreal, Canada

Eden Chlamtáč (11)
Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Lily Chung (42)
MIT, Cambridge, MA, USA

Vincent Cohen-Addad (1)
Google Research, Zürich, Switzerland

Amin Coja-Oghlan (54)
Department of Computer Science, TU Dortmund, Germany

Joshua Cook (47, 55)
Department of Computer Science, University of Texas Austin, TX, USA

Pierluigi Crescenzi (29)
Gran Sasso Science Institute, L'Aquila, Italy

Mark de Berg (27)
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Chengyuan Deng (58)
Department of Computer Science, Rutgers University, New Brunswick, NJ, USA

Lindsey Deryckere (17)
School of Computer Science, The University of Sydney, Australia

Krishnamoorthy Dinesh (43)
Dept. of Computer Science and Engineering, Indian Institute of Technology, Palakkad, India

Ilan Doron-Arad (22)
Computer Science Department, Technion, Haifa, Israel

Anita Dürr (18)
Department of Computer Science, ETH Zürich, Switzerland

Talya Eden (62)
Bar-Ilan University, Ramat Gan, Israel

Charilaos Efthymiou (33)
Computer Science, University of Warwick, Coventry, UK

Nicolas El Maalouly (18)
Department of Computer Science, ETH Zürich, Switzerland

Andreas Emil Feldmann (9)
Department of Computer Science, University of Sheffield, UK

Renato Ferreira Pinto Jr. (61)
University of Waterloo, Canada

Yuval Filmus (36)
Technion - Israel Institute of Technology, Haifa, Israel

Josefine Foos (19)
Research Institute for Discrete Mathematics and Hausdorff Center for Mathematics, University of Bonn, Germany

Zachary Friggstad (13)
Department of Computing Science, University of Alberta, Canada

Karthik Gajulapalli (65)
Georgetown University, Washington, DC, USA

Andreas Galanis (64)
Department of Computer Science, University of Oxford, UK

Jane Gao (54)
Department of Combinatorics and Optimization, University of Waterloo, Canada

Jie Gao (58)
Department of Computer Science, Rutgers University, New Brunswick, NJ, USA

Evangelia Gergatsouli (26)
University of Wisconsin - Madison, WI, USA

Shayan Oveis Gharan (40)
University of Washington, Seattle, WA, USA

Arijit Ghosh (48)
Indian Statistical Institute, Kolkata, India

Lior Gishboliner (46)
ETH Zürich, Switzerland

Leslie Ann Goldberg (64)
Department of Computer Science, University of Oxford, UK

Alexander Golovnev (10, 65)
Georgetown University, Washington, D. C., USA

Roy Gotlib (49)
Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel

Jacob Gray (56)
University of Massachusetts, Amherst, MA, USA

Elena Grigorescu (8, 59)
Department of Computer Science, Purdue University, West Lafayette, IN, USA

Logan Grout (14)
Operations Research and Information Engineering, Cornell University, Ithaca, NY, USA

Siyao Guo (10)
Shanghai Frontiers Science Center of Artificial Intelligence and Deep Learning, NYU Shanghai, China

Anupam Gupta (12)
Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

Meghal Gupta (32)
University of California Berkeley, CA, USA

Venkatesan Guruswami (50)
Department of EECS, University of California, Berkeley, CA, USA

Andreas Göbel (38)
Hasso Plattner Institute, University of Potsdam, Germany

Max Hahn-Klimroth (54)
Department of Computer Science, TU Dortmund, Germany

Thomas P. Hayes (33)
Dept. of Computer Science and Engineering, University at Buffalo, NY, USA

Yahli Hecht (51)
School of Computer Science, Tel Aviv University, Israel

Stephan Held (19)
Research Institute for Discrete Mathematics and Hausdorff Center for Mathematics, University of Bonn, Germany

Jakob Bæk Tejs Houen (62)
BARC, University of Copenhagen, Denmark

Felix Höhne (16)
Fraunhofer Institute for Industrial Mathematics ITWM, Kaiserslautern, Germany

Sharat Ibrahimpur (14)
Department of Mathematics, London School of Economics and Political Science, UK

Russell Impagliazzo (31)
Department of Computer Science, University of California San Diego, La Jolla, CA, USA

Fernando Granha Jeronimo (60)
Institute for Advanced Study, Princeton, NJ, USA

Yaonan Jin (39)
Columbia University, New York, NY, USA

Valentine Kabanets (31)
School of Computing Science, Simon Fraser University, Burnaby, Canada

Michael Kapralov (57)
School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland

George Karakostas (5)
Department of Computing & Software, McMaster University, Hamilton, Canada

Emin Karayel (35)
Department of Computer Science, Technische Universität München, Germany

Karthik C. S. (1)
Rutgers University, New Brunswick, NJ, USA

Danish Kashaev (20)
Centrum Wiskunde & Informatica, Amsterdam, The Netherlands

Tali Kaufman (49)
Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel

Chandrima Kayal (63)
Indian Statistical Institute, Kolkata, India

Yiduo Ke (28)
Northwestern University, Evanston, IL, USA

Samir Khuller (28)
Northwestern University, Evanston, IL, USA

Bojana Kodric (29)
Ca' Foscari University of Venice, Italy

Stavros G. Kolliopoulos (5)
Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Greece

Swastik Kopparty (52)
Department of Computer Science and Department of Mathematics, University of Toronto, Canada

Ariel Kulik (22)
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Amit Kumar (12)
Computer Science and Engineering Department, Indian Institute of Technology, Delhi, India

Nithish Kumar (8)
Department of Computer Science, Purdue University, West Lafayette, IN, USA

Nick Kushnir (46)
School of Mathematics, Tel Aviv University, Israel

Lap Chi Lau (4)
David R. Cheriton School of Computer Science, University of Waterloo, Canada

Chin Ho Lee (44)
Harvard University, Cambridge, MA, USA

Joon Lee (54)
Communication Theory Laboratory, École Polytechnique Fédérale de Lausanne, Switzerland

Itai Leigh (36)
Tel-Aviv University, Israel

Johannes Lengler (30)
Department of Computer Science, ETH Zürich, Switzerland

Reut Levi (42)
Efi Arazi School of Computer Science, Reichman University, Herzliya, Israel

Shilun Li (50)
Department of Mathematics, University of California, Berkeley, CA, USA

Matej Lieskovský (9)
Faculty of Mathematics and Physics, Computer Science Institute of Charles University, Prague, Czech Republic

Young-San Lin (8)
Melbourne Business School, Australia

Kasper Lindberg (40)
University of Washington, Seattle, WA, USA

Quanquan C. Liu (28)
Northwestern University, Evanston, IL, USA

John C. S. Lui (43)
Dept. of Computer Science and Engineering, Chinese University of Hong Kong, China

Sepideh Mahabadi (25)
Microsoft Research, Redmond, WA, USA

Yury Makarychev (11)
Toyota Technological Institute at Chicago (TTIC), IL, USA

Joel Manning (45)
Carnegie Mellon University, Pittsburgh, PA, USA

Anders Martinsson (30)
Department of Computer Science, ETH Zürich, Switzerland

Jeremy McMahan (26)
University of Wisconsin - Madison, WI, USA

Dor Minzer (51)
Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA, USA

Gopinath Mishra (48)
University of Warwick, Coventry, UK

Rajat Mittal (63)
Indian Institute of Technology Kanpur, India

Morteza Monemizadeh (6)
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Dana Moshkovitz (55)
Department of Computer Science, University of Texas Austin, TX, USA

Ramin Mousavi (13)
Department of Computing Science, University of Alberta, Canada

Tamalika Mukherjee (59)
Purdue University, West Lafayette, IN, USA

Kamesh Munagala (2)
Department of Computer Science, Duke University, Durham, NC, USA

Meiram Murzabulatov (66)
Computer Science Department, School of Digital Sciences, Nazarbayev University, Astana, Kazakhstan

Saachi Mutreja (56)
University of California, Berkeley, CA, USA

Noela Müller (54)
Department of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands

Vishvajeet N (52)
School of Informatics, University of Edinburgh,UK

Satyajeet Nagargoje (65)
Georgetown University, Washington, DC, USA

Shyam Narayanan (25, 62)
Massachusetts Institute of Technology, Cambridge, MA, USA

Debmalya Panigrahi (12)
Computer Science, Duke University, Durham, NC, USA

Marcus Pappik (38)
Hasso Plattner Institute, University of Potsdam, Germany

Manaswi Paraashar (63)
Aarhus University, Denmark

Kalen Patton (23)
School of Mathematics, Georgia Tech, Atlanta, GA, USA

Chris Peikert (37)
University of Michigan, Ann Arbor, MI, USA; Algorand, Inc., Boston, MA, USA

Will Perkins (38)
School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA

Spencer Peters (10)
Cornell University, Ithaca, NY, USA

Kalina Petrova (30)
Department of Computer Science, ETH Zürich, Switzerland

Milind Prabhu (21)
University of Michigan, Ann Arbor, MI, USA

Edward Pyne (42, 44)
MIT, Cambridge, MA, USA

Kent Quanrud (24)
Department of Computer Science, Purdue University, West Lafayette, IN, USA

Tim Randolph (39)
Columbia University, New York, NY, USA

Sofya Raskhodnikova (66)
Boston University, MA, USA

Michael Raskin (29)
LaBRI, CNRS, University of Bordeaux, France

Malte Renken (29)
Technical University of Berlin, Germany

Artur Riazanov (36)
École Polytechnique Fédérale de Lausanne, Switzerland

Dragos Ristache (66)
Boston University, MA, USA

Maurice Rolvien (54)
Department of Computer Science, TU Dortmund, Germany

Will Rosenbaum (62)
Amherst College, MA, USA

Ron D. Rothblum (47)
Faculty of Computer Science, Technion, Haifa, Israel

Matteo Russo (23)
DIAG, Sapienza Università di Roma, Italy

Arpan Sadhukhan (27)
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Muli Safra (51)
School of Computer Science, Tel Aviv University, Israel

Govind S. Sankar (2)
Department of Computer Science, Duke University, Durham, NC, USA

Swagato Sanyal (63)
Indian Institute of Technology Kharagpur, India

Sidhant Saraogi (65)
Georgetown University, Washington, DC, USA

Nitin Saurabh (63)
Indian Institute of Technology Hyderabad, India

Patrick Schnider (30)
Department of Computer Science, ETH Zürich, Switzerland

Guido Schäfer (20)
Centrum Wiskunde & Informatica, Amsterdam, The Netherlands; ILLC, University of Amsterdam, The Netherlands

Saeed Seddighin (1)
Toyota Technological Institute at Chicago, IL, USA

Rocco A. Servedio (39)
Columbia University, New York, NY, USA

Jiří Sgall (9)
Faculty of Mathematics and Physics, Computer Science Institute of Charles University, Prague, Czech Republic

Hadas Shachnai (22)
Computer Science Department, Technion, Haifa, Israel

Asaf Shapira (46)
School of Mathematics, Tel Aviv University, Israel

Noah G. Singer (15)
Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

Sahil Singla (23)
School of Computer Science, Georgia Tech, Atlanta, GA, USA

Paulina Smolarova (64)
Department of Computer Science, University of Oxford, UK

Dmitry Sokolov (36)
École Polytechnique Fédérale de Lausanne, Switzerland

Frits Spieksma (27)
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Yannik Kyle Dustin Spitzley (19)
Research Institute for Discrete Mathematics and Hausdorff Center for Mathematics, University of Bonn, Germany

Srikanth Srinivasan (41)
Department of Computer Science, Aarhus University, Denmark

Raphael Steiner (30)
Department of Computer Science, ETH Zürich, Switzerland

Noah Stephens-Davidowitz (10)
Cornell University, Ithaca, NY, USA

Madhu Sudan (41)
School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA

Erin Taylor (2)
Department of Computer Science, Duke University, Durham, NC, USA

Harsha Tirumala (56)
Rutgers University, Piscataway, NJ, USA

Christos Tzamos (26)
University of Wisconsin - Madison, WI, USA; University of Athens, Greece

Jakub Tětek (62)
BARC, University of Copenhagen, Denmark

Seeun William Umboh (17)
School of Computing and Information Systems, The University of Melbourne, Australia

Salil Vadhan (44)
Harvard University, Cambridge, MA, USA

Ali Vakilian (11)
Toyota Technological Institute at Chicago (TTIC), IL, USA

Rob van Stee (16)
University of Siegen, Germany

Eric Vigoda (33)
Department of Computer Science, University of California, Santa Barbara, CA, USA

Ilya Volkovich (31)
Computer Science Department, Boston College, Chestnut Hill, MA, USA

Chen Wang (58)
Department of Computer Science, Rutgers University, New Brunswick, NJ, USA

Pengxiang Wang (56)
University of Michigan, Ann Arbor, MI, USA

Robert Wang (4)
David R. Cheriton School of Computer Science, University of Waterloo, Canada

Weihang Wang (3)
University of Illinois, Urbana-Champaign, IL, USA

Simon Weber (30)
Department of Computer Science, ETH Zürich, Switzerland

Emo Welzl (30)
Department of Computer Science, ETH Zürich, Switzerland

Zhiwei Steven Wu (45)
Carnegie Mellon University, Pittsburgh, PA, USA

Lasse Wulf (18)
Institute of Discrete Mathematics, TU Graz, Austria

Huacheng Yu (57)
Department of Computer Science, Princeton University, NJ, USA

Viktor Zamaraev (29)
University of Liverpool, UK

Rachel Yun Zhang (32)
Massachusetts Institute of Technology, Cambridge, MA, USA

Xusheng Zhang (53)
Pennsylvania State University, University Park, PA, USA

Hong Zhou (4)
School of Mathematics and Statistics, Fuzhou University, China

Samson Zhou (45, 59)
University of California Berkeley, CA, USA; Rice University, Houston, TX, USA

Daniel Štefankovič (33)
Department of Computer Science, University of Rochester, NY, USA

On Complexity of 1-Center in Various Metrics

Amir Abboud
Weizmann Institute of Science, Rehovot, Israel

MohammadHossein Bateni
Google Research, Mountain View, CA, USA

Vincent Cohen-Addad
Google Research, Zürich, Switzerland

Karthik C. S.
Rutgers University, New Brunswick, NJ, USA

Saeed Seddighin
Toyota Technological Institute at Chicago, IL, USA

Abstract

We consider the classic 1-center problem: Given a set P of n points in a metric space find the point in P that minimizes the maximum distance to the other points of P. We study the complexity of this problem in d-dimensional ℓ_p-metrics and in edit and Ulam metrics over strings of length d. Our results for the 1-center problem may be classified based on d as follows.

- **Small *d*.** Assuming the hitting set conjecture (HSC), we show that when $d = \omega(\log n)$, no subquadratic algorithm can solve the 1-center problem in any of the ℓ_p-metrics, or in the edit or Ulam metrics.
- **Large *d*.** When $d = \Omega(n)$, we extend our conditional lower bound to rule out sub*quartic* algorithms for the 1-center problem in edit metric (assuming Quantified SETH). On the other hand, we give a $(1+\epsilon)$-approximation for 1-center in the Ulam metric with running time $\widetilde{O_\varepsilon}(nd + n^2\sqrt{d})$.

We also strengthen some of the above lower bounds by allowing approximation algorithms or by reducing the dimension d, but only against a weaker class of algorithms which list all requisite solutions. Moreover, we extend one of our hardness results to rule out subquartic algorithms for the well-studied 1-median problem in the edit metric, where given a set of n strings each of length n, the goal is to find a string in the set that minimizes the sum of the edit distances to the rest of the strings in the set.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Facility location and clustering; Theory of computation → Problems, reductions and completeness; Theory of computation → Unsupervised learning and clustering

Keywords and phrases Center, Clustering, Edit metric, Ulam metric, Hamming metric, Fine-grained Complexity, Approximation

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.1

Category APPROX

Related Version *Full Version*: https://arxiv.org/abs/2112.03222

Funding *Amir Abboud*: This project has received funding from the European Research Council (ERC) under the European Union's Horizon Europe research and innovation programme (grant agreement No 101078482). Additionally, the author is supported by an Alon scholarship and a research grant from the Center for New Scientists at the Weizmann Institute of Science.
Karthik C. S.: This work was supported by Subhash Khot's Simons Investigator Award and by a grant from the Simons Foundation, Grant Number 825876, Awardee Thu D. Nguyen and by the National Science Foundation under Grant CCF-2313372.
Saeed Seddighin: Supported by a Google research gift.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 1; pp. 1:1–1:19
LIPICS Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Given a set of points P in a metric space, finding the point that "best" represents P is a fundamental question in both discrete and continuous optimization. Motivated by applications ranging from machine learning to computational biology, this question has naturally received a large amount of attention through the years.

The objective can be phrased in various ways: In the median problem, the goal is to find the point p that minimizes the sum of the distances to the points in P; in the mean problem, it is the point that minimizes the sum of distances squared; while in the center problem, it is the point p that minimizes the maximum distance from a point of P to p. When the metric is the ℓ_2 (Euclidean) metric, the question of computing the geometric median dates back to the 17th century, when Torricelli was looking for a solution for the case $|P| = 3$, and to whom Fermat described an explicit solution. More recently, the question of computing the center has also become central in applications arising, e.g., in machine learning [11], to compute the minimum enclosing ball of a set of points, or in computational biology, to find a good representative of a set of strings (representing molecular sequences) (e.g., [30]). This fundamental computational geometry problem which has applications to various domains, is the problem we consider in this paper.

Formally, in the (often referred to as the *discrete*) 1-center problem, the input is a set of points P in a metric space, and the goal is to find a point of P that minimizes the maximum distance to the points in P. When doing data summarization or compression, the discrete version often makes more sense: Given a set of, say n strings, taking the most representative string among the input strings, or at least in the set of grammatically (or semantically) meaningful strings is much more insightful than taking an arbitrary string as representative. This also applies more globally, outputting a data element that has been observed provides a better summary than a data element that has been forged by the algorithm and that may be unlikely to exist in the real-world. From a computational complexity standpoint, this problem can be easily solved in time $O(|P|^2 f(d))$ where $f(d)$ is the time required to compute the distance of two points. This can be done by enumerating all possible choices for the center; and for each choice computing the distance from each point in P; then outputting the best center. However, is this naïve algorithm the best we can do?

The computational geometry community has done extensive work on the above question since the 80s. For metrics such as ℓ_1 or ℓ_2, computing the center has received a large deal of attention. When the dimension is assumed to be a constant, there exist *barely subquadratic* algorithms for the ℓ_2 metric, while there exists near-linear time algorithms for the ℓ_1 case (for a discussion on this we refer the reader to [35]). For the case of string metrics, such as Ulam or Edit distance metrics, nothing better than the $O(|P|^2 f(d))$ (where d is the string length) "brute-force" algorithm is known.

Understanding how fast the 1-center problem can be solved in these different metrics is not only interesting from a computational complexity point of view, but also from the perspective of an improved understanding of the geometry of these metrics. For example, is the geometry of the ℓ_1 metric "more amenable" for designing algorithms than the ℓ_2 one? Is the Edit distance metric hard for such problems? We also believe that understanding the geometry of the Ulam and Edit distance metrics, which one may interpret as generalization of the Hamming metric, is not only a very basic computational geometry question, but also would likely lead to better algorithms for these widely-studied problems. We thus ask:

How fast can the 1-center problem be solved or approximated in ℓ_p-metrics and stringology metrics such as Ulam or Edit distance?

1.1 Our Results

In this paper, we take a step towards answering the above question by providing lower and upper bounds on solving the 1-center problem in ℓ_p, Ulam, and Edit distance metrics.

Assuming the Hitting Set Conjecture (HSC), we provide a strong conditional lower bound for the 1-center problem, in a pleathora of metrics.

▶ **Theorem 1** (see Theorem 9, Corollary 12, and Corollary 14 for formal statement). *Assuming* HSC*, no algorithm running in time $n^{2-o(1)}$ can given as input a set of points/strings P of dimension/length d solve the discrete 1-center problem in Edit/Ulam/ℓ_p metric, where $|P| = n$, $d = \widetilde{\Omega}(\log n)$, and $p \in \mathbb{R}_{\geq 1} \cup \{0\}$.*

Moreover, by assuming a stronger complexity theoretic assumption we can strengthen this lower bound in the case when $d = \text{poly}(n)$ for the Edit metric. For the sake of presentation, we state our result below when $d = n$.

▶ **Theorem 2** (see Theorem 15). *Assuming Quantified SETH, no algorithm running in time $n^{4-o(1)}$ can given as input a set of n strings of length $d := n$ each, solve the discrete 1-center problem in Edit metric.*

It's worth emphasizing that the above lower bound for the edit metric is a rare quartic lower bound in fine-grained complexity. It's true that, conceptually, it's not unexpected because there's a quadratic hardness from the 1-center problem and a quadratic hardness from edit distance, so we would expect the combined problem to be quartic. But we find it noteworthy that this actually works on the technical level because complexity theory is full of notorious examples where such "semi-stitching techniques" completely fail (for example KRW games [20]).

Note that we cannot expect such lower bounds for the 1-center problem in ℓ_p-metrics when $d = n$, as one can compute all pairwise distances within a point-set in subcubic time using fast matrix multiplication.

Next, we complement the lower bounds by the following subcubic approximation scheme for the 1-center in the Ulam metric.

▶ **Theorem 3.** *There exists a $1 + \epsilon$ approximation algorithm for the 1-center under Ulam metric that runs in time $\widetilde{O}_\epsilon(nd + n^2\sqrt{d})$.*

It is worth emphasizing here that for the (discrete) 1-center problem in any metric space, an arbitrary point in the input is a 2-approximate solution. Also note that exact 1-center in Ulam metric can be solved in $O(n^2 d)$ time. It remains an open problem to show a conditional lower bound of $n^{3-o(1)}$ for computing the 1-center in the Ulam metric for n strings each of length n.

Finally, we strengthen some of the lower bounds above, but against a weaker class of algorithms, specifically, against algorithms which list all requisite solutions. Using the ideas in [35, 10], assuming HSC, we rule out subquadratic algorithms that can list all optimal solutions to the 1-center problem in the Euclidean metric even for very low $d = o(\log n)$ dimensions. At a high level, this result contrasts with both ℓ_1 and ℓ_∞ metrics where the 1-center in $o(\log n)$ dimensions can be solved in $n^{1+o(1)}$ time.

▶ **Theorem 4** (see Theorem 20 for formal statement). *Assuming* HSC*, there is no $n^{2-o(1)}$-time algorithm listing all optimal solutions to the 1-center problem in $7^{\log^* n}$ dimensions in the Euclidean metric.*

In the same spirit as above, by applying the distributed PCP framework [2, 31] we extend the lower bound in Theorem 1 against approximation algorithms which list all approximately optimal 1-centers.

▶ **Theorem 5** (see Theorem 22 for formal statement). *Assuming* HSC*, there is some $\delta > 0$, such that no $n^{2-o(1)}$-time algorithm can given as input a set of points/strings P of dimension/length d, list all $(1+\delta)$-approximate solutions to the 1-center problem in the Edit/Ulam/ℓ_p metric, where $|P| = n$, $d = \widetilde{\Omega}(\log n)$, and $p \in \mathbb{R}_{\geq 1} \cup \{0\}$.*

One may compute all pairwise distances in $\widetilde{O}(n^2)$ time for the inputs in Theorems 4 and 5, and then obtain the list of all optimal and approximately optimal solutions efficiently. Our theorems above say that one cannot do much better. It remains an intriguing open problem to extend the above two conditional lower bounds but against standard decision algorithms. We note that this involves breaking some technical barriers and in particular, developing techniques that go beyond the dimensionality reduction techniques of [35, 10] and the distributed PCP framework [2, 31] respectively.

We close this subsection by a short discussion about the Discrete 1-median problem in ℓ_p-metrics and string metrics. For the case when $d = n$, we can prove a result similar to Theorem 2 (see Remark 18). On the other hand for ℓ_p-metrics, one cannot prove a result similar to Theorem 1 for the 1-median problem, because the 1-median problem in Hamming and ℓ_1-metrics admits a near linear time algorithm and for the Euclidean metric, it is even unclear if the problem is in NP! (see discussion in [18].) Also note that by subsampling coordinates, we can approximate 1-median in all ℓ_p-metrics to $(1+\varepsilon)$ factor, for any $\varepsilon > 0$ in near linear time.

1.2 Related Work

We now review the related work on the 1-center problem, and the related 1-median problem. Both problems may be considered in the *discrete* or *continuous* settings. The discrete[1] version asks the center or median to be picked from an input set of points, while in the continuous version, the "center" is an arbitrary element of the metric. See [13] for a discussion on these two settings.

Below, we mainly discuss 1-center problem in stringology metrics as the literature on related work in ℓ_p metrics is too vast to survey (but the interested reader may look at [12, 25, 23, 16] and the references therein).

Metrics arising in stringology

We now review results on the 1-center problem in metric spaces arising from stringology applications. Let Σ be an alphabet, often the binary alphabet $\{0,1\}$. Consider the set of strings $\Sigma^* = \Sigma^L$ of length L, with a metric distance $D : \Sigma^* \times \Sigma^* \mapsto \mathbb{R}$. Researchers have mainly considered the following metrics defined over this space:

- **Edit distance** (ED or Levenshtein distance): The minimum number of single-character insertions, deletions, and substitutions required to change one string to the other.
- **Hamming distance or ℓ_1 over binary alphabets** (HD): A special case of edit distance, where only substitutions are allowed.
- **Ulam distance** (UD): Same as edit distance with the restriction that the input strings may not contain any character more than once.

[1] Sometimes called the *medoid* problem in contrast to *median*, *generalized* median, or *Steiner* string.

For most of the above metrics, one need to incorporate into the running times obtained for simpler metric such as ℓ_1 or ℓ_2 the time it takes to compute the exact or approximate distance between any two points of the space. Naumovitz et al. [28] show how to approximate UD within factor $1+\varepsilon$ in time $\widetilde{O}(d/\eta+\sqrt{d})$ if the distance is η. This result is tight up to log factors.

Turning back to the 1-center and 1-median problem, note the *discrete* versions can be solved exactly via $O(n^2)$ distance computations, giving trivial $\widetilde{O}(n^2 d)$-time algorithms for the case of UD. In Section 4, we show that this can be improved to $\widetilde{O}(n^2\sqrt{d})$ for $1+\varepsilon$ approximation if we combine two algorithms [26, 28] for computations of UD. Note that a 2-approximation is trivial, as we can output any string as the hub in the case of 1-center or a random string in the case of 1-median.

Recently [9] made progress on obtaining better approximation algorithms for the continuous 1-Median problem in UD, where the median can be picked from anywhere in space, by presenting the first polynomial-time constant-factor approximation algorithm with approximation guarantee smaller than 2 as well as an exact algorithm for the case where the input contains *three* strings. They observe that if the average distance to median is $\Omega(d)$, picking the best string as the median already gives an approximation better than two. Now the problem is reduced to the above case if the total cost is mostly due to a small subset of characters. Otherwise, they argue that one can deduce the relative ordering of a good portion of the optimal median by looking at pairs of characters whose relative order is consistent in *most* input strings.

Note that in the continuous case, for constant d or constant n, the median and center problems are both solvable in polynomial time for string problems. De la Higuera and Casacuberta [15] prove that median and center are both NP-hard. Nicolas and Rivals [29] lift the restrictions and show that median and center are both NP-hard and $W[1]$-hard (when parameterized by n, the number of strings), even for binary alphabets. Prior to these works, NP-hardness of median was only known for ED when the substitutions have specific costs for each pair of characters [33]. Li et al. [24, 25] give a PTAS for the HD 1-center problem,[2] augmenting the LP-based PTAS for super-logarithmic d [5]. The HD 1-center problem is known to be NP-hard [16, 23].[3] Previously the best polynomial-time approximation ratio was $\frac{4}{3}+\varepsilon$ in general [23, 19], with an exact algorithm known for constant d (optimal value) [34].

1.3 Organization of the Paper

In Section 2, we provide the formal definition of 1-center problem and the various hypotheses used in the paper. In Section 3, we provide conditional lower bounds against exact algorithms that compute 1-center when $d=\omega(\log n)$. Next, in Section 4, we provide a subcubic approximation algorithm for 1-center in Ulam metric when $d=n$. Finally, in Section 5, we provide some hardness of approximation results (Theorem 5).

2 Preliminaries

▶ **Definition 6** (Discrete 1-center). *Let (X,Δ) be a metric space. Given a set of points $P\subseteq X$ in the metric space, find x in P which minimizes the maximum distance to every other point.*

[2] This is the *closest string* problem. They also give a PTAS for the *closest substring* problem, which assumes that the cost of deletions from the input strings (∞ for HD) is zero.

[3] Note that median is solvable exactly for HD.

Perhaps the most popular assumption for proving conditional lower bounds for polynomial time problems is the Orthogonal Vectors Hypothesis (OVH) that is implied by the Strong Exponential Time Hypothesis (SETH). Unfortunately, the logical structure of these problems makes reductions to our 1-center problems difficult. This was observed already by Abboud, Vassilevska Williams, and Wang [3] in the context of 1-center in *graphs* (known as the Graph Radius problem) and has lead them to introduce the hitting set conjecture (HSC): a stronger variant of OVH that facilitates reductions to problems with different structure. A formal barrier for establishing HSC (and similarly also any hardness results for 1-center) under SETH was presented by Carmosino et al. [8].

▶ **Definition 7** (HSC). *For every $\varepsilon > 0$ there exists $c > 1$ such that no algorithm running in time $n^{2-\varepsilon}$ can, given as input two collections of n-many subsets $\mathcal{A}$ and $\mathcal{B}$ of the universe $U := [c \log n]$, determine if there exists S in $\mathcal{A}$ which has non-empty intersection with every subset in $\mathcal{B}$.*

The difference between HSC and OVH is in the quantifiers: $\exists\forall$ versus $\exists\exists$. Studying the *polyline simplification* problem, Bringmann and Chaudhury [6] proposed a further strengthening with more quantifiers. Just like OVH is implied by SETH, an assumption about k-SAT, so too can HSC and its generalizations with more quantifiers be based on the hardness of a quantified version of k-SAT; an assumption called *Quantified-SETH*. Interestingly, the previous papers using Quantified-SETH [6, 1] only needed its special case where the quantifier structure is $\forall\exists$; whereas in this paper we benefit from a $\exists\forall\exists$ structure that has one more alternation.

The specific hardness assumption (implied by Quantified SETH) that we need is the following; we refer to [6, 1] for further discussion on Quantified-SETH and to [3, 36] for further discussion on HSC and on the need for assumptions with other quantifier structures.

▶ **Definition 8** ($\exists\forall\exists\exists$OVH). *For every $\varepsilon > 0$ there exists $c > 1$ such that no algorithm running in time $n^{4-\varepsilon}$ can, given as input four collections of n-many subsets $\mathcal{A}, \mathcal{B}, \mathcal{C}$, and $\mathcal{D}$ of the universe $U := [c \log n]$, determine if there exists S_A in $\mathcal{A}$ such that for all S_B in $\mathcal{B}$ there exist $S_C \in \mathcal{C}$ and $S_D \in \mathcal{D}$ such that the intersection $S_A \cap S_B \cap S_C \cap S_D = \emptyset$ is empty.*

3 Exact Lower Bounds for 1-center

In this section, we prove conditional lower bounds for the 1-center problem. We start with some high-level remarks about the reductions and our contributions.

Previous work (for example [31, 14]) has already designed reductions from SETH and OVH to *closest pair* kinds of questions for the metrics we consider, and our work can be viewed as lifting these results to the 1-center question. As discussed in Section 2 this requires a new starting assumption (either Quantified SETH or the Hitting Set Conjecture) that has a different structure. Thus, technically, the main contribution is to adapt the gadgetry of previous work into new reductions with a different structure. In some cases, fundamental difficulties arise and we can only resolve them by requiring that the algorithm lists all solutions.

In all our reductions, we first reduce to the *Discrete 1-center with Facilities*, where given a set of clients $C \subseteq X$ and a set of facilities $F \subseteq X$ in the metric space, the goal is to find x in F which minimizes the maximum distance to every point in C. We then reduce a hard instance (F, C) of the Discrete 1-center with Facilities problem to an instance P of the standard Discrete 1-center problem (without facilities) by adding a few additional coordinates to points in $F \cup C$ and then introducing a new point/string s such that it is far from every point in C (in comparison to its distance from the points/strings in F). Thus we ensured that the 1-center of $P := F \cup C \cup \{s\}$ must be from F. Nevertheless, for the sake of compactness, this two step reduction in the proofs of this section is sometimes written as a one step reduction.

This section is organized as follows. In Section 3.1, we show the conditional subquadratic time lower bounds for 1-center in various metrics (Theorem 1). Next, in Section 3.2, we show the conditional subquartic time lower bound for 1-center in edit metric (also Theorem 2) and explain how to adapt it for 1-median. Finally, in Section 3.3, we show that there are no subquadratic listing algorithms for Euclidean 1-center even in low dimensions (Theorem 4).

3.1 Subquadratic Lower Bounds for 1-center when $d = \omega(\log n)$ in String and ℓ_p-metrics

In this subsection, we show that subquadratic time algorithms for 1-center do not exist in ℓ_p-metrics, Ulam metic, and edit metric, when $d = \omega(\log n)$.

▶ **Theorem 9** (Subquadratic Hardness of 1-center in ℓ_p-metrics)**.** *Let $p \in \mathbb{R}_{\geq 1} \cup \{0\}$. Assuming* HSC*, for every $\varepsilon > 0$, there exists $c > 1$ such that no algorithm running in time $n^{2-\varepsilon}$ can, given as input a point-set $P \subseteq \{0,1\}^d$, solve the discrete 1-center problem in ℓ_p-metric, where $|P| = n$ and $d = c \log n$.*

Proof. Let $(\mathcal{A} := (S_1, \ldots, S_n), \mathcal{B} := (T_1, \ldots, T_n), U)$ be an instance arising from HSC. We construct a point-set $P \subseteq \{0,1\}^d$ where $|P| = 2n + 1$ and $d = 5 \cdot |U| + 2$. We build the two maps $\tau_{\mathcal{A}} : \mathcal{A} \to \{0,1\}^d, \tau_{\mathcal{B}} : \mathcal{B} \to \{0,1\}^d$ and a special point $s \in \{0,1\}^d$ and the point-set P is then simply defined to be the union of $\{s\}$ and the images (range) of $\tau_{\mathcal{A}}$ and $\tau_{\mathcal{B}}$, i.e.,

$$P := \{\tau_{\mathcal{A}}(S) \mid S \in \mathcal{A}\} \cup \{\tau_{\mathcal{B}}(T) \mid T \in \mathcal{B}\} \cup \{s\}.$$

Let $U := \{u_1, \ldots, u_m\}$. We define our special point s as follows:

$$\forall i \in [5m+2],\ s_i := \begin{cases} 0 & \text{if } 1 \leq i \leq 3m \\ 1 & \text{if } 3m+1 \leq i \leq 5m+2 \end{cases}$$

For any $S \in \mathcal{A}$ we define $\tau_{\mathcal{A}}(S)$ as follows:

$$\forall i \in [5m+2],\ \tau_{\mathcal{A}}(S)_i := \begin{cases} 1 & \text{if } u_i \in S \text{ and } 1 \leq i \leq m \\ 0 & \text{if } u_i \notin S \text{ and } 1 \leq i \leq m \\ 0 & \text{if } u_{i-m} \in S \text{ and } m+1 \leq i \leq 2m \\ 1 & \text{if } u_{i-m} \notin S \text{ and } m+1 \leq i \leq 2m \\ 0 & \text{if } 2m+1 \leq i \leq 4m+1 \\ 1 & \text{if } 4m+2 \leq i \leq 5m+2 \end{cases}$$

For any $T \in \mathcal{B}$ we define $\tau_{\mathcal{B}}(T)$ as follows:

$$\forall i \in [5m+2],\ \tau_{\mathcal{B}}(T)_i := \begin{cases} 1 & \text{if } u_i \in T \text{ and } 1 \leq i \leq m \\ 0 & \text{if } u_i \notin T \text{ and } 1 \leq i \leq m \\ 0 & \text{if } m+1 \leq i \leq 2m \\ 0 & \text{if } u_{i-2m} \in T \text{ and } 2m+1 \leq i \leq 3m \\ 1 & \text{if } u_{i-2m} \notin T \text{ and } 2m+1 \leq i \leq 3m \\ 0 & \text{if } 3m+1 \leq i \leq 5m+2 \end{cases}$$

Notice that for any S, S' in $\mathcal{A}$ and T in $\mathcal{B}$, we have

$$\begin{aligned}\|\tau_{\mathcal{A}}(S)-\tau_{\mathcal{B}}(T)\|_p &= (|S|+|T|-2\cdot|S\cap T|+m-|S|+m-|T|+m+1)^{1/p}\\ &= (3m+1-2\cdot|S\cap T|)^{1/p}\,.\end{aligned}$$

$$\|\tau_{\mathcal{A}}(S)-\tau_{\mathcal{A}}(S')\|_p \le (2m)^{1/p}\,.$$

$$\|\tau_{\mathcal{A}}(S)-s\|_p = (2m+1)^{1/p}\,.$$

$$\|\tau_{\mathcal{B}}(T)-s\|_p = (3m+2)^{1/p}\,.$$

Suppose there exists S in $\mathcal{A}$ such that it intersects with every subset T in $\mathcal{B}$ then $\tau_{\mathcal{A}}(S)$ has distance strictly less than $(3m+1)^{1/p}$ with $\tau_{\mathcal{B}}(T)$ for every T in $\mathcal{B}$. Additionally, $\tau_{\mathcal{A}}(S)$ has distance at most $(2m)^{1/p}$ with $\tau_{\mathcal{A}}(S')$ for any $S' \in \mathcal{A}$ and distance $(2m+1)^{1/p}$ with s. Therefore, $\tau_{\mathcal{A}}(S)$ is at distance at most $(3m)^{1/p}$ from every point in P.

On the other hand, if for every S in $\mathcal{A}$ there exists T in $\mathcal{B}$ such that S and T are disjoint, then we show that for any point x in P there is a point y in P such that $\|x-y\|_p \ge (3m+1)^{1/p}$. Suppose $x := \tau_{\mathcal{B}}(T)$ for some $T \in \mathcal{B}$ then we have x is at distance $(3m+2)^{1/p}$ from s. Similarly if $x := s$ then it is at distance $(3m+2)^{1/p}$ from every $\tau_{\mathcal{B}}(T)$ for all $T \in \mathcal{B}$. Finally, if $x := \tau_{\mathcal{A}}(S)$ for some $S \in \mathcal{A}$ then from the soundness assumption we have that there exists T in $\mathcal{B}$ such that S and T are disjoint. Thus, x is at distance $(3m+1)^{1/p}$ from $\tau_{\mathcal{B}}(T)$. ◀

▶ Remark 10. For the ℓ_∞-metric, we can solve Discrete 1-center problem in $O(nd^2)$ time as follows. Given input point-set P, for every coordinate $i \in [d]$, determine a farthest pair of points (a_i, b_i) in the point-set when restricted to that coordinate. Note that discrete 1-center cost of P is equal to the cost of the discrete 1-center of the point-set $\{a_1, ..., a_d, b_1, ..., b_d\}$ when the center can be picked anywhere in P. Thus we can solve discrete 1-center in the ℓ_∞-metric in $O(nd^2)$ time, which is near linear time as long as $d = n^{o(1)}$.

The quadratic lower bound for 1-center in Ulam metric follows from the below lemma.

▶ **Lemma 11.** *Let Π_d denote the set of all permutations over $[d]$. For every $d \in \mathbb{N}$, there is a function $\eta : \{0,1\}^d \to \Pi_{2d}$, such that for all $a, b \in \{0,1\}^d$ the following holds:*

$$\mathsf{ed}(\eta(a), \eta(b)) = 2\cdot\|a-b\|_0.$$

Moreover, for any $a \in \{0,1\}^d$, $\eta(a)$ can be computed in $O(d)$ time.

Proof. Let $a \in \{0,1\}^d$. We define $\eta(a)$ as follows:

$$\forall i \in [2d],\ \eta(a)[i] = \begin{cases} i & \text{if } i = 2k-1 \text{ and } a_k = 0 \text{ for some } k \in \mathbb{N}\\ i & \text{if } i = 2k \text{ and } a_k = 0 \text{ for some } k \in \mathbb{N}\\ i+1 & \text{if } i = 2k-1 \text{ and } a_k = 1 \text{ for some } k \in \mathbb{N}\\ i-1 & \text{if } i = 2k \text{ and } a_k = 1 \text{ for some } k \in \mathbb{N}\end{cases}$$

Fix some $k \in [d]$ and $a, b \in \{0,1\}^d$. If $a_k = b_k$ then notice that $\eta(a)[2k] = \eta(b)[2k]$ and $\eta(a)[2k-1] = \eta(b)[2k-1]$. If $a_k \ne b_k$ then $\eta(a)[2k] = \eta(b)[2k-1]$ and $\eta(a)[2k-1] = \eta(b)[2k]$. Since the characters do not repeat, we have that the optimal distance is obtained by swapping which amounts to two edit operations. ◀

▶ **Corollary 12** (Subquadratic Hardness of 1-center in Ulam metric). *Assuming* HSC*, for every $\varepsilon > 0$ there exists $c > 1$ such that no algorithm running in time $n^{2-\varepsilon}$ can given as input a set P of n many permutations of $[d]$, solve the discrete 1-center problem in Ulam metric, where $|P| = n$ and $d = c\log n$.*

The quadratic lower bound for 1-center in Edit metric follows from the below lemma.

▶ **Lemma 13.** *For every $d \in \mathbb{N}$, there is a function $\eta : \{0,1\}^d \to \{0,1\}^{d'}$, such that for all $a, b \in \{0,1\}^d$ the following holds:*

$$\mathsf{ed}(\eta(a), \eta(b)) = \|a - b\|_0.$$

Moreover, for any $a \in \{0,1\}^d$, $\eta(a)$ can be computed in $O(d \log d)$ time.

Proof. Let $l_1, l_2, \ldots, l_d$ be d strings of length $10 \log d$ each made by realizing $10 \log d$ 0/1 bits uniformly at random. It follows that with high probability, the hamming distance as well as the edit distance of each pair l_i, l_j $(i \neq j)$ is $\Omega(\log d)$ [22]. For a string a, we define $\eta(a)$ in the following way: we make a string of size $d(10 \log d + 1)$ which consists of d consecutive blocks. Each block i starts with a_i and is followed by l_i. By putting all the blocks next to each other we obtain a string of size $d(10 \log d + 1)$ which we denote by $\eta(a)$. We prove in the following that $\mathsf{ed}(\eta(a), \eta(b)) = \|a - b\|_0$ holds for each pair of strings a and b.

$\mathsf{ed}(\eta(a), \eta(b)) \leq \|a - b\|_0$ immediately follows from the fact that by only toggling the first characters of some block of $\eta(a)$ we can turn $\eta(a)$ into $\eta(b)$ and this transformation only costs $\|a - b\|_0$. Note that we only toggle the first characters of the blocks whose corresponding characters in a and b are not the same.

Now, assume for the sake of contradiction that $\mathsf{ed}(\eta(a), \eta(b)) < \|a - b\|_0$ holds. This implies that for at least $d - \|a - b\|_0 + 1$ many blocks of a, the transformation cost is 0. In other words, for each of these blocks, there is a substring of length $10 \log d + 1$ in $\eta(b)$ which is completely the same as that block. Since the blocks are generated randomly, this can only happen if for some i, the i'th block of $\eta(a)$ is transformed into the i'th block of $\eta(b)$ and $a_i = b_i$. Thus, this implies that for at least $d - \|a - b\|_0 + 1$ different values of i we have $a_i = b_i$ which is contradiction. ◀

▶ **Corollary 14** (Subquadratic Hardness of 1-center in Edit metric). *Assuming* HSC*, for every $\varepsilon > 0$ there exists $c > 1$ such that no algorithm running in time $n^{2-\varepsilon}$ can given as input a point-set $P \subseteq \{0,1\}^d$ solve the discrete 1-center problem in edit metric, where $|P| = n$ and $d = c \log n \log \log n$.*

Next we prove much higher lower bounds for the Edit metric when d is larger.

3.2 Subquartic Lower Bound for 1-center when $d = n$ in Edit metric

We now present our lower bound under Quantified SETH which offers a conceptual novelty since as discussed in Section 2 it is the first time (to our knowledge) that more than two quantifier alternations are utilized.

▶ **Theorem 15** (Subquartic Hardness of 1-center in Edit metric). *Assuming Quantified SETH, for every $\varepsilon > 0$ no algorithm running in time $n^{4-\varepsilon}$ can given as input a point-set $P \subseteq \{0,1\}^n$ solve the discrete 1-center problem in edit metric, where $|P| = n$.*

Proof. Let us first reduce to the 1-center problem with facilities where there are two sets of binary strings, a set of clients C and a set of facilities F and the goal is to decide if there is a string in F that has ED at most τ to all strings in C. Given an instance $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$ of $\exists\forall\exists\exists$OVH we construct C and F as follows.

First we will use the following lemma that follows from the existing reductions from OVH to ED [4, 7] (the latter reference gets the alphabet size down to 2).

▶ **Lemma 16** ([7]). *There are two linear time algorithms such that: each algorithm takes a set (A or B) of n binary vectors of length d and constructs (independently of the other) a binary string (s_A or s_B) of length $O(nd)$ with the following property for a fixed value τ that only depends on n, d: $ED(s_A, s_B) < \tau$ if there is a pair of orthogonal vectors $v_A \in A, v_B \in B$ and $ED(s_A, s_B) \geq \tau$ otherwise.*

For a set $X \subseteq [d]$ let $v(X) \in \{0,1\}^d$ be the natural encoding of the set as a binary vector where $v(X)[i] = 1$ iff $i \in X$. Note that two vectors are orthogonal iff the two corresponding sets are disjoint.

Now, for each set $S_A \in \mathcal{A}$ define the set of n vectors $A = \{v(X) \mid S_C \in \mathcal{C}, X = S_A \cap S_C\}$ representing the n sets that result from intersecting S_A with any set in $\mathcal{C}$. Similarly, for each set $S_B \in \mathcal{B}$ define the set of n vectors $B = \{v(X) \mid S_D \in \mathcal{D}, X = S_B \cap S_D\}$.

It follows that there is an orthogonal pair $v_A \in A, v_B \in B$ iff there exist $S_C \in \mathcal{C}$ and $S_D \in \mathcal{D}$ such that $S_A \cap S_B \cap S_C \cap S_D = \emptyset$. Therefore, if we use the algorithms in the above lemma to encode each set A with a string s_A and add it to the set of facilities F, and also encode each set B with a string s_B and add it into the set of clients C we get the reduction we are after: By the definition of the ∃∀∃∃OVH problem, there is a string $s_A \in F$ such that for all strings $s_B \in C$ we have $ED(s_A, s_B) < \tau$ if and only if the given ∃∀∃∃OVH instance is a yes-instance.

Finally, we reduce to the basic 1-center problem (without facilities). Suppose that the strings in F, C have length m. We simply construct an instance $P \subseteq \{0,1\}^{4m}$ of 1-center as follows.

$$P := \{1^m \circ f \circ 0^{2m} \mid f \in F\} \cup \{1^m \circ c \circ 1^{2m} \mid c \in C\} \cup \{0^{4m}\}.$$

The following simple facts about the ED of the transformed strings show that the optimal center in P must be from $\{1^m \circ f \circ 0^{2m} \mid f \in F\}$ and its cost would be smaller than $2m + \tau$ iff the original ∃∀∃∃OVH instance is a yes-instance.

▷ Claim 17. Let x, y be two binary strings of length m with ED exactly t.

- $ED(1^m \circ x \circ 0^{2m}, 1^m \circ y \circ 0^{2m}) \leq m$.
- $ED(1^m \circ x \circ 0^{2m}, 0^{4m}) \leq 2m$.
- $ED(1^m \circ x \circ 0^{2m}, 1^m \circ y \circ 1^{2m}) = 2m + t$.
- $ED(1^m \circ x \circ 1^{2m}, 0^{4m}) \geq 3m$.

The first and second items follow from the straightforward alignment of the strings. The fourth item follows because $ED(0^\ell, 1^\ell) = \ell$. The third item requires a bit more care. First, to see that the ED is at most $2m + t$ consider the alignment that maps x to y optimally at cost t and then maps the other parts in the straightforward way at cost $2m$. Now suppose for contradiction that there was a better alignment. This alignment must match one of the new letters (from the transformation) to x or y; otherwise it would yield an alignment between x, y at cost smaller than t. But any alignment that matches the 1 letters on the left to x or y can be corrected so that the 1^m parts on the left are matched to each other, without affecting the cost. Similarly, any matching between the letters on the right to x or y can be corrected without increasing the cost. Suppose that a 0 from the right is matched to y. This implies that one of the 1's to the right of y must be deleted (because there are no longer enough 0's in the other string to get substituted with all of them), and a corrected alignment that instead substitutes the 0 with a 1 (reducing the number of such deletions by one) and leaves the mate in y unmatched does not have a higher cost. We refer the reader to [7] for more formal proofs of such claims. ◀

▶ Remark 18. The above reduction to Edit also work for the 1-median problem but with two key differences. The first and main difference is that, since we take the sum instead of the max, the cost in the objective may now be affected by non-orthogonal pairs and it is no longer sufficient to have gadgets that give distance $< \tau$ or $\geq \tau$ depending on the orthogonality. Instead, we need gadgets that guarantee that the distance is either $< \tau$ or exactly τ. Fortunately, such requirements can be accomplished, see e.g. Theorem 4 in [4]. The second difference is that we do not need the $\forall$ quantifier in the starting assumption; the sum is powerful enough to support the (standard) $\exists\exists\exists\exists$ structure type. Therefore, the lower bounds for 1-median can be based on the standard SETH rather than the Quantified-SETH.

3.3 Subquadratic Lower Bounds for 1-center in Low dimensional Euclidean space

In this subsection, we show that an algorithm with subquadratic running time does not exist in the low dimensional Euclidean metric for the 1-center problem. Our proof essentially adopts ideas developed in [35, 10]. We note that this result is surprising as there is a near linear time algorithm for 1-center in the low dimensional ℓ_1-metric.

▶ Remark 19. For the ℓ_1-metric, we can solve Discrete 1-center problem in $O(n2^{2d})$ time by using the isometric embedding of the ℓ_1-metric to the ℓ_∞-metric [27], and then noting Remark 10.

▶ **Theorem 20.** *Assuming* HSC*, there exists a constant $\eta > 1$ such that for every $\varepsilon > 0$, no algorithm running in time $n^{2-\varepsilon}$ can given as input a point-set $P \subseteq \mathbb{R}^d$ and a positive real α output all points in P whose 1-center cost in the Euclidean metric is at most α, where $|P| = n$, $d = \eta^{\log^* n}$, and representing each vector requires at most $\widetilde{O}(\log n)$ bits.*

Proof of Theorem 20. We prove the theorem statement by contradiction. Suppose for some $\varepsilon > 0$ there is an algorithm $\mathcal{T}$ running in time $n^{2-\varepsilon}$ that can given as input a point-set $P \subseteq \mathbb{R}^d$ and a positive real α output all points in P whose 1-center cost in the Euclidean metric is at most α, where $|P| = n$, $d = \eta^{\log^* n}$, and each vector is of at most $k \log n$ bit entries (for some constant integer k).

Let $(\mathcal{A} := (S_1, \ldots, S_n), \mathcal{B} := (T_1, \ldots, T_n), U)$ be an instance arising from HSC, where $|U| = c \log n$. We think of each set in $\mathcal{A}$ and $\mathcal{B}$ as its characteristic vector in $\{0,1\}^{c \log n}$. We show how we can decide this instance in $n^{2-\frac{\varepsilon}{2}}$ time using $\mathcal{T}$, thus contradicting HSC.

We need the following theorem from Chen [10].

▶ **Theorem 21** (Chen [10])**.** *Let b, ℓ be two sufficiently large integers. There is a reduction $\psi_{b,\ell} : \{0,1\}^{b\cdot\ell} \to \mathbb{Z}^\ell$ and a set $V_{b,\ell} \subseteq \mathbb{Z}$, such that for every $x, y \in \{0,1\}^{b\cdot\ell}$,*

$$x \cdot y = 0 \Leftrightarrow \psi_{b,\ell}(x) \cdot \psi_{b,\ell}(y) \in V_{b,\ell}$$

and

$$0 \leq \psi_{b,\ell}(x)_i < \ell^{6^{\log^*(b)}\cdot b}$$

for all possible x and $i \in [\ell]$. Moreover, the computation of $\psi_{b,\ell}(x)$ takes $\mathrm{poly}(b \cdot \ell)$ time, and the set $V_{b,\ell}$ can be constructed in $O\left(\ell^{O(6^{\log^*(b)}\cdot b)} \cdot \mathrm{poly}(b \cdot \ell)\right)$ time.

We use the above theorem with $\ell = 7^{\log^* n}$ and $b = |U|/\ell$. Note that if $\ell = 7^{\log^* n}$ then $\log\left(\ell^{6^{\log^*(b)}\cdot b}\right) = o(\log n)$. All of the below construction details appears in [35, 10] and we skip many of the calculations and claim proofs hereafter. Our contributions are mainly in using these previously known constructions in a new way to prove the theorem statement. In particular, for every $t \in V_{b,\ell}$ we create an instance $(P_t \subseteq \mathbb{R}^{(\ell+1)^2+3}, \alpha := \sqrt{2n^5-1})$ of 1-center as follows.

For every[4] $S_i \in \mathcal{A}$ (resp. $T_j \in \mathcal{B}$) we first define a point $p_i^t \in \mathbb{Z}^{\ell+1}$ (resp. $q_j^t \in \mathbb{Z}^{\ell+1}$) as follows:

$$p_i^t := (\psi_{b,\ell}(S_i), t) \text{ (resp. } q_j^t := (\psi_{b,\ell}(T_j), -1)).$$

It is then easy to verify that $S_i \cap T_j = \emptyset$ if and only if there exists some $t \in V_{b,\ell}$ such that $\langle p_i^t, q_j^t \rangle = 0$. Next for every $p_i^t \in \mathbb{Z}^{\ell+1}$ (resp. $q_j^t \in \mathbb{Z}^{\ell+1}$) we define $\widetilde{p}_i^t \in \mathbb{Z}^{(\ell+1)^2}$ (resp. $\widetilde{q}_j^t \in \mathbb{Z}^{(\ell+1)^2}$) as follows:

$$\forall a, b \in [\ell+1],\ \widetilde{p}_i^t(a,b) := p_i^t(a) \cdot p_i^t(b) \text{ (resp. } \widetilde{q}_j^t(a,b) := -q_j^t(a) \cdot q_j^t(b)).$$

It is then straightforward to verify that $\langle p_i^t, q_j^t \rangle = 0$ if and only if $\langle \widetilde{p}_i^t, \widetilde{q}_j^t \rangle \geq 0$.

Finally, we have our pointset $P_t \in \mathbb{R}^{(\ell+1)^2+3}$ defined as follows:

$$P_t := \underbrace{\left\{ \left(\widetilde{p}_i^t, \sqrt{n^5 - \|\widetilde{p}_i^t\|_2^2}, 0, 0 \right) \middle| S_i \in \mathcal{A} \right\}}_{P_t^{\mathcal{A}}} \bigcup \underbrace{\left\{ \left(-\widetilde{q}_j^t, 0, \sqrt{n^5 - \|\widetilde{q}_j^t\|_2^2}, \sqrt{n^5} \right) \middle| T_j \in \mathcal{B} \right\}}_{P_t^{\mathcal{B}}} \cup \{\vec{0}\},$$

where $\vec{0} = (0, 0, \ldots, 0)$.

It can then be verified that $\langle \widetilde{p}_i^t, \widetilde{q}_j^t \rangle \geq 0$ if and only if the distance between $\left(\widetilde{p}_i^t, \sqrt{n^5 - \|\widetilde{p}_i^t\|_2^2}, 0 \right)$ and $\left(-\widetilde{q}_j^t, 0, \sqrt{n^5 - \|\widetilde{q}_j^t\|_2^2} \right)$ is at least $\sqrt{2n^5}$; otherwise their distance is at most $\sqrt{2n^5-1}$. Also note that any pair of points in $P_t^{\mathcal{A}}$ or any pair of points in $P_t^{\mathcal{B}}$ are at distance at most $\sqrt{2n^5-1}$ from each other. Finallt, note that the distance between any point in $P_t^{\mathcal{B}}$ and $\vec{0}$ is exactly $\sqrt{2n^5}$ and the distance between any point in $P_t^{\mathcal{A}}$ and $\vec{0}$ is exactly $\sqrt{n^5}$.

We run $\mathcal{T}$ on $(P_t, \alpha := \sqrt{2n^5-1})$ for every $t \in V_{b,\ell}$. Let $\mathcal{O}_t \subseteq P_t$ be the output of running $\mathcal{T}$ on (P_t, α). In other words for every $t \in V_{b,\ell}$ and every $p \in \mathcal{O}_t$ we have that for every $p' \in P_t$, $\|p - p'\|_2 \leq \sqrt{2n^5-1}$.

We claim that there exists S in $\mathcal{A}$ such that it intersects with every subset T in $\mathcal{B}$ if and only if there exists $i \in [n]$ such that for all $t \in V_{b,\ell}$, we have $\left(\widetilde{p}_i^t, \sqrt{n^5 - \|\widetilde{p}_i^t\|_2^2}, 0 \right) \in \mathcal{O}_t$.

Suppose there exists S_{i^*} in $\mathcal{A}$ such that it intersects with every subset T in $\mathcal{B}$. Fix $t \in V_{b,\ell}$. We have that $\left(\widetilde{p}_{i^*}^t, \sqrt{n^5 - \|\widetilde{p}_{i^*}^t\|_2^2}, 0 \right)$ is at distance at most $\sqrt{2n^5-1}$ from every other point in $P_t^{\mathcal{A}}$ just from construction. Suppose there is $\left(-\widetilde{q}_j^t, 0, \sqrt{n^5 - \|\widetilde{q}_j^t\|_2^2} \right) \in P_t^{\mathcal{B}}$ such that their distance is greater than $\sqrt{2n^5-1}$ then from the construction, their distance must be $\sqrt{2n^5}$, which implies that $S_{i^*} \cap T_j = \emptyset$, a contradiction.

On the other hand, if for every S in $\mathcal{A}$ there exists T in $\mathcal{B}$ such that S and T are disjoint then we show that for any point $\left(\widetilde{p}_i^t, \sqrt{n^5 - \|\widetilde{p}_i^t\|_2^2}, 0 \right)$ there exists $t \in V_{b,\ell}$ such that $\left(\widetilde{p}_i^t, \sqrt{n^5 - \|\widetilde{p}_i^t\|_2^2}, 0 \right) \notin \mathcal{O}_t$. Fix $i \in [n]$. Let $T_j \in \mathcal{B}$ such that $S_i \cap T_j = \emptyset$. Let $t^* := \psi_{b,\ell}(S_i) \cdot \psi_{b,\ell}(T_j)$. From Theorem 21 we have that $t^* \in V_{b,\ell}$. Thus $\left(\widetilde{p}_i^{t^*}, \sqrt{n^5 - \|\widetilde{p}_i^{t^*}\|_2^2}, 0 \right)$ and $\left(-\widetilde{q}_j^{t^*}, 0, \sqrt{n^5 - \|\widetilde{q}_j^{t^*}\|_2^2} \right)$ in P_{t^*} are at distance at least $\sqrt{2n^5}$ and thus $\left(\widetilde{p}_i^{t^*}, \sqrt{n^5 - \|\widetilde{p}_i^{t^*}\|_2^2}, 0 \right) \notin \mathcal{O}_{t^*}$.

Finally, note that the total run time was $O(n^{2-\varepsilon} \cdot |V_{b,\ell}|) = O(n^{2-\varepsilon} \log n) < n^{2-\frac{\varepsilon}{2}}$. ◀

[4] Recall that we think of S_i and T_j through their characteristic vector.

4 An $n^{2.5}$ time $1+\epsilon$ Approximation Algorithm for 1-Center in Ulam Metric when $d = n$

In this section, we consider the 1-center problem under Ulam metric. More precisely, we consider a problem where n strings $s_1, s_2, \ldots, s_n$ are given as input and our goal is to find a string s_k such that the maximum distance of s_k from the rest of the strings is minimized. Our focus here is on the Ulam metric.

We assume throughout this section that the length of all strings is equal to d. Our algorithm for this case is two-fold. Let o be the value of the solution (i.e., the maximum distance of the center of the strings to the rest of the strings is exactly equal to o). If o is lower bounded by $\sqrt{d}$, previous work on Ulam distance gives us a $1+\epsilon$ approximate solution for center in the following way: We iterate over all pairs of strings and each time we estimate their Ulam distance via the algorithm of Naumovitz, Saks, and Seshadhri [28] for approximating the Ulam distance of each pair. When the Ulam distance of two strings is equal to u, their algorithm takes time $\widetilde{O}_\epsilon(d/u + \sqrt{d})$ to $1+\epsilon$ approximate the solution. Thus, we only run their algorithm up to a runtime of $\widetilde{O}_\epsilon(\sqrt{d})$ to either obtain a $1+\epsilon$ approximate solution for the Ulam distance or verify that the Ulam distance is smaller than $\sqrt{d}$. It follows that if $o \geq \sqrt{d}$ this information is enough for us to approximate the 1-center problem within a factor $1+\epsilon$ and the runtime of the algorithm is bounded by $\widetilde{O}_\epsilon(n^2\sqrt{d})$. Thus, it only remains to design an algorithm for the low-distance regime.

From here on, we assume that $o \leq \sqrt{d}$. In this case, we take an arbitrary string (say s_1) and compute the Ulam distance of that string to all the other strings. In addition to this, we also keep track of the changes that convert s_1 into all the strings. It follows that since $o \leq \sqrt{d}$, the distance of s_1 to all the strings is bounded by at most $2\sqrt{d}$. Thus, via the transformations we compute in this step, we would be able to make a transformation from any s_i to any s_j with at most $4\sqrt{d}$ operations (we can combine the transformation from s_1 to s_i and the transformation from s_1 to s_j). Using this information, we can determine the exact Ulam distance of every string s_i to every string s_j in the following way:

We start with the non-optimal transformation from s_i to s_j that uses at most $4\sqrt{d}$ operations. We then split the characters from $[1, \ldots, d]$ into buckets such that in each bucket all the characters are next to each other and they appear in the same order in the two strings. To be more precise, consider the following procedure: color each character of s_i and s_j which is touched in the transformation (deleted, added, or changed) in red and the rest blue. Each set of consecutive blue characters and each single red character makes a bucket. It follows that because there is a transformation from s_i to s_j with at most $4\sqrt{d}$ operations, the total number of buckets would be bounded by $O(\sqrt{d})$. Moreover, there exists an optimal transformation wherein either all characters of each buckets are deleted/inserted or all characters of each bucket remain intact. This implies that we can compress the two strings into smaller strings by replacing each bucket with a single character. The insertion and deletion of these special characters then has a cost proportional to the size of the bucket. This way, the size of the two strings would be bounded by $O(\sqrt{d})$ and thus we can compute the Ulam distance of the two strings in time $\widetilde{O}(\sqrt{d})$. Therefore, we can compute the center of the strings in time $\widetilde{O}_\epsilon(nd + n^2\sqrt{d})$.

▶ **Theorem 3.** *There exists a $1+\epsilon$ approximation algorithm for the 1-center under Ulam metric that runs in time $\widetilde{O}_\epsilon(nd + n^2\sqrt{d})$.*

Proof of Theorem 3. The outline of the algorithm along with its runtime analysis is given earlier. Here we prove that the approximation factor of the algorithm is bounded by $1+\epsilon$. In case $o \geq \sqrt{d}$, we use the $1+\epsilon$ approximation algorithm of Naumovitz, Saks, and Seshadhri [28]

Algorithm 1 1-center of n strings under Ulam metric.

```
Data: $s_1, s_2, \ldots, s_n$
Result: 1-center
$o \leftarrow \infty$;
for $i \leftarrow 1$ to $n$ do
    $mx \leftarrow -1$;
    for $j \leftarrow 1$ to $n$ do
        Run [28] on $s_i$ and $s_j$ up to $\widetilde{O}_\epsilon(\sqrt{d})$ steps;
        if the algorithm terminates then
            $mx \leftarrow \max\{mx, \text{the output of the algorithm}\}$;
        end
    end
    $o \leftarrow \min\{o, mx\}$;
end
if $o \neq -1$ then
    return o;
end
else
    $o \leftarrow \infty$;
    for $i \leftarrow 1$ to $n$ do
        $tr_i \leftarrow$ optimal transformation between $s_1$ and $s_i$;
    end
    for $i \leftarrow 1$ to $n$ do
        $mx \leftarrow -1$;
        for $j \leftarrow 1$ to $n$ do
            $(s_i^*, s_j^*) \leftarrow$ compressed versions of $(s_i, s_j)$ based on $tr_i$ and $tr_j$;
            $mx \leftarrow \max\{mx, \text{the Ulam distance of } s_i^* \text{ and } s_j^*\}$;
        end
        $o \leftarrow \min\{o, mx\}$;
    end
    return $o$;
end
```

for each pair of strings up to a runtime of $\widetilde{O}_\epsilon(\sqrt{d})$. If the algorithm gives an estimation before we terminate it, we take the value into account when determining the maximum distance for the strings involved. It follows that since $o \geq \sqrt{d}$, then for each string s_x, there is one string s_y whose distance to s_x is at least $\sqrt{d}$ and thus the maximum distance we determine for each string is a $1+\epsilon$ approximation of the optimal value.

Next, we show that in case $o \leq \sqrt{d}$, our algorithm determines the Ulam distance of each pair exactly and thus solves the 1-center problem correctly. In order to determine the Ulam distance between s_i and s_j, we begin with a transformation of size at most $4\sqrt{d}$ between the two strings. We then mark all the characters that are either deleted or inserted in this transformation and all the characters that are next to these characters. We then compress the two strings in the following way: each marked character becomes a single character with the same value. Each maximal interval of unmarked characters that are next to each other also become a single character whose value represents the entire interval. Therefore, the compressed strings have $O(\sqrt{d})$ characters each. It follows that the Ulam distance of the compressed strings is exactly equal to the Ulam distance of the original strings. Moreover, even though for the compressed strings the operations have arbitrary costs, we can still solve Ulam distance in time proportional to the length of the strings which results in an algorithm with runtime $\widetilde{O}(\sqrt{d})$ for computing Ulam distance between each pair. ◀

5 Hardness of Approximation of 1-center in String and ℓ_p-metrics

In this section, we prove hardness of approximation results for the 1-center problem. A background on error correcting codes is detailed in Appendix A.

▶ **Theorem 22** (Subquadratic Inapproximability of 1-center in ℓ_p-metrics)**.** *Let $p \in \mathbb{R}_{\geq 1} \cup \{0\}$. Assuming* HSC*, for every $\varepsilon > 0$ there exists $\delta > 0$ such that no algorithm running in time $n^{2-\varepsilon}$ can given as input a point-set $P \subseteq \{0,1\}^d$ and a positive real α output all points in P whose 1-center cost in the ℓ_p-metric is at most $\alpha \cdot (1+\delta)$, where $|P| = n$ and $d = O_\varepsilon(\log n)$.*

Proof. Fix $p \in \mathbb{R}_{\geq 1} \cup \{0\}$. We prove the theorem statement by contradiction. Suppose for some $\varepsilon > 0$ there is an algorithm $\mathcal{T}$ running in time $n^{2-\varepsilon}$ that can for every $\delta > 0$, given as input a point-set $P \subseteq \mathbb{R}^d$ and a positive real α output all points in P whose 1-center cost in the ℓ_p-metric is at most $\alpha \cdot (1+\delta)$, where $|P| = n$ and $d = O_\varepsilon(\log n)$ (dependency on ε will become clear later in the proof).

Let $(\mathcal{A} := (S_1, \ldots, S_n), \mathcal{B} := (T_1, \ldots, T_n), U)$ be an instance arising from HSC, where $|U| = c \log n$. We think of each set in $\mathcal{A}$ and $\mathcal{B}$ as its characteristic vector in $\{0,1\}^{c \log n}$. We show how we can decide this instance in $n^{2-\frac{\varepsilon}{2}}$ time using $\mathcal{T}$, thus contradicting HSC.

The construction below is exactly the same as the one suggested by Rubinstein [31]. We however, use this construction to fit our purposes of proving lower bound for the 1-center problem.

Algebrization. Fix $T = 2c/\varepsilon$. Let q be the smallest prime greater than T (i.e., $q < 2 \cdot T$). Let $m := c \log n$. Let $C^1_{m/T}$ and $C^2_{m/T}$ be the codes guaranteed in Theorem 26 over $\mathbb{F}_{q^2}$ with block length $\ell \leq \lambda m/T$.

Let $\widetilde{C} \subseteq C^2_{m/T}$ such that $\omega \in \widetilde{C}$ if and only if $\omega\ |_{[m/T]} = \vec{0}$ (i.e., ω has a zero entry in each of the first m/T coordinates). For every $\omega \in \widetilde{C}$ we define two functions $\tau^\omega_{\mathcal{A}}, \tau^\omega_{\mathcal{B}} : \{0,1\}^m \to \{0,1\}^r$, where $r := q^{4(T+2)} \times \ell$. We can thus index every $i \in [r]$ using elements in $\mathbb{F}^T_{q^2} \times \mathbb{F}^T_{q^2} \times [\ell]$.

Fix $x \in \{0,1\}^m$. Let $x = (x^1, \ldots, x^T) \in \{0,1\}^m$ where for all $i \in [T]$ we have $x^i \in \{0,1\}^{m/T}$. We define $\tau^\omega_{\mathcal{A}}(x)$ coordinate wise. Fix $\zeta \in [r]$ and we think of ζ as follows:

$$\zeta = \big((\mu^{\mathcal{A}}_1, \ldots, \mu^{\mathcal{A}}_{T+2}), (\mu^{\mathcal{B}}_1, \ldots, \mu^{\mathcal{B}}_{T+2}), j\big) \in \mathbb{F}^{T+2}_{q^2} \times \mathbb{F}^{T+2}_{q^2} \times [\ell]. \tag{1}$$

We define $\tau^\omega_{\mathcal{A}}(x)_\zeta$ to be 1 if and only if:

$$\sum_{i \in [T+2]} \mu^{\mathcal{A}}_i \cdot \mu^{\mathcal{B}}_i = \omega(j) \quad \text{and} \quad \forall i \in [T],\ \mu^{\mathcal{A}}_i = C^1_{m/T}(x^i)_j, \text{ and } \mu^{\mathcal{A}}_{T+1} = 0,\ \mu^{\mathcal{A}}_{T+2} = C^1_{m/T}(\mathbb{1}^{m/T})_j.$$

Similarly, we define $\tau^\omega_{\mathcal{B}}(x)$ coordinate wise. Fix $\zeta \in [r]$ and we think of ζ as in (1). We define $\tau^\omega_{\mathcal{B}}(x)_\zeta$ to be 1 if and only if:

$$\sum_{i \in [T+2]} \mu^{\mathcal{A}}_i \cdot \mu^{\mathcal{B}}_i = \omega(j) \quad \text{and} \quad \forall i \in [T],\ \mu^{\mathcal{B}}_i = C^1_{m/T}(x^i)_j \text{ and } \mu^{\mathcal{B}}_{T+1} = C^1_{m/T}(\mathbb{1}^{m/T})_j,\ \mu^{\mathcal{B}}_{T+2} = 0.$$

Construction. For every $S_i \in \mathcal{A}$, we define $s_i^\omega := \tau_{\mathcal{A}}^\omega(S_i)$. Further, we define $\widetilde{s}_i^\omega = (s_i^\omega, \mathbb{1}^r - s_i^\omega, \mathbb{1}^{2r}) \in \{0,1\}^{4r}$. For every $T_j \in \mathcal{B}$, we define $t_j^\omega := \tau_{\mathcal{B}}^\omega(T_j)$. Further, we define $\widetilde{t}_j^\omega = (\mathbb{1}^r - t_j^\omega, t_j^\omega, 0^{2r}) \in \{0,1\}^{4r}$.

We define the point-set P_ω to be $P_\omega^{\mathcal{A}} := \{\widetilde{s}_i^\omega \mid S_i \in \mathcal{A}\} \cup P_\omega^{\mathcal{B}} := \{\widetilde{t}_j^\omega \mid T_j \in \mathcal{B}\} \cup \{\mathbb{1}^{4r}\}$. Let $\alpha := (2q^{4(T+2)} - 4q^{2(T+1)}) \cdot \ell + 2r + \ell$. Let $\delta := 1/(4q^{4T} - 4q^{2T-2} + 1)$.

Analysis. We run $\mathcal{T}$ on (P_ω, α) for every $\omega \in \widetilde{C}$. Let $\mathcal{O}_\omega \subseteq P_\omega$ be the output of running $\mathcal{T}$ on (P_ω, α). In other words for every $\omega \in \widetilde{C}$ and every $s \in \mathcal{O}_t$ we have that for every $s' \in P_\omega$, $\|s - s'\|_p \leq (1+\delta)^{1/p} \cdot \alpha^{1/p}$.

We claim that there exists S in $\mathcal{A}$ such that it intersects with every subset T in $\mathcal{B}$ if and only if there exists $i \in [n]$ such that for all $\omega \in \widetilde{C}$, we have $\widetilde{s}_i^\omega \in \mathcal{O}_\omega$.

Suppose there exists S_{i^*} in $\mathcal{A}$ such that it intersects with every subset T in $\mathcal{B}$. Fix $\omega \in \widetilde{C}$. We have that $\widetilde{s}_{i^*}^\omega$ is at distance at most $(2r)^{1/p}$ from every other point in $P_\omega^{\mathcal{A}}$ just from construction. Suppose there is $\widetilde{t}_j^\omega \in P_\omega^{\mathcal{B}}$ such that their distance is greater than $\alpha^{1/p}$ then from the construction, their distance must be at least $(1+\delta)^{1/p} \cdot \alpha^{1/p}$, which implies that $S_{i^*} \cap T_j = \emptyset$, a contradiction.

On the other hand, if for every S in $\mathcal{A}$ there exists T in $\mathcal{B}$ such that S and T are disjoint then we show that for any point $\widetilde{s}_i^\omega$ there exists $\omega \in \widetilde{C}$ such that $\widetilde{s}_i^\omega \notin \mathcal{O}_\omega$. Fix $i \in [n]$. Let $T_j \in \mathcal{B}$ such that $S_i \cap T_j = \emptyset$. Let $\omega^* := \sum_{e \in [T]} C^1_{m/T}(S_i^e) \cdot C^1_{m/T}(T_j^e)$. From Theorem 26 we have that $\omega^* \in \widetilde{C}$. Thus $\widetilde{s}_i^{\omega^*}$ and $\widetilde{t}_j^{\omega^*}$ in P_{ω^*} are at distance at least $(1+\delta)^{1/p} \cdot \alpha^{1/p}$ and thus $\widetilde{s}_i^{\omega^*} \notin \mathcal{O}_{\omega^*}$. Also, note that for all $\omega \in \widetilde{C}$, we have that every point in $P_\omega^{\mathcal{B}}$ is at distance at least $(3r)^{1/p}$ from $\mathbb{1}^{4r}$.

Finally, note that the total run time was $O(n^{2-\varepsilon} \cdot |\widetilde{C}|) = O(n^{2-\varepsilon+\frac{c}{T}}) < n^{2-\frac{\varepsilon}{2}}$. ◀

We remark that the above construction is the same as the one in [31] albeit for a different problem (nearest neighbors problem).

Theorem 22 readily extend to the edit metric from the below statement and to the Ulam metric from Lemma 11.

▶ **Lemma 23** (Rubinstein [31]). *For large enough $d \in \mathbb{N}$, there is a function $\eta : \{0,1\}^d \to \{0,1\}^{d'}$, where $d' = O(d \log d)$, such that for all $a, b \in \{0,1\}^d$ the following holds for some constant $\lambda > 0$:*

$$|\mathsf{ed}(\eta(a), \eta(b)) - \lambda \cdot \log d \cdot \|a - b\|_0| = o(d').$$

Moreover, for any $a \in \{0,1\}^d$, $\eta(a)$ can be computed in $2^{o(d)}$ time.

References

1 Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. Scheduling lower bounds via AND subset sum. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 4:1–4:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.4.

2 Amir Abboud, Aviad Rubinstein, and R. Ryan Williams. Distributed PCP theorems for hardness of approximation in P. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 25–36, 2017. doi:10.1109/FOCS.2017.12.

3 Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. 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 377–391. SIAM, 2016. doi:10.1137/1.9781611974331.ch28.

4 Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 51–58. ACM, 2015. doi:10.1145/2746539.2746612.

5 Amir Ben-Dor, Giuseppe Lancia, Jennifer Perone, and R. Ravi. Banishing bias from consensus sequences. In Alberto Apostolico and Jotun Hein, editors, *Combinatorial Pattern Matching, 8th Annual Symposium, CPM 97, Aarhus, Denmark, June 30 - July 2, 1997, Proceedings*, volume 1264 of *Lecture Notes in Computer Science*, pages 247–261. Springer, 1997. doi:10.1007/3-540-63220-4_63.

6 Karl Bringmann and Bhaskar Ray Chaudhury. Polyline simplification has cubic complexity. In Gill Barequet and Yusu Wang, editors, *35th International Symposium on Computational Geometry, SoCG 2019, June 18-21, 2019, Portland, Oregon, USA*, volume 129 of *LIPIcs*, pages 18:1–18:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.SoCG.2019.18.

7 Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 79–97. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.15.

8 Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In Madhu Sudan, editor, *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 261–270. ACM, 2016. doi:10.1145/2840728.2840746.

9 Diptarka Chakraborty, Debarati Das, and Robert Krauthgamer. Approximating the median under the ulam metric. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 761–775. SIAM, 2021. doi:10.1137/1.9781611976465.48.

10 Lijie Chen. On the hardness of approximate and exact (bichromatic) maximum inner product. *Theory Comput.*, 16:1–50, 2020. doi:10.4086/toc.2020.v016a004.

11 Kenneth L. Clarkson, Elad Hazan, and David P. Woodruff. Sublinear optimization for machine learning. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 449–457. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.50.

12 Michael B. Cohen, Yin Tat Lee, Gary L. Miller, Jakub Pachocki, and Aaron Sidford. Geometric median in nearly linear time. 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 9–21. ACM, 2016. doi:10.1145/2897518.2897647.

13 Vincent Cohen-Addad, Karthik C. S., and Euiwoong Lee. On approximability of clustering problems without candidate centers. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2635–2648. SIAM, 2021. doi:10.1137/1.9781611976465.156.

14 Roee David, Karthik C. S., and Bundit Laekhanukit. On the complexity of closest pair via polar–pair of point–sets. *SIAM J. Discrete Math.*, 33(1):509–527, 2019. doi:10.1137/17M1128733.

15 Colin de la Higuera and Francisco Casacuberta. Topology of strings: Median string is np-complete. *Theoretical Computer Science*, 230(1):39–48, 2000. doi:10.1016/S0304-3975(97)00240-5.

16 Moti Frances and Ami Litman. On covering problems of codes. *Theory Comput. Syst.*, 30(2):113–119, 1997. doi:10.1007/s002240000044.

17 Arnaldo Garcia and Henning Stichtenoth. On the asymptotic behaviour of some towers of function fields over finite fields. *Journal of Number Theory*, 61(2):248–273, 1996. doi:10.1006/jnth.1996.0147.

18 M. R. Garey, Ronald L. Graham, and David S. Johnson. Some np-complete geometric problems. In Ashok K. Chandra, Detlef Wotschke, Emily P. Friedman, and Michael A. Harrison, editors, *Proceedings of the 8th Annual ACM Symposium on Theory of Computing, May 3-5, 1976, Hershey, Pennsylvania, USA*, pages 10–22. ACM, 1976. doi:10.1145/800113.803626.

19 Leszek Gasieniec, Jesper Jansson, and Andrzej Lingas. Efficient approximation algorithms for the hamming center 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 905–906. ACM/SIAM, 1999. URL: http://dl.acm.org/citation.cfm?id=314500.315081.

20 Mauricio Karchmer, Ran Raz, and Avi Wigderson. Super-logarithmic depth lower bounds via direct sum in communication coplexity. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference, Chicago, Illinois, USA, June 30 - July 3, 1991*, pages 299–304. IEEE Computer Society, 1991. doi:10.1109/SCT.1991.160273.

21 Karthik C. S., Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. *J. ACM*, 66(5):33:1–33:38, 2019. doi:10.1145/3325116.

22 Marcos Kiwi and José Soto. On a speculated relation between chvátal–sankoff constants of several sequences. *Combinatorics, Probability and Computing*, 18(4):517–532, 2009.

23 J. Kevin Lanctôt, Ming Li, Bin Ma, Shaojiu Wang, and Louxin Zhang. Distinguishing string selection problems. *Inf. Comput.*, 185(1):41–55, 2003. doi:10.1016/S0890-5401(03)00057-9.

24 Ming Li, Bin Ma, and Lusheng Wang. Finding similar regions in many sequences. *J. Comput. Syst. Sci.*, 65(1):73–96, 2002. doi:10.1006/jcss.2002.1823.

25 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.

26 Michael Mitzenmacher and Saeed Seddighin. Dynamic algorithms for LIS and distance to monotonicity. 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 671–684. ACM, 2020. doi:10.1145/3357713.3384240.

27 Ashley Montanaro. Metric Embeddings. http://people.maths.bris.ac.uk/~csxam/presentations/embeddings.pdf, 2008. [Online; accessed 12-December-2008].

28 Timothy Naumovitz, Michael Saks, and C Seshadhri. Accurate and nearly optimal sublinear approximations to ulam distance. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2012–2031. SIAM, 2017.

29 François Nicolas and Eric Rivals. Complexities of the centre and median string problems. In *Proceedings of the 14th Annual Conference on Combinatorial Pattern Matching*, CPM'03, pages 315–327, Berlin, Heidelberg, 2003. Springer-Verlag.

30 François Nicolas and Eric Rivals. Hardness results for the center and median string problems under the weighted and unweighted edit distances. *Journal of discrete algorithms*, 3(2-4):390–415, 2005.

31 Aviad Rubinstein. Hardness of approximate nearest neighbor search. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1260–1268. ACM, 2018. doi:10.1145/3188745.3188916.

32 Kenneth W. Shum, Ilia Aleshnikov, P. Vijay Kumar, Henning Stichtenoth, and Vinay Deolalikar. A low-complexity algorithm for the construction of algebraic-geometric codes better than the Gilbert-Varshamov bound. *IEEE Trans. Information Theory*, 47(6):2225–2241, 2001. doi:10.1109/18.945244.

33 Jeong Seop Sim and Kunsoo Park. The consensus string problem for a metric is np-complete. *J. Discrete Algorithms*, 1(1):111–117, 2003. doi:10.1016/S1570-8667(03)00011-X.

34 Nikola Stojanovic, Piotr Berman, Deborah Gumucio, Ross C. Hardison, and Webb Miller. A linear-time algorithm for the 1-mismatch problem. In Frank K. H. A. Dehne, Andrew Rau-Chaplin, Jörg-Rüdiger Sack, and Roberto Tamassia, editors, *Algorithms and Data Structures, 5th International Workshop, WADS '97, Halifax, Nova Scotia, Canada, August 6-8, 1997, Proceedings*, volume 1272 of *Lecture Notes in Computer Science*, pages 126–135. Springer, 1997. doi:10.1007/3-540-63307-3_53.

35 Ryan Williams. On the difference between closest, furthest, and orthogonal pairs: Nearly-linear vs barely-subquadratic complexity. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1207–1215. SIAM, 2018. doi:10.1137/1.9781611975031.78.

36 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018*, pages 3447–3487. World Scientific, 2018.

A Error Correcting Codes

An error correcting code C over alphabet Σ is a function $C : \Sigma^m \to \Sigma^\ell$ where m and ℓ are positive integers which are referred to as the *message length* and *block length* of C respectively. Intuitively, the function C encodes an original message of length m to an encoded message of length ℓ. The *rate* of a code $\rho(C)$ is defined as the ratio between its message length and its block length, i.e., $\rho(C) = {}^m/_\ell$. The *relative distance* of a code, denoted by $\delta(C)$, is defined as $\min\limits_{x \neq y \in \Sigma^m} \delta(C(x), C(y))$ where $\delta(C(x), C(y))$ is the *relative Hamming distance* between $C(x)$ and $C(y)$, i.e., the fraction of coordinates on which $C(x)$ and $C(y)$ disagree.

In this paper, we require our codes to have some special algebraic properties which have been shown to be present in algebraic geometric codes [17]. First, we will introduce a couple of additional definitions.

▶ **Definition 24** (Systematicity). *Given $s \in \mathbb{N}$, a code $C : \Sigma^m \to \Sigma^\ell$ is* s-systematic *if there exists a size-s subset of $[\ell]$, which for convenience we identify with $[s]$, such that for every $x \in \Sigma^s$ there exists $w \in \Sigma^m$ in which $x = C(w)\mid_{[s]}$.*

▶ **Definition 25** (Degree-t Closure). *Let Σ be a finite field. Given two codes $C : \Sigma^m \to \Sigma^\ell, C' : \Sigma^{m'} \to \Sigma^\ell$ and positive integer t, we say that C' is a degree-t closure of C if, for every $w_1, \ldots, w_r \in \Sigma^m$ and $P \in \mathbb{F}[X_1, \ldots, X_r]$ of total degree at most t, it holds that $\omega := P(C(w_1), \ldots, C(w_r))$ is in the range of C', where $\omega \in \Sigma^\ell$ is defined coordinate-wise by the equation $\omega_i := P(C(w_1)_i, \ldots, C(w_r)_i)$.*

Below we provide a self-contained statement of the result we need; it follows from Theorem 7 of [32], which gives an efficient construction of the algebraic geometric codes based on [17]'s explicit towers of function fields.

▶ **Theorem 26** ([17, 32]). *There is a constant $\lambda > 0$ such that for any prime $q \geq 7$, there are two code families $\mathcal{C}^1 = \{C_n^1\}_{n\in\mathbb{N}}$, $\mathcal{C}^2 = \{C_n^2\}_{n\in\mathbb{N}}$ such that the following holds for all $n \in \mathbb{N}$,*

- *C_n^1 and C_n^2 are n-systematic code with alphabet $\mathbb{F}_{q^2}$,*
- *C_n^1 and C_n^2 have block length less than λn.*
- *C_n^2 has relative distance $\geq {}^1/_2$,*
- *C_n^2 is a degree-2 closure of C_n^1, and,*
- *Any codeword in C_n^1 or C_n^2 can be computed in poly(n) time.*

We point the interested reader to [21] for a proof sketch of the above theorem.

Probabilistic Metric Embedding via Metric Labeling

Kamesh Munagala
Department of Computer Science, Duke University, Durham, NC, USA

Govind S. Sankar
Department of Computer Science, Duke University, Durham, NC, USA

Erin Taylor
Department of Computer Science, Duke University, Durham, NC, USA

Abstract

We consider probabilistic embedding of metric spaces into ultra-metrics (or equivalently to a constant factor, into hierarchically separated trees) to minimize the expected distortion of any pairwise distance. Such embeddings have been widely used in network design and online algorithms. Our main result is a polynomial time algorithm that approximates the optimal distortion on any instance to within a constant factor. We achieve this via a novel LP formulation that reduces this problem to a probabilistic version of uniform metric labeling.

2012 ACM Subject Classification Theory of computation → Random projections and metric embeddings

Keywords and phrases Metric Embedding, Approximation Algorithms, Ultrametrics

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.2

Category APPROX

Funding This work is supported by NSF grant CCF-2113798.

1 Introduction

Embedding a finite metric space into simpler spaces such as trees, ultrametrics, and Euclidean spaces (called "target metrics") has a wide range of applications, and has been widely studied. In such an embedding, the distance between any pair of points should be at least as large as in the original space, while being at most a factor of α larger, where α is termed the *distortion* of the embedding. The goal is to design an embedding into a given target metric whose distortion is as small as possible. We will denote by n the number of points in the metric space.

A lot of attention has focused on probabilistic embeddings, which construct a distribution over metrics from the target space, for instance, a distribution over trees or ultrametrics. The goal is now to bound the *expected distortion* for any pair of points relative to their distance in the original metric space. Probabilistic embeddings typically allow for much lower values of distortion. Indeed, when the target metric is a tree or an ultrametric, deterministic embeddings have distortion $\Omega(n)$ [2], while probabilistic embeddings have distortion $\alpha = O(\log n)$ [20].

In this paper, we consider the problem of embedding metrics into a distribution over ultrametrics, defined in Section 2. To within a constant factor on distortion, these metrics embed into *hierarchically separated trees* (HSTs), and such embeddings have found myriad uses in network design, data analysis and online algorithms. This is because most network design problems such as Steiner tree or facility location are NP-Hard in general metric spaces, but amenable to polynomial time algorithms on trees. If the objective function is a linear combination of distances, the solution on the distribution over HSTs yields a α approximation algorithm for the input metric space, where α is the distortion of the embedding. Indeed, recent breakthroughs in developing competitive algorithms for the celebrated randomized k-server problem proceed via probabilistic embedding into HSTs [16, 4]. Other applications include analysis of hierarchical clusterings [10, 31, 17], and approximation algorithms for group Steiner trees [21], buy-at-bulk network design [3], and metric labeling [27].

In this context, it is known that in the worst case over input metric spaces, the distortion of embedding into a distribution over ultrametrics is $\alpha = O(\log n)$ [20], and this bound is tight [25, 5]. However, these algorithms typically use fixed ball-growing procedures oblivious to the actual metric. It is conceivable that any given metric can be embedded with much lower distortion than such procedures imply. For instance, an entirely different algorithm can embed doubling metrics into ultrametrics with distortion $O(\log \Delta)$, where Δ is the spread[1] of the point set; this can be significantly better if $\Delta = o(n)$ or independent of n [26]. Further, many real-world social graphs have small diameter, often independent of the network size n [32, 30] and again, it is conceivable that specifically tailored algorithms can embed these better than what the worst case bounds imply.

In this paper, we therefore ask: *Can we achieve the best possible (in terms of distortion) probabilistic embedding of a given discrete metric space into ultrametrics in polynomial time?*

1.1 Result

Our main result is positive:

▶ **Theorem 1.** *Given any n-point metric space that can be embedded into a distribution over ultrametrics with optimal distortion α, there is an algorithm with expected polynomial running time*[2] *that can find an embedding of distortion $16 \cdot \alpha$.*

Prior to our work, the best published approximation factor was $O(\log n)$ [20], which is also the tight existence result. In contrast, we provide a nearly tight *computational* result, which also yields improved instance-dependent approximation factors for problems like metric labeling and buy-at-bulk network design, where the factor of $O(\log n)$ in the approximation ratio improves to $O(\alpha)$.

1.2 Technique

Our algorithm for proving Theorem 1 proceeds via constructing an LP relaxation. In this LP relaxation, the variables $\gamma^r_{jj'}$ represent the probability that j, j' are separated in the HST at radius (or level) r. This LP is our main non-trivial contribution. Note that though there are LP formulations that embed metrics into trees [11, 27], we do not know how to solve them to a $o(\log n)$ approximation factor.

We then use the randomized rounding technique for uniform metric labeling in [27] in decreasing order of radius to construct a distribution over centers and assignments, while relaxing $\gamma^r_{jj'}$ by a factor of 2. In the uniform metric labeling problem, the goal is to assign one of k labels to the vertices of a graph with edge weights so that the total weight of the edges whose end-points have different labels is minimized. In the algorithm of [27], they encode the probability that two endpoints of an edge are separated as a random variable

[1] Spread is the ratio of the largest to smallest distance in the metric space.

[2] For any $\delta > 0$, the algorithm can easily be converted to a $16 + \delta$ approximation in deterministic polynomial time, with the polynomial depending on n and $\log \frac{\Delta}{\delta}$, where Δ is the ratio of largest to smallest distance in the metric space.

in their LP, and then minimize an expected cost over these random variables. Our main idea is that the event whether two nodes belong to the same subtree in the HST can be similarly treated as a random variable in our LP. The expected "cost" of the labeling is the contribution of these variables to the distortion, and these variables can be randomly rounded via the same ideas as for metric labeling.

Note that the algorithm for general metric labeling [27] – where the labels lie in a metric space and the weight of the edge is multiplied by the distance between the corresponding labels – proceeds via embedding the metric over labels into an ultrametric. We effectively reduce in the reverse direction and show that metric embedding reduces to (uniform) metric labeling!

1.3 Related Work

The main technique for probabilistically embedding metrics into ultrametrics is low-diameter decompositions. These involve decomposing the input metric space into small diameter components via sampling radii from a suitable distribution and randomly partitioning based on these radii. Starting with a result of [5], a sequence of results showed improving bounds for such embeddings [6, 11, 20], culminating in the optimal distortion bound of $O(\log n)$. Better results are known for special types of metrics [26]. All these decompositions proceed by deriving absolute bounds on the probability that a pair of nodes end up in different partitions; in contrast, we encode this probability as variables in an LP formulation.

Trees are more general spaces than ultrametrics, and the seminal work of [2] initiated the study of embedding metrics into trees. Though the worst case bound for embedding into a distribution over trees remains $\Theta(\log n)$, better results are known for embedding specific classes of metrics such as the shortest path metrics of k-outerplanar or small pathwidth graphs [13, 23, 29]. The work of [11] provides an LP formulation for computing the optimal distortion for embedding into a distribution over trees; however, their separation oracle – the minimum communication cost spanning tree problem – is unlikely to admit a $o(\log n)$ approximation. Our LP for ultrametrics, in contrast, is inspired by stochastic optimization and metric labeling and shows a constant approximation.

The above works provide worst case guarantees on the distortion of the embedding, while we provide an approximation result. Though other prior work has considered such approximation guarantees [19, 24, 1, 15], these works focus on *deterministic* embeddings, while ours consider approximations for probabilistic embeddings. The work of [1] provides a polynomial time algorithm for optimally embedding a metric into a single ultrametric, and the work of [15] shows an improved running time for obtaining such an embedding. However, the optimal distortion for embedding into a single ultrametric might be $\Omega(n)$, while it is $O(\log n)$ for embedding into a distribution. This result motivates the need for algorithms that approximate the optimal probabilistic embedding. We note that the algorithm of [1] does not extend to probabilistic embeddings.

Our LP is also similar in spirit to those in [28]. They give a 2-approximation to separating decompositions (see [28] for formal definitions) by writing an LP using variables similar to our variables γ for the separation probabilities. The key observation we make is that unlike in their setting where there is a fixed separation probability, we need to optimize over the separation probabilities at all levels of the HST simultaneously.

Stochastic Optimization. Our algorithms involve rounding a linear programming relaxation to the optimal solution. This LP is inspired by similar LPs approximate stochastic optimization, particularly those for stochastic knapsack [18], multi-armed bandits [22], Bayesian auctions [9, 12, 8], and stochastic matching [14]. The novel aspects of our work is the formulation of probabilistic embeddings as a stochastic optimization problem, and viewing the uniform metric labeling algorithm as a stochastic rounding procedure [27].

2 Terminology

▶ **Definition 2** (Metric space). *A metric space (N, d) is a finite set of n points N endowed with a distance function $d : N \times N \to \mathbb{R}^+ \cup \{0\}$. This distance function has the following properties:*

- $d(x, x) = 0$ *for all* $x \in N$*;*
- $d(x, y) = d(y, x)$ *for all* $x, y \in N$*; and*
- $d(x, z) \leq d(y, z) + d(x, y)$ *for all* $x, y, z \in N$*.*

▶ **Definition 3** (Embedding). *Given two metric spaces (N, d) and (T, d_T), an embedding from N to T is a function $f : N \to T$.*

With an abuse of notation, for $x, y \in N$, we use $d_T(x, y)$ to refer to $d_T(f(x), f(y))$.

Our goal is to *embed* a given n point metric space into a probability distribution over ultrametrics. An ultrametric and probabilistic embedding are defined below.

▶ **Definition 4** (Ultrametric). *A metric space (N, d) is an* ultrametric *if for all points $x, y, z \in N$, we have $d(x, z) \leq \max(d(x, y), d(y, z))$.*

▶ **Definition 5** (Probabilistic embedding). *Given a metric space (N, d), an embedding is a distribution over ultrametrics (N, d_T), where ultrametric T is chosen with probability p_T. Let F denote this distribution and S denote its support. The embedding should be* non-contractive*, meaning that*

$$\forall x, y \in N, \quad \forall T \in S, \; d_T(x, y) \geq d(x, y).$$

Further, this embedding has distortion α (where $\alpha \geq 1$) if

$$\forall x, y \in N, \mathsf{E}_{T \sim F}[d_T(x, y)] \leq \alpha \cdot d(x, y).$$

Hierarchically Separated Trees. It is convenient to consider a specific type of ultrametric termed *hierarchically separated trees.* These are defined as follows.

▶ **Definition 6** (exact c-HST). *A metric (N, d) is an exact c-HST (for $c > 1$) if the elements of N are the leaves of a rooted tree T, all of whose leaves are at the same level. Each internal node v of T is associated with a number δ_v. These numbers increase by a factor of exactly c as we move up the tree, so that $\delta_v = c \cdot \delta_u$ whenever u is a child of v. Given leaves $x, y \in N$, let z be their least common ancestor in T. Then $d(x, y) = \delta_z$.*

The diameter of a c-HST with root r is δ_r. Note that a c-HST with diameter D decomposes into c-HSTs with diameter D/c, where points in different parts are separated by distance exactly D.

The following result shows that it suffices to consider embedding into exact c-HSTs.

▶ **Lemma 7** ([7]). *Given a metric space and its embedding into a distribution over ultrametrics with distortion α, there is an embedding into a distribution over exact c-HSTs with distortion $\alpha \cdot c$.*

3 Linear Programming Relaxation

The LP relaxation is not obvious, and we will present it in some detail. We are given a metric space (N, d) whose smallest distance is 1 and largest distance is Δ. We will use exact c-HSTs for $c = 2$; using any other value of c only yields a worse approximation factor, and our presentation is simplified without the parameter c. By losing a factor of 2, we assume that the optimal solution embeds this metric into a distribution over exact 2-HSTs. Let this embedding have distortion $q \le 2\alpha$ if the optimal distortion for probabilistically embedding into ultrametrics is α. Our goal therefore is to approximate q in polynomial time.

Given this optimal probabilistic embedding into 2-HSTs, let M denote the set of possible δ_z values in powers of 2. Note that $|M| = O(\log \Delta)$, since $1 \le \delta_r \le 2\Delta$ without loss of generality.[3]

Consider some 2-HST in the optimal embedding of (N, d) and some $r \in M$. For any sub-tree T' whose root z has $\delta_z = r$, all nodes in T' have distance in the 2-HST at most r from some node $i \in T'$. Since this embedding is non-contractive, we have $d(i, j) \le r$ for all $j \in T'$. We arbitrarily pick one i in this subtree as the "representative" of this subtree and "assign" all other nodes in the subtree to i. Therefore, each node j is assigned to one representative i at each level $r \in M$ in the tree and this satisfies $d(i, j) \le r$. Now we can define a graph $G_r(N, E_r)$ for each $r \in M$, where there is an edge $(i, j) \in E_r$ if and only if $d(i, j) \le r$. Let $B_i(r) = \{j \in N, d(i, j) \le r\}$.

For $(i, j) \in E_r$, define a variable x^r_{ij} as the probability that j's representative at level r is i. Similarly, let $z^r_{ijj'}$ be the probability that both j, j' have i as their level r representative, where we assume $(i, j) \in E_r$ and $(i, j') \in E_r$. Finally, let $\gamma^r_{jj'}$ be the probability that j and j' do not share a level r representative.

The LP relaxation with variables $x^r_{ij}, z^r_{ijj'}$, and $\gamma^r_{jj'}$, and is shown in Figure 1.

$$
\text{(LP1)}\qquad
\begin{aligned}
&\text{Minimize } q && (1)\\
\forall j, j', \quad & \sum_{r \in M} r \cdot \gamma^r_{jj'} \le q \cdot d(j, j') && (2)\\
\forall i, r, j, j' \in B_i(r) \quad & \min(x^r_{ij}, x^r_{ij'}) \ge z^r_{ijj'} && (3)\\
\forall j, j', r \quad & \sum_{i: j, j' \in B_i(r)} z^r_{ijj'} \ge 1 - \gamma^r_{jj'} && (4)\\
\forall j, r \quad & \sum_{i: j \in B_i(r)} x^r_{ij} = 1 && (5)\\
\forall j, j', r < d(j, j') \quad & \gamma^r_{jj'} = 1 && (6)\\
\forall i, j, j', r \quad & \gamma^r_{jj'}, x^r_{ij}, z^r_{ijj'} \ge 0. && (7)
\end{aligned}
$$

Figure 1 Linear program relaxation for embedding into 2-HSTs.

[3] To see this, consider any HST in the support of the optimal probabilistic embedding. We can contract all the internal nodes v of the HST with $\delta_v \ge 2\Delta$ into one node r with $\delta_r = 2\Delta$. Clearly, this preserves the non-contractivity property, since no pair of vertices are more than Δ far apart. Furthermore, this can only decrease the expected distortion of any edge.

▶ **Lemma 8.** *LP1 is feasible and is a 2-approximation to the distortion of the optimal probabilistic embedding into ultrametrics.*

Proof. Consider an optimal embedding of the input metric into 2-HSTs with distortion q. As mentioned before, since q is a factor 2 approximation to optimal distortion of embedding into ultrametrics, this means the objective is a 2-approximation. We only need to show that there is a feasible solution to the LP with objective at most q.

Now, consider any 2-HST in the support of the optimal embedding. We interpret the variables as described before. To interpret Equation (2), note that if (j, j') have least common level r, they are separated at levels $r/2, r/4, r/8$, and so on. Therefore, the contribution this 2-HST makes to the LHS of Equation (2) is $r/2 + r/4 + \cdots \leq r$. Taking expectation over all 2-HSTs in the optimal embedding, we have $\mathsf{E}[r] \leq q \cdot d(i, j)$, where q is the distortion of this embedding. This shows that the constraint holds.

Equation (6) captures that the embedding is non-contractive: If $d(j, j') < r$, then they are separated at level r. The remaining constraints are interpreted as follows. Equation (3) says j, j' are co-assigned to i implies they were both individually assigned to i; Equation (5) says each $j \in N$ has a representative at each level r; and Equation (4) says that the probabilities that j, j' are co-assigned and not co-assigned at level r sum to at least 1.

One feasible solution to the LP is to assign each node j to itself at all levels, so that $x^r_{jj} = 1$ for all $j \in N, r \in M$. Then $z^r_{ijj'} = 0$ and $\gamma^r_{jj'} = 1$ for all $j \neq j'$. This is feasible when $q = 2\Delta$. ◀

4 Rounding and HST Construction

We first present a procedure in Algorithm 1 that generates partitions separately for each level r. Essentially, our algorithm solves $|M|$ instances of metric labeling, one for each level r. This step adapts the rounding scheme in [27] for uniform metric labeling. Each "label" is a possible representative at level r. The expected cost of this metric labeling is then used to bound the expected distortion of the embedding.

To construct the HST itself, we inductively compute the tree in the following manner: At level 2Δ, every node is assigned the same representative. For every lower level r, the set of nodes assigned to the same representative at level $\frac{r}{2}$ belong to the same subtree. These nodes are then assigned new representatives at level r based on the metric labeling.

Algorithm 1 Rounding to Create Partitions.

```
for $r \in M$ in decreasing order do
    $S \leftarrow N$; $P^r_i = \emptyset$ for all $i \in N$
    while $S \neq \emptyset$ do
        Choose a center $i \in N$ uniformly at random independent of past choices.
        Choose $\ell^r_i \in [0, 1]$ uniformly at random independent of past choices.
        For each $j \in S \cap B_i(r)$, if $x^r_{ij} \geq \ell^r_i$, assign $j$ to $P^r_i$ and remove $j$ from $S$.
    end while
end for
```

We next combine these partitions into a 2-HST in Algorithm 2.

Analysis

We first consider Algorithm 1. The lemma below follows directly from the analysis of the rounding algorithm for uniform metric labeling in [27]. For completeness, we provide the proof here.

Algorithm 2 Combining Partitions and Constructing the 2-HST.

```
Place a root node $w$ at the highest level with $\delta_w = 2\Delta$ and set $S_w = N$.
for $r \in M$ in decreasing order do
    for each node $w$ at the previous (parent) level with set $S_w$ do
        for each $i$ with $P_i^r \cap S_w \neq \emptyset$ do
            Place a child node $v$ with set $S_v = P_i^r \cap S_w$ and $\delta_v = 2r$
        end for
    end for
end for
```

▶ **Lemma 9** (Lemma 3.2 in [27]). *Consider some pair j, j' with $d(j, j') = R$. Consider some level $r \geq R$. For any $(j, j') \in E(G)$,*

$$\Pr[j, j' \textit{ separated at level } r] \leq 2\gamma_{jj'}^r.$$

Proof. We assume $|C| = n \geq 2$; the result is trivial for $n = 1$. Since we fix a phase r, we will omit the superscript r from the proof below.

Suppose that both $j, j' \in S$ at some point in time. Then j is assigned to i with probability $\frac{x_{ij}}{n}$, where $\frac{1}{n}$ is the probability that i is chosen; conditioned on this, j is assigned to i if $\ell_i \leq x_{ij}$. Therefore, j is assigned to some center at this step with probability is $\sum_{i:j \in B_i} \frac{x_{ij}}{n} = \frac{1}{n}$.

Similarly, conditioned on both $j, j' \in S$, the probability with which they are assigned to center $i \in C$ is $\frac{\min(x_{ij}, x_{ij'})}{n} \geq \frac{z_{ijj'}}{n}$, so this pair is co-assigned with probability at least $\frac{\sum_i z_{ijj'}}{n} \geq \frac{1-\gamma_{jj'}}{n}$. This is therefore a lower bound on the probability with which both j and j' get assigned this step.

By the inclusion-exclusion principle on the pair (j, j'), conditioned on both $j, j' \in S$, the probability with which either j or j' gets assigned is:

$$\Pr[\text{Either } j \text{ or } j' \text{ assigned } | j, j' \in S] \leq \frac{1}{n} + \frac{1}{n} - \frac{1-\gamma_{jj'}}{n} = \frac{1+\gamma_{jj'}}{n}. \tag{8}$$

Since $n \geq 2$ and $\gamma_{jj'} \leq 1$, the RHS above is at most 1. Therefore, the probability that both $j, j' \in S$ at time t is at least $\left(1 - \frac{1+\gamma_{jj'}}{n}\right)^{t-1}$. Conditioned on this event, they are co-assigned at time step t with probability at least $\frac{1-\gamma_{jj'}}{n}$. Therefore, we have

$$\Pr[j, j' \text{ not separated }] \geq \sum_{t=1}^{\infty} \left(1 - \frac{1+\gamma_{jj'}}{n}\right)^{t-1} \cdot \frac{1-\gamma_{jj'}}{n} = \frac{1-\gamma_{jj'}}{1+\gamma_{jj'}}.$$

Noting that $\Pr[j, j' \text{ separated}] \leq 1 - \frac{1-\gamma_{jj'}}{1+\gamma_{jj'}} \leq 2\gamma_{jj'}$, this completes the proof. ◀

The following two lemmas will now complete the proof of Theorem 1.

▶ **Lemma 10.** *The construction in Algorithm 1 and Algorithm 2 is non-contractive.*

Proof. Suppose $d(j, j') = R$. Consider some level $r \in M \cap [R/4, R/2]$. For this value of r, there is no center i such that $j, j' \in B_i(r)$. Therefore, j, j' lie in different partitions at this level. Observing that Algorithm 2 sets $\delta_v = 2r$ for nodes v at level r, their common ancestor u must have $\delta_u \geq 4r \geq R$. Therefore, the embedding is non-contractive. ◀

▶ **Lemma 11.** *In the output of Algorithm 2, the expected distortion of any distance is at most $8q$. This implies a 16 approximation to the optimal embedding into a distribution over ultrametrics.*

Proof. Consider some pair j, j' with $d(j, j') = R$. Consider some level $r \geq R$. By Lemma 9,

$$\Pr[j, j' \text{ separated at level } r] \leq 2\gamma^r_{jj'}.$$

If $r < R$, note that $\gamma^r_{jj'} = 1$, so the above inequality trivially holds.

If r^* is the highest level at which j, j' are separated, the distance in the embedding is $4r^*$. As an upper bound, we simply add a distance of $4r$ for all levels r at which j, j' are cut. This yields:

$$\mathsf{E}[\text{ Distance in embedding between } (j, j')] \leq \sum_{r \in M} 4r \cdot 2\gamma^r_{jj'} \leq 8q \cdot d(j, j').$$

Since q itself is a 2 approximation to the optimal distortion of embedding into ultrametrics, this implies a 16 approximation to the distortion of embedding into a distribution over ultrametrics. ◀

Running Time. Since each j gets assigned with probability $1/n$ each step, the expected number of steps is $O(n \log n)$ per level, and there are $O(\log \Delta)$ levels. Suppose we stop the process after $c \cdot n \ln \frac{\Delta}{\delta}$ steps. Then for large constant c, the probability that at some level, all j have not been assigned is at most $\frac{\delta}{\Delta}$. In this event, we pretend the ultrametric distorts all distances to Δ. The expected distortion now becomes a $16 + \delta$ approximation. This completes the proof of Theorem 1.

5 Conclusion

We have in effect reduced probabilistic metric embeddings to metric labeling, performing the reverse of the reduction in [27] that reduces metric labeling to metric embedding. There are some open questions that arise from this work. First, is there an *exact* polynomial time algorithm for embedding into ultrametrics or even a PTAS? Second, can similar results be obtained for the more general problem of embedding into tree metrics?

References

1 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 Trans. Algorithms*, 4(4), August 2008.

2 Noga Alon, Richard M. Karp, David Peleg, and Douglas West. A graph-theoretic game and its application to the k-server problem. *SIAM Journal on Computing*, 24(1):78–100, 1995.

3 B. Awerbuch and Y. Azar. Buy-at-bulk network design. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 542–547, 1997. doi:10.1109/SFCS.1997.646143.

4 Nikhil Bansal, Niv Buchbinder, Aleksander Madry, and Joseph (Seffi) Naor. A polylogarithmic-competitive algorithm for the k-server problem. *J. ACM*, 62(5), November 2015.

5 Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 184–193, 1996. doi:10.1109/SFCS.1996.548477.

6 Yair Bartal. On approximating arbitrary metrices by tree metrics. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 161–168, New York, NY, USA, 1998. Association for Computing Machinery. doi:10.1145/276698.276725.

7 Yair Bartal, Nathan Linial, Manor Mendel, and Assaf Naor. On metric Ramsey-type phenomena. *Annals of Mathematics*, 162(2):643–709, 2005.

8 Sayan Bhattacharya, Gagan Goel, Sreenivas Gollapudi, and Kamesh Munagala. Budget constrained auctions with heterogeneous items. 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 379–388. ACM, 2010.

9 Yang Cai, Constantinos Daskalakis, and S. Matthew Weinberg. Optimal multi-dimensional mechanism design: Reducing revenue to welfare maximization. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 130–139. IEEE Computer Society, 2012.

10 Gunnar Carlsson and Facundo Mémoli. Characterization, stability and convergence of hierarchical clustering methods. *Journal of Machine Learning Research*, 11(47):1425–1470, 2010. URL: `http://jmlr.org/papers/v11/carlsson10a.html`.

11 Moses Charikar, Chandra Chekuri, Ashish Goel, Sudipto Guha, and Serge Plotkin. Approximating a finite metric by a small number of tree metrics. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, FOCS '98, page 379, USA, 1998. IEEE Computer Society.

12 Shuchi Chawla, Jason D. Hartline, David L. Malec, and Balasubramanian Sivan. Multi-parameter mechanism design and sequential posted pricing. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 311–320. ACM, 2010.

13 Chandra Chekuri, Anupam Gupta, Ilan Newman, Yuri Rabinovich, and Alistair Sinclair. Embedding k-outerplanar graphs into l1. *SIAM Journal on Discrete Mathematics*, 20(1):119–136, 2006. `doi:10.1137/S0895480102417379`.

14 Ning Chen, Nicole Immorlica, Anna R Karlin, Mohammad Mahdian, and Atri Rudra. Approximating matches made in heaven. In *International Colloquium on Automata, Languages, and Programming*, pages 266–278. Springer, 2009.

15 Vincent Cohen-Addad, C. S. Karthik, and Guillaume Lagarde. On efficient low distortion ultrametric embedding. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

16 Aaron Coté, Adam Meyerson, and Laura Poplawski. Randomized k-server on hierarchical binary trees. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 227–234, New York, NY, USA, 2008. Association for Computing Machinery.

17 Sanjoy Dasgupta. A cost function for similarity-based hierarchical clustering. 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 118–127. ACM, 2016. `doi:10.1145/2897518.2897527`.

18 Brian C Dean, Michel X Goemans, and Jan Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Mathematics of Operations Research*, 33(4):945–964, 2008.

19 Kedar Dhamdhere, Anupam Gupta, and R Ravi. Approximation algorithms for minimizing average distortion. *Theory of Computing Systems*, 39(1):93–111, 2006.

20 Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004. Special Issue on STOC 2003. `doi:10.1016/j.jcss.2004.04.011`.

21 Naveen Garg, Goran Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group steiner tree problem. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '98, pages 253–259, USA, 1998. Society for Industrial and Applied Mathematics.

22 Sudipto Guha and Kamesh Munagala. Approximation algorithms for budgeted learning problems. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 104–113, 2007.

23 Anupam Gupta, Ilan Newman, Yuri Rabinovich, and Alistair Sinclair. Cuts, trees and ℓ_1-embeddings of graphs. *Combinatorica*, 24(2):233–269, April 2004. `doi:10.1007/s00493-004-0015-x`.

24 Alexander Hall and Christos Papadimitriou. Approximating the distortion. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 111–122. Springer, 2005.

25 Makoto Imase and Bernard M. Waxman. Dynamic steiner tree problem. *SIAM Journal on Discrete Mathematics*, 4(3):369–384, 1991. doi:10.1137/0404033.

26 Piotr Indyk, Avner Magen, Anastasios Sidiropoulos, and Anastasios Zouzias. Online embeddings. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 246–259, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

27 Jon Kleinberg and Éva Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. *J. ACM*, 49(5):616–639, September 2002.

28 Robert Krauthgamer and Tim Roughgarden. Metric clustering via consistent labeling. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 809–818, USA, 2008. Society for Industrial and Applied Mathematics.

29 James R. Lee and Anastasios Sidiropoulos. Pathwidth, trees, and random embeddings. *arXiv e-prints*, page arXiv:0910.1409, October 2009. doi:10.48550/arXiv.0910.1409.

30 Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: An approach to modeling networks. *J. Mach. Learn. Res.*, 11:985–1042, March 2010.

31 Aurko Roy and Sebastian Pokutta. Hierarchical clustering via spreading metrics. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 2324–2332, Red Hook, NY, USA, 2016. Curran Associates Inc.

32 Duncan J. Watts and Steven H. Strogatz. Collective dynamics of “small-world” networks. *Nature*, 393(6684):440–442, 1998.

Approximating Submodular k-Partition via Principal Partition Sequence

Karthekeyan Chandrasekaran
University of Illinois, Urbana-Champaign, IL, USA

Weihang Wang
University of Illinois, Urbana-Champaign, IL, USA

Abstract

In submodular k-partition, the input is a submodular function $f : 2^V \to \mathbb{R}_{\geq 0}$ (given by an evaluation oracle) along with a positive integer k and the goal is to find a partition of the ground set V into k non-empty parts $V_1, V_2, \ldots, V_k$ in order to minimize $\sum_{i=1}^{k} f(V_i)$. Narayanan, Roy, and Patkar [18] designed an algorithm for submodular k-partition based on the principal partition sequence and showed that the approximation factor of their algorithm is 2 for the special case of graph cut functions (which was subsequently rediscovered by Ravi and Sinha [22]). In this work, we study the approximation factor of their algorithm for three subfamilies of submodular functions – namely monotone, symmetric, and posimodular and show the following results:

1. The approximation factor of their algorithm for monotone submodular k-partition is $4/3$. This result improves on the 2-factor that was known to be achievable for monotone submodular k-partition via other algorithms. Moreover, our upper bound of $4/3$ matches the recently shown lower bound under polynomial number of function evaluation queries [23]. Our upper bound of $4/3$ is also the first improvement beyond 2 for a certain graph partitioning problem that is a special case of monotone submodular k-partition.
2. The approximation factor of their algorithm for symmetric submodular k-partition is 2. This result generalizes their approximation factor analysis beyond graph cut functions.
3. The approximation factor of their algorithm for posimodular submodular k-partition is 2.

We also construct an example to show that the approximation factor of their algorithm for arbitrary submodular functions is $\Omega(n/k)$.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.3

Category APPROX

Related Version *Full Version*: `https://arxiv.org/abs/2305.01069`

Funding *Karthekeyan Chandrasekaran*: Supported in part by NSF grants CCF 1814613 and 1907937
Weihang Wang: Supported in part by NSF grants CCF 1814613 and 1907937.

Acknowledgements Karthekeyan Chandrasekaran would like to thank Chandra Chekuri for asking about the approximation factor of the principal partition sequence based approach for symmetric submodular k-partition.

1 Introduction

A set function $f : 2^V \to \mathbb{R}$ is submodular if $f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$ for every $A, B \subseteq V$. An evaluation oracle for a function $f : 2^V \to \mathbb{R}$ takes a subset $A \subseteq V$ as input and returns $f(A)$. We consider the *submodular k-partition* problem defined as follows: The input consists of a non-negative submodular function $f : 2^V \to \mathbb{R}_{\geq 0}$ on a finite ground set V via an evaluation oracle and an integer $k \geq 2$. The goal is to find a partition $V_1, V_2, \ldots, V_k$ of V into k non-empty parts in order to minimize $\sum_{i=1}^{k} f(V_i)$. Namely, the goal is to compute

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 3; pp. 3:1–3:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

$$\min \left\{ \sum_{i \in [k]} f(V_i) : \; V_1, V_2, \ldots, V_k \text{ is a partition of } V, \; V_i \neq \emptyset \; \forall i \in [k] \right\}.$$

Throughout, we will assume that the input submodular function is non-negative and denote the size of the ground set V by n. If $k = 2$, then the problem reduces to the classic submodular minimization problem. We emphasize that our focus is on submodular k-partitioning when k is part of input (see [4] for a discussion of the problem for fixed constant k). Submodular k-partition formulates several interesting partitioning problems and we will discuss some of the special cases below. For arbitrary submodular functions, the problem is NP-hard [10], does not admit a $(2 - \epsilon)$-approximation assuming polynomial number of function evaluation queries [23], does not admit a $n^{1/(\log \log n)^c}$-approximation for every constant c assuming the Exponential Time Hypothesis [6], and the best approximation factor that is known is $O(k)$ [19,26].

In this work, we will be interested in the submodular k-partition problem for subfamilies of submodular functions – namely monotone, symmetric, and posimodular submodular functions. A set function $f : 2^V \to \mathbb{R}$ is

1. monotone if $f(B) \geq f(A)$ for every $A \subseteq B \subseteq V$,
2. symmetric if $f(A) = f(V - A)$ for every $A \subseteq V$, and
3. posimodular if $f(A) + f(B) \geq f(A - B) + f(B - A)$ for every $A, B \subseteq V$.

If the input submodular function is monotone/symmetric/posimodular, then we call the associated submodular k-partition problem as monotone/symmetric/posimodular submodular k-partition. We note that monotone submodular functions and symmetric submodular functions are also posimodular[1]. Hence, posimodular submodular k-partition problem generalizes both monotone submodular k-partition and symmetric submodular k-partition problems. We now discuss the approximation status of symmetric/monotone/posimodular submodular k-partition and some of their well-known special cases (see Table 1 for a summary of approximation factors of symmetric/monotone/posimodular submodular k-partition achieved by different approaches).

Monotone submodular k-partition. Special cases of monotone submodular k-partition problem include matroid k-partition and coverage k-partition – the submodular functions of interest here are matroid rank functions and coverage functions respectively. Matroid k-partition captures several interesting problems: e.g., (1) partition the columns of a given matrix into k non-empty parts to minimize the sum of the dimension of the subspace spanned by the parts and (2) partition the edges of a given graph into k non-empty parts to *maximize* the sum of the number of connected components formed by the parts. Coverage k-partition also captures several interesting problems: e.g., (3) partition the vertices of a given graph into k non-empty parts $V_1, V_2, \ldots, V_k$ in order to minimize $\sum_{i=1}^{k} f(V_i)$, where $f(P)$ is the number of edges incident to the vertex subset $P \subseteq V$. To gain a better understanding of the difficulty in solving/approximating monotone submodular k-partition, we encourage the reader to briefly think about the concrete special case of matrix column partitioning problem (i.e., problem (1) described above which is seemingly a linear algebra problem) before reading further.

[1] In fact, monotone functions are posimodular since $f(A) \geq f(A - B)$ and $f(B) \geq f(B - A)$ for every $A, B \subseteq V$. Symmetric submodular functions are posimodular: for every $A, B \subseteq V$, we have that $f(A) + f(B) = f(V - A) + f(B) \geq f((V - A) \cup B) + f((V - A) \cap B) = f(V - (A - B)) + f(B - A) = f(A - B) + f(B - A)$.

Monotone submodular k-partition is NP-hard [23]. Moreover, it admits a simple (and fast) $(2-1/k)$-approximation algorithm that will be denoted henceforth as the *cheapest singleton partitioning algorithm*: return the partition $V_1 := \{v_1\}, V_2 := \{v_2\}, \ldots, V_{k-1} := \{v_{k-1}\}, V_k := V - \{v_1, \ldots, v_{k-1}\})$, where the n elements of the ground set are ordered as $v_1, \ldots, v_n$ such that $f(\{v_1\}) \leq f(\{v_2\}) \leq \ldots \leq f(\{v_n\})$. Santiago [23] showed that this is a 2-approximation. His analysis can be extended to show that it is in fact a $(2-1/k)$-approximation[2] and this is the best possible approximation factor for this algorithm[3]. Alternatively, the greedy splitting algorithm presented in [26] achieves a $(2-2/k)$-approximation. On the inapproximability front, Santiago [23] showed that there does not exist an algorithm that makes polynomial number of function evaluation queries to obtain a $(4/3-\epsilon)$-approximation for every constant $\epsilon > 0$.

Symmetric submodular k-partition. Well-known special cases of symmetric submodular k-partition problem are graph k-cut and hypergraph k-partition – the submodular functions of interest here are the cut functions of an explicitly given graph and hypergraph respectively. Graph k-cut is NP-complete [10] and does not have a polynomial-time $(2-\epsilon)$-approximation for every constant $\epsilon > 0$ under the Small Set Expansion Hypothesis [12]. There are several known approaches to achieve a 2-approximation for graph k-cut – (i) greedy splitting approach [24], (ii) Gomory-Hu tree based approach [25], (iii) extreme sets based approach [13], (iv) principal partition sequence based approach [2, 18, 22], and (v) covering-LP based approach [7, 15, 21]. Greedy splitting, Gomory-Hu tree, and extreme sets based approaches lead to a $(2-2/k)$-approximation while the principal partition sequence and the covering-LP based approaches lead to a $(2-2/n)$-approximation for graph k-cut. The principal partition sequence and the covering-LP based approaches for graph k-cut have also been shown to be related to each other [7]. The principal partition sequence based approach is the main algorithm of interest to our work and we will discuss it in detail in Section 2.

For the more general problem of symmetric submodular k-partition, two of the approaches discussed in the previous paragraph for graph k-cut have been generalized to obtain 2-approximations – the greedy splitting approach [26] and the Gomory-Hu tree approach lead to a $(2-2/k)$-approximation. Analyzing the approximation factor of the principal partition sequence based approach for symmetric submodular k-partition was one of the driving motivations of our work. On the inapproximability front, Santiago [23] showed that there does not exist an algorithm that makes polynomial number of function evaluation queries to obtain a $(2-\epsilon)$-approximation for every constant $\epsilon > 0$.

[2] The cheapest singleton partitioning algorithm returns a solution whose cost is $f(V - V_k) + \sum_{i=1}^{k-1} f(\{v_i\}) \leq f(V) + \sum_{i=1}^{k-1} f(\{v_i\}) \leq f(V) + (1-1/k)\sum_{i=1}^{k} f(\{v_i\}) \leq (2-1/k)\max\{f(V), \sum_{i=1}^{k} f(\{v_i\})\}$ while the cost of an optimum k-partition is at least $\max\{f(V), \sum_{i=1}^{k} f(\{v_i\})\}$. The lower bound on the cost of the optimum k-partition $V_1^*, \ldots, V_k^*$ is because $\sum_{i=1}^{k} f(V_i^*) \geq f(V)$ by non-negativity and submodularity and moreover, if the optimum partition is indexed such that $\min\{j \in [n] : v_j \in V_i^*\} \leq \min\{j \in [n] : v_j \in V_{i+1}^*\}$ for all $i \in [k-1]$, then $f(V_i^*) \geq f(\{v_i\})$ by monotonicity and hence, $\sum_{i=1}^{k} f(V_i^*) \geq \sum_{i=1}^{k} f(\{v_i\})$.

[3] The best possible approximation factor for the cheapest singleton partitioning algorithm is $2-1/k$ as seen from this example: Let f be the rank function of a partition matroid on a k-partition $\{S_1, \ldots, S_k\}$ of the ground set S where $|S_i| \geq 2$ for all $i \in [k]$. Then, the algorithm may return $\{\{s_1\}, \{s_2\}, \ldots, \{s_{k-1}\}, S - \{s_1, \ldots, s_{k-1}\}\}$, where $s_i \in S_i$ for all $i \in [k-1]$ and this k-partition has objective value $2k-1$, whereas the partition $\{S_1, \ldots, S_k\}$ has objective value k.

Posimodular submodular k-partition. The only natural family of posimodular submodular functions that we are familiar with are symmetric submodular functions and monotone submodular functions as well as their positive linear combinations. As mentioned before, posimodular submodular k-partition is a unified generalization of symmetric submodular k-partition and monotone submodular k-partition. To the best of authors' knowledge, posimodular submodular k-partition has not been studied in the literature before and there are no specialized algorithms or approximation factor analysis of existing algorithms for posimodular submodular k-partition. A slight modification to the analysis of the greedy splitting algorithm presented in [26] shows that their algorithm achieves a $(3 - 2/k)$-approximation for posimodular submodular k-partition – we refrain from presenting this analysis in the interests of brevity. On the inapproximability front, since symmetric submodular functions are also posimodular submodular, the lower bound for symmetric submodular k-partition also holds for posimodular submodular k-partition, i.e., there does not exist an algorithm for posimodular submodular k-partition that makes polynomial number of function evaluation queries to obtain a $(2 - \epsilon)$-approximation for every constant $\epsilon > 0$.

1.1 Our Results

In this work, we investigate Narayanan, Roy, and Patkar's [18] principal partition sequence based algorithm for submodular k-partition. They showed that their algorithm achieves a 2-approximation for graph k-cut (which was subsequently rediscovered by Ravi and Sinha [22]). We show the following results:

1. Their algorithm achieves a 4/3-approximation for monotone submodular k-partition. This result improves on the 2-factor that is known to be achievable via two different algorithms: the cheapest singleton partitioning algorithm and the greedy splitting algorithm. Moreover, our upper bound of 4/3 matches the lower bound shown by Santiago [23]. We will discuss the significance of our upper bound result shortly.
2. Their algorithm achieves a 2-approximation for symmetric submodular k-partition. This factor matches the 2-factor that is known to be achievable via two other algorithms: the greedy splitting algorithm and the Gomory-Hu tree based algorithm, and also matches the lower bound [12, 23]. Our contribution here is generalizing the analysis of [18, 22] to beyond graph cut functions.
3. Their algorithm achieves a 2-approximation for posimodular submodular k-partition. This result improves on the 3-factor that is known to be achievable via the greedy splitting algorithm and matches the lower bound of 2 shown by Santiago [23].

See Table 1 for a comparison. Graph k-cut is the well-studied special case of symmetric submodular k-partition/posimodular submodular k-partition, so we include that as the last column in the table for comparison. Approximation factors in the row corresponding to principal partition sequence are the main results of this work. In the last row of the table, we include the known lower bounds on the approximation factor for comparison. The lower bound for graph cut function is assuming the Small Set Expansion Hypothesis [12] while the rest of the lower bounds are assuming polynomial number of function evaluation queries. Dashes in the table indicate that either the approach does not extend or there has been no analysis of the approximation factor of the approach for the subfamily.

We complement our upper bounds on the approximation factor of their algorithm with matching lower bound constructions for each subfamily of submodular functions. Our results show that the principal partition sequence based algorithm achieves the best possible approximation factor for broad subfamilies of submodular functions, thus illustrating the power and applicability of this algorithm. On the other hand, we show that the approximation factor of their algorithm for arbitrary submodular functions is $\Omega(n/k)$ via a lower bound construction. This construction shows that their principal partition sequence based algorithm cannot directly improve the approximation factor for submodular k-partition beyond the current best $O(k)$.

We briefly discuss the significance of our 4/3-approximation result for monotone submodular k-partition. Firstly, prior to our results, there were no known families of submodular functions for which the submodular k-partition problem could be approximated to a factor better than 2. Our result for monotone submodular functions breaks this 2-factor barrier for a broad family of submodular functions. Secondly, our result for monotone submodular k-partition leads to a new approximation result even for a graph partitioning problem that we describe now. For a graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}_+$, consider functions $d, f : 2^V \to \mathbb{R}_+$ defined by $d(S) := w(\delta(S))$ and $f(S) := w(E[S]) + w(\delta(S))$ for every $S \subseteq V$, where $\delta(S)$ denotes the set of edges with exactly one end-vertex in S, $E[S]$ denotes the set of edges with both end-vertices in S, and $w(F) := \sum_{e\in F} w(e)$ for every $F \subseteq E$. The function d is the cut function of the graph and is symmetric submodular. The function f is the coverage function of the graph and is monotone submodular. Submodular k-partition for the function d is known as graph k-cut and it is known that graph k-cut does not admit a $(2-\epsilon)$-approximation under the Small Set Expansion Hypothesis [12]. In contrast, our results show that coverage k-partition in graphs – i.e., submodular k-partition for the function f – admits a 4/3-approximation. We note that coverage k-partition in graphs is NP-hard [23] and its approximability is an intriguing open question.

Table 1 Approximation factors of symmetric/monotone/posimodular submodular k-partition using different approaches. Result in the first row marked with an asterisk follows by slight modifications to the known analysis of the approximation factor for symmetric submodular functions given in [26].

	Monotone Submodular Function	Symmetric Submodular Function	Posimodular Submodular Function	Graph Cut Function
Greedy splitting	$2-2/k$ [26]	$2-2/k$ [26]	$3-2/k$ [26]*	$2-2/k$ [24]
Extreme Sets	—	—	—	$2-2/k$ [13]
Gomory-Hu tree	—	$2-2/k$ [Folklore]	—	$2-2/k$ [25]
Covering-LP	—	—	—	$2-2/k$ [7, 15]
Cheapest Singleton Partitioning	$2-1/k$ [23]	—	—	—
Principal Partition Sequence	$4/3-4/(9n+3)$ (this work)	$2-2/n$ (this work)	$2-2/(n+1)$ (this work)	$2-2/n$ [18, 22]
Lower Bound	$4/3-\epsilon$ [23]	$2-\epsilon$ [23]	$2-\epsilon$ [23]	$\mathbf{2-\epsilon}$ [12]

Organization. We discuss the principal partition sequence based algorithm in Section 2. We analyze the approximation factor of the algorithm with matching lower bound constructions for monotone submodular k-partition in 3. Due to space limitations, we refer the reader to the full version of the work [5] for the analysis of the approximation factor of the constructionsalgorithm with matching lower bound constructions for symmetric submodular k-partition and posimodular submodular k-partition. We exhibit an instance of submodular k-partition where the algorithm achieves an approximation factor of $\Omega(n/k)$ in Section 4 and conclude with certain open questions in Section 5.

1.2 Related work

The principal partition sequence based algorithm for submodular k-partition was introduced by Narayanan, Roy, and Patkar [18]. We will formally define the principal partition sequence of a submodular function and describe their algorithm in Section 2. They analyzed the approximation factor of their algorithm for two variants of k-partitioning problems in hypergraphs. These two variants are not special cases of symmetric/monotone/posimodular submodular k-partition and are not of direct interest to our work. However, we describe these variants to highlight the versatility of the principal partition sequence based approach and also to shed light on the results of Narayanan, Roy, and Patkar's work which do not seem to be well-known in the literature. Given a hypergraph $H = (V, E)$, a hyperedge cost function $c : E \to \mathbb{R}_+$, and an integer k, the goal is to find a partition $\mathcal{P} := \{V_1, V_2, \ldots, V_k\}$ of V into k non-empty parts that minimizes an objective of interest:

1. If the objective is the sum of cost of hyperedges that intersect at least two parts of $\mathcal{P}$, then the problem is known as *hypergraph k-cut.*

2. If the objective is the sum of *cost of hyperedges relative to the partition* $\mathcal{P}$, where the cost of a hyperedge e relative to $\mathcal{P}$ is $c(e)(\ell - 1)$ with ℓ being the number of parts of $\mathcal{P}$ intersected by e, then the problem is known as *normalized coverage k-partition*[4].

Narayanan, Roy, and Patkar [18] showed that their principal partition sequence based algorithm achieves a $r(1 - 1/n)$-approximation for hypergraph k-cut, where r is the size of the largest hyperedge and n is the number of vertices in the input hypergraph, and achieves a $(2 - 2/n)$-approximation for normalized coverage k-partition. A consequence (of both of their results) is that the principal partition sequence based algorithm achieves a $(2 - 2/n)$-approximation for graph k-cut. Their principal partition sequence based algorithm for graph k-cut is equivalent to the Lagrangean relaxation approach suggested by Barahona [2]. The approximation factor of the principal partition sequence based algorithm for graph k-cut being at most 2 was rediscovered by Ravi and Sinha [22] and for hypergraph k-cut being at most r was rediscovered by Baïou and Barahona [1].

We mention that a slight modification to the analysis of the greedy splitting algorithm presented in [26] shows that the greedy splitting algorithm achieves a $(2-2/k)$-approximation for normalized coverage k-partition and hypergraph k-partition. We note that hypergraph k-partition is a special case of symmetric submodular k-partition and is different from hypergraph k-cut (for definition of hypergraph k-partition, see discussion of symmetric submodular k-partition at the beginning of the introduction). On the inapproximability front, it is known that hypergraph k-cut does not admit an approximation factor of $n^{1/(\log \log n)^c}$, where c is a constant, assuming the Exponential Time Hypothesis [6]. The best inapproximability result for the more general submodular k-partition problem (mentioned in the introduction) follows from this inapproximability for hypergraph k-cut.

[4] We introduce this nomenclature because the problem is equivalent to finding a partition $V_1, V_2, \ldots, V_k$ of the ground set V in order to minimize $\sum_{i=1}^{k} f(V_i) - f(V)$, where $f : 2^V \to \mathbb{R}_+$ is an explicitly given coverage function (every coverage function can be uniquely represented using a hypergraph [3]). We consider the subtraction of $f(V)$ as normalizing the objective since it is a trivial lower bound on the sum of the function values of the parts: $\sum_{i=1}^{k} f(V_i) \geq f(V)$ holds for every k-partition $V_1, V_2, \ldots, V_k$ since f is a coverage function.

2 Principal partition sequence based algorithm

In this section, we recall the principal partition sequence based algorithm for submodular k-partition designed by Narayanan, Roy, and Patkar [18]. We begin with some notation. Throughout this work, a *partition* of a set S is defined to be a collection of *nonempty* pairwise disjoint subsets of S whose union is S, and a *k-partition* of a set S is defined to be a partition of S with exactly k parts. For two distinct partitions $\mathcal{P}$ and $\mathcal{Q}$ of a set S, if every part of $\mathcal{Q}$ is completely contained in some part of $\mathcal{P}$ then we say that $\mathcal{Q}$ *refines* $\mathcal{P}$ (equivalently, $\mathcal{P}$ is a coarsening of $\mathcal{Q}$). For two distinct partitions $\mathcal{P}$ and $\mathcal{Q}$ of a set S, we will say that $\mathcal{Q}$ is obtained from $\mathcal{P}$ by refining only one part of $\mathcal{P}$ if there exists a part $P \in \mathcal{P}$ such that $P \notin \mathcal{Q}$ and every part $Q \in \mathcal{Q}$ satisfies either $Q \subsetneq P$ or $Q \in \mathcal{P}$ (i.e., either Q is a proper subset of the part P or Q is a part of the partition $\mathcal{P}$); we will denote such a part $P \in \mathcal{P}$ as the part refined by $\mathcal{Q}$.

Let $f : 2^V \to \mathbb{R}$ be a set function on ground set V. For a collection $\mathcal{P}$ of subsets of V, we write $f(\mathcal{P}) := \sum_{P\in\mathcal{P}} f(P)$. We will say that a partition $\mathcal{P} = \{P_1, \ldots, P_k\}$ is an optimal k-partition if $f(\mathcal{P}) \le f(\mathcal{Q})$ for every k-partition $\mathcal{Q}$ of V. We define the function $g_{f,\mathcal{P}} : \mathbb{R}_{\ge 0} \to \mathbb{R}$ for a partition $\mathcal{P}$ of the ground set V and the function $g_f : \mathbb{R}_{\ge 0} \to \mathbb{R}$ as follows:

$$\begin{aligned} g_{f,\mathcal{P}}(b) &:= f(\mathcal{P}) - b|\mathcal{P}| \text{ and} \\ g_f(b) &:= \min\{g_{f,\mathcal{P}}(b) : \mathcal{P} \text{ is a partition of } V\}. \end{aligned}$$

We drop the subscript f and instead write $g_{\mathcal{P}}$ and g respectively, if the function f is clear from context. By definition, the function g_f is piece-wise linear. It can be shown that g_f has at most $|V| - 1$ breakpoints. The next theorem shows that if the function $f : 2^V \to \mathbb{R}$ is submodular, then there exists a sequence of partitions achieving the g_f function values at the breakpoints that have a nice structure; moreover, the breakpoints and such a sequence of partitions can be computed in polynomial time given access to the evaluation oracle of the submodular function f. We emphasize that the theorem holds for arbitrary submodular functions (which may not be non-negative valued).

▶ **Theorem 1** ([16, 18]). *Let $f : 2^V \to \mathbb{R}$ be a submodular function on a ground set V. Then, there exists a sequence $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_r$ of partitions of V and values $b_1, b_2, \ldots, b_{r-1}$ such that*

1. *$\mathcal{P}_1 = \{V\}$ and $\mathcal{P}_r = \{\{v\} : v \in V\}$,*
2. *For each $j \in [r-1]$, the partition $\mathcal{P}_{j+1}$ is obtained from $\mathcal{P}_j$ by refining only one part of $\mathcal{P}_j$,*
3. *$b_1 < b_2 < \ldots < b_{r-1}$,*
4. *$g(b_j) = g_{\mathcal{P}_j}(b_j) = g_{\mathcal{P}_{j+1}}(b_j)$ for each $j \in [r-1]$ and*
5. *$g(b) = g_{\mathcal{P}_1}(b)$ for all $b \in (-\infty, b_1]$,*
   $g(b) = g_{\mathcal{P}_{j+1}}(b)$ for all $b \in [b_j, b_{j+1}]$ for each $j \in [r-2]$, and
   $g(b) = g_{\mathcal{P}_r}(b)$ for all $b \in [b_{r-1}, \infty)$.

Moreover, such a sequence $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_r$ of partitions of V and values $b_1, b_2, \ldots, b_{r-1}$ can be computed in polynomial time given access to the evaluation oracle of the submodular function f.

For a submodular function $f : 2^V \to \mathbb{R}$, we will denote a sequence of partitions $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_r$ and the sequence of values $b_1, b_2, \ldots, b_{r-1}$ satisfying the conditions given in Theorem 1 as a *principal partition sequence* and the *critical value sequence* of f, respectively. We note that this definition differs from those in [16, 18] owing to the reversed indexing order and the imposition of condition 2 – we note that the proofs given in those papers also show that condition 2 holds (also see [22]). The principal partition sequence of submodular functions is known in the literature as *principal lattice of partitions* of submodular functions since there exists a lattice structure associated with the sequence of partitions. We choose to call it as principal partition sequence in this work since the sequence suffices for our purpose. For more on principal lattice of partitions of submodular functions and their computation, we refer the reader to [2, 8, 9, 11, 14, 16–18, 20].

We now discuss the principal partition sequence based algorithm for submodular k-partition that was proposed by Narayanan, Roy, and Patkar [18]. This algorithm computes a principal partition sequence satisfying all conditions in Theorem 1. If the sequence contains a partition that has exactly k parts, then the algorithm returns this k-partition. Otherwise, the algorithm returns a k-partition obtained by refining the partition in the sequence that has the largest number of parts that is less than k. The refinement is based on the partition in the sequence that has the fewest number of parts that is more than k. The formal description of the refinement is given in Algorithm 1. Since the sequence $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_r$ satisfying the conditions of Theorem 1 can be computed in polynomial time, Algorithm 1 can indeed be implemented to run in polynomial time. By design, the algorithm returns a k-partition. The remainder of this work will focus on analyzing the approximation factor of the algorithm.

Algorithm 1 Principal partition sequence based algorithm for submodular k-partition.

```
Input: A submodular function $f : 2^V \to \mathbb{R}$ given by evaluation oracles and an integer $k \geq 2$.
Output: A $k$-partition $\mathcal{P}$ of $V$.
Use Theorem 1 to compute a principal partition sequence $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_r$ of the submodular function $f$ satisfying all conditions stated in that theorem.
if $|\mathcal{P}_j| = k$ for some $j \in [r]$ then
  Return $\mathcal{P} := \mathcal{P}_j$.
end if
Let $i \in \{2, 3, \ldots, r\}$ so that $|\mathcal{P}_{i-1}| < k < |\mathcal{P}_i|$.
Let $S \in \mathcal{P}_{i-1}$ be the part refined by $\mathcal{P}_i$ and $\mathcal{P}'$ be the parts of $\mathcal{P}_i$ contained in $S$.
Let $\mathcal{P}' = \{B_1, \ldots, B_{|\mathcal{P}'|}\}$ such that $f(B_1) \leq f(B_2) \leq \ldots \leq f(B_{|\mathcal{P}'|})$.
Return $\mathcal{P} := (\mathcal{P}_{i-1} \backslash \{S\}) \cup \left\{B_1, B_2, \ldots, B_{k-|\mathcal{P}_{i-1}|}\right\} \cup \left\{\bigcup_{j=k-|\mathcal{P}_{i-1}|+1}^{|\mathcal{P}'|} B_j\right\}$.
```

To construct examples that exhibit tight lower bound on the approximation factor of Algorithm 1, we will need the following proposition that identifies a special case under which the principal partition is unique and consists only of two partitions – namely, the partition into singletons and the partition that consists of only one part. We refer the reader to the full version of this work for the proof [5].

▶ **Proposition 2.** *Let $f : 2^V \to \mathbb{R}_{\geq 0}$ be a non-negative submodular function. Suppose that for every partition $\mathcal{P} \neq \mathcal{Q}, \{V\}$ where $\mathcal{Q} := \{\{v\} : v \in V\}$, the function f satisfies*

$$\frac{f(\mathcal{P}) - f(V)}{|\mathcal{P}| - 1} > \frac{f(\mathcal{Q}) - f(V)}{|V| - 1}.$$

Then, the principal partition sequence of f is $\{V\}, \mathcal{Q}$.

3 Approximation factor analysis

In this section, we analyze the approximation factor of Algorithm 1 for various subfamilies of submodular functions. We state and prove certain lemmas that will be useful for all submodular functions (Lemmas 3 and 4). We analyze the approximation factor of Algorithm 1 for monotone submodular functions in Section 3.1. We refer the reader to the full version of this work [5] for the analysis of the approximation factor of Algorithm 1 for symmetric submodular functions and for posimodular submodular functions.

Our first lemma identifies a special case in which Algorithm 1 returns an optimum k-partition. This special case was also identified by Narayanan, Roy, and Patkar. We note that the following lemma holds for arbitrary submodular functions (which may not be non-negative).

▶ **Lemma 3** ([18]). *Let $k \geq 2$ be an integer, $f : 2^V \to \mathbb{R}$ be a submodular function on a ground set V, and $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_r$ be a principal partition sequence of the submodular function f satisfying the conditions of Theorem 1. If there exists $j \in [r]$ such that $|\mathcal{P}_j| = k$, then $\mathcal{P}_j$ is an optimal k-partition.*

Proof. Let $\mathcal{P}^*$ be a k-partition of V that minimizes $f(\mathcal{P}^*)$ and let b_j be the value where $g(b_j) = g_{\mathcal{P}_j}(b_j)$. Then,

$$f(\mathcal{P}^*) - b_j \cdot k = g_{\mathcal{P}^*}(b_j) \geq g(b_j) = g_{\mathcal{P}_j}(b_j) = f(\mathcal{P}_j) - b_j|\mathcal{P}_j| = f(\mathcal{P}_j) - b_j \cdot k,$$

and hence, $f(\mathcal{P}^*) \geq f(\mathcal{P}_j)$. Therefore, $\mathcal{P}_j$ is indeed an optimal k-partition. ◀

In order to address the case where there is no $j \in [r]$ such that $|\mathcal{P}_j| = k$, we need the following lemma that shows two lower bounds on the optimum value. The first lower bound in the lemma holds for arbitrary submodular functions (which may not be non-negative) while the second lower bound holds for non-negative submodular functions.

▶ **Lemma 4.** *Let $k \geq 2$ be an integer, $f : 2^V \to \mathbb{R}$ be a submodular function on a ground set V, $\mathcal{P}^*$ be a k-partition of V that minimizes $f(\mathcal{P}^*)$, and $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_r$ be a principal partition sequence of the submodular function f satisfying the conditions of Theorem 1. Suppose $|\mathcal{P}_j| \neq k$ for all $j \in [r]$. Let $\mathcal{P}_{i-1}, \mathcal{P}_i$ be the partitions such that $|\mathcal{P}_{i-1}| < k < |\mathcal{P}_i|$. Then,*
(i) *$f(\mathcal{P}^*) \geq \frac{|\mathcal{P}_i|-k}{|\mathcal{P}_i|-|\mathcal{P}_{i-1}|} f(\mathcal{P}_{i-1}) + \frac{k-|\mathcal{P}_{i-1}|}{|\mathcal{P}_i|-|\mathcal{P}_{i-1}|} f(\mathcal{P}_i)$ and*
(ii) *$f(\mathcal{P}^*) \geq f(\mathcal{P}_{i-1})$ if f is non-negative.*

Proof. We prove the two lower bounds below.
(i) Let b_{i-1} be the value such that $g_{\mathcal{P}_{i-1}}(b_{i-1}) = g_{\mathcal{P}_i}(b_{i-1})$. Then, we have

$$f(\mathcal{P}_{i-1}) - b_{i-1}|\mathcal{P}_{i-1}| = f(\mathcal{P}_i) - b_{i-1}|\mathcal{P}_i| \implies b_{i-1} = \frac{f(\mathcal{P}_i) - f(\mathcal{P}_{i-1})}{|\mathcal{P}_i| - |\mathcal{P}_{i-1}|}. \tag{1}$$

By condition 4 of Theorem 1, we also have

$$\begin{aligned} f(\mathcal{P}^*) - b_{i-1} \cdot k = g_{\mathcal{P}^*}(b_{i-1}) \geq g(b_{i-1}) = g_{\mathcal{P}_i}(b_{i-1}) = f(\mathcal{P}_i) - b_{i-1}|\mathcal{P}_i| \\ \implies f(\mathcal{P}^*) \geq f(\mathcal{P}_i) + b_{i-1}(k - |\mathcal{P}_i|). \end{aligned} \tag{2}$$

Combining (1) and (2), we get

$$\begin{aligned} f(\mathcal{P}^*) &\geq f(\mathcal{P}_i) + \frac{f(\mathcal{P}_i) - f(\mathcal{P}_{i-1})}{|P_i| - |P_{i-1}|}(k - |\mathcal{P}_i|) \\ &= \frac{|\mathcal{P}_i| - k}{|\mathcal{P}_i| - |\mathcal{P}_{i-1}|} f(\mathcal{P}_{i-1}) + \frac{k - |\mathcal{P}_{i-1}|}{|\mathcal{P}_i| - |\mathcal{P}_{i-1}|} f(\mathcal{P}_i). \end{aligned}$$

(ii) Let $P_1^*, \ldots, P_k^*$ be the parts of $\mathcal{P}^*$ (in arbitrary order). Let $k' := |\mathcal{P}_{i-1}|$. We know that $k' < k$. Consider the k'-partition $\mathcal{Q}$ obtained as $Q_1 := P_1^*, Q_2 := P_2^*, \ldots, Q_{k'-1} := P_{k'-1}^*, Q_{k'} := \cup_{j=k'}^{k} P_j^*$. Then,

$$f(\mathcal{P}^*) = \sum_{i=1}^{k} f(P_i^*) \geq \left(\sum_{i=1}^{k'-1} f(P_i^*) \right) + f(\cup_{j=k'}^{k} P_j^*) = \sum_{i=1}^{k'} f(Q_i) = f(\mathcal{Q}).$$

The inequality above is due to submodularity and non-negativity. The partition $\mathcal{Q}$ is a k'-partition. By Lemma 3, the partition $\mathcal{P}_{i-1}$ is an optimal k'-partition. Hence,

$$f(\mathcal{Q}) \geq f(\mathcal{P}_{i-1}).$$

The above two inequalities together imply that $f(\mathcal{P}^*) \geq f(\mathcal{P}_{i-1})$. ◀

3.1 Monotone submodular functions

In this section, we bound the approximation factor of Algorithm 1 for monotone submodular k-partitioning. The following is the main theorem of this section.

▶ **Theorem 5.** *The approximation factor of Algorithm 1 for non-negative monotone submodular k-partitioning is $\frac{4}{3} - \frac{4}{9n+3}$, where n is the size of the ground set.*

The asymptotic approximation factor of 4/3 achieved by Algorithm 1 is the best possible for non-negative monotone submodular k-partition: for every constant $\epsilon > 0$, there does not exist an algorithm that achieves a $(4/3 - \epsilon)$-approximation using polynomial number of function evaluation queries [23]. We will also exhibit examples to show the tightness of the approximation factor for Algorithm 1 after proving the theorem. The proof of Theorem 5 follows from Lemma 3 and Lemma 6 shown below.

▶ **Lemma 6.** *Let $k \geq 2$ be an integer, $f : 2^V \to \mathbb{R}_{\geq 0}$ be a non-negative monotone submodular function on a ground set V of size n, $\mathcal{P}^*$ be a k-partition of V that minimizes $f(\mathcal{P}^*)$, and $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_r$ be a principal partition sequence of the submodular function f satisfying the conditions of Theorem 1. Suppose $|\mathcal{P}_j| \neq k$ for all $j \in [r]$. Then, the partition $\mathcal{P}$ returned by Algorithm 1 satisfies*

$$f(\mathcal{P}) \leq \left(\frac{4}{3} - \frac{4}{9n+3} \right) f(\mathcal{P}^*).$$

Proof. Let $\mathcal{P}_{i-1}, \mathcal{P}_i$ be the partitions such that $|\mathcal{P}_{i-1}| < k < |\mathcal{P}_i|$. Let S and $\mathcal{P}' = \{B_1, B_2, \ldots, B_{|\mathcal{P}'|}\}$ be as in Algorithm 1.

Firstly, since $\cup_{j=k-|\mathcal{P}_{i-1}|+1}^{|\mathcal{P}'|} B_j \subseteq S$ and f is monotone, we have that

$$f\left(\cup_{j=k-|\mathcal{P}_{i-1}|+1}^{|\mathcal{P}'|} B_j \right) \leq f(S).$$

Secondly, by our choice of $B_1, B_2, \ldots, B_{k-|\mathcal{P}_{i-1}|}$, we know that

$$\sum_{j=1}^{k-|\mathcal{P}_{i-1}|} f(B_j) \leq \frac{k - |\mathcal{P}_{i-1}|}{|\mathcal{P}'|} f(\mathcal{P}').$$

Hence,

$$\begin{aligned}
f(\mathcal{P}) &= f(\mathcal{P}_{i-1}) - f(S) + \sum_{j=1}^{k-|\mathcal{P}_{i-1}|} f(B_j) + f\left(\bigcup_{j=k-|\mathcal{P}_{i-1}|+1}^{|\mathcal{P}'|} B_j\right) \\
&\le f(\mathcal{P}_{i-1}) + \sum_{j=1}^{k-|\mathcal{P}_{i-1}|} f(B_j) \\
&\le f(\mathcal{P}_{i-1}) + \frac{k-|\mathcal{P}_{i-1}|}{|\mathcal{P}'|} f(\mathcal{P}') \\
&= f(\mathcal{P}_{i-1}) + \frac{k-|\mathcal{P}_{i-1}|}{|\mathcal{P}_i|-|\mathcal{P}_{i-1}|+1}(f(\mathcal{P}_i)+f(S)-f(\mathcal{P}_{i-1})),
\end{aligned}$$

where the last equality follows from the fact that $f(\mathcal{P}_{i-1}) - f(S) + f(\mathcal{P}') = f(\mathcal{P}_i)$ and $|\mathcal{P}'| = |\mathcal{P}_i| - |\mathcal{P}_{i-1}| + 1$. Rearranging, we have that

$$\begin{aligned}
f(\mathcal{P}) &\le \frac{f(\mathcal{P}_{i-1})}{|\mathcal{P}_i|-|\mathcal{P}_{i-1}|+1}(|\mathcal{P}_i|-k+1) + \frac{f(\mathcal{P}_i)}{|\mathcal{P}_i|-|\mathcal{P}_{i-1}|+1}(k-|\mathcal{P}_{i-1}|) \\
&\qquad + \frac{f(S)}{|\mathcal{P}_i|-|\mathcal{P}_{i-1}|+1}(k-|\mathcal{P}_{i-1}|) \\
&= \left(\frac{|\mathcal{P}_i|-|\mathcal{P}_{i-1}|}{|\mathcal{P}_i|-|\mathcal{P}_{i-1}|+1}\right)\left(\frac{f(\mathcal{P}_{i-1})}{|\mathcal{P}_i|-|\mathcal{P}_{i-1}|}(|\mathcal{P}_i|-k+1) + \frac{f(\mathcal{P}_i)+f(S)}{|\mathcal{P}_i|-|\mathcal{P}_{i-1}|}(k-|\mathcal{P}_{i-1}|)\right) \\
&\le \left(\frac{|\mathcal{P}_i|-|\mathcal{P}_{i-1}|}{|\mathcal{P}_i|-|\mathcal{P}_{i-1}|+1}\right)\left(f(\mathcal{P}^*) + \frac{f(\mathcal{P}_{i-1})}{|\mathcal{P}_i|-|\mathcal{P}_{i-1}|} + \frac{f(S)}{|\mathcal{P}_i|-|\mathcal{P}_{i-1}|}(k-|\mathcal{P}_{i-1}|)\right) \\
&\le \left(\frac{|\mathcal{P}_i|-|\mathcal{P}_{i-1}|}{|\mathcal{P}_i|-|\mathcal{P}_{i-1}|+1}\right)\left(f(\mathcal{P}^*) + \frac{f(\mathcal{P}_{i-1})}{|\mathcal{P}_i|-|\mathcal{P}_{i-1}|}(1+k-|\mathcal{P}_{i-1}|)\right). \qquad (3)
\end{aligned}$$

where the second inequality above is by by Lemma 4(i) and the third inequality above is because $f(S) \le f(\mathcal{P}_{i-1})$. Inequality (3) implies that

$$\begin{aligned}
f(\mathcal{P}) &\le \left(\frac{|\mathcal{P}_i|-|\mathcal{P}_{i-1}|}{|\mathcal{P}_i|-|\mathcal{P}_{i-1}|+1}\right)\left(f(\mathcal{P}^*) + f(\mathcal{P}_{i-1}) + \frac{f(\mathcal{P}_{i-1})}{|\mathcal{P}_i|-|\mathcal{P}_{i-1}|}(1+k-|\mathcal{P}_i|)\right) \\
&\le \left(1 - \frac{1}{|\mathcal{P}_i|-|\mathcal{P}_{i-1}|+1}\right)(f(\mathcal{P}^*)+f(\mathcal{P}_{i-1})) \quad \text{(since } k < |\mathcal{P}_i|\text{)} \\
&\le 2f(\mathcal{P}^*),
\end{aligned}$$

where the last inequality is because $f(\mathcal{P}_{i-1}) \le f(\mathcal{P}^*)$ by Lemma 4(ii). The above analysis shows that the approximation factor is at most 2. We tighten the analysis now. As a consequence of the above inequality, we may assume that $f(\mathcal{P}^*) \neq 0$ because if $f(\mathcal{P}^*) = 0$, then the returned k-partition $\mathcal{P}$ also satisfies $f(\mathcal{P}) = 0$ and thus, is optimal. Let $c := f(\mathcal{P}_{i-1})/f(\mathcal{P}^*)$. By Lemma 4(ii), we have that $f(\mathcal{P}_{i-1}) \le f(\mathcal{P}^*)$ and hence, $c \in [0,1]$. For convenience, we define $A := k - |\mathcal{P}_{i-1}|$ and $B := |\mathcal{P}_i| - k$ and note that $A, B \ge 1$. Using this notation, we may rewrite inequality (3) as

$$\begin{aligned}
f(\mathcal{P}) &\le \left(\frac{A+B}{A+B+1}\right)\left(f(\mathcal{P}^*) + \frac{1+A}{A+B} f(\mathcal{P}_{i-1})\right) \\
&= \left(\frac{A+B}{A+B+1}\right)\left(1 + \frac{1+A}{A+B}\cdot c\right) f(\mathcal{P}^*). \qquad (4)
\end{aligned}$$

By Lemma 4(i), we have

$$f(\mathcal{P}^*) \ge \left(\frac{B}{A+B}\right) f(\mathcal{P}_{i-1}) + \left(\frac{A}{A+B}\right) f(\mathcal{P}_i) = \left(\frac{B}{A+B}\right) c f(\mathcal{P}^*) + \left(\frac{A}{A+B}\right) f(\mathcal{P}_i).$$

Rearranging, we have

$$f(\mathcal{P}_i) \le \left(1 - \frac{B}{A+B} \cdot c\right)\left(\frac{A+B}{A}\right) f(\mathcal{P}^*) = \left(\frac{A+B}{A} - \frac{B}{A} \cdot c\right) f(\mathcal{P}^*).$$

Since $\mathcal{P}$ is obtained by coarsening $\mathcal{P}_i$, we have $f(\mathcal{P}) \le f(\mathcal{P}_i)$ by submodularity and non-negativity of f. This implies

$$f(\mathcal{P}) \le \left(\frac{A+B}{A} - \frac{B}{A} \cdot c\right) f(\mathcal{P}^*). \tag{5}$$

Combining inequalities (4) and (5), we have

$$\frac{f(\mathcal{P})}{f(\mathcal{P}^*)} \le \max_{c \in [0,1]} \min \left\{ \left(\frac{A+B}{A+B+1}\right)\left(1 + \frac{1+A}{A+B} \cdot c\right), \frac{A+B}{A} - \frac{B}{A} \cdot c \right\}. \tag{6}$$

Thus, in order to upper bound the approximation factor, it suffices to upper bound the right hand side of inequality (6). Since $\left(\frac{A+B}{A+B+1}\right)\left(1 + \frac{1+A}{A+B} \cdot c\right)$ and $\frac{A+B}{A} - \frac{B}{A} \cdot c$ are both linear in c, with the former increasing and the latter decreasing as a function of c, the value

$$\max_{c \in \mathbb{R}} \min \left\{ \left(\frac{A+B}{A+B+1}\right)\left(1 + \frac{1+A}{A+B} \cdot c\right), \frac{A+B}{A} - \frac{B}{A} \cdot c \right\}$$

is achieved when the two terms are equal. Setting $\left(\frac{A+B}{A+B+1}\right)\left(1 + \frac{1+A}{A+B} \cdot c^*\right) = \frac{A+B}{A} - \frac{B}{A} \cdot c^*$ and solving for c^*, we get

$$c^* = \frac{\frac{A+B}{A} - \frac{A+B}{A+B+1}}{\frac{1+A}{A+B+1} + \frac{B}{A}} = \frac{\frac{B}{A} + \frac{1}{A+B+1}}{\frac{1+A}{A+B+1} + \frac{B}{A}}.$$

Plugging $c = c^*$ into $\frac{A+B}{A} - \frac{B}{A} \cdot c$ yields

$$\begin{aligned}
\frac{f(\mathcal{P})}{f(\mathcal{P}^*)} &\le \max_{c \in [0,1]} \min \left\{ \frac{A+B}{A+B+1}\left(1 + \frac{1+A}{A+B} \cdot c\right), \frac{A+B}{A} - \frac{B}{A} \cdot c \right\} \\
&\le \frac{A+B}{A} - \frac{B}{A} \cdot c^* = 1 + \frac{B}{A}(1 - c^*) = 1 + \frac{B}{A}\left(1 - \frac{\frac{B}{A} + \frac{1}{A+B+1}}{\frac{1+A}{A+B+1} + \frac{B}{A}}\right) \\
&= 1 + \frac{AB}{A + A^2 + AB + B^2 + B} \\
&\le 1 + \frac{AB}{3AB + A + B} \quad (\text{since } A^2 + B^2 \ge 2AB) \\
&= \frac{4}{3} - \frac{1}{3} \cdot \frac{A+B}{3AB/2 + 3AB/2 + A + B} \\
&\le \frac{4}{3} - \frac{1}{3} \cdot \frac{A+B}{3AB/2 + 3(A^2 + B^2)/4 + A + B} \quad (\text{since } AB \le \frac{A^2 + B^2}{2}) \\
&= \frac{4}{3} - \frac{1}{3} \cdot \frac{1}{3(A+B)/4 + 1} \\
&\le \frac{4}{3} - \frac{4}{9n+3}.
\end{aligned}$$

The last inequality above is because $A + B = |\mathcal{P}_i| - |\mathcal{P}_{i-1}| \le n - 1$. ◀

▶ Remark 7. The approximation factor of Algorithm 1 for non-negative monotone submodular functions is at least $4/3 - 4/(9n+3)$. We show this for $n = 3$ using the following example: Let $V = \{a, b, c\}$, $k = 2$, and $f : 2^V \to \mathbb{R}_{\geq 0}$ be defined by

$$f(\emptyset) = 0,\ f(\{a\}) = 1,\ f(\{b\}) = f(\{c\}) = 1 + \epsilon,$$
$$f(\{a, b\}) = f(\{a, c\}) = \frac{3}{2} + \epsilon,\ f(\{b, c\}) = f(V) = 2 + 2\epsilon.$$

Submodularity and monotonicity of f can be verified by considering all possible subsets. Moreover, the principal partition sequence of this instance is $\{V\}, \{\{a\}, \{b\}, \{c\}\}$. Thus, Algorithm 1 returns the 2-partition $\{\{a\}, \{b, c\}\}$, whose objective value is $3+2\epsilon$. An optimum 2-partition is $\{\{c\}, \{a, b\}\}$, whose objective value is $5/2 + \epsilon$. Thus, the approximation factor is $\frac{3+2\epsilon}{5/2+\epsilon} \to \frac{6}{5}$ as $\epsilon \to 0$. We note that for $n = 3$, the approximation factor guaranteed by Theorem 5 is $\frac{4}{3} - \frac{4}{9n+3} = \frac{6}{5}$.

We conclude the section by showing that there exist monotone submodular functions for which the approximation factor of Algorithm 1 is at least 4/3 asymptotically (i.e., as $n \to \infty$).

▶ **Lemma 8.** *For every odd positive integer $n \geq 5$, there exists a function $k = k(n)$ (i.e., k is a function of n) and an instance of non-negative monotone submodular k-partition over an n-element ground set such that the approximation factor of Algorithm 1 on that instance is arbitrarily close to $4/3 - 4/(3n + 3)$.*

Proof. Let $n \geq 5$ be an arbitrary odd number, $k = \frac{n+1}{2}$, and let $V = \{v_1, \ldots, v_n\}$ be the ground set. Moreover, let $U := \{v_1, \ldots, v_{\frac{n-1}{2}}\}$ and $D := \{v_{\frac{n+1}{2}}, \ldots, v_n\}$ so that $V = U \uplus D$. Let $g : 2^U \to \mathbb{R}_{\geq 0}$ be a function over the ground set U defined by

$$g(S) = \begin{cases} \frac{1}{2} + \frac{1}{2} \cdot |S| & \text{if } \emptyset \neq S \subseteq U, \\ 0 & \text{if } S = \emptyset. \end{cases}$$

and $f : 2^V \to \mathbb{R}_{\geq 0}$ be defined by

$$f(S) := \min\left\{g(S \cap U) + (1 + \epsilon)|S \cap D|,\ \frac{n+1}{2}\right\}\ \forall\ S \subseteq V,$$

where $\epsilon > 0$ is infinitesimally small. The function f satisfies $f(\emptyset) = 0$, $f(V) = \frac{n+1}{2}$, $f(U) = \frac{n+1}{4}$, $f(D) = \frac{n+1}{2}$, $f(\{v\}) = 1$ for each $v \in U$, and $f(\{v\}) = 1 + \epsilon$ for each $u \in D$. We will use $\mathcal{Q}$ to denote the partition of V into n singleton sets. The following claims show properties about the function f. We refer the reader to the full version of this work for the proof [5].

▷ Claim 9. The function f is submodular and monotone.

▷ Claim 10. For every partition $\mathcal{P} \neq \mathcal{Q}, \{V\}$, the function f satisfies

$$\frac{f(\mathcal{P}) - f(V)}{|\mathcal{P}| - 1} > \frac{f(\mathcal{Q}) - f(V)}{n - 1}.$$

By Proposition 2, Algorithm 1 returns the partition $\{\{v_1\}, \ldots, \{v_{\frac{n-1}{2}}\}, D\}$ (because $\{v_1\}, \ldots, \{v_{\frac{n-1}{2}}\}$ are the $k - 1$ singleton sets that minimize the f values), whose objective is $\frac{n-1}{2} + \frac{n+1}{2} = n$. The partition $\{\{v_1, \ldots, v_{\frac{n+1}{2}}\}, \{v_{\frac{n+3}{2}}\}, \ldots, \{v_n\}\}$ has objective $\frac{n+1}{4} + (1 + \epsilon) + (1 + \epsilon)\frac{n-1}{2} = \frac{3n+3}{4} + \frac{(n+1)\epsilon}{2}$. The approximation factor is at least

$$\frac{n}{\frac{3n+3}{4} + \frac{(n+1)\epsilon}{2}} \to \frac{4}{3} - \frac{4}{3n + 3} \text{ (as } \epsilon \to 0).$$

This completes the proof of Lemma 8. ◀

4 Lower bound for arbitrary submodular functions

In this section, we present an instance of submodular k-partition where Algorithm 1 achieves an approximation factor of $\Omega(n/k)$. We emphasize that the submodular function in our instance is not symmetric/monotone/posimodular.

Let $V = \{v_0, v_1, \ldots, v_{n-1}\}$ be the ground set. We define a digraph $D = (V, E(D))$ and a hypergraph $H = (V, E(H))$ on the same vertex set V as follows (see Fig.1):

$$E(D) = \{v_0 v_i : i \in [n-1]\} \text{ and}$$
$$E(H) = \{\{v_1, v_2, \ldots, v_{n-1}\}\}.$$

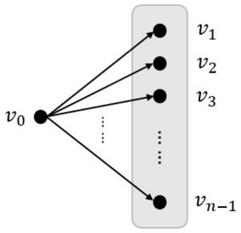


Figure 1 Example in Section 4. The arcs belong to the digraph D and the hyperedge $\{v_1, \ldots, v_{n-1}\}$ belongs to the hypergraph H.

For every subset $S \subseteq V$, we will use $d_D^{in}(S)$ to denote the number of arcs in D whose tails are in $\bar{S}$ and heads are in S. We will use $d_H(S)$ to denote the number of hyperedges in H that have at least one vertex in S and one vertex in $\bar{S}$. Next, we define a set function $f : 2^V \to \mathbb{R}_{\geq 0}$ by

$$f(S) := a \cdot d_D^{in}(S) + d_H(S) \quad \forall S \subseteq V,$$

where $a \gg 1$ is a large constant. We note that f is submodular because it is a positive linear combination of two submodular functions (and it is not monotone/symmetric/posimodular).

▷ Claim 11. The principal partition sequence of f is $\{V\}, \{\{v_i\} : i \in \{0, 1, \ldots, n-1\}\}$.

Proof. For convenience, we will use $\mathcal{Q}$ to denote the partition of V into singletons. By Proposition 2 and the fact that $f(V) = 0$, it suffices to prove that for every partition $\mathcal{P}$ of V such that $\mathcal{P}$ is not $\mathcal{Q}$ or $\{V\}$, we have that

$$\frac{f(\mathcal{P})}{|\mathcal{P}| - 1} > \frac{f(\mathcal{Q})}{n-1}. \tag{7}$$

Let $P_0 \in \mathcal{P}$ be the part that contains v_0. Then, we have that $f(P_0) \geq 1$ if $P_0 \neq \{v_0\}$ and $f(P_0) = 0$ otherwise. For each part $P \in \mathcal{P}$ that does not contain v_0, we have that $f(P) \geq 1+a$ if $|P| = 1$ and $f(P) \geq 2+a$ if $|P| \geq 2$. Since $\mathcal{P} \neq \mathcal{Q}$, we have that either $P_0 \neq \{v_0\}$ or at least one of the parts $P \in \mathcal{P} \backslash \{P_0\}$ has size $|P| \geq 2$. Thus, $f(\mathcal{P}) = \sum_{P \in \mathcal{P}} f(P) \geq (1+a)(|\mathcal{P}|-1)+1$. Moreover, we have $f(\mathcal{Q}) = (1+a)(n-1)$ and hence

$$\frac{f(\mathcal{P})}{|\mathcal{P}| - 1} \geq \frac{(1+a)(|\mathcal{P}|-1)+1}{|\mathcal{P}|-1} = 1 + a + \frac{1}{|\mathcal{P}|-1} > 1 + a = \frac{(1+a)(n-1)}{n-1} = \frac{f(\mathcal{Q})}{n-1}.$$

This proves inequality (7). ◁

▷ Claim 12. The approximation factor of Algorithm 1 on input (f, k) is $\Omega(n/k)$.

Proof. We note that $f(\{v_0\}) = 0$ and $f(\{v_i\}) = 1 + a$ for all $i \in [n-1]$. By Claim 11, on input (f, k), Algorithm 1 returns a partition $\mathcal{P}$ consisting of the $k-1$ singleton parts that minimizes f among all singleton sets and the complement of the union of these $k-1$ singleton parts. Therefore, the returned partition $\mathcal{P}$ contains $\{v_0\}$ as a part and thus

$$f(\mathcal{P}) = \sum_{P \in \mathcal{P}: P \neq \{v_0\}} f(P) \geq f(V - \{v_0\}) = a(n-1).$$

The second inequality follows from submodularity of the function f. Consider the k-partition $\{\{v_1\}, \{v_2\}, \ldots, \{v_{k-1}\}, V - \{v_1, \ldots, v_{k-1}\}\}$, which has objective $(1+a)(k-1)+1$. This implies that the optimum k-partition $\mathcal{P}^*$ satisfies $f(\mathcal{P}^*) \leq (1+a)(k-1)+1$. Thus, the approximation factor of the solution returned by Algorithm 1 is

$$\frac{f(\mathcal{P})}{f(\mathcal{P}^*)} \geq \frac{a(n-1)}{(1+a)(k-1)+1} \to \frac{n-1}{k-1} \text{ as } a \to \infty. \qquad \triangleleft$$

5 Conclusion

The principal partition sequence of submodular functions was shown to exist by Narayanan [16]. The principal partition sequence of submodular functions is known in the literature as *principal lattice of partitions* of submodular functions since there exists a lattice structure associated with the sequence of partitions [2, 8, 9, 11, 14, 17, 20]. We chose to call it as principal partition sequence in this work since the sequence suffices for our purpose. Narayanan, Roy, and Patkar [18] used the principal partition sequence to design an algorithm for submodular k-partition. They analyzed the approximation factor of their algorithm for certain subfamilies of submodular functions that arise from hypergraphs. In this work, we investigated the approximation factor of their algorithm for three broad subfamilies of submodular functions – namely monotone, symmetric, and posimodular submodular functions. Our results show that the principal partition sequence based algorithm achieves the best possible asymptotic approximation factor for all these three subfamilies. A novelty of our contributions is the improvement in the approximability of monotone submodular k-partition from 2 to 4/3, thus matching the inapproximability threshold. It would be interesting to pin down the approximability of special cases of monotone submodular k-partition – e.g., matroid k-partition and coverage k-partition which are interesting by themselves since they capture several interesting partitioning problems.

References

1 M. Baïou and F. Barahona. Packing Hypertrees and the k-cut problem in Hypergraphs. In *International Conference on Learning and Intelligent Optimization*, LION, pages 521–534, 2023.

2 F. Barahona. On the k-cut problem. *Operations Research Letters*, 26(3):99–105, 2000.

3 D. Chakrabarty and Z. Huang. Testing coverage functions. In *Automata, Languages, and Programming*, ICALP, pages 170–181, 2012.

4 K. Chandrasekaran and C. Chekuri. Hypergraph k-cut for fixed k in deterministic polynomial time. *Mathematics of Operations Research*, 2022. Prelim. version in FOCS 2020: pages 810–821.

5 K. Chandrasekaran and W. Wang. Approximating submodular k-partition via principal partition sequence. arXiv, 2023. arXiv:2305.01069.

6 C. Chekuri and S. Li. On the hardness of approximating the k-way hypergraph cut problem. *Theory Comput.*, 16:1–8, 2020.

7 C. Chekuri, K. Quanrud, and C. Xu. LP Relaxation and Tree Packing for Minimum k-Cut. *SIAM Journal on Discrete Mathematics*, 34(2):1334–1353, 2020.

8 W. Cunningham. Optimal attack and reinforcement of a network. *J. ACM*, 32(3):549–561, July 1985.

9 M. P. Desai, H. Narayanan, and S. B. Patkar. The realization of finite state machines by decomposition and the principal lattice of partitions of a submodular function. *Discrete Appl. Math.*, 131:299–310, 2003.

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 V. Kolmogorov. A Faster Algorithm for Computing the Principal Sequence of Partitions of a Graph. *Algorithmica*, pages 394–412, 2010.

12 P. Manurangsi. Inapproximability of Maximum Biclique Problems, Minimum k-Cut and Densest At-Least-k-Subgraph from the Small Set Expansion Hypothesis. *Algorithms*, 11(1):10, 2018.

13 H. Nagamochi and Y. Kamidoi. Minimum cost subpartitions in graphs. *Information Processing Letters*, 102(2):79–84, 2007.

14 K. Nagano, Y. Kawahara, and S. Iwata. Minimum average cost clustering. In *Advances in Neural Information Processing Systems*, volume 23 of *NIPS*, 2010.

15 J. Naor and Y. Rabani. Tree packing and approximating k-cuts. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, pages 26–27, 2001.

16 H. Narayanan. The principal lattice of partitions of a submodular function. *Linear Algebra and its Applications*, 144:179–216, 1991.

17 H. Narayanan. Submodular functions and electrical networks (2nd edition), 1997. URL: `https://www.ee.iitb.ac.in/~hn/book/SubmodularFunction_2ed.pdf`.

18 H. Narayanan, S. Roy, and S. Patkar. Approximation algorithms for min-k-overlap problems using the principal lattice of partitions approach. *Journal of Algorithms*, 21(2):306–330, 1996.

19 K. Okumoto, T. Fukunaga, and H. Nagamochi. Divide-and-conquer algorithms for partitioning hypergraphs and submodular systems. *Algorithmica*, 62(3):787–806, 2012.

20 S. B. Patkar and H. Narayanan. Improving graph partitions using submodular functions. *Discrete Appl. Math.*, 131:535–553, 2003.

21 K. Quanrud. Fast and Deterministic Approximations for k-Cut. In *Proceedings of Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, APPROX, pages 23:1–23:20, 2019.

22 R. Ravi and A. Sinha. Approximating k-cuts using network strength as a lagrangean relaxation. *European Journal of Operational Research*, 186(1):77–90, 2008.

23 R. Santiago. New approximations and hardness results for submodular partitioning problems. In *Proceedings of International Workshop on Combinatorial Algorithms*, IWOCA, pages 516–530, 2021.

24 H. Saran and V. Vazirani. Finding k Cuts within Twice the Optimal. *SIAM Journal on Computing*, 24(1):101–108, 1995.

25 V. Vazirani. *Approximation algorithms.* Springer, Berlin, Germany, 2003.

26 L. Zhao, H. Nagamochi, and T. Ibaraki. Greedy splitting algorithms for approximating multiway partition problems. *Mathematical Programming*, 102(1):167–183, 2005.

Experimental Design for Any p-Norm

Lap Chi Lau
David R. Cheriton School of Computer Science, University of Waterloo, Canada

Robert Wang
David R. Cheriton School of Computer Science, University of Waterloo, Canada

Hong Zhou[1]
School of Mathematics and Statistics, Fuzhou University, China

Abstract

We consider a general p-norm objective for experimental design problems that captures some well-studied objectives (D/A/E-design) as special cases. We prove that a randomized local search approach provides a unified algorithm to solve this problem for all nonnegative integer p. This provides the first approximation algorithm for the general p-norm objective, and a nice interpolation of the best known bounds of the special cases.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Theory of computation → Rounding techniques

Keywords and phrases Approximation Algorithm, Optimal Experimental Design, Randomized Local Search

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.4

Category APPROX

Related Version *Full Version*: https://arxiv.org/abs/2305.01942

Funding *Lap Chi Lau*: Supported in part by NSERC Discovery Grant.
Robert Wang: Supported in part by NSERC Discovery Grant and Canada Graduate Scholarship.
Hong Zhou: Supported in part by Fuzhou University startup funding (XRC-22056) and Natural Science Foundation of Fujian Province of China (2023J01414).

Acknowledgements We thank Mohit Singh for bringing the Φ_p objective function to our attention. We also thank anonymous reviewers of an earlier version of this manuscript for helpful suggestions.

1 Introduction

In experimental design problems, we are given vectors $v_1, \ldots, v_n \in \mathbb{R}^d$ and a budget $k \geq d$, and the goal is to choose a subset S of k vectors so that $\sum_{i \in S} v_i v_i^\top$ optimizes some objective function that measures the "diversity" of the input data. The most popular and well-studied objective functions are:

- D-design: Maximizing $(\det(\sum_{i \in S} v_i v_i^\top))^{\frac{1}{d}}$.
- A-design: Minimizing $\frac{1}{d}\operatorname{tr}((\sum_{i \in S} v_i v_i)^{-1})$.
- E-design: Maximizing $\lambda_{\min}(\sum_{i \in S} v_i v_i^\top)$.

Experimental design problems have a long history and wide applications, from statistics to machine learning to numerical linear algebra to graph algorithms. For more information on these applications, we refer the reader to [13, 11, 1, 7, 6] and the references therein.

Although the objectives of D/A/E-design look quite different, we observe that there is a natural generalization using eigenvalues that captures all three objectives as special cases.

[1] Corresponding author

▶ **Definition 1** (p-Norm of Inverse Eigenvalues). *Given a d-dimensional real-symmetric matrix A with eigenvalues $\lambda_1, \ldots, \lambda_d > 0$ and a natural number $0 \leq p \leq \infty$, we define*

$$\Phi_p(A) := \left(\frac{1}{d} \operatorname{tr}\left(A^{-p}\right)\right)^{\frac{1}{p}} = \left(\frac{1}{d} \sum_{i=1}^{d} \lambda_i^{-p}\right)^{\frac{1}{p}}, \tag{1}$$

with $\Phi_0(A) := \lim_{p \to 0^+} \Phi_p(A)$ and $\Phi_\infty(A) := \lim_{p \to +\infty} \Phi_p(A)$.

Given $p \geq 0$, we refer to the experimental design problem with the objective function Φ_p as Φ_p-design. To see that Φ_p-design is a generalization of D/A/E-design, let $A = \sum_{i \in S} v_i v_i^T$ and note that:

- For $p = \infty$, $\Phi_\infty(A) = \lambda_{\max}(A^{-1}) = 1/\lambda_{\min}(A)$, which is the inverse of the E-design objective;
- For $p = 1$, $\Phi_1(A) = \frac{1}{d}\operatorname{tr}(A^{-1})$ is exactly the A-design objective;
- For $p = 0$, $\Phi_0(A)$ is the inverse of the D-design objective, as

$$\Phi_0(A) = \lim_{p \to 0^+} \left(\frac{1}{d} \sum_{i=1}^{d} \lambda_i^{-p}\right)^{1/p} = \left(\prod_{i=1}^{d} \lambda_i^{-1}\right)^{1/d} = \det(A)^{-1/d},$$

  where the second equality is a well-known fact (see, e.g., Exercise 28 in Chapter 5 of [16]).

It is known that $\Phi_p(A)$ is convex in A for any given $0 \leq p \leq \infty$, and so the following is a natural convex programming relaxation for Φ_p-design:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & \Phi_p\left(\sum_{i=1}^{n} x(i) \cdot v_i v_i^\top\right) \\ \text{subject to} \quad & \sum_{i=1}^{n} x(i) \leq k, \\ & 0 \leq x(i) \leq 1, \qquad \text{for } 1 \leq i \leq n. \end{aligned} \tag{2}$$

To the best of our knowledge, there are no known approximation algorithms for the general Φ_p-design problem, other than the special cases $p = 0, 1, \infty$ which we summarize as follows (the notation $x \gtrsim y$ denotes that $x \geq cy$ for some large enough constant c):

- There is a $(1+\varepsilon)$-approximation algorithm for Φ_0-design (D-design) when $k \gtrsim d/\varepsilon$ [13, 9, 11, 6].
- There is a $(1+\varepsilon)$-approximation algorithm for Φ_1-design (A-design) when $k \gtrsim d/\varepsilon$ [9, 11, 6].
- There is a $(1+\varepsilon)$-approximation algorithm for Φ_∞-design (E-design) when $k \gtrsim d/\varepsilon^2$ [1, 7].

These results are tight in the sense that they match the known integrality gap lower bound of the convex programming relaxation (2) (see [11] for integrality gap examples).

Note that there is a d/ε vs d/ε^2 gap between the relaxations for D/A-design ($p = 0, 1$) and for E-design ($p = \infty$). The main question that we study in this paper is: How does the integrality gap of the convex programming relaxation (2) change with varying value of p? In particular, where does the transition from d/ε to d/ε^2 happen?

1.1 Main Result

Our main result is that, when $k \gtrsim \min\{dp/\varepsilon, d/\varepsilon^2\}$, there is a $(1+\varepsilon)$-approximation algorithm for Φ_p-design.

▶ **Theorem 2.** *Given an integer $p \geq 1$, let $\mathsf{x} \in [0,1]^n$ be an optimal fractional solution to (2). For any $\varepsilon \in (0,1)$, let $\gamma = \max\{\varepsilon, 1/p\}$, if $k \gtrsim d/(\gamma\varepsilon)$, then there is a randomized polynomial time algorithm that returns an integral solution $Z = \sum_{i=1}^n z(i) \cdot v_i v_i^\top$ with $z(i) \in \{0,1\}$ for $1 \leq i \leq n$ such that*

$$\Phi_p\left(\sum_{i=1}^n z(i) \cdot v_i v_i^\top\right) \leq (1+\varepsilon) \cdot \Phi_p\left(\sum_{i=1}^n x(i) \cdot v_i v_i^\top\right) \quad \text{and} \quad \sum_{i=1}^n z(i) \leq k,$$

with probability at least $1 - O\left(\frac{(d^2+d/\gamma)^2}{\varepsilon^2 p^2} \cdot e^{-\Omega(\gamma\sqrt{d})}\right) - e^{-\Omega(\varepsilon d/\gamma)}$.

▶ Remark. Theorem 2 can be generalized to all real $p \geq 1$, see Remark 11 for more details. The $p = 0$ case can also be covered by Theorem 2, but with a different analysis from [6].

This is the first approximation algorithm for Φ_p-design for general p. Theorem 2 shows that Φ_p-design for constant p admits as good an approximation algorithm as for D/A-design, and there is a unifying algorithm to achieve this guarantee.

Note that, when $p \to +\infty$, Φ_p becomes the E-design objective and $\gamma = \max\{\varepsilon, 1/p\} = \varepsilon$. Thus, Theorem 2 provides a nice interpolation between the d/ε bound for D/A-design and the d/ε^2 bound for E-design.

We further remark that our results can be generalized to the weighted setting to handle multiple budget/knapsack constraints as in [6], but we omit the details to keep the presentation cleaner as they are the same as in [6].

The proof of Theorem 2 is built on the randomized local search approach in [7] and [6], but several new technical ideas are needed to handle higher moments that are introduced by the higher p-norm. In Section 2, we will review the background and previous work, present the algorithm and the overall structure of the analysis, and explain the new ideas in this work. Then, in Section 3, we will present the details of the Φ_p experimental design.

1.2 Discussions and Future Directions

Our proof of Theorem 2 does not address the range $p \in [0,1)$. However, with a similar analysis as in [6], exactly the same algorithm in this paper can achieve a $(1+\varepsilon)$-approximation for $p = 0$ (i.e., D-design) when $k \gtrsim d/\varepsilon$. It would be interesting to see whether the randomized local search approach can be extended to Φ_p-design with $p \in (0,1)$. An obstacle to the current analysis is that the Lieb-Thirring inequality used in the proof of Lemma 10 goes in the wrong direction for $p \in (0,1)$.

If we plot the minimum required k for achieving the $(1+\varepsilon)$-approximation for the Φ_p-design as a function of p, Theorem 2 has ruled out a sharp transition of the curve at $p = 1$, from d/ε to d/ε^2. However, we do not know whether Theorem 2 is tight or not, in particular, in the range of $p \geq 1/\varepsilon$ (see Figure 1 for a demonstration). It would be interesting to fully characterize the whole curve.

1.3 Related Work

In this paper, we focus on generalizing the D/A/E-design with the p-norm objective in (1). However, there are other different ways to extend D/A/E-design. For example, the Bayesian framework of experimental design [2] extends the problem by adding a fixed matrix B (which encodes some prior information, e.g., a multiple of identity) to the covariance matrix before applying the objective function. When $B = 0$, we recover classical experimental design problems. Tantipongpipat [14] provided an approximation algorithm for Bayesian A-design

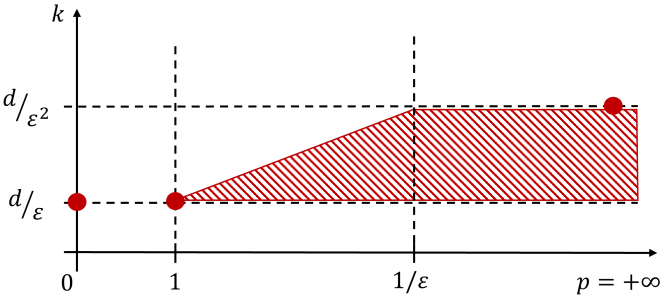


Figure 1 Known results on the minimum k required to achieve $(1+\varepsilon)$-approximation for the Φ_p-design (ignoring constant factors). The exact curve should lie within the shadowed area.

problem when B is a multiple of identity. Dereziński et. al. [3] studied approximation algorithms for Bayesian A/D-design (together with C/V-design, i.e., two other objectives) with a general PSD matrix B. Another example is using elementary symmetric polynomials to generalize A/D-design [10]. Nikolov et. al. [11] extended their approximation algorithm for A-design to tackle this family of generalized ratio objectives.

All these are interesting generalizations of D/A/E-design. It would be interesting to see whether the randomized local search approach in [7, 6] and this paper can be applied to these settings to obtain better results.

2 The Framework

In this section, we first review the randomized local search approach in Section 2.1. Then, we present the full algorithm in Section 2.2 and state the main technical theorem. Then, in Section 2.3, we provide the overall proof plan and the precise statements for analyzing the randomized exchange algorithm, and then put together the statements to prove the main technical theorem. Finally, we discuss the main difficulty and the new ideas needed to analyze the Φ_p objective in Section 3.

2.1 Randomized Local Search Approach

The proof of Theorem 2 is built on the randomized local search approach in [6], which is based on the regret minimization framework developed in [1] for experimental design problems and the randomized spectral rounding techniques in [7]. We review this approach in this subsection.

In [6], the first step is to solve (2) for D/A/E-design to obtain a solution $x \in \mathbb{R}^n$, and then to normalize the vectors v_i's so that $\sum_{i=1}^n x(i) \cdot v_i v_i^\top = I$ for using the regret minimization framework in [1]. Then, the rounding algorithm starts from a random initial solution set S_0 that is independently sampled according to x. Using the density matrix A_t maintained by the regret minimization framework at each step t, the algorithm randomly chooses a pair of vectors v_{i_t} and v_{j_t} with the following probability distributions:

$$\Pr(i_t = i) \propto \big(1 - x(i)\big) \cdot \Big(1 - \alpha\langle v_i v_i^\top, A_t^{1/2}\rangle\Big) \quad \text{and} \quad \Pr(j_t = j) \propto x(j) \cdot \Big(1 + \alpha\langle v_j v_j^\top, A_t^{1/2}\rangle\Big), \tag{3}$$

and set $S_{t+1} \leftarrow S_t - v_{i_t} + v_{j_t}$. Using the above randomized local search strategy, it can be shown that the size of the solution is expected to stay around k, while the potential function from the regret minimization framework [1] related to the minimum eigenvalue is expected to improve as long as the minimum eigenvalue is less than $1 - \varepsilon$. Freedman's martingale inequality and a new concentration inequality for non-martingales are used in [7] to prove that all these quantities are close to their expected values with high probability. This randomized local search algorithm provides a $(1 \pm \varepsilon)$-approximate solution for D/A/E-design when $k \gtrsim d/\varepsilon^2$.

The main contribution in [6] is to prove that the randomized local search algorithm can be adapted to achieve $(1 \pm \varepsilon)$-approximation for D/A-design when $k \gtrsim d/\varepsilon$, thus providing a unifying approach to achieve the optimal integrality gap for D/A/E-design. Essentially, the algorithm is the same as the one for E-design but only require that the solution to have minimum eigenvalue $3/4$ rather than $1 - \varepsilon$. For the analysis, the randomized local search algorithm is conceptually divided into two phases. In the first phase, the algorithm will find a solution with minimum eigenvalue at least $3/4$ with high probability when $k \gtrsim d/\varepsilon$. In the second phase, the minimum eigenvalue will maintain to be at least $1/4$ with high probability, and the objective value for D/A-design will improve to $(1 \pm \varepsilon)$ times the optimal objective value in polynomial time with high probability. The analysis of the first phase follows directly from earlier work on spectral rounding in [7]. The analysis of the second phase includes two main parts: (1) to show that, in expectation, the probability distributions in (3) for E-design are also good for making progress towards D/A-design objectives; (2) to show that the progress in the objective value is concentrated around the expectation with a martingale concentration argument. The condition that the minimum eigenvalue is at least $1/4$ is very important in both parts in the second phase, and the optimality conditions for the convex program (2) is crucially used in the second part in the second phase.

In this paper, we will extend the algorithm and the analysis described above and show that the randomized local search algorithm provides a unifying approach for Φ_p-design for all p.

2.2 The Algorithm

We present the full algorithm for Φ_p-design in this subsection.

Algorithm 1 Randomized Exchange Algorithm.

```
Input: $m$ vectors $u_1, \dots, u_m \in \mathbb{R}^d$, a budget $k \geq d$, an accuracy parameter $\varepsilon \in (0,1)$.

1. Solve the convex programming relaxation (2) and obtain an optimal solution $x \in [0,1]^m$ with at most $d^2 + 1$ fractional entries, i.e. $|\{i \in [m] \mid 0 < x(i) < 1\}| \leq d^2 + 1$. Let $X = \sum_{i=1}^m x(i) \cdot u_i u_i^\top$.
2. Preprocessing: Let $v_i \leftarrow X^{-1/2} u_i$ for all $i \in [m]$, so that $\sum_{i=1}^m x(i) \cdot v_i v_i^\top = I_d$.
3. Initialization: $t \leftarrow 1$, $S_0 \leftarrow \emptyset$, $\gamma = \max\{\frac{\varepsilon}{6}, \frac{1}{6p}\}$, $\kappa = \max\{\frac{\varepsilon}{2}, \frac{1}{2p}\}$, $M \leftarrow \frac{d}{\gamma} + d^2 + 1$, and $\alpha \leftarrow \frac{\sqrt{d}}{\gamma}$.
4. Add $i$ into $S_0$ independently with probability $x(i)$ for each $i \in [m]$. Let $Y_1 \leftarrow \sum_{i \in S_0} u_i u_i^\top$ and $Z_1 \leftarrow \sum_{i \in S_0} v_i v_i^\top$.
5. While the termination condition $(\mathrm{tr}(Y_t^{-p}))^{\frac{1}{p}} \leq (1+\varepsilon)(\mathrm{tr}(X^{-p}))^{\frac{1}{p}}$ is not satisfied and $t \leq \frac{2M}{\gamma} + \frac{2M}{\varepsilon p}$, do the following:
   a. $S_t \leftarrow \mathrm{Exchange}(S_{t-1})$.
   b. Set $Y_{t+1} \leftarrow \sum_{i \in S_t} u_i u_i^\top$, $Z_{t+1} \leftarrow \sum_{i \in S_t} v_i v_i^\top$ and $t \leftarrow t+1$.
6. Return $S_{t-1}$ as the solution.
```

The main algorithm is almost the same as the one in [6], but with two additional parameters γ, κ that will depend on the value of p. For A-design (when $p = 1$), the algorithm in [6] is just a special case of the randomized exchange algorithm with parameter $\gamma = \frac{1}{8}$ and $\kappa = 1$. In this paper, the parameter γ is used to adjust the learning rate α of the regret minimization framework and the parameter κ will be used in the exchange subroutine.

■ **Algorithm 2** Exchange Subroutine.

```
Input: the current solution set $S_{t-1}$.

1. Compute the action matrix $A_t := (\alpha Z_t - c_t I)^{-2}$, where $Z_t = \sum_{i \in S_{t-1}} v_i v_i^\top$ and $c_t$ is the
   unique scalar such that $A_t \succ 0$ and $\operatorname{tr}(A_t) = 1$.
2. Define $S'_t := \{ i \in S_{t-1} \mid \alpha \langle v_i v_i^\top, A_t^{1/2} \rangle \le \frac{1}{2}$ and $\langle v_i v_i^\top, Z_t^{-1} \rangle \le \kappa \}$.
3. Sample $j_t \in [m] \backslash S_{t-1}$ from the following probability distribution

     $\Pr(j_t = j) = \frac{x(j)}{M} \cdot \big(1 + \alpha \langle v_j v_j^\top, A_t^{1/2} \rangle\big)$, for $j \in [m] \backslash S_{t-1}$ and
     $\Pr(j_t = \emptyset) = 1 - \sum_{j \in [m] \backslash S_{t-1}} \frac{x(j)}{M} \cdot \big(1 + \alpha \langle v_j v_j^\top, A_t^{1/2} \rangle\big)$.

4. Sample $i_t \in S'_{t-1}$ from the following probability distribution

     $\Pr(i_t = i) = \frac{1 - x(i)}{M} \cdot \big(1 - \alpha \langle v_i v_i^\top, A_t^{1/2} \rangle\big)$, for $i \in S'_{t-1}$ and
     $\Pr(i_t = \emptyset) = 1 - \sum_{i \in S'_{t-1}} \frac{1 - x(i)}{M} \cdot \big(1 - \alpha \langle v_i v_i^\top, A_t^{1/2} \rangle\big)$.

5. Return $S_t := S_{t-1} \cup \{j_t\} \backslash \{i_t\}$.
```

The exchange subroutine is also almost the same as in [6]. The key difference is to use the new parameter κ to further restrict the set of vectors that are allowed to remove from the current solution.

▶ Remark 3. Using the new analysis in this paper, the same algorithm in [6] (without changing the parameters γ and κ) can achieve $(1+\varepsilon)$-approximation for Φ_p-design when $k \gtrsim 2^{O(p)} d/\varepsilon$, with an exponential dependence on p in the budget requirement, much worse than $k \gtrsim \min\{dp/\varepsilon, d/\varepsilon^2\}$ in Theorem 2.

Main Technical Theorem

To prove Theorem 2, we will prove that the randomized exchange algorithm is a bicriteria approximation algorithm, such that it returns a solution that is $(1+\varepsilon)$-approximate in the Φ_p objective and the size of the solution is not much larger than k.

▶ **Theorem 4.** *Given $\varepsilon \in (0,1)$, if $k \gtrsim \frac{d}{\gamma\varepsilon}$, then the randomized exchange algorithm returns a solution set S within $\frac{2M}{\gamma} + \frac{2M}{\varepsilon p}$ iterations such that*

$$\left(\operatorname{tr} \left(\Big(\sum_{i \in S} u_i u_i^\top \Big)^{-p} \right) \right)^{\frac{1}{p}} \le (1+\varepsilon) \cdot \left(\operatorname{tr} \left(X^{-p} \right) \right)^{1/p}$$

with probability at least $1 - O\big(\frac{M^2}{\varepsilon^2 p^2} \cdot e^{-\Omega(\gamma\sqrt{d})}\big)$, where X is an optimal fractional solution to (2). Moreover, the solution set S satisfies $|S| \le (1+\varepsilon)k + O\big(\frac{d}{\gamma} + \frac{d}{\kappa}\big)$ with probability at least $1 - e^{-\Omega(\varepsilon d / \min\{\gamma,\kappa\})}$.

With Theorem 4, we can prove Theorem 2 by first noticing that both γ and κ are chosen in the order of $\Theta(\max\{\varepsilon, 1/p\})$ and then turning the bicriteria approximation result into a true approximation with a scaling argument as in [6]. We omit the standard proof and refer the readers to [6] for more details.

2.3 The Proof Plan

In this subsection, we provide the overall plan and the precise statements for analyzing the randomized exchange algorithm, and then put together the statements to obtain the main technical theorem.

2.3.1 Well-Defined Algorithm

First, we prove that the randomized exchange algorithm is well-defined. In particular, we need to show that a fractional optimal solution to the convex relaxation (2) with at most $O(d^2)$ fractional entries can be found in polynomial time, and also the probability distributions in the exchange subroutine are well-defined for $M = d^2 + \frac{d}{\gamma} + 1$. These can be established using the same arguments (with a different value for γ) as in Lemma 4.2 and Claim 4.4 in [6]. Note that the modified exchange subroutine does not affect these arguments.

The following simple observation (Observation 4.3 in [6]) will be useful in the analysis of the algorithm.

▶ **Observation 5.** *For any $t \geq 0$, it holds that $i \in S_t$ for all i with $x(i) = 1$ and $j \in [n] \backslash S_t$ for all j with $x(j) = 0$. This further implies that $\Pr(i_t = i) = 0$ for all i with $x(i) \in \{0, 1\}$ and $\Pr(j_t = j) = 0$ for all j with $x(j) \in \{0, 1\}$.*

2.3.2 Solution Size Bound

Then, we show that the algorithm returns a solution set S of size not much larger than k with high probability.

▶ **Theorem 6** (Variant of Theorem 3.12 of [7]). *Let $\alpha = \sqrt{d}/\gamma$ and κ be the parameters used in the randomized exchange algorithm. Suppose that the solution S_t of the randomized exchange algorithm satisfies $\lambda_{\min}\left(\sum_{i \in S_t} v_i v_i^\top\right) < 1$ for all $t \in [\tau]$. Then, for any given $\delta \in [0, 1]$,*

$$\Pr\left[|S_\tau| \leq (1+\delta) \cdot \sum_{i=1}^{n} x(i) + \left(\frac{12d}{\gamma} + \frac{2d}{\kappa}\right)\right] \geq 1 - \exp\left(-\Omega\left(\frac{\delta d}{\min\{\gamma, \kappa\}}\right)\right).$$

When $\kappa = 1$, the above theorem follows directly from the one-sided spectral rounding result in [7]. With smaller κ, we are restricting the set of vectors that can be swapped out from the current solution. This would increase the chance of not removing a vector, and thus increasing the size of the solution. Fortunately, we can show that the increase of the solution size can be bounded by an additive d/κ term. The proof idea is similar to the one in [7], and the main difference is a modified bound on the expected change of size. Please refer to the appendices of the full arxiv version [5] for more details.

2.3.3 Approximation Guarantee

The most technical part of the proof is to establish the approximation guarantee. We follow the analysis in [6] to conceptually divide the execution of the algorithm into two phases as described in Section 2.1.

In the first phase, we show that the minimum eigenvalue will reach $1 - 2\gamma$ in $O(M/\gamma)$ iterations with high probability, which follows from the spectral rounding result in [7].

▶ **Proposition 7** (Counterpart of Proposition 4.5 in [6]). *The probability that the randomized exchange algorithm has terminated successfully within $2M/\gamma$ iterations or there exists $\tau_1 \leq 2M/\gamma$ with $\lambda_{\min}(Z_{\tau_1}) \geq 1 - 2\gamma$ is at least $1 - \exp(-\Omega(\sqrt{d}))$.*

In the second phase, the minimum eigenvalue will be at least $1 - 5\gamma$ during the next $\Theta\left(\frac{M}{\varepsilon p}\right)$ iterations with good probability.

▶ **Proposition 8** (Counterpart of Proposition 4.6 in [6]). *Suppose $\lambda_{\min}(Z_{\tau_1}) \geq 1 - 2\gamma$ for some τ_1. In the randomized exchange algorithm, the probability that $\lambda_{\min}(Z_t) \geq 1 - 5\gamma$ for all $\tau_1 \leq t \leq \tau_1 + \frac{2M}{\varepsilon p}$ is at least $1 - \frac{4M^2}{\varepsilon^2 p^2} \cdot e^{-\Omega(\gamma\sqrt{d})}$.*

Both of the proofs of Proposition 7 and Proposition 8 follow same arguments in [6] (with a new parameter γ). We remark that the modified exchange subroutine with a restricted S'_t does not affect these arguments, as removing less vectors only helps to improve the minimum eigenvalue of the solution. We omit the proofs and refer the readers to [6].

The main technical contribution in this paper is to prove that the Φ_p objective will improve to at most $(1 + \varepsilon)$ times the optimal value during the second phase when the minimum eigenvalue is at least $1 - 5\gamma$.

▶ **Theorem 9.** *Given $\varepsilon \in (0, 1)$, if $p \leq 1/\varepsilon$ and $k \gtrsim \frac{pd}{\varepsilon}$ for some $\varepsilon \in (0, 1)$, then the probability that the following three events happen simultaneously during the execution of the randomized exchange algorithm is at most $\exp(-\Omega(\gamma\sqrt{d}))$.*

1. *$\lambda_{\min}(Z_{\tau_1}) \geq 1 - 2\gamma$ for some τ_1;*
2. *$\lambda_{\min}(Z_t) \geq 1 - 5\gamma$ for all $\tau_1 \leq t \leq \tau_1 + \frac{2M}{\varepsilon p}$;*
3. *the randomized exchange algorithm has not terminated by time $\tau_1 + \frac{2M}{\varepsilon p}$.*

2.3.4 Proof of Theorem 4

We put together the statements in this subsection to obtain Theorem 4.

Proof of Theorem 4. We start with analyzing the approximation guarantee in the theorem.

Firstly, consider the easier case $p \geq 1/\varepsilon$, which implies that $\gamma = \varepsilon/6$. By Proposition 7, there exists $\tau_1 \leq 2M/\gamma$ such that $\lambda_{\min}(Z_{\tau_1}) \geq (1 - \varepsilon/3)$ with probability $1 - \exp(-\Omega(\sqrt{d}))$. We note that $\lambda_{\min}(Z_{\tau_1}) \geq (1 - \varepsilon/3)$ is equivalent to $Y_{\tau_1} \succcurlyeq (1 - \varepsilon/3)X$, which is sufficient to establish

$$(\operatorname{tr}(Y_{\tau_1}^{-p}))^{1/p} \leq (1 + \varepsilon)(\operatorname{tr}(X^{-p}))^{1/p},$$

i.e., the approximation guarantee in the theorem. We remark that we do not need the assumption $k \gtrsim d/(\gamma\varepsilon)$ in the proof of this case.

Then, we consider the case of $p \leq 1/\varepsilon$ and define the bad events for the randomized exchange algorithm:

- B_1: the algorithm has not terminated successfully within $2M/\gamma$ iterations and $\tau_1 > 2M/\gamma$ where τ_1 is the first time such that $\lambda_{\min}(Z_{\tau_1}) \geq 1 - 2\gamma$.
- B_2: there exists some $\tau_1 \leq t \leq \tau_1 + \frac{2M}{\varepsilon p}$ such that $\lambda_{\min}(Z_t) < 1 - 5\gamma$.
- B_3: the termination condition $(\operatorname{tr}(Y_t^{-p}))^{\frac{1}{p}} \leq (1 + \varepsilon)(\operatorname{tr}(X^{-p}))^{\frac{1}{p}}$ is not satisfied for all $\tau_1 \leq t \leq \tau_1 + \frac{2M}{\varepsilon p}$.

If none of the bad events happens, then either the algorithm has terminated successfully within $2M/\gamma$ iterations or the termination condition will be satisfied at some time $t \leq \tau_1 + \frac{2M}{\varepsilon p} \leq \frac{2M}{\gamma} + \frac{2M}{\varepsilon p}$. So, the probability that the randomized exchange algorithm has not satisfied the termination condition within $\frac{2M}{\gamma} + \frac{2M}{\varepsilon p}$ iterations is upper bounded by

$$\begin{aligned}\Pr[B_1 \cup B_2 \cup B_3] &= \Pr[B_1] + \Pr[B_2 \cap \neg B_1] + \Pr[B_3 \cap \neg B_2 \cap \neg B_1] \\ &\leq O\left(e^{-\Omega(\sqrt{d})}\right) + O\left(\frac{M^2}{\varepsilon^2 p^2} \cdot e^{-\Omega(\gamma\sqrt{d})}\right) + O\left(e^{-\Omega(\gamma\sqrt{d})}\right) \\ &\leq O\left(\frac{M^2}{\varepsilon^2 p^2} \cdot e^{-\Omega(\gamma\sqrt{d})}\right),\end{aligned}$$

where $\Pr[B_1]$ is bounded in Proposition 7, $\Pr[B_2 \cap \neg B_1]$ is bounded in Proposition 8, and $\Pr[B_3 \cap \neg B_2 \cap \neg B_1]$ is bounded in Theorem 9 (note that we need the assumption $p \leq 1/\varepsilon$ and $k \gtrsim pd/\varepsilon$ here). The termination condition implies the approximation guarantee directly.

Finally, we consider the size of the returned solution. Note that if $\lambda_{\min}(Z_t) \geq 1$ then $Y_t \succcurlyeq X$, which further implies that the termination condition is met at time t. Hence, we can assume $\lambda_{\min}(Z_t) < 1$ before the algorithm terminates. Therefore, we can apply Theorem 6 to conclude that the returned solution S satisfies $|S| \leq (1+\varepsilon)k + O\big(\frac{d}{\gamma} + \frac{d}{\kappa}\big)$ with probability at least $1 - \exp(-\Omega(\frac{\varepsilon d}{\min\{\gamma,\kappa\}}))$. ◀

2.4 New Ideas

The key in proving Theorem 9 is to bound the change of the objective value after an exchange. For A-design ($p = 1$), there is a simple inequality bounding the change of the objective as

$$\operatorname{tr}\left(\left(Y - vv^\top + ww^\top\right)^{-1}\right) \leq \operatorname{tr}\left(Y^{-1}\right) + \underbrace{\frac{v^\top Y^{-2} v}{1 - \langle vv^\top, Y^{-1}\rangle} - \frac{w^\top Y^{-2} w}{1 + \langle ww^\top, Y^{-1}\rangle}}_{\text{progress}}.$$

For general Φ_p-design, the change of the Φ_p function under rank-two updates is considerably more complicated. Using Sherman-Morrison formula and Lieb-Thirring inequality, we can bound the change of the Φ_p objective (in fact, the p-th power of the Φ_p objective) as follows:

$$\begin{aligned}&\operatorname{tr}\left((Y + ww^\top - vv^\top)^{-p}\right) \\ &\quad\leq \operatorname{tr}(Y^{-p}) + \sum_{i=1}^{p} \binom{p}{i} \left((-1)^i \frac{(w^\top Y^{-1} w)^{i-1} \cdot w^\top Y^{-p-1} w}{(1 + w^\top Y^{-1} w)^i} + \frac{(v^\top Y^{-1} v)^{i-1} \cdot v^\top Y^{-p-1} v}{(1 - v^\top Y^{-1} v)^i}\right).\end{aligned}$$

There are many higher order terms introduced by the p-norm, and dealing with these is the main technical difficulty in this paper.

As discussed in Remark 3, if we use the same algorithm in [6], with some careful manipulations including applying Hölder's inequality appropriately, we can achieve $(1+\varepsilon)$-approximation but with the much worst requirement that $k \gtrsim 2^{O(p)} d/\varepsilon$. The reason is that removing some "influential" vectors (even with relatively small probability) from the current solution will blow up the expectation of the change of the objective function due to the higher order terms in the above inequality.

To overcome this issue, we introduce the parameter κ and modify the randomized exchange algorithm by restricting those vectors (in S_t') that are allowed to swap out of the current solution. This helps us to effectively bound those higher order terms in the above inequality about the change of the objective function. But, with smaller κ, we are restricting the set of vectors that can be swapped out from the current solution. This would increase the chance of not removing a vector, and thus increasing the size of the solution. Fortunately, we can show that the increase of the solution size can be bounded by an additive d/κ term as described in Theorem 6.

The analysis for A-design in [6] contains two parts: (1) bound the expected progress; (2) prove the concentration of the total progress. The condition that the minimum eigenvalue is at least 1/4 is very important in both parts, and the optimality conditions for the convex program (2) is crucially used in the concentration argument. Interestingly, the optimality condition of the convex program (2) is also crucial in bounding the expected progress for Φ_p objective with higher p. Much effort in this paper is used to get the expectations of the objective value right, while it was relatively easy for D/A-design (when $p = 0, 1$).

3 The Analysis of Φ_p Objective

The goal of this section is to prove Theorem 9. Since we are focusing on the case of $p \le 1/\varepsilon$ in Theorem 9, we can assume without loss of generality that

$$\gamma = \max\left\{\frac{\varepsilon}{6}, \frac{1}{6p}\right\} = \frac{1}{6p} \qquad \text{and} \qquad \kappa = \max\left\{\frac{\varepsilon}{2}, \frac{1}{2p}\right\} = \frac{1}{2p}$$

in the remaining of this section. For ease of notation, we also assume the start time of the second phase is $\tau_1 = 1$. To analyze progress of the algorithm in terms of the objective function, we upper bound the change of $\mathrm{tr}(Y_t^{-p})$ after a swap using the following lemma, for which we will provide a proof in Section 3.1.

▶ **Lemma 10.** *Let $Y \succ 0$ be a d-dimensional positive definite matrix and $p \ge 1$ be an integer. For any $w \in \mathbb{R}^d$ and $v \in \mathbb{R}^d$ such that $v^\top Y^{-1} v < 1$,*

$$\begin{aligned}
&\mathrm{tr}\left((Y + ww^\top - vv^\top)^{-p}\right) \\
&\quad \le \mathrm{tr}(Y^{-p}) + \sum_{i=1}^{p} \binom{p}{i} \left((-1)^i \frac{(w^\top Y^{-1} w)^{i-1} \cdot w^\top Y^{-p-1} w}{(1 + w^\top Y^{-1} w)^i} + \frac{(v^\top Y^{-1} v)^{i-1} \cdot v^\top Y^{-p-1} v}{(1 - v^\top Y^{-1} v)^i} \right).
\end{aligned}$$

In the randomized exchange algorithm, we swap vectors u_{i_t} and u_{j_t} in each iteration where u_{i_t} is in the current solution. Thus, $u_{i_t}^\top Y_t^{-1} u_{i_t} \le 1$ always holds, and in fact it will be clear later that $u_{i_t}^\top Y_t^{-1} u_{i_t}$ is strictly less than 1. Hence, Lemma 10 can be applied repeatedly to obtain that, for any $\tau \ge 1$,

$$\begin{aligned}
\mathrm{tr}(Y_{\tau+1}^{-p}) - \mathrm{tr}(Y_1^{-p}) \le \sum_{t=1}^{\tau} \Bigg(&\underbrace{\sum_{i=1}^{p} \binom{p}{i} \frac{(u_{i_t}^\top Y_t^{-1} u_{i_t})^{i-1} \cdot u_{i_t}^\top Y_t^{-p-1} u_{i_t}}{(1 - u_{i_t}^\top Y_t^{-1} u_{i_t})^i}}_{\text{loss}} \\
&- \underbrace{\sum_{i=1}^{p} \binom{p}{i} (-1)^{i+1} \frac{(u_{j_t}^\top Y_t^{-1} u_{j_t})^{i-1} \cdot u_{j_t}^\top Y_t^{-p-1} u_{j_t}}{(1 + u_{j_t}^\top Y_t^{-1} u_{j_t})^i}}_{\text{gain}} \Bigg).
\end{aligned} \tag{4}$$

We define gain g_t, loss l_t, and progress Γ_t in the t-th iteration as follows

$$\begin{aligned}
g_t &:= \sum_{i=1}^{p} \binom{p}{i} (-1)^{i+1} \frac{(u_{j_t}^\top Y_t^{-1} u_{j_t})^{i-1} \cdot u_{j_t}^\top Y_t^{-p-1} u_{j_t}}{(1 + u_{j_t}^\top Y_t^{-1} u_{j_t})^i}, \\
l_t &:= \sum_{i=1}^{p} \binom{p}{i} \frac{(u_{i_t}^\top Y_t^{-1} u_{i_t})^{i-1} \cdot u_{i_t}^\top Y_t^{-p-1} u_{i_t}}{(1 - u_{i_t}^\top Y_t^{-1} u_{i_t})^i}, \\
\Gamma_t &:= g_t - l_t.
\end{aligned}$$

In Section 3.2, we will prove that if x is a fractional optimal solution to (2) and Z_t has lower-bounded minimum eigenvalue and the objective value of the current solution is far from optimal, then the expected progress in the t-th iteration is large. Then, in Section 3.3, we will prove that the total progress is concentrated around its expectation. Finally, we complete the proof of Theorem 9 in Section 3.4.

3.1 Change of Objective Value in One Step

In [6], a rank-two update formula is used to compute the change of the objective value in one step when $p = 1$. For general Φ_p-design, the rank-two update formula becomes considerably more complicated, and instead we do the update in two smaller steps: We first use a rank-one update to add u_{j_t} to the current solution, then use another rank-one update to remove u_{i_t} from the current solution.

Proof of Lemma 10. Let $Y_1 = Y + ww^\top$. By Sherman-Morrison formula [12], it holds that

$$\operatorname{tr}(Y_1^{-p}) = \operatorname{tr}\left(\left(Y^{-1} - \frac{Y^{-1}ww^\top Y^{-1}}{1 + w^\top Y^{-1}w}\right)^p\right) = \operatorname{tr}\left(\left(Y^{-1/2}\left(I - \frac{Y^{-1/2}ww^\top Y^{-1/2}}{1 + w^\top Y^{-1}w}\right)Y^{-1/2}\right)^p\right).$$

Then, we can apply Lieb-Thirring inequality [8] to show that

$$\operatorname{tr}(Y_1^{-p}) \le \operatorname{tr}\left(Y^{-p/2}\left(I - \frac{Y^{-1/2}ww^\top Y^{-1/2}}{1 + w^\top Y^{-1}w}\right)^p Y^{-p/2}\right) = \operatorname{tr}\left(Y^{-p}\left(I - \frac{Y^{-1/2}ww^\top Y^{-1/2}}{1 + w^\top Y^{-1}w}\right)^p\right).$$

Expanding by the binomial theorem,

$$\begin{aligned}\operatorname{tr}(Y_1^{-p}) &\le \sum_{i=0}^{p}(-1)^i\binom{p}{i}\operatorname{tr}\left(Y^{-p}\left(\frac{Y^{-1/2}ww^\top Y^{-1/2}}{1 + w^\top Y^{-1}w}\right)^i\right)\\ &= \operatorname{tr}(Y^{-p}) + \sum_{i=1}^{p}(-1)^i\binom{p}{i}\frac{(w^\top Y^{-1}w)^{i-1}\cdot w^\top Y^{-p-1}w}{(1 + w^\top Y^{-1}w)^i}.\end{aligned} \tag{5}$$

For $Y_2 = Y_1 - vv^\top$, we can apply similar argument to show that

$$\operatorname{tr}(Y_2^{-p}) \le \operatorname{tr}(Y_1^{-p}) + \sum_{i=1}^{p}\binom{p}{i}\frac{(v^\top Y_1^{-1}v)^{i-1}\cdot v^\top Y_1^{-p-1}v}{(1 - v^\top Y_1^{-1}v)^i}.$$

Notice that $Y_1 = Y + ww^\top \succcurlyeq Y$ and $v^\top Y^{-1}v < 1$, thus it holds that

$$\operatorname{tr}(Y_2^{-p}) \le \operatorname{tr}(Y_1^{-p}) + \sum_{i=1}^{p}\binom{p}{i}\frac{(v^\top Y^{-1}v)^{i-1}\cdot v^\top Y^{-p-1}v}{(1 - v^\top Y^{-1}v)^i}. \tag{6}$$

The lemma follows by combining (5) and (6). ◀

▶ Remark 11. Lemma 10 can be generalized to all real $p \ge 1$ by invoking Newton's generalized binomial theorem in the proof. To guarantee the convergence of the generalized binomial theorem, we need to ensure a stronger condition $v^\top Y^{-1}v \le \frac{1}{2}$. This is not an issue for our application, as our algorithm always removes vectors from the restricted set S_t', which guarantees that $v^\top Y^{-1}v \le \frac{1}{2}$ is satisfied. Given the new version of Lemma 10 with real p, we can generalize the main result in this paper (i.e., Theorem 2) to all real $p \ge 1$ with essentially the same analysis.

3.2 Expected Progress

To analyze the expected progress, we need to use the following two lemmas. The first one is an implication of the lower-bounded minimum eigenvalue condition, which is an analog of Lemma 4.13 in [6], and we refer the readers to the appendices of the arxiv version of this paper [5] for a proof.

▶ **Lemma 12.** *For $\gamma \le \frac{1}{6}$, if $Z_t \succcurlyeq (1-5\gamma)I$, then*

$$\langle v_i v_i^\top, Z_t^{-1} \rangle \le \alpha \langle v_i v_i^\top, A_t^{\frac{1}{2}} \rangle \le \alpha \lambda_{\min}(Z_t) \langle v_i v_i^\top, Z_t^{-1} \rangle \qquad \forall i \in [n].$$

The other one is an implication of the optimality condition of (2). The proof of the lemma is similar to the one in [6] for A-design, see the appendices of the full arxiv version [5] for more details.

▶ **Lemma 13.** *Let $x \in [0,1]^n$ be an optimal fractional solution of the convex programming relaxation (2) for the p-norm problem. Then, for each $1 \le i \le n$ with $0 < x(i) < 1$,*

$$\langle X^{-p-1}, u_i u_i^\top \rangle \le \frac{1}{k} \cdot \operatorname{tr}(X^{-p}).$$

Now, we are ready to lower bound the expected progress. We will first handle the expected loss and expected gain separately in Lemma 14 and Lemma 15. Then, combine the two parts to lower bound the expected progress in Lemma 17. For simplicity, we denote $\mathbb{E}_t[\cdot]$ as the conditional expectation given what had happened up to time t, that is, $\mathbb{E}[\cdot \mid S_{t-1}]$.

3.2.1 Expected Loss

The minimum eigenvalue lower bound (Lemma 12), the optimality condition (Lemma 13), and the introduction of the new parameter κ in the randomized exchange algorithm are all crucial in the following lemma.

▶ **Lemma 14** (Expected Loss). *Let S_{t-1} be the solution set at time t and $Z_t = \sum_{i \in S_{t-1}} v_i v_i^\top$ for $1 \le t \le \tau$. Suppose x is an optimal solution of (2), $\lambda_{\min}(Z_t) \in [1-5\gamma, 1)$, $\gamma = 1/6p$, and $\kappa = 1/2p$. Then*

$$\mathbb{E}_t[l_t] \le \frac{p}{M}\Big(\operatorname{tr}(Y_t^{-p}) - \langle X_{S_{t-1}}, Y_t^{-p-1} \rangle\Big) + O\Big(\frac{p^2 d}{kM}\Big) \cdot \operatorname{tr}(X^{-p}),$$

where we denote $X_S := \sum_{i \in S} x(i) u_i u_i^\top$ for any set $S \subseteq [n]$.

Proof. There are p terms in the loss term l_t. We deal with the linear term and higher order terms separately. Consider the linear term:

$$\begin{aligned}\mathbb{E}_t\left[\frac{p \cdot u_{i_t}^\top Y_t^{-p-1} u_{i_t}}{1 - u_{i_t}^\top Y_t^{-1} u_{i_t}}\right] &= \sum_{i \in S'_{t-1}} \frac{1-x(i)}{M}\Big(1 - \alpha \langle v_i v_i^\top, A_t^{1/2} \rangle\Big) \cdot \frac{p \cdot u_i^\top Y_t^{-p-1} u_i}{1 - u_i^\top Y_t^{-1} u_i} \\ &= \sum_{i \in S'_{t-1}} \frac{1-x(i)}{M}\Big(1 - \alpha \langle v_i v_i^\top, A_t^{1/2} \rangle\Big) \cdot \frac{p \cdot u_i^\top Y_t^{-p-1} u_i}{1 - \langle v_i v_i^\top, Z_t^{-1} \rangle},\end{aligned}$$

where the second line follows by the definitions of Y_t and Z_t, which implies that

$$\langle v_i v_i^\top, Z_t^{-1} \rangle = u_i^\top Y_t^{-1} u_i. \tag{7}$$

Note that $\gamma = 1/6p \leq 1/6$. Thus, we can apply the first inequality in Lemma 12 and then relax S'_{t-1} to S_{t-1} to obtain that

$$\begin{aligned}\mathbb{E}_t\left[\frac{p \cdot u_{i_t}^\top Y_t^{-p-1} u_{i_t}}{1 - u_{i_t}^\top Y_t^{-1} u_{i_t}}\right] &\leq \frac{p}{M} \sum_{i \in S_{t-1}} (1 - x(i)) \cdot u_i^\top Y_t^{-p-1} u_i \\ &= \frac{p}{M}\Big(\operatorname{tr}(Y_t^{-p}) - \langle X_{S_{t-1}}, Y_t^{-p-1}\rangle\Big). \qquad (8)\end{aligned}$$

Then, we consider the remaining $p-1$ higher order loss terms.

$$\begin{aligned}&\underbrace{\mathbb{E}_t\left[\sum_{l=2}^{p} \binom{p}{l} \frac{(u_{i_t}^\top Y_t^{-1} u_{i_t})^{l-1} \cdot u_{i_t}^\top Y_t^{-p-1} u_{i_t}}{(1 - u_{i_t}^\top Y_t^{-1} u_{i_t})^l}\right]}_{\textcircled{1}} \\ &\qquad = \sum_{i \in S'_{t-1}} \frac{1 - x(i)}{M} \Big(1 - \alpha\langle v_i v_i^\top, A_t^{1/2}\rangle\Big) \sum_{l=2}^{p} \binom{p}{l} \frac{(u_i^\top Y_t^{-1} u_i)^{l-1} \cdot u_i^\top Y_t^{-p-1} u_i}{(1 - u_i^\top Y_t^{-1} u_i)^l}.\end{aligned}$$

Again, by applying Lemma 12, it holds that

$$\textcircled{1} \leq \sum_{i \in S'_{t-1}} \frac{1 - x(i)}{M} \cdot u_i^\top Y_t^{-p-1} u_i \cdot \sum_{l=2}^{p} \binom{p}{l} \frac{(u_i^\top Y_t^{-1} u_i)^{l-1}}{(1 - u_i^\top Y_t^{-1} u_i)^{l-1}}.$$

Notice that

$$\lambda_{\min}(Z_t) \geq 1 - 5\gamma \qquad \Longleftrightarrow \qquad Y_t \succcurlyeq (1 - 5\gamma)X. \qquad (9)$$

Thus, by the assumption $\lambda_{\min}(Z_t) \geq 1 - 5\gamma$, it follows that

$$\textcircled{1} \leq \frac{(1 - 5\gamma)^{-p-1}}{M} \sum_{i \in S'_{t-1}} (1 - x(i)) \cdot u_i^\top X^{-p-1} u_i \cdot \sum_{l=2}^{p} \binom{p}{l} \frac{(u_i^\top Y_t^{-1} u_i)^{l-1}}{(1 - u_i^\top Y_t^{-1} u_i)^{l-1}}.$$

Using the fact that x is a fractional optimal solution to (2) and then applying Lemma 13, it holds that

$$\begin{aligned}\textcircled{1} &\leq \frac{(1 - 5\gamma)^{-p-1}}{kM} \cdot \operatorname{tr}(X^{-p}) \sum_{i \in S'_{t-1}} (1 - x(i)) \cdot \sum_{l=2}^{p} \binom{p}{l} \frac{(u_i^\top Y_t^{-1} u_i)^{l-1}}{(1 - u_i^\top Y_t^{-1} u_i)^{l-1}} \\ &\leq \frac{(1 - 5\gamma)^{-p-1}}{kM} \cdot \operatorname{tr}(X^{-p}) \sum_{l=2}^{p} \binom{p}{l} \sum_{i \in S'_{t-1}} u_i^\top Y_t^{-1} u_i \cdot \frac{(u_i^\top Y_t^{-1} u_i)^{l-2}}{(1 - u_i^\top Y_t^{-1} u_i)^{l-1}}.\end{aligned}$$

Due to the definition of the set S'_{t-1} and (7), it holds that $u_i^\top Y_t^{-1} u_i \leq \kappa$ for all $i \in S'_{t-1}$. Thus,

$$\textcircled{1} \leq \frac{(1 - 5\gamma)^{-p-1}}{kM} \cdot \operatorname{tr}(X^{-p}) \sum_{l=2}^{p} \binom{p}{l} \sum_{i \in S'_{t-1}} u_i^\top Y_t^{-1} u_i \cdot \left(\frac{\kappa}{1 - \kappa}\right)^{l-2} \cdot \frac{1}{1 - \kappa}.$$

Since $\sum_{i \in S'_{t-1}} u_i^\top Y_t^{-1} u_i \leq \langle Y_t, Y_t^{-1}\rangle = d$, we can further upper bound the $\textcircled{1}$ term by

$$\begin{aligned}\textcircled{1} &\leq \frac{(1 - 5\gamma)^{-p-1} d}{kM} \cdot \operatorname{tr}(X^{-p}) \cdot \sum_{l=2}^{p} \binom{p}{l} \left(\frac{\kappa}{1 - \kappa}\right)^{l} \cdot \frac{1 - \kappa}{\kappa^2} \\ &= \frac{(1 - 5\gamma)^{-p-1} d}{kM} \cdot \operatorname{tr}(X^{-p}) \cdot \frac{1 - \kappa}{\kappa^2} \left(\frac{1}{(1 - \kappa)^p} - 1 - \frac{p\kappa}{1 - \kappa}\right) \\ &\leq \frac{(1 - 5\gamma)^{-p-1} d}{kM} \cdot \operatorname{tr}(X^{-p}) \cdot \frac{1}{\kappa^2} \cdot \frac{1}{(1 - \kappa)^p}. \qquad (10)\end{aligned}$$

Combining (8) and (10), the expected loss can be bounded by

$$\begin{aligned}\mathbb{E}_t[l_t] &\le \frac{p}{M}\Big(\operatorname{tr}(Y_t^{-p}) - \langle X_{S_{t-1}}, Y_t^{-p-1}\rangle\Big) + \frac{(1-5\gamma)^{-p-1}d}{\kappa^2(1-\kappa)^p kM}\cdot \operatorname{tr}(X^{-p})\\ &\le \frac{p}{M}\Big(\operatorname{tr}(Y_t^{-p}) - \langle X_{S_{t-1}}, Y_t^{-p-1}\rangle\Big) + O\Big(\frac{p^2 d}{kM}\Big)\cdot \operatorname{tr}(X^{-p}),\end{aligned}$$

where the last inequality follows by the choice of γ and κ. ◀

3.2.2 Expected Gain

The analysis of the expected gain is slightly more complicated since the sampling probability does not cancel out with the denominator term in g_t as nicely as in the analysis of loss l_t, and so we will divide into two cases and use different arguments. Again, the lower bound of the minimum eigenvalue (Lemma 12) and the optimality condition (Lemma 13) are crucial in the following lemma, the proof of which is deferred to Appendix A due to the space limit.

▶ **Lemma 15** (Expected Gain). *Let S_{t-1} be the solution set at time t and $Z_t = \sum_{i\in S_{t-1}} v_i v_i^\top$ for $1 \le t \le \tau$. Suppose x is an optimal solution to* (2)*, $\lambda_{\min}(Z_t) \in [1-5\gamma, 1)$, and $\gamma = 1/6p$. Then*

$$\mathbb{E}_t[g_t] \ge \frac{p}{M}\Big(\langle X, Y_t^{-p-1}\rangle - \langle X_{S_{t-1}}, Y_t^{-p-1}\rangle\Big) - O\Big(\frac{p^2 d}{kM}\Big)\cdot \operatorname{tr}(X^{-p}).$$

3.2.3 Expected Progress

Finally, we apply Hölder's inequality appropriately to compare the gain term and loss term.

▶ **Lemma 16.** *Given positive definite matrices $A, B \in \mathbb{S}^d_{++}$ and an integer $p \ge 1$, it holds that*

$$\langle A, B^{-p-1}\rangle \ge \left(\frac{\operatorname{tr}(B^{-p})}{\operatorname{tr}(A^{-p})}\right)^{1/p}\cdot \operatorname{tr}(B^{-p}).$$

Proof. Let $A = \sum_{i=1}^d a_i v_i v_i^\top$ be the eigendecomposition of A, and $B = \sum_{i=1}^d b_i w_i w_i^\top$ be the eigendecomposition of B. Then,

$$\begin{aligned}\operatorname{tr}(B^{-p}) = \sum_{i=1}^d \frac{1}{b_i^p} &= \sum_{1\le i,j\le d} \frac{1}{b_i^p}\langle v_i, w_j\rangle^2\\ &= \sum_{1\le i,j\le d} \frac{a_j^{p/(p+1)}}{b_i^p}\langle v_i, w_j\rangle^{2p/(p+1)}\cdot \frac{1}{a_j^{p/(p+1)}}\langle v_i, w_j\rangle^{2/(p+1)}\\ &\le \Big(\sum_{1\le i,j\le d} \frac{a_j}{b_i^{p+1}}\langle v_i, w_j\rangle^2\Big)^{p/(p+1)}\cdot \Big(\sum_{1\le i,j\le d}\frac{1}{a_j^p}\langle v_i, w_j\rangle^2\Big)^{1/(p+1)}\\ &= \langle A, B^{-p-1}\rangle^{p/(p+1)}\cdot \operatorname{tr}(A^{-p})^{1/(p+1)},\end{aligned}$$

where the inequality follows by Hölder's inequality. Then, the lemma follows by taking the $(p+1)/p$'s power of both sides and rearranging the terms. ◀

Now, we lower bound the expected progress by combining Lemma 14 and Lemma 15.

▶ **Lemma 17.** *Let S_{t-1} be the solution set at time t and $Z_t = \sum_{i \in S_{t-1}} v_i v_i^\top$ for $1 \leq t \leq \tau$. Suppose x is an fractional optimal solution to (2), and $\lambda_{\min}(Z_t) \in [1-5\gamma, 1)$, $(\mathrm{tr}(Y_t^{-2}))^{1/p} \geq \lambda \cdot (\mathrm{tr}\left(X^{-2}\right))^{1/p}$ for some $\lambda \geq 1$ and for $1 \leq t \leq \tau$. Then*

$$\mathbb{E}_t[\Gamma_t] \geq \left(\frac{p(\lambda-1)\lambda^p}{M} - O\Big(\frac{p^2 d}{kM}\Big)\right) \cdot \mathrm{tr}(X^{-p}).$$

In particular, when $\lambda > 1+\varepsilon$ and $k \gtrsim \frac{pd}{\varepsilon}$, the expected progress is positive.

Proof. Combining the expected loss Lemma 14 and the expected gain Lemma 15, it follows that

$$\mathbb{E}_t[\Gamma_t] \geq \frac{p}{M}\Big(\langle X, Y_t^{-p-1}\rangle - \mathrm{tr}(Y_t^{-p})\Big) - O\Big(\frac{p^2 d}{kM}\Big) \cdot \mathrm{tr}(X^{-p}).$$

Applying Lemma 16, we derive that

$$\mathbb{E}_t[\Gamma_t] \geq \frac{p}{M}\left(\left(\frac{\mathrm{tr}(Y_t^{-p})}{\mathrm{tr}(X^{-p})}\right)^{1/p} \cdot \mathrm{tr}(Y_t^{-p}) - \mathrm{tr}(Y_t^{-p})\right) - O\Big(\frac{p^2 d}{kM}\Big) \cdot \mathrm{tr}(X^{-p}).$$

By the assumption that $(\mathrm{tr}(Y_t^{-p}))^{1/p} \geq \lambda(\mathrm{tr}(X^{-p}))^{1/p}$, or equivalently $\mathrm{tr}(Y_t^{-p}) \geq \lambda^p \, \mathrm{tr}(X^{-p})$, we arrive at the final bound that

$$\mathbb{E}_t[\Gamma_t] \geq \frac{p(\lambda-1)}{M} \cdot \mathrm{tr}(Y_t^{-p}) - O\Big(\frac{p^2 d}{kM}\Big) \cdot \mathrm{tr}(X^{-p}) \geq \left(\frac{p(\lambda-1)\lambda^p}{M} - O\Big(\frac{p^2 d}{kM}\Big)\right) \cdot \mathrm{tr}(X^{-p}). ◀$$

3.3 Martingale Concentration Argument

In this subsection, we prove that the total progress is concentrated around the expectation. The proof uses the minimum eigenvalue assumption and the optimality characterization in Lemma 13 to bound the variance of the random process. The proof idea is similar to the one in [6], but we need some additional efforts to take care of the higher order terms that are introduced by higher p-norm. We defer the detailed proof to Appendix A.

▶ **Lemma 18.** *For any $\eta > 0$, it holds that*

$$\Pr\left[\sum_{t=1}^{\tau} \Gamma_t \leq \sum_{t=1}^{\tau} \mathbb{E}_t[\Gamma_t] - \eta \bigcap \min_{1 \leq t \leq \tau} \lambda_{\min}(Z_t) \geq 1-5\gamma\right] \leq \exp\left(-\Omega\left(\frac{\eta^2 kM}{\tau p^3 \sqrt{d}(\mathrm{tr}(X^{-p}))^2 + \eta p M \, \mathrm{tr}(X^{-p})}\right)\right).$$

3.4 Proof of Theorem 9

We are ready to prove Theorem 9. Let $\tau = \frac{2M}{\varepsilon p}$. We want to upper bound the probability that the following three events happen simultaneously:

- E_1: The randomized exchange algorithm entered the second phase, i.e., $\lambda_{\min}(Z_1) \geq 1-2\gamma$ using the notation in this subsection.
- E_2: $\min_{1 \leq t \leq \tau} \lambda_{\min}(Z_t) \geq 1-5\gamma$.
- E_3: The second phase of the algorithm has not terminated by time τ.

Suppose the event E_3 happens. Then $\lambda = \min_{1 \leq t \leq \tau+1} (\mathrm{tr}(Y_t^{-p}))^{1/p} / (\mathrm{tr}(X^{-p}))^{1/p} > (1+\varepsilon)$. If the event E_2 also happens, then Lemma 17 implies that

$$\begin{aligned}\sum_{t=1}^{\tau} \mathbb{E}_t\left[\Gamma_t\right] &\geq \tau \cdot \left(\frac{p(\lambda-1)\lambda^p}{M} - O\Big(\frac{p^2 d}{kM}\Big)\right) \cdot \mathrm{tr}(X^{-p}) \\ &\geq \frac{2M}{\varepsilon p} \cdot \left(\frac{\varepsilon p}{M} - \frac{\varepsilon p}{2M}\right) \cdot \mathrm{tr}(X^{-p}) \geq \mathrm{tr}\left(X^{-p}\right), \end{aligned} \tag{11}$$

where the second inequality holds for $k \gtrsim pd/\varepsilon$ with large enough constant.

On the other hand, the initial solution of the second phase satisfies $Z_1 \succcurlyeq (1-2\gamma)I$ (follows by E_1), which implies that $Y_1 \succcurlyeq (1-2\gamma)X$. Thus, $\operatorname{tr}(Y_1^{-p}) \leq (1-2\gamma)^{-p}\operatorname{tr}(X^{-p}) \leq (1-1/3p)^{-p}\operatorname{tr}(X^{-p}) \leq 3\operatorname{tr}(X^{-p})/2$, for $p \geq 1$. When the event E_2 happens, we know from Lemma 12 that $\langle v_{i_t} v_{i_t}^\top, Z_t^{-1}\rangle \leq \alpha \cdot \langle v_{i_t} v_{i_t}^\top, A_t^{\frac{1}{2}}\rangle < \frac{1}{2}$, and so we can apply (4) to deduce that

$$\operatorname{tr}(Y_{\tau+1}^{-p}) \leq \operatorname{tr}(Y_1^{-p}) - \sum_{t=1}^{\tau}\Gamma_t \leq \frac{3}{2}\cdot\operatorname{tr}\left(X^{-p}\right) - \sum_{t=1}^{\tau}\Gamma_t.$$

Since the algorithm has not terminated by time τ,

$$\operatorname{tr}(X^{-p}) \leq \operatorname{tr}(Y_{\tau+1}^{-p}) \quad\Longrightarrow\quad \sum_{t=1}^{\tau}\Gamma_t \leq \frac{1}{2}\cdot\operatorname{tr}\left(X^{-p}\right). \tag{12}$$

Combining (11) and (12), $E_1 \cap E_2 \cap E_3$ implies a large deviation of the progress from the expectation such that

$$\sum_{t=1}^{\tau}\Gamma_t - \sum_{t=1}^{\tau}\mathbb{E}_t[\Gamma_t] < -\frac{1}{2}\cdot\operatorname{tr}(X^{-p}).$$

Thus, we can apply Lemma 18 with $\eta = \frac{1}{2}\cdot\operatorname{tr}(X^{-p})$ and $\tau = \frac{2M}{\varepsilon p}$ to conclude that

$$\begin{aligned}
\Pr\left[E_1 \cap E_2 \cap E_3\right] &\leq \Pr\left[\sum_{t=1}^{\tau}\Gamma_t < \sum_{t=1}^{\tau}\mathbb{E}_t\left[\Gamma_t\right] - \frac{1}{2}\cdot\operatorname{tr}(X^{-p}) \bigcap E_2\right] \\
&\leq \exp\left(-\Omega\left(\frac{(\operatorname{tr}(X^{-p}))^2\cdot kM}{\left(\frac{2M}{\varepsilon p}\right)p^3\sqrt{d}\cdot(\operatorname{tr}(X^{-p}))^2 + pM\cdot(\operatorname{tr}(X^{-p}))^2}\right)\right) \\
&\leq \exp\left(-\Omega\left(\frac{\varepsilon k}{p^2\sqrt{d}}\right)\right) \leq \exp\left(-\Omega\left(\frac{\sqrt{d}}{p}\right)\right) = \exp\left(-\Omega\left(\gamma\sqrt{d}\right)\right),
\end{aligned}$$

where the last inequality holds by the assumption $k \gtrsim pd/\varepsilon$, and the last equality follows as $\gamma = 1/6p$.

References

1 Zeyuan Allen-Zhu, Yuanzhi Li, Aarti Singh, and Yining Wang. Near-optimal discrete optimization for experimental design: a regret minimization approach. *Math. Program.*, 186(1-2, Ser. A):439–478, 2021. doi:10.1007/s10107-019-01464-2.

2 Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: a review. *Statist. Sci.*, 10(3):273–304, 1995.

3 Michal Derezinski, Feynman T. Liang, and Michael W. Mahoney. Bayesian experimental design using regularized determinantal point processes. In Silvia Chiappa and Roberto Calandra, editors, *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 3197–3207. PMLR, 2020. URL: http://proceedings.mlr.press/v108/derezinski20a.html.

4 David A. Freedman. On tail probabilities for martingales. *Ann. Probability*, 3:100–118, 1975. doi:10.1214/aop/1176996452.

5 Lap Chi Lau, Robert Wang, and Hong Zhou. Experimental design for any p-norm. *CoRR*, abs/2305.01942, 2020. arXiv:2305.01942.

6 Lap Chi Lau and Hong Zhou. A local search framework for experimental design. *SIAM J. Comput.*, 51(4):900–951, 2022. doi:10.1137/20M1386542.

7 Lap Chi Lau and Hong Zhou. A spectral approach to network design. *SIAM J. Comput.*, 51(4):1018–1064, 2022. doi:10.1137/20M1330762.

8 Elliott H. Lieb and Walter E. Thirring. *Inequalities for the moments of the eigenvalues of the Schrödinger Hamiltonian and their relation to Sobolev inequalities*, pages 269–304. Princeton University Press, 1976.

9 Vivek Madan, Mohit Singh, Uthaipon Tantipongpipat, and Weijun Xie. Combinatorial algorithms for optimal design. In Alina Beygelzimer and Daniel Hsu, editors, *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pages 2210–2258, Phoenix, USA, 25–28 June 2019. PMLR.

10 Zelda Mariet and Suvrit Sra. Elementary symmetric polynomials for optimal experimental design. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 2136–2145, Red Hook, NY, USA, 2017. Curran Associates Inc.

11 Aleksandar Nikolov, Mohit Singh, and Uthaipon Tao Tantipongpipat. Proportional volume sampling and approximation algorithms for A-optimal design. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pages 1369–1386, Philadelphia, PA, USA, 2019. Society for Industrial and Applied Mathematics.

12 Jack Sherman and Winifred J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *Ann. Math. Statistics*, 21:124–127, 1950. doi:10.1214/aoms/1177729893.

13 Mohit Singh and Weijun Xie. Approximate positive correlated distributions and approximation algorithms for D-optimal design. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 2240–2255, USA, 2018. Society for Industrial and Applied Mathematics.

14 Uthaipon Tantipongpipat. λ-regularized A-optimal design and its approximation by λ-regularized proportional volume sampling. *CoRR*, abs/2006.11182, 2020. arXiv:2006.11182.

15 Joel A. Tropp. Freedman's inequality for matrix martingales. *Electron. Commun. Probab.*, 16:262–270, 2011. doi:10.1214/ECP.v16-1624.

16 Richard L. Wheeden and Antoni Zygmund. *Measure and integral.* Pure and Applied Mathematics (Boca Raton). CRC Press, Boca Raton, FL, second edition, 2015. An introduction to real analysis.

A Omitted Proofs in Section 3

Proof of Lemma 15 (Expected Gain). We consider two separate cases:

$$
\begin{aligned}
&S_1 := \{j \in [m] \backslash S_{t-1} \mid \alpha \langle v_j v_j^\top, A_t^{1/2} \rangle \geq p \cdot u_j^\top Y_t^{-1} u_j\} \quad \text{and} \\
&S_2 := \{j \in [m] \backslash S_{t-1} \mid \alpha \langle v_j v_j^\top, A_t^{1/2} \rangle < p \cdot u_j^\top Y_t^{-1} u_j\}.
\end{aligned}
$$

Then, the expected gain can be written as

$$
\begin{aligned}
\mathbb{E}_t[g_t] = &\sum_{j \in S_1} \frac{x(j)}{M} \cdot \left(1 + \alpha \langle v_j v_j^\top, A_t^{1/2} \rangle\right) \cdot \sum_{l=1}^{p} \binom{p}{l} (-1)^{l+1} \frac{(u_j^\top Y_t^{-1} u_j)^{l-1} \cdot u_j^\top Y_t^{-p-1} u_j}{(1 + u_j^\top Y_t^{-1} u_j)^l} &(13)\\
&+ \sum_{j \in S_2} \frac{x(j)}{M} \cdot \left(1 + \alpha \langle v_j v_j^\top, A_t^{1/2} \rangle\right) \cdot \sum_{l=1}^{p} \binom{p}{l} (-1)^{l+1} \frac{(u_j^\top Y_t^{-1} u_j)^{l-1} \cdot u_j^\top Y_t^{-p-1} u_j}{(1 + u_j^\top Y_t^{-1} u_j)^l}. &(14)
\end{aligned}
$$

We analyze (13) and (14) separately. First, we consider (13) and rearrange the g_t term a bit, which will be easier for the analysis of (13).

$$\begin{aligned} g_t &= \frac{u_{j_t}^\top Y_t^{-p-1} u_{j_t}}{u_{j_t}^\top Y_t^{-1} u_{j_t}} \cdot \sum_{i=1}^{p} \binom{p}{i} (-1)^{i+1} \left(\frac{u_{j_t}^\top Y_t^{-1} u_{j_t}}{1 + u_{j_t}^\top Y_t^{-1} u_{j_t}} \right)^i \\ &= \frac{u_{j_t}^\top Y_t^{-p-1} u_{j_t}}{u_{j_t}^\top Y_t^{-1} u_{j_t}} \left(1 - \left(1 - \frac{u_{j_t}^\top Y_t^{-1} u_{j_t}}{1 + u_{j_t}^\top Y_t^{-1} u_{j_t}} \right)^p \right) \qquad (15) \\ &= \frac{u_{j_t}^\top Y_t^{-p-1} u_{j_t}}{u_{j_t}^\top Y_t^{-1} u_{j_t}} \left(1 - \frac{1}{(1 + u_{j_t}^\top Y_t^{-1} u_{j_t})^p} \right). \qquad (16) \end{aligned}$$

Thus, we can rewrite (13) as

$$(13) = \sum_{j \in S_1} \frac{x(j)}{M} \cdot \left(1 + \alpha \langle v_j v_j^\top, A_t^{1/2} \rangle \right) \cdot \frac{u_j^\top Y_t^{-p-1} u_j}{u_j^\top Y_t^{-1} u_j} \left(1 - \frac{1}{(1 + u_j^\top Y_t^{-1} u_j)^p} \right).$$

By the definition of S_1, it holds that

$$(13) \geq \sum_{j \in S_1} \frac{x(j)}{M} \cdot u_j^\top Y_t^{-p-1} u_j \cdot \frac{1 + p \cdot u_j^\top Y_t^{-1} u_j}{u_j^\top Y_t^{-1} u_j} \left(1 - \frac{1}{(1 + u_j^\top Y_t^{-1} u_j)^p} \right).$$

Let $x = u_j^\top Y_t^{-1} u_j > 0$ (as $Y_t \succ 0$). Then, it holds that

$$\frac{1 + px}{x} \cdot \left(1 - \frac{1}{(1+x)^p} \right) = \frac{1 + px}{x} \cdot \frac{(1+x)^p - 1}{(1+x)^p} = \frac{px(1+x)^p + (1+x)^p - 1 - px}{x(1+x)^p} \geq p.$$

Thus,

$$(13) \geq \frac{p}{M} \sum_{j \in S_1} x(j) \cdot u_j^\top Y_t^{-p-1} u_j. \qquad (17)$$

Then, we consider (14). As in the proof of Lemma 14, we separate (14) into two parts, ② concerning the linear term and ③ concerning the remaining $p - 1$ higher order terms.

$$\begin{aligned} (14) = & \underbrace{\sum_{j \in S_2} \frac{x(j)}{M} \cdot \left(1 + \alpha \langle v_j v_j^\top, A_t^{1/2} \rangle \right) \cdot \frac{p \cdot u_j^\top Y_t^{-p-1} u_j}{1 + u_j^\top Y_t^{-1} u_j}}_{\textcircled{2}} \\ & \underbrace{- \sum_{j \in S_2} \frac{x(j)}{M} \cdot \left(1 + \alpha \langle v_j v_j^\top, A_t^{1/2} \rangle \right) \cdot \sum_{l=2}^{p} \binom{p}{l} (-1)^l \frac{(u_j^\top Y_t^{-1} u_j)^{l-1} \cdot u_j^\top Y_t^{-p-1} u_j}{(1 + u_j^\top Y_t^{-1} u_j)^l}}_{\textcircled{3}} . \end{aligned}$$

The linear term ② is easy to bound, we can control it by Lemma 12 (combined with (7)) so that

$$\textcircled{2} \geq \frac{p}{M} \sum_{j \in S_2} x(j) \cdot u_j^\top Y_t^{-p-1} u_j. \qquad (18)$$

Then, we upper bound the higher order terms ③ (notice that ③ does not contain the minus sign). To upper bound ③, for each $j \in S_2$, we can assume $\sum_{l=2}^{p} \binom{p}{l} (-1)^l \frac{(u_j^\top Y_t^{-1} u_j)^{l-1} \cdot u_j^\top Y_t^{-p-1} u_j}{(1 + u_j^\top Y_t^{-1} u_j)^l} \geq 0$ without loss of generality, as otherwise we can simply ignore the j-th term. With this assumption and by the definition of S_2, it follows that

$$\begin{aligned}\textcircled{3} &\leq \frac{1}{M}\sum_{j\in S_2} x(j)(1+pu_j^\top Y_t^{-1}u_j)\sum_{l=2}^{p}\binom{p}{l}(-1)^l\frac{(u_j^\top Y_t^{-1}u_j)^{l-1}\cdot u_j^\top Y_t^{-p-1}u_j}{(1+u_j^\top Y_t^{-1}u_j)^l}\\ &= \frac{1}{M}\sum_{j\in S_2} x(j)u_j^\top Y_t^{-p-1}u_j\cdot(1+pu_j^\top Y_t^{-1}u_j)\sum_{l=2}^{p}\binom{p}{l}(-1)^l\frac{(u_j^\top Y_t^{-1}u_j)^{l-1}}{(1+u_j^\top Y_t^{-1}u_j)^l}.\end{aligned}$$

Using the assumption $\lambda_{\min}(Z_t)\geq 1-5\gamma$ and (9), it holds that

$$\textcircled{3} \leq \frac{(1-5\gamma)^{-p-1}}{M}\sum_{j\in S_2} x(j)u_j^\top X^{-p-1}u_j\cdot(1+pu_j^\top Y_t^{-1}u_j)\sum_{l=2}^{p}\binom{p}{l}(-1)^l\frac{(u_j^\top Y_t^{-1}u_j)^{l-1}}{(1+u_j^\top Y_t^{-1}.u_j)^l}.$$

Since x is an optimal solution to (2), we can apply Lemma 13 and derive that

$$\begin{aligned}\textcircled{3} &\leq \frac{(1-5\gamma)^{-p-1}}{kM}\cdot\operatorname{tr}(X^{-p})\sum_{j\in S_2} x(j)\cdot(1+pu_j^\top Y_t^{-1}u_j)\sum_{l=2}^{p}\binom{p}{l}(-1)^l\frac{(u_j^\top Y_t^{-1}u_j)^{l-1}}{(1+u_j^\top Y_t^{-1}u_j)^l}\\ &= \frac{(1-5\gamma)^{-p-1}}{kM}\cdot\operatorname{tr}(X^{-p})\cdot\\ &\qquad \sum_{j\in S_2} x(j)\cdot\frac{1+pu_j^\top Y_t^{-1}u_j}{u_j^\top Y_t^{-1}u_j}\left(\left(1-\frac{u_j^\top Y_t^{-1}u_j}{1+u_j^\top Y_t^{-1}u_j}\right)^p-1+\frac{pu_j^\top Y_t^{-1}u_j}{1+u_j^\top Y_t^{-1}u_j}\right).\end{aligned}$$

Let $x=u_j^\top Y_t^{-1}u_j$, we want to upper bound

$$\begin{aligned}&\frac{1+px}{x}\left(\left(1-\frac{x}{1+x}\right)^p-1+\frac{px}{1+x}\right)=\\ &\qquad\qquad \frac{1}{x}\left(\left(1-\frac{x}{1+x}\right)^p-1+\frac{px}{1+x}\right)+p\left(\left(1-\frac{x}{1+x}\right)^p-1+\frac{px}{1+x}\right).\end{aligned}$$

For any $y\in[0,1]$, it holds that $(1-y)^p\leq 1-py+\binom{p}{2}y^2$. Thus, it follows that

$$\begin{aligned}&\frac{1}{x}\left(\left(1-\frac{x}{1+x}\right)^p-1+\frac{px}{1+x}\right)+p\left(\left(1-\frac{x}{1+x}\right)^p-1+\frac{px}{1+x}\right)\\ &\leq \frac{1}{x}\cdot\binom{p}{2}\cdot\frac{x^2}{(1+x)^2}+p\cdot\frac{px}{1+x}=\binom{p}{2}\cdot\frac{x}{1+x}+\frac{p^2x}{1+x}\leq 2p^2x.\end{aligned}$$

Therefore, the $\textcircled{3}$ term can be further bounded by

$$\begin{aligned}\textcircled{3} &\leq \frac{2(1-5\gamma)^{-p-1}\cdot p^2}{kM}\cdot\operatorname{tr}(X^{-p})\sum_{j\in S_2} x(j)\cdot u_j^\top Y_t^{-1}u_j\\ &\leq \frac{2(1-5\gamma)^{-p-1}\cdot p^2}{kM}\cdot\operatorname{tr}(X^{-p})\langle X, Y_t^{-1}\rangle.\end{aligned}$$

Using (9), it holds that

$$\textcircled{3} \leq \frac{2(1-5\gamma)^{-p-2}\cdot p^2 d}{kM}\cdot\operatorname{tr}(X^{-p})\leq O\left(\frac{p^2d}{kM}\right)\cdot\operatorname{tr}(X^{-p}), \tag{19}$$

where the last inequality follows by the choice of γ.

Combining (17), (18) and (19), we can lower bound the expected gain by

$$\begin{aligned}\mathbb{E}_t[g_t] &\geq \frac{p}{M}\left(\sum_{j\in[m]\setminus S_{t-1}} x(j)u_j^\top Y_t^{-p-1}u_j\right)-O\left(\frac{p^2d}{kM}\right)\cdot\operatorname{tr}(X^{-p})\\ &= \frac{p}{M}\left(\langle X, Y_t^{-p-1}\rangle-\langle X_{S_{t-1}}, Y_t^{-p-1}\rangle\right)-O\left(\frac{p^2d}{kM}\right)\cdot\operatorname{tr}(X^{-p}).\end{aligned}$$ ◀

Proof of Lemma 18. We define two sequences of random variables $\{X_t\}_t$ and $\{Y_t\}_t$, where $X_t := \mathbb{E}_t[\Gamma_t] - \Gamma_t$ and $Y_t := \sum_{l=1}^{t} X_l$. It is easy to check that $\{Y_t\}_t$ is a martingale with respect to $\{S_t\}_t$. We will use Freedman's inequality (see, e.g., [4, 15]) to bound the probability $\Pr[Y_\tau \geq \eta \cap \min_{1\leq t\leq \tau} \lambda_{\min}(Z_t) \geq 1-5\gamma]$.

In the following, we first show that if the event $\min_{1\leq t\leq\tau}\lambda_{\min}(Z_t) \geq 1-5\gamma$ happens, then we can upper bound X_t and $\mathbb{E}_t[X_t^2]$ so that we can apply Freedman's inequality. To upper bound X_t, we first prove an upper bound on g_t and l_t.

Note that, if the event $\lambda_{\min}(Z_t) \geq 1-5\gamma$ happens, then $Y_t \succeq (1-5\gamma)X$, which implies

$$u_{i_t}^\top Y_t^{-p-1} u_{i_t} \leq (1-5\gamma)^{-p-1} u_{i_t}^\top X^{-p-1} u_{i_t} \quad \text{and} \quad u_{j_t}^\top Y_t^{-p-1} u_{j_t} \leq (1-5\gamma)^{-p-1} u_{j_t}^\top X^{-p-1} u_{j_t}.$$

By Observation 5, for all the exchange pairs i_t, j_t, it holds that $x(i_t), x(j_t) \in (0,1)$. Thus, we can apply Lemma 13 to show that $\langle X^{-p-1}, u_{i_t}u_{i_t}^\top\rangle \leq \frac{1}{k}\cdot \mathrm{tr}(X^{-p})$ and $\langle X^{-p-1}, u_{j_t}u_{j_t}^\top\rangle \leq \frac{1}{k}\cdot \mathrm{tr}(X^{-p})$. Therefore, since $\gamma = 1/6p$,

$$u_{i_t}^\top Y_t^{-p-1} u_{i_t} \leq O\Big(\frac{1}{k}\Big)\cdot \mathrm{tr}(X^{-p}) \quad \text{and} \quad u_{j_t}^\top Y_t^{-p-1} u_{j_t} \leq O\Big(\frac{1}{k}\Big)\cdot \mathrm{tr}(X^{-p}). \tag{20}$$

We first give a deterministic bound on g_t. Let $x = u_{j_t}^\top Y_t^{-p-1} u_{j_t}$, according to (15), the gain term g_t can be written as

$$g_t = u_{j_t}^\top Y_t^{-p-1} u_{j_t} \cdot \frac{1}{x} \cdot \Big(1 - \Big(1 - \frac{x}{1+x}\Big)^p\Big).$$

Since $(1-y)^p \geq 1 - py$ for $y \in [0,1]$ and $p \geq 1$, we can bound g_t by

$$0 \leq g_t \leq u_{j_t}^\top Y_t^{-p-1} u_{j_t} \cdot \frac{1}{x}\cdot\frac{px}{1+x} \leq p \cdot u_{j_t}^\top Y_t^{-p-1} u_{j_t} \leq O\Big(\frac{p}{k}\Big)\cdot \mathrm{tr}(X^{-p}),$$

where the last inequality follows from (20).

Then, we give an deterministic bound on l_t. By the definition of S'_{t-1} and (7), it holds that $0 < u_{i_t}^\top Y_t^{-1} u_{i_t} = \langle v_{i_t} v_{i_t}^\top, Z_t^{-1}\rangle \leq \kappa$. Thus, we can bound l_t by

$$\begin{aligned} 0 \leq l_t = \sum_{i=1}^{p} \binom{p}{i} \frac{(u_{i_t}^\top Y_t^{-1} u_{i_t})^{i-1} \cdot u_{i_t}^\top Y_t^{-p-1} u_{i_t}}{(1 - u_{i_t}^\top Y_t^{-1} u_{i_t})^i} &\leq u_{i_t}^\top Y_t^{-p-1} u_{i_t} \cdot \sum_{i=1}^{p} \binom{p}{i} \frac{\kappa^{i-1}}{(1-\kappa)^i} \\ &= u_{i_t}^\top Y_t^{-p-1} u_{i_t} \cdot \frac{1}{\kappa}\cdot\Big(\frac{1}{(1-\kappa)^p} - 1\Big). \end{aligned}$$

For $\kappa = \frac{1}{2p}$, we can control l_t such that

$$l_t \leq O(p)\cdot u_{i_t}^\top Y_t^{-p-1} u_{i_t} \leq O\Big(\frac{p}{k}\Big)\cdot \mathrm{tr}(X^{-p}),$$

where the last inequality follows from (20).

With the above bounds on g_t and l_t, we can control the size of the martingale increment by

$$|X_t| = |\mathbb{E}_t[\Gamma_t] - \Gamma_t| \leq g_t + l_t \leq O\Big(\frac{p}{k}\Big)\cdot \mathrm{tr}(X^{-p}).$$

Next, we upper bound $\mathbb{E}_t[X_t^2]$ by

$$\mathbb{E}_t[X_t^2] \leq |X_t|\cdot \mathbb{E}_t[|X_t|] \leq O\Big(\frac{p}{k}\Big)\cdot \mathrm{tr}(X^{-p})\cdot\Big(\mathbb{E}_t[g_t] + \mathbb{E}_t[l_t]\Big).$$

Using Lemma 14, we bound the expected loss term by

$$\begin{aligned}\mathbb{E}_t[l_t] &\le \frac{p}{M}\Big(\operatorname{tr}(Y_t^{-p}) - \langle X_{S_{t-1}}, Y_t^{-p-1}\rangle\Big) + O\Big(\frac{p^2 d}{kM}\Big)\cdot\operatorname{tr}(X^{-p})\\ &\le O\Big(\frac{p}{M}\Big)\cdot\operatorname{tr}(X^{-p}) + O\Big(\frac{p^2 d}{kM}\Big)\cdot\operatorname{tr}(X^{-p})\\ &\le O\Big(\frac{p}{M}\Big)\cdot\operatorname{tr}(X^{-p}),\end{aligned}$$

where the second inequality follows by the assumption that $\lambda_{\min}(Z_t) \ge 1 - 5\gamma$ happens (and (9)), the last inequality follows by the assumption on k.

Then, with the expression of g_t in (15), we write the expected gain as

$$\mathbb{E}_t[g_t] = \sum_{j\in[m]\setminus S_{t-1}} \frac{x(j)}{M}\cdot\Big(1+\alpha\langle v_j v_j^\top, A_t^{\frac{1}{2}}\rangle\Big)\cdot\frac{u_j^\top Y_t^{-p-1}u_j}{u_j^\top Y_t^{-1}u_j}\left(1-\left(1-\frac{u_j^\top Y_t^{-1}u_j}{1+u_j^\top Y_t^{-1}u_j}\right)^p\right).$$

Using the fact that $(1-y)^p \ge 1 - py$ for $y \in [0,1]$ and $p \ge 1$, the expected gain can be bounded by

$$\mathbb{E}_t[g_t] \le \sum_{j\in[m]\setminus S_{t-1}} \frac{p}{M}\cdot x(j)\cdot u_j^\top Y_t^{-p-1}u_j\cdot\frac{1+\alpha\langle v_j v_j^\top, A_t^{\frac{1}{2}}\rangle}{1+u_j^\top Y_t^{-1}u_j}.$$

By (7), $u_j^\top Y_t^{-1}u_j = \langle v_j v_j^\top, Z^{-1}\rangle$. Then, by the second inequality in Lemma 12, it holds that

$$\mathbb{E}_t[g_t] \le \frac{p}{M}\cdot\alpha\lambda_{\min}(Z_t)\cdot\sum_{j=1}^m x(j)\cdot u_j^\top Y_t^{-p-1}u_j \le \frac{p}{M}\cdot\alpha\cdot\langle X, Y_t^{-p-1}\rangle,$$

where the last inequality holds as $\lambda_{\min}(Z_t) < 1$ before the termination of the algorithm. By the assumption that $\lambda_{\min}(Z_t) \ge 1 - 5\gamma$ happens (and (9)) and the choice of $\alpha = \sqrt{d}/\gamma$ and $\gamma = 1/6p$, we obtain the bound that

$$\mathbb{E}_t[g_t] \le \frac{p\sqrt{d}}{\gamma(1-5\gamma)^{p+1}M}\cdot\operatorname{tr}(X^{-p}) \le O\Big(\frac{p^2\sqrt{d}}{M}\Big)\cdot\operatorname{tr}(X^{-p}).$$

Therefore,

$$\mathbb{E}_t[X_t^2] \le O\Big(\frac{p}{k}\Big)\cdot O\Big(\frac{p^2\sqrt{d}}{M}\Big)\cdot\big(\operatorname{tr}(X^{-p})\big)^2 = O\Big(\frac{p^3\sqrt{d}}{kM}\Big)\cdot\big(\operatorname{tr}(X^{-p})\big)^2,$$

which implies

$$W_t := \sum_{l=1}^t \mathbb{E}_l[X_l^2] \le O\Big(\frac{\tau p^3\sqrt{d}}{kM}\Big)\cdot\big(\operatorname{tr}(X^{-p})\big)^2, \quad \forall t\in[\tau].$$

Finally, we can apply Freedman's martingale inequality Theorem (see, e.g., [4, 15]) with

$$R = O\Big(\frac{p}{k}\Big)\cdot\operatorname{tr}(X^{-p}) \qquad\text{and}\qquad \sigma^2 = O\Big(\frac{p^3\sqrt{d}\tau}{kM}\Big)\cdot\big(\operatorname{tr}(X^{-p})\big)^2$$

to conclude that

$$\begin{aligned}&\Pr\left[Y_\tau \ge \eta \bigcap \min_{1\le t\le\tau}\lambda_{\min}(Z_t) \ge 1-5\gamma\right]\\ \le\ &\Pr[\exists t\in[\tau] : Y_t \ge \eta \cap W_t \le \sigma^2] \le \exp\left(-\frac{\eta^2/2}{\sigma^2 + R\eta/3}\right)\\ =\ &\exp\left(-\Omega\left(\frac{\eta^2 kM}{\tau p^3\sqrt{d}(\operatorname{tr}(X^{-p}))^2 + \eta pM\operatorname{tr}(X^{-p})}\right)\right).\end{aligned}$$

The lemma follows by noting that $\sum_{t=1}^{\tau}\Gamma_t \le \sum_{t=1}^{\tau}\mathbb{E}_t[\Gamma_t] - \eta$ is equivalent to $Y_\tau \ge \eta$. ◄

Approximation Algorithms for Maximum Weighted Throughput on Unrelated Machines

George Karakostas
Department of Computing & Software, McMaster University, Hamilton, Canada

Stavros G. Kolliopoulos[1]
Department of Informatics and Telecommunications,
National and Kapodistrian University of Athens, Greece

Abstract

We study the classic weighted maximum throughput problem on unrelated machines. We give a $(1 - 1/e - \varepsilon)$-approximation algorithm for the preemptive case. To our knowledge this is the first ever approximation result for this problem. It is an immediate consequence of a polynomial-time reduction we design, that uses any ρ-approximation algorithm for the single-machine problem to obtain an approximation factor of $(1 - 1/e)\rho - \varepsilon$ for the corresponding unrelated-machines problem, for any $\varepsilon > 0$. On a single machine we present a PTAS for the non-preemptive version of the problem for the special case of a constant number of distinct due dates or distinct release dates. By our reduction this yields an approximation factor of $(1 - 1/e) - \varepsilon$ for the non-preemptive problem on unrelated machines when there is a constant number of distinct due dates or release dates on each machine.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases scheduling, maximum weighted throughput, unrelated machines, approximation algorithm, PTAS

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.5

Category APPROX

Funding *Stavros G. Kolliopoulos*: Travel support was provided by the Faculty of Science of the National and Kapodistrian University of Athens.

1 Introduction

We study the classic scheduling problem of maximizing weighted throughput. We are given a set $\mathcal{J}$ of n jobs, where job j has release date r_j, due date d_j and nonnegative weight w_j. The jobs are to be scheduled on a set $\mathcal{M}$ of m machines so that no two jobs are ever processed simultaneously on the same machine. Job j completes when it has been processed for an amount equal to its processing time. If *preemption* is allowed, the processing of a job may be interrupted and resumed later at no cost. The objective is to maximize the total weight of jobs that complete by their due dates. This basic problem has been studied for decades, but for most of its versions we have no tight approximability results.

We employ the standard 3-field notation $\alpha|\,\beta|\,\gamma$ of [18], where α stands for the machine environment, β for the job characteristics and γ for the objective function. When $\beta = pmtn$ preemption is allowed. When $\alpha = 1$ there is a single machine; $\alpha = P$ stands for m *identical* parallel machines: job j has processing time p_j on every machine i. When $\alpha = R$ the m machines are *unrelated:* job j has processing time p_{ij} on machine i. In this paper we focus on the most general machine environment of unrelated machines. The problem of maximizing weighted throughput on unrelated machines without preemption is denoted by $R|\,r_j|\,\sum w_j\bar{U}_j$.

[1] Corresponding author

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 5; pp. 5:1–5:17
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We provide a polynomial-time reduction that shows that a ρ-approximation algorithm for the single-machine problem $1|\, r_j,\, \beta|\, \sum w_j \bar{U}_j$ yields a $((1-1/e)\rho - \varepsilon)$-approximation for $R|r_j, \beta| \sum w_j \bar{U}_j$, for any $\varepsilon > 0$ (cf. Theorem 2). For the preemptive version and using different ideas, Kalyanasundaram and Pruhs gave a 6-approximate reduction from $P|r_j, pmtn| \sum w_j \bar{U}_j$ to the single-machine problem [13]. Pruhs and Woeginger [19] designed an FPTAS for $1|\, r_j, pmtn|\, \sum w_j \bar{U}_j$, using a pseudopolynomial algorithm by Lawler [17]. Therefore, our framework produces a $(1 - 1/e - \varepsilon)$-approximation algorithm for $R|\, r_j,\, pmtn|\, \sum w_j \bar{U}_j$. To our knowledge this is the first approximation result for preemptive throughput maximization on unrelated machines (see also [7]). Our reduction is based on a simple idea. We write a Configuration LP for the parallel-machine setting and use the single-machine algorithm as an approximate separation oracle for the dual. Except for the definition of what constitutes a configuration, the LP and its dual are essentially the same as the ones in [8]. In the latter a Knapsack subroutine was used as a separation oracle. We round the fractional primal LP solution using the dependent rounding method of [21, 9]. Despite its simplicity, our reduction is powerful enough to allow machine-dependent release and due dates: on machine i, job j can be processed in the interval $[r_{ij}, d_{ij}]$. Moreover, for the preemptive case, our algorithm produces a schedule on the unrelated machines with the desirable property of no migrations: every job is processed in its entirety on a single machine. For the non-preemptive case, our approach shows the existence of a $((1-1/e)\rho_* - \varepsilon)$-approximation algorithm, where ρ_* is the approximability threshold of $1|\, r_j|\, \sum w_j \bar{U}_j$. Despite decades of research, the latter threshold is yet to be determined; we only know that $\rho_* \leq 1/2$ [4, 2]. The best known bound for the unweighted case (i.e., $w_j = 1, \forall j$) is 0.6448 [12]. Currently, our reduction does not improve on the best known non-preemptive algorithms for $R|\, r_j|\, \sum w_j \bar{U}_j$, which achieve a ratio of $1/2 - \varepsilon$ [2, 4]. Since the single-machine problem is far from well-understood, special cases have received considerable attention. Our method can translate all existing optimal algorithms or approximation schemes for special cases of the single-machine problem to an $(1 - 1/e - \varepsilon)$ guarantee on unrelated machines. See Corollary 1 for a list of results.

The unrelated machines setting can be further extended by adding constraints on the grouping of jobs. We provide one such extension. Every job j has a *type* t_j from a finite set T of types. Machines need to undergo preparation to accommodate different job types. For every machine i, we are thus given a set $E_i \subseteq 2^T$ that specifies the allowed combinations of types that may be processed on i in a feasible schedule. Denote this problem as $R|\, r_j,\, \text{types}\, |\, \sum_j w_j \bar{U}_j$. Using a ρ-approximate oracle for $1|\, r_j,\, \beta|\, \sum w_j \bar{U}_j$ we obtain a $((1-1/e)\rho - \varepsilon)$ guarantee for $R|\, r_j,\, \beta,\, \text{types}\, |\, \sum_j w_j \bar{U}_j$ in time polynomial in $\sum_{i \in \mathcal{M}} |E_i|$. To avoid clutter we present in Theorem 2 the result where all jobs have the same type and provide the details for the more general setting in the Appendix.

Our algorithm for $R|\, r_j, pmtn|\, \sum w_j \bar{U}_j$ uses the FPTAS of [17, 19] for the preemptive case on a single machine as a black box. For the single-machine non-preemptive case, the dynamic program of Lawler [17] can produce an FPTAS only for the case of *similarly ordered* release and due dates, i.e., $d_i < d_j$ implies $r_i \leq r_j$ for each pair of jobs i and j. The dynamic program in [17] yields thus an FPTAS for the non-preemptive version when all jobs have a common due date or a common release date. It takes significant effort and new ideas to overcome the restriction of similarly ordered release and due dates. We show how to do this and obtain a PTAS for the single-machine, non-preemptive version, when there is a constant number of distinct due dates (or release dates) (cf. Theorems 8 and 9). At a high level our PTAS combines the FPTAS of [17] with the ideas from the PTAS of Khan et al. [15] for the Generalized Assignment Problem with a constant number of bins. More specifically, our approach can be summarized as follows (we restrict our description to the case of a constant number of distinct due dates; the arguments are symmetric for the constant number of release dates). By guessing the jobs straddling the due dates, we split the schedule into

Table 1 Summary of our main results for weighted throughput on unrelated machines; ρ_* is the approximability threshold of $1|\,r_j|\,\sum w_j\bar{U}_j$.

Preemption	m	# distinct d_j	# distinct r_j	Previous	This work
$\surd$	any	any	any	–	$1-1/e-\varepsilon$ (Cor. 1)
–	any	any	any	$1/2-\varepsilon$ [2, 4]	$(1-1/e)\rho_*-\varepsilon$ (Thm. 2)
–	1	$O(1)$	any	$1/2-\varepsilon$ [2, 4]	PTAS (Thm. 8)
–	1	any	$O(1)$	$1/2-\varepsilon$ [2, 4]	PTAS (Thm. 9)
–	any	$O(1)$	any	$1/2-\varepsilon$ [2, 4]	$1-1/e-\varepsilon$ (Cor. 1)
–	any	any	$O(1)$	$1/2-\varepsilon$ [2, 4]	$1-1/e-\varepsilon$ (Cor. 1)

bins of two different kinds: (i) bins with jobs whose release dates occur before the bin, and (ii) bins with jobs whose release date may occur inside the bin. By partitioning the jobs of $\mathcal{J}$, we split the problem into a constant number of subproblems each defined on a disjoint subset of jobs and where each subproblem contains only one bin of the second kind. This allows us to apply the ideas of [17] on the bin of kind (ii), and the GAP ideas of [15] on the bins of kind (i) *simultaneously*. The combination of all "parallel" subproblems defines the dynamic programming state of our PTAS.

By the aforementioned reduction our PTAS yields a $(1-1/e-\varepsilon)$-approximation on unrelated machines when on every machine the number of distinct due dates or release dates is constant. Even for these special cases, we are not aware of any previous results that improve upon the $1/2-\varepsilon$ ratio of [2, 4]. We summarize our main results in Table 1.

Other related work. The single-machine problem without release dates $1|\,|\,\sum_j w_j\bar{U}_j$ is equivalent in optimality with the problem of minimizing the total weight of late jobs. The latter was one of Karp's 21 NP-complete problems [14]. Without release dates, preemption is of no use; therefore, $1|\,pmtn|\,\sum_j w_j\bar{U}_j$ is also NP-complete. Several versions of the unweighted problem are hard. $1|\,r_j|\,\sum_j \bar{U}_j$ is strongly NP-complete [10], and $R|\,r_j|\,\sum_j \bar{U}_j$ is MAX SNP-hard [3]. $P|\,pmtn|\,\sum \bar{U}_j$ is NP-complete [16]. Chuzhoy et al. [6] provide a $(1-1/e-\varepsilon)$-approximation algorithm for the Job Interval Selection Problem, in which for every job we are given an explicit list of allowed intervals and the intervals selected for all scheduled jobs must be disjoint. The objective is to maximize the number of scheduled jobs. Because of the explicit list requirement, the running time of the algorithm of [6] becomes pseudopolynomial when applied to $R|\,r_j|\,\sum_j \bar{U}_j$. Hyatt-Denesik et al. [11] provide a PTAS that works for unweighted jobs on identical machines, when m, the number of distinct release dates, and the number of distinct due dates, are all bounded by a constant. A number of further results for identical machines are given in [4, 12].

The paper is structured as follows. In Section 2 we provide the reduction from the unrelated machines to the single-machine problem and the improved approximation bounds that are obtained as a consequence. In Section 3 we present the PTAS for the single-machine case when the number of distinct due dates is fixed. In Section 4 we outline the necessary modifications to the algorithm from Section 3 in order to obtain a PTAS for the single-machine when the number of distinct release dates is fixed.

2 Reduction to the single-machine case

In this section we provide a polynomial-time approximation-preserving reduction from the problem with machine-dependent release and due dates, $R|\,r_{ij},\,d_{ij},\,\beta|\,\sum w_j\bar{U}_j$, to $1|\,r_j,\,\beta|\,\sum w_j\bar{U}_j$. We provide first a Configuration LP relaxation for $R|\,r_{ij},\,d_{ij},\,\beta|\,\sum w_j\bar{U}_j$.

For machine $i \in \mathcal{M}$ we define a set $\mathcal{C}(i)$ of configurations. A configuration $C \in \mathcal{C}(i)$ is a schedule of a subset $J \subseteq \mathcal{J}$ of jobs on machine i such that (i) all jobs in J respect their release and due dates on i (ii) there is no unnecessary idle time (iii) the β-constraints are met. $\mathcal{C}(i)$ is a finite set whose size can be naively bounded by $(n!)2^n$. We slightly abuse notation and we also view a configuration C as the set of jobs in the corresponding schedule. Without loss of generality we can assume that $\mathcal{C}(i) \cap \mathcal{C}(j) = \varnothing$ for $i, j \in \mathcal{M}$, $i \neq j$. The configuration LP, here denoted (CLP), is the formulation (1) of [8] and has a variable x_C for each machine i and configuration $C \in \mathcal{C}(i)$:

$$\max \sum_{j \in \mathcal{J}} w_j \left(\sum_{i \in \mathcal{M}} \sum_{C \in \mathcal{C}(i): j \in C} x_C \right) \tag{CLP}$$

$$\sum_{C \in \mathcal{C}(i)} x_C \leq 1 \qquad \forall i \in \mathcal{M} \tag{1}$$

$$\sum_{i \in \mathcal{M}} \sum_{C \in \mathcal{C}(i): j \in C} x_C \leq 1 \qquad \forall j \in \mathcal{J} \tag{2}$$

$$x_C \geq 0 \qquad \forall i \in \mathcal{M}, \forall C \in \mathcal{C}(i) \tag{3}$$

The set of constraints (1) ensures that to each machine is assigned at most one configuration. Constraints (2) ensure that each job is assigned at most once. Clearly, an integer solution to (CLP) corresponds to a feasible schedule for $R|\,\beta|\,\sum w_j \bar{U}_j$. For a configuration C, let $w(C) := \sum_{j \in C} w_j$. The dual of (CLP) is the following:

$$\min \sum_{i \in \mathcal{M}} y_i + \sum_{j \in \mathcal{J}} z_j \tag{D-CLP}$$

$$y_i + \sum_{j \in C} z_j \geq w(C) \qquad \forall i \in \mathcal{M}, \forall C \in \mathcal{C}(i) \tag{4}$$

$$y, z \geq 0 \tag{5}$$

In what follows, we will use the notion of a *ρ-approximate separation oracle* for (D-CLP) (see e.g., [8]), i.e., a polynomial-time algorithm that, given values y, z, either returns a violated constraint, or guarantees that values $y/\rho, z$ satisfy constraints (4)-(5). It is well-known (cf. Lemma 2.2 in [8]) that such an oracle, combined with binary search on the optimal value, implies a polynomial-time $(\rho - \delta)$-approximate algorithm for solving (D-CLP), and hence (CLP), for any constant $\delta > 0$ (without any constraint violations). The facet complexity ϕ of (D-CLP) is $O(n \log w_{\max})$, where $w_{\max} = \max_{j \in \mathcal{J}} w_j$. Since the coefficients of the objective function are all 1, by well-known arguments the optimal value of (D-CLP) can be represented by $O(n^2\phi)$ bits and hence binary search on the optimum runs in polynomial-time (see Corollary 10.2a in [20]). Our reduction is based on the following simple fact.

▶ **Lemma 1.** *If there is a polynomial-time ρ-approximation algorithm A_β for the problem* $1|\,r_j,\,\beta|\,\sum w_j \bar{U}_j$, *then there is a ρ-approximate separation oracle for* (D-CLP).

Proof. Given a candidate solution (y, z) to (D-CLP), the separation oracle has to solve $|M|$ instances of $1|\,\beta|\,\sum w_j \bar{U}_j$, one instance I_i for each $i \in \mathcal{M}$.

We define instance I_i. We have a single machine and the n jobs in $\mathcal{J}$ where job j has processing time p_{ij}, release date r_{ij}, due date d_{ij}. Job j has a (possibly negative) *value* $v_j := w_j - z_j$. There is a violated constraint (4) corresponding to machine i iff there is a feasible schedule for I_i where the total value of the on-time jobs exceeds y_i. Any such schedule can contain only jobs of positive value; therefore, we discard any job j with $v_j \leq 0$. If there are no jobs of positive value, then the constraint cannot be violated for any configuration in $\mathcal{C}(i)$ and we move to examine the next instance I_{i+1}.

We run algorithm A_β on instance I_i. A_β will return a feasible schedule whose value is at least ρ-times the maximum value, for $\rho \leq 1$. If this value exceeds y_i we have identified a violated constraint. Else the vectors $(y/\rho, z)$ satisfy all constraints (4) for machine i. ◀

▶ **Theorem 2.** *If there is a polynomial-time ρ-approximation algorithm A_β for the problem $1|\, r_j,\, \beta|\, \sum w_j \bar{U}_j$, then there is a $((1 - 1/e)\rho - \varepsilon)$-approximation algorithm for $R|\, r_{ij},\, d_{ij},\, \beta|\, \sum w_j \bar{U}_j$ where $\varepsilon > 0$ is any constant of our choice.*

Proof. By Lemma 1 and Lemma 2.2 in [8] the ρ-approximation algorithm A_β for $1|\beta|\sum w_j \bar{U}_j$, can be used to compute a $(\rho - \delta)$-approximate solution x^* to (CLP), for some $\delta > 0$. It remains to round the fractional solution x^* to an integer one. We use the dependent rounding method of [21, 9] in the setting where we have a collection of m star graphs, each corresponding to a machine $i \in M$ and the subset of configurations from $C(i)$ that have been assigned by a non-zero amount by x^* to i. We sample independently from m distributions, one for each star, to produce an integer assignment $\hat{x}$ of configurations to the machines. Each distribution and the corresponding sampling method are designed using the technique of [21], so that, with probability 1, each machine gets one configuration and the assignment of configurations to a machine respects negative correlation. Moreover, the expectation of $\hat{x}_C$ equals x^*_C. Adapting slightly the analysis for the maximum coverage problem in [21] we obtain that the expected total weight of the jobs that appear in at least one configuration in the support of $\hat{x}$ is at least $(1 - 1/e)$ the value of the x^* solution. For the sake of completeness we include in Appendix A the relevant details from [21]. Every configuration chosen by $\hat{x}$ corresponds to a feasible schedule on the corresponding machine. To obtain the final schedule, for every job j that appears in more that one configurations we arbitrarily delete all but one of its occurrences. Note that every machine gets exactly one configuration by $\hat{x}$, therefore the occurrences of j happen each on a different machine. Since x^* is $(\rho - \delta)$-approximate we obtain the theorem. ◀

The upcoming corollary collects applications of Theorem 2 to settings where an (F)PTAS exists for the single machine case. The *additive laxity* of a job j is defined as $d_j - p_j - r_j$. For each application we provide the reference for the single-machine result. Note that the algorithm in [1] finds an optimal solution.

▶ **Corollary 1.** *There is a polynomial-time $(1 - 1/e - \varepsilon)$-approximation algorithm for $R|\, r_{ij},\, d_{ij},\, \beta|\, \sum_j w_j \bar{U}_j$ where jobs have machine-dependent release and due dates and β can stand for the following:* (i) *preemption is allowed [17, 19]* (ii) *on every machine the number of distinct release dates or the number of distinct due dates is constant (Theorems 8, 9)* (iii) *on every machine the number of distinct processing times is constant and all jobs have unit weights [11]* (iv) *for every machine i and job j, $p_{ij} = p_i$, i.e., all jobs have the same processing time on machine i [1]* (v) *on every machine the jobs have equal additive laxity [5]* (vi) *on every machine release dates and due dates are similarly ordered [17, 19].*

In all the results of Corollary 1, we used as a lower bound for the optimum the value of (CLP), i.e., the value of a (fractional) schedule in which a job may be simultaneously executed on more than one machine. On the other hand the algorithm of Theorem 2 produces an integer solution where every scheduled job runs completely on one machine, i.e., in the preemptive case it produces a schedule with no migrations. The following is immediate.

▶ **Corollary 2.** *The approximation ratio of* $(1 - 1/e - \varepsilon)$ *for* $R|\, r_{ij},\, d_{ij},\, pmtn|\, \sum_j w_j \bar{U}_j$ *holds both for the problem version where a job may be simultaneously executed on two or more machines and for the version where this is not allowed.*

The above can be extended to the setting where jobs have different types, see Appendix B.

3 Constant number of distinct due dates

In this section we provide a PTAS for the setting where we have a single machine and the number of distinct due dates is constant.

3.1 Preliminaries

For a set of jobs S, we denote by $w(S) = \sum_{j \in S} w_j$ its total weight, $p(S) = \sum_{j \in S} p_j$ its total processing time. Wlog only jobs that meet their due dates are processed in a feasible schedule. Given a schedule σ in which a set S of jobs is processed we denote by $w(\sigma)$ the quantity $w(S)$. For a time interval B we denote its length by $|B|$. The jobs *scheduled in* B are jobs that start and finish in the interval. We slightly abuse notation and denote by $w(B)$ the total weight of the jobs scheduled in B if an associated schedule is clear from the context.

We assume that the due dates of the n jobs take a value out of k possible numbers $D_1 \leq D_2 \leq \ldots \leq D_k$. Let $D_0 := 0$. We assume that time 0 coincides with the earliest release date. Wlog the time horizon $T = D_k$. For an integer m, the notation $m \in (i, k]$ (resp. $m \in [i, k]$) stands for $m = i+1, \ldots, k$ (resp. $m = i, \ldots, k$). Similarly $m \in [i, k)$ is a shorthand for $m = i, \ldots, k-1$. We denote by R_i, $i \in [1, k]$, the set of jobs j s.t. $D_{i-1} \leq r_j < D_i$.

3.2 A structure lemma

In this section we devise structural properties of a near-optimal solution and show how to pre-compute in polynomial time some of its features.

Let OPT be an optimal solution for the problem $1|\, r_j|\, \sum w_j \bar{U}_j$. Wlog $w(OPT) \geq w_{\max}$. Let $\varepsilon_1 > 0$ be any constant. We apply the classic knapsack rounding in order to reduce the job weights within a polynomial range: by rounding all job weights down to the closest multiple of $\theta := \varepsilon_1 w_{\max}/n$, the total weight is still at least $(1 - \varepsilon_1)OPT$, but now all job weights are between 1 and $O(n/\varepsilon_1)$ in units of θ. In addition, we can impose the following special structure:

- We shift all scheduled jobs later, to start as late as possible: the schedule consists of contiguous (i.e., without idle time) blocks of jobs, each finishing at a due date (with the last block finishing at D_k).
- In every interval $[D_{i-1}, D_i)$, we can rearrange the jobs that start and finish within the interval to be scheduled in non-decreasing order of release dates. Note that any jobs from the set R_i that are scheduled in the interval, are scheduled last.

A job j *straddles* time t if j starts at or before t and finishes after t. Let $S_1, S_2, \ldots, S_{k-1}$ be the jobs straddling due dates $D_1, \ldots, D_{k-1}$, and let s_i, f_i be the starting and finishing times of straddler S_i. Note that due to the structure of OPT, some due dates may not have a straddler, just like D_0, D_k; in this case $S_i = \emptyset$. Wlog we may assume that no job straddles more than one due date. We will assume that we know the straddlers by enumerating all $O(n^k)$ possibilities.

Let $K_m = (f_{m-1}, s_m)$ be the interval between the end of S_{m-1} and the beginning of S_m, and $|K_m| \leq D_m - D_{m-1}$ its length. K_m is the concatenation of intervals $B_m^1, B_m^2, \ldots, B_m^m$ in this order, so that B_m^h contains only the jobs of R_h scheduled in K_m. We refer to these intervals as *bins.* Note that the jobs in each B_m^h are scheduled al late as possible, and each bin straddles only release dates. Figure 1 shows bins $B_m^1, \dots, B_m^{m-2}, B_m^{m-1}, B_m^m$ of K_m. The following lemma will allow us to enumerate the bin sizes in polynomial time for the PTAS.

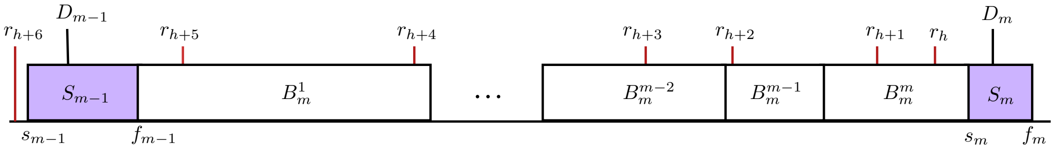


Figure 1 Bins $B_m^1, \dots, B_m^{m-2}, B_m^{m-1}, B_m^m$ of K_m. Bin B_m^m contains jobs in R_m, i.e., with release dates $r_h, r_{h+1}, \dots, r_{h+5}$. Jobs with release date r_{h+6} will be scheduled in B_m^{m-1}.

▶ **Lemma 3.** *Let $\varepsilon > 0$ be any given constant. Let be $J \subseteq \mathcal{J}$ be a set of jobs feasibly scheduled in an interval I without idle time where I does not contain a due date. Then there is $\hat{J} \subseteq J$, and a subinterval $\hat{I}$ of I s. t.* (i) *$\hat{I}$ has the same right endpoint as I,* (ii) *$w(\hat{J}) \geq (1-3\varepsilon)w(J)$ and $p(\hat{J}) \leq |\hat{I}| \leq |I|$,* (iii) *the jobs in $\hat{J}$ can be feasibly scheduled without idle time in $\hat{I}$, and* (iv) *$|\hat{I}|$ takes values from a set of size $O(n^{1/\varepsilon^2} \log_{1+\varepsilon} |I|)$.*

Proof. Let σ be the feasible schedule of the jobs of J in I. The set $\hat{J}$ and the corresponding feasible schedule $\hat{\sigma}$ are derived in two phases. We adapt the methods in [15] to account for the presence of release dates in I. In Phase 1, we apply Theorem 11 of [15] for a single bin that corresponds to interval I. We establish the existence of subsets X and Y of J. Define $J' = J - X - Y$. By Theorem 11, for every $j \in J'$, $p_j \leq \varepsilon(|I| - p(X))$. We produce a schedule σ' of $X \cup J'$ by deleting from σ the jobs of Y and replacing them by idle time. Schedule σ' executes in the same interval I as σ. By Theorem 11, $w(X \cup J') \geq (1-\varepsilon)w(J)$.

In Phase 2, we extend the trimming method of [15] to show that there is $R \subseteq J'$ s.t. $w(R) \leq (2\varepsilon/(1-\varepsilon))w(J')$ and there is an integer β s.t. $p(J'-R) \leq (1+\varepsilon)^\beta \leq |I| - p(X)$. See Appendix D.3. We set $\hat{J} = X \cup (J' - R)$. We have that $w(\hat{J}) \geq (1-\varepsilon)(1-2\varepsilon/(1-\varepsilon))w(J) = (1-3\varepsilon)w(J)$. The schedule $\hat{\sigma}$ results from σ' by deleting all jobs in R and shifting jobs as needed to the right to remove any idle time. This is feasible, since I doesn't contain any due dates. It suffices to set $\hat{I}$ to an interval that ends at the right endpoint of I such that $|\hat{I}|$ equals $p(\hat{J})$. By Theorem 11, there are $O(n^{1/\varepsilon^2})$ choices for the set X and accordingly that many choices for $p(X)$. The value $p(J'-R)$ can be upper-bounded by a number that takes $O(\log_{1+\varepsilon} |I|)$ many values. Therefore $|\hat{I}|$ takes values from a set of size $O(n^{1/\varepsilon^2} \log_{1+\varepsilon} |I|)$. ◀

We apply Lemma 3 with $\varepsilon := \varepsilon_2$ to the jobs scheduled in OPT in each of the $O(k^2)$ bins B_m^i, $m \in [1,k]$, $i \in [1,m]$. There are sets X_m^i and integers a_m^i such that there is a near-optimal feasible schedule that assigns to each bin B_m^i a set S_m^i of "small" jobs in addition to X_m^i so that $p(S_m^i) \leq (1+\varepsilon_2)^{a_m^i}$. By enumerating all $O(n^{O(k^2/\varepsilon_2^2)}(\log_{1+\varepsilon_2} D_k)^{k^2})$ possibilities, we can assume that we have determined all the X_m^i's and a_m^i's.

Let $\widehat{OPT}$ be the solution of value at least $(1-3\varepsilon_2)(1-\varepsilon_1)w(OPT)$ obtained from OPT by replacing each bin B_m^i by a (smaller) bin $\hat{B}_m^i$ with known size $|\hat{B}_m^i| = p(X_m^i) + (1+\varepsilon_2)^{a_m^i}$. By Lemma 3 $|\hat{B}_m^i| \leq |B_m^i|$. Then K_m has been replaced by a (smaller) $\hat{K}_m$ of size $\hat{K}_m = \sum_{i=1}^m |\hat{B}_m^i|$, and all scheduled jobs are shifted as late as possible, while maintaining $S_1, S_2, \dots, S_{k-1}$ as straddlers. Figure 2 shows $\widehat{OPT}$.

Expansion of X_m^i. Let $|\hat{B}_m^i|' := |\hat{B}_m^i| - p(X_m^i)$ be the bin space available for jobs other than X_m^i in $\widehat{OPT}$, i.e., job sets J_m^i s.t. $p(J_m^i) \leq |\hat{B}_m^i|'$. In order to be able to apply the techniques of [15] later, we need to keep in $\widehat{OPT}$ only jobs of J_m^i that satisfy a property

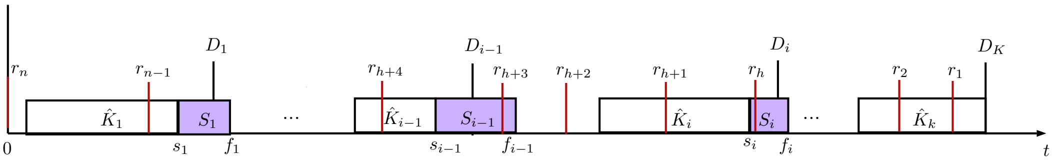


Figure 2 An approximate optimal schedule $\widehat{OPT}$. Note that the length of $\hat{K}_i$ is at most the length of the original K_i. Release dates are in non-increasing order. R_i contains all jobs with release dates $r_h, r_{h+1}, r_{h+2}, r_{h+3}$.

stronger than the property $p_j \le \varepsilon(|B^i_m| - p(X^i_m))$ guaranteed by Lemma 3 for all jobs of J^i_m. We apply the structure theorem of [15] (Theorem 11 in Appendix D) on each job set $J^i_{i+1}, J^i_{i+2}, \ldots, J^i_k$ for each $i \in [1, k)$, to define job sets W^i_m and U^i_m with $|W^i_m| = O(1/\varepsilon_3^2)$, such that $w(\cup_{i,m} U^i_m) \le \varepsilon_3 \widehat{OPT}$ and

$$p_j \le \varepsilon_3(|\hat{B}^i_m|' - p(W^i_m)),\ \forall j \in J^i_m - W^i_m - U^i_m.$$

Note that bins $B^i_i, i = 1, 2, \ldots, k$ do not participate in this process, and, therefore, $W^i_i := \emptyset$. We expand the sets X^i_m to include W^i_m, i.e., we reset $X^i_m := X^i_m \cup W^i_m$ for all i and m. The sets W^i_m can be enumerated in $O(n^{k^2/\varepsilon_3^2})$ time.

▶ **Definition 1.** *A job j is* large *if $j \in \bigcup_{i,m} X^i_m$.*

▶ **Definition 2.** *A job j is* small *for bin $\hat{B}^i_m$ if j is not large and $p_j \le \varepsilon_3(|\hat{B}^i_m|' - p(W^i_m))$.*

Since $|\cup_{i,m} X^i_m| = O(k^2/(\min\{\varepsilon_2, \varepsilon_3\})^2)$, for the rest of the paper we will assume that we have "guessed" all sets X^i_m of large jobs and their placement in bins. Keeping in $\widehat{OPT}$ only the large and small jobs, as defined in Definitions 1 and 2, $\widehat{OPT}$ still achieves total weight at least $(1 - \varepsilon_3)(1 - 3\varepsilon_2)(1 - \varepsilon_1)OPT$.

Calculation of times s_i, f_i. Since we can enumerate them in polynomial time, in what follows we will assume that we have "guessed" sizes $|\hat{K}_i|$, $i = 1, \ldots, k$. Starting from D_k, and going backwards in time, we use these sizes to calculate times $f_{k-1}, s_{k-1}, f_{k-2}, s_{k-2}, \ldots, f_1, s_1$. We should only be careful in case $s_i - |\hat{K}_i| - p(S_{i-1}) < D_i - D_{i-1}$, i.e., when the beginning of $\hat{K}_i$ is more than $p(S_{i-1})$ time units after D_{i-1}. In this case, we don't allow S_{i-1} to be scheduled after D_{i-1} (as a non-straddler), but we set $s_{i-1} := D_{i-1}$, and continue.

The following lemma summarizes the properties of $\widehat{OPT}$:

▶ **Lemma 4** (Structure Lemma). *There is a feasible schedule $\widehat{OPT}$ of weight at least $(1 - \varepsilon_3)(1 - 3\varepsilon_2)(1 - \varepsilon_1)OPT$, such that:*

- *Job weights are between 1 and $O(n/\varepsilon_1)$.*
- *There are k known straddlers S_i, with known starting and finishing times s_i, f_i, $i \in [1, k]$.*
- *Jobs are scheduled in intervals $\hat{K}_i \subseteq (f_{i-1}, s_i)$ contiguously as late as possible, and in Earliest Release Date (ERD) order. Each $\hat{K}_i$ is adjacent to S_i, and its size $|\hat{K}_i|$ is known.*
- *Only large and small jobs are scheduled; the large jobs can be computed in polynomial-time.*

3.3 A PTAS for $\widehat{OPT}$

We order the jobs in Latest Release Date (LRD) ordering, i.e., in non-increasing release date order $r_1 \ge r_2 \ge \ldots \ge r_n = 0$. The algorithm will schedule the jobs one at a time, and a job scheduled in some bin $\hat{B}^i_m$ of $\hat{K}_i$ is scheduled right *before* the jobs already scheduled in $\hat{B}^i_m$. There are at most k^2 different bins overall. Recall that all job weights are at most $O(n/\varepsilon_1)$, hence there are $O((n^2/\varepsilon_1^2)^{k^2})$ different combinations of total weights for the bins.

3.3.1 Intuition

When there is only one due date D_1, the DP of [17] for non-preemptive maximum throughput is applicable, because its requirement of similarly ordered release and due dates is met. Crucially, the correctness of the algorithm is based on DP subproblems computing the minimum total schedule time needed to achieve a total weight target (cf. Appendix C). Unfortunately, this approach cannot work with more than one due date, because for a given R_i, the notion of minimum total scheduling time is not well-defined for all its bins *simultaneously*.

For a different approach that allows for simultaneous scheduling across all $\hat{K}_m$'s, one may turn to [15], who give a very interesting PTAS for the Generalized Assignment Problem (GAP) with a *constant* number of bins (cf. Appendix D). When discretizing the state space of the dynamic program the algorithm of [15] augments bin capacities by a $(1+\varepsilon)$ factor. It then employs trimming (see Appendix D.2) to remove a low-weight contiguous set of jobs, which exists by the Pigeonhole Principle, so that the remaining jobs fit in the bins with the original capacities restored. In our case, increasing the bin capacities may entail that in some $\hat{K}_m$, jobs may end up scheduled to start *before* their release dates in (f_{m-1}, s_m). (We remind the reader that $f_{m-1} > D_{m-1}$ is the completion time of the $(m-1)$th straddler and $s_m \leq D_m$ is the start time of the mth straddler). Unfortunately, the trimming part of [15] may fail when applied to such a block of jobs: it can only work if the jobs removed are the latest ones in the block so that *all* the previous ones can be shifted later and meet their release date. On the other hand, if all release dates of the jobs in the block occur *before* D_{m-1}, then the whole procedure works as for GPA, without any problems. The two key observations for deriving a PTAS for our problem are the following:

Obs1 We can break the problem into a *constant* number of subproblems that schedule disjoint subsets of jobs: we are going to have one subproblem for each set $R_i, i = 1, 2, \ldots, k$.

Obs2 The subproblem for R_i consists of at most k bins $\hat{B}^i_i, \hat{B}^i_{i+1}, \ldots, \hat{B}^i_k$, each being a "slice" of $\hat{K}_i, \hat{K}_{i+1}, \ldots, \hat{K}_k$ (recall that in each $\hat{K}_m$, jobs in R_g are scheduled before jobs in R_h when $g < h$). Note that the release dates of jobs in R_i do not affect the scheduling of bins $\hat{B}^i_{i+1}, \ldots, \hat{B}^i_k$, since these release dates occur before D_i and all those intervals start after D_i; for these bins the PTAS of [15] for GAP is applicable, when the bin capacities $|\hat{B}^i_{i+1}|, \ldots, |\hat{B}^i_k|$ are given. This leaves bin $\hat{B}^i_i$, which is scheduled using the minimum total processing time idea of [17], while respecting the total processing time and weight targets of the rest of the bins and so that the release dates of R_i are respected.

Obs1 above together with the total processing times, $|\hat{B}^i_i|, \ldots, |\hat{B}^i_k|$, for the bins for each R_i, allows us to combine all "parallel" subproblems in a common DP state, computed by k "parallel" applications of Obs2.

3.3.2 Scheduling of the jobs

Recall that we have "guessed" the large jobs X^i_m that have already been slotted for each bin $\hat{B}^i_m$. We consider jobs one-by-one for scheduling in a Latest Release Date (LRD) ordering. If large and small jobs have the same release date, the large jobs are scheduled first. Recall that Lemma 3 allowed us to "guess" bin sizes $|\hat{B}^i_m|$. Let $|\hat{B}^i_m|' := |\hat{B}^i_m| - p(X^i_m)$ be the bin space available for jobs small for $\hat{B}^i_m$, as defined in Definition 2. Note that given the (guessed) $|\hat{B}^i_m|'$ and W^i_m the algorithm can determine whether a job j is small for bin $\hat{B}^i_m$.

Resource augmentation. We increase the bin capacities allocated to small jobs $|\hat{B}^i_m|'$ in each bin $\hat{B}^i_m, m > i$, and set $|\hat{B}^i_m|'' := (1+\varepsilon_3)|\hat{B}^i_m|'$, and round the job processing times according to the resource augmentation scheme of [15] (cf. Appendix D.1). The bin capacity allocated to the large jobs of the bin is not augmented. Now small jobs may have different processing times for different bins; let p_{mj} be the processing time of a small job $j \in R_i$ for bin $\hat{B}^i_m$ ($p_{mj} = \infty$ if j cannot be scheduled in $\hat{B}^i_m$, e.g., when it is not small for this bin, according to definition 2). Let $\widehat{OPT}_a$ be the maximum throughput schedule for the resource-augmented instance. The following lemma is a direct corollary of Lemma 1 in [15].

▶ **Lemma 5.** *Schedule $\widehat{OPT}$ is feasible for the resource-augmented instance, hence $w(\widehat{OPT}_a) \geq w(\widehat{OPT})$.*

DP states and state transitions. Recall that jobs are scheduled one by one, in LRD order. The DP table records whether a given state is feasible. It is initialized with value ∞, indicating infeasibility, everywhere except for base-case-state $(0, 0, \dots, 0)$ that is initialized to value 1, indicating feasibility. Table entries that cannot be reached by legitimate state transitions as described below will remain infeasible throughout the execution.

Assume that the first $j-1$ jobs have been considered, and we are now considering job $j \in R_i$. For brevity, we present the portion (substate) $\mathcal{S}^i(j-1)$ of the DP state $\mathcal{S}(j-1) = (\mathcal{S}^1(j-1), \mathcal{S}^2(j-1), \dots, \mathcal{S}^k(j-1))$ that corresponds to the i-th "parallel" subproblem, defined for jobs in R_i. It has the form of the following tuple:

$$\mathcal{S}^i(j-1) = (Z^{j-1}_i, W^{j-1}_i, Z^{j-1}_{i+1}, W^{j-1}_{i+1}, \dots, Z^{j-1}_k, W^{j-1}_k),$$

where Z^{j-1}_m is the processing time used by the jobs in $\{1, 2, \dots, j-1\} \cap R_i$ scheduled in $\hat{B}^i_m$, and W^{j-1}_m is their total weight. Scheduling job $j \in R_i$ produces the following transitions $\mathcal{S}(j-1) \to \mathcal{S}(j)$:

- Large job $j \in X^i_m$ is scheduled in $\hat{B}^i_m$, $m \geq i$: There is a transition to state $\mathcal{S}(j)$ with

  $$\mathcal{S}^i(j) = (Z^{j-1}_i, W^{j-1}_i, \dots, Z^{j-1}_m + p_{mj}, W^{j-1}_m + w_j, \dots, Z^{j-1}_k, W^{j-1}_k).$$

- Job j is scheduled as small in a $\hat{B}^i_m$, $m > i$: We check whether the following conditions hold: (i) d_j is not violated, (ii) large jobs in X^i_m with release dates smaller than r_j can still be feasibly scheduled in $\hat{B}^i_m$, (iii) $Z^{j-1}_m + p_{mj} \leq |\hat{B}^i_m|''$, and (iv) there isn't already a feasible state $\mathcal{T}(j-1)$ with

  $$\mathcal{T}^i(j-1) = (T, W^{j-1}_i, \dots, Z^{j-1}_m + p_{mj}, W^{j-1}_m + w_j, \dots, Z^{j-1}_k, W^{j-1}_k)$$

  such that $T < Z^{j-1}_i$. Note that large jobs with release dates at least r_j have already been scheduled before j, so given Z^i_m it is easy to check condition (iii) above. If all conditions hold, then there is a transition to state $\mathcal{S}(j)$ with

  $$\mathcal{S}^i(j) = (Z^{j-1}_i, W^{j-1}_i, \dots, Z^{j-1}_m + p_{mj}, W^{j-1}_m + w_j, \dots, Z^{j-1}_k, W^{j-1}_k),$$

  otherwise j is not scheduled in $\hat{B}^i_m$.

- Job j is scheduled as small in $\hat{B}^i_i$: We check whether the following conditions hold: (i) large jobs in X^i_i with release dates smaller than r_j can still be feasibly scheduled in LRD order in $\hat{B}^i_i$, (ii) $Z^{j-1}_i + p_{ij} \leq |\hat{B}^i_i|''$, and (iii) the scheduling of j doesn't violate r_j. If all conditions hold, then we consider the state $\mathcal{T}(j-1)$ with

  $$\mathcal{T}^i(j-1) = (Z, W^{j-1}_i + w_j, Z^{j-1}_{i+1}, W^{j-1}_{i+1}, \dots, Z^{j-1}_k, W^{j-1}_k)$$

  that, after scheduling the $j-1$ first jobs, achieves total weight $W^{j-1}_i + w_j$ in $\hat{B}^i_i$, while the rest of the state is the same as $\mathcal{S}(j-1)$. If state $\mathcal{T}(j-1)$ is not feasible, then we schedule

j in $\hat{B}_i^i$ as in the previous case. Otherwise, Z is well defined as a sum of processing times. Then there is a transition to state $\mathcal{S}(j)$ with

$$\mathcal{S}^i(j) = \begin{cases} (Z_i^{j-1} + p_{ij}, W_i^{j-1} + w_j, Z_{i+1}^{j-1}, W_{i+1}^{j-1}, \ldots, Z_k^{j-1}, W_k^{j-1}), & Z_i^{j-1} + p_{ij} < Z \\ \mathcal{T}^i(j-1), & Z_i^{j-1} + p_{ij} \geq Z. \end{cases} \tag{6}$$

- Job j is not scheduled: In this case $\mathcal{S}(j) = \mathcal{S}(j-1)$.

After the feasibility of all DP states $\mathcal{S}(j) = (\mathcal{S}^1(j), \mathcal{S}^2(j), \ldots, \mathcal{S}^k(j))$ has been computed for all j, we keep as a solution the feasible state $\mathcal{S}(n)$ with the maximum sum of total weights in all bins, as well as the schedule S that achieves it.

▶ **Lemma 6.** *The DP algorithm correctly outputs a feasible schedule S that is as good as $\widehat{OPT}_a$, if the latter exists.*

Proof. First note that $\widehat{OPT}_a$ complies with the structure of Lemma 4. Let $\widehat{OPT}_a(j)$ be the sub-schedule of $\widehat{OPT}_a$ containing only jobs $1, \ldots, j$ in the LRD order. Note that $\widehat{OPT}_a(j)$ continues to comply with the structure of Lemma 4. Let $\bar{Z}_m^i(j), \bar{W}_m^i(j)$ be the total processing time and total weight scheduled in $\hat{B}_m^i$ by $\widehat{OPT}_a(j)$.

▷ Claim 7. For $j = 0, 1, \ldots, n$, there is a feasible state $\mathcal{S}(j)$ with substates

$$\mathcal{S}^i(j) = (Z_i^i(j), W_i^i(j), Z_{i+1}^i(j), W_{i+1}^i(j), \ldots, Z_k^i(j), W_k^i(j)),\ i = 1, \ldots, k$$

corresponding to jobs in R_i, produced by the DP after considering jobs $1, 2, \ldots, j$, such that

$$Z_m^i(j) = \bar{Z}_m^i(j), W_m^i(j) = \bar{W}_m^i(j),\ \forall i, m : m > i, \ \text{ and } \ Z_i^i(j) \leq \bar{Z}_i^i(j), W_i^i(j) = \bar{W}_i^i(j),\ \forall i.$$

Proof. The proof is by induction on j. For the base case $j = 0$, the feasible DP state $\mathcal{S}(0) = (0, 0, \ldots, 0)$ proves the claim trivially true. We assume that the claim is true up to job $j-1$, and we consider the case $j \in R_i$. Let $\mathcal{S}(j-1)$ be the state which by the inductive hypothesis corresponds to $\widehat{OPT}_a(j-1)$. There are three cases to consider.

If j is not scheduled in the transition from $\widehat{OPT}_a(j-1)$ to $\widehat{OPT}_a(j)$ then setting $\mathcal{S}(j) = \mathcal{S}(j-1)$ satisfies the claim.

If $\widehat{OPT}_a(j)$ is obtained from $\widehat{OPT}_a(j-1)$ by scheduling j in $\hat{B}_m^i$ for some $m > i$, then the transition from the state $\mathcal{S}(j-1)$ to a state $\mathcal{S}(j)$ with the same placement of j as $\widehat{OPT}_a(j-1)$ is feasible, since it is feasible for $\widehat{OPT}_a(j)$, and $\mathcal{S}(j)$ satisfies the claim.

If $\widehat{OPT}_a(j)$ is obtained from $\widehat{OPT}_a(j-1)$ by scheduling j in $\hat{B}_i^i$, then the transition from state $\mathcal{S}(j-1)$ of the inductive hypothesis to a state $\mathcal{S}(j)$ when the same placement of j as in $\widehat{OPT}_a(j-1)$ is tried, can follow one of two possibilities: (i) If there is another feasible state $\mathcal{T}(j-1)$ with Z_i^i value T smaller than $Z_i^i(j-1) + p_j \leq \bar{Z}_i^i(j-1) + p_j$, W_i^i value U equal to $W_i^i(j-1) + w_j$, and all other values equal to the values of $\mathcal{S}(j-1)$, then the transition followed by the DP (bottom branch of (6)) sets $\mathcal{S}(j) = \mathcal{T}(j-1)$ (which means that j is not scheduled in $\mathcal{S}(j)$), and the claim is satisfied. (ii) If there is no such state $\mathcal{T}(j-1)$, then by the inductive hypothesis $Z_i^i(j-1) \leq \bar{Z}_i^i(j-1)$ which implies that $Z_i^i(j-1) + p_j \leq \bar{Z}_i^i(j-1) + p_j$. In addition, the transition of the DP for this case (top branch of (6)) is feasible, since it was feasible for $\widehat{OPT}_a(j-1)$, and the new state $\mathcal{S}(j)$ it produces again satisfies the claim. ◁

The claim proves that there is a path of feasible states of the DP that achieves the same total weight as $\widehat{OPT}_a(j)$ for all j, and, therefore, there is a feasible state $\mathcal{S}(n)$ of total weight equal to $w(\widehat{OPT}_a(n)) = w(\widehat{OPT}_a)$. ◀

Trimming. Schedule S corresponds to scheduling in a resource-augmented set of bins $\hat{B}^i_m, m > i$. Following [15], we apply trimming (cf. Appendix D.2) to each one of them with $\gamma := 1 + \varepsilon_3$, $\delta := \varepsilon_3, \varepsilon := \varepsilon_3$, for a total loss of at most $2\varepsilon_3\widehat{OPT}_a$ total weight. The resulting schedule is feasible for the original problem without speed-up, and its total weight is at least $(1 - \varepsilon)OPT$, where $\varepsilon \leq 1 - (1 - \varepsilon_1)(1 - 3\varepsilon_2)(1 - 3\varepsilon_3)$.

Running time. Let us focus first on the subproblem for a fixed R_i. There are $C := O(n^{O(k/\varepsilon^2)}(\log_{1+\varepsilon} D_k)^k)$ choices for all the large sets and the bin sizes. For each choice we run the DP for resource-augmented bin sizes, with rounded weight values, which gives a total of $O((n^2/\varepsilon^2)^k \cdot C)$ choices for the coordinates, except for the first one, of the substate $\mathcal{S}^i()$. By Claim 7, for every such combination of coordinates we keep one substate. To place job $j \in R_i$ we examine $O(k)$ transitions. Taking into account that there are k sets $R_1, \dots, R_k$, we obtain a total running time of $O((n \log D_k)^{O(k^2/\varepsilon^2)})$.

▶ **Theorem 8.** *There is a PTAS for the maximum weighted throughput problem on a single machine with a constant number of distinct due dates.*

Theorem 8 is used in part (ii) of Corollary 1.

4 Constant number of distinct release dates

The ideas behind Theorem 8 can easily be applied to provide a PTAS for the case of a constant number of release dates. Let $d_{\max} = \max_{j \in \mathcal{J}} d_j$. The role of due dates is now played by the k different release dates $0 = r_1 \leq r_2 \leq \ldots \leq r_k$, that define k intervals $[r_i, r_{i+1})$, $i \in [1, k-1]$, and $[r_k, d_{\max})$. Note that release dates are now ordered in non-decreasing order. Let S_i be the straddler of r_i, $i \in [2, k]$ and $S_1 = \emptyset$.

After the rounding of the job weights, we consider an optimal schedule OPT where jobs are now shifted as *early* as possible. Intervals $K_m = (f_m, s_{m+1})$ between straddlers S_m, S_{m+1} are defined as before, and Lemma 3 applies to show the existence of a subset of the scheduled jobs of K_m that can now be scheduled in (smaller) intervals $\hat{K}_m$ starting also at f_m and with polynomially many possible sizes. This is ensured by Lemma 3, modified to apply to bins straddling only due dates, and by scheduling jobs within the bins in Earliest Due Date (EDD) order and as early as possible. When empty space is created within a bin, we shift jobs to the left on the time axis as needed so as to eliminate idle time. The only additional change to Lemma 4 is that now jobs within each interval are scheduled in EDD order.

The PTAS description is virtually the same, with the role of R_i played by D_i, the set of jobs with due dates in $[r_i, r_{i+1})$. Bins $\hat{B}^i_m$, $i \in [m, k]$, of $\hat{K}_m$ are defined similarly to Section 3.3, where $\hat{B}^i_m$ contains jobs scheduled in $\hat{K}_m$ with due date in $\hat{K}_i$. These bins appear from left to right in $\hat{K}_m$, i.e., in order of increasing i. The DP transitions are also defined similarly, to always respect (i) the due dates, (ii) the bin capacities, and (iii) the invariant that the jobs from D_i that are scheduled in $\hat{B}^i_i$ occupy the minimum processing time, *given* the processing time target for bins $\hat{B}^i_m$, $m \in [1, i)$, and the total weight targets for bins $\hat{B}^i_m$, $m \in [1, i]$. By the latter invariant, Lemma 6 applies also to this case. Hence we have:

▶ **Theorem 9.** *There is a PTAS for the maximum weighted throughput problem on a single machine with a constant number of distinct release dates.*

Theorem 9 is used in part (ii) of Corollary 1.

References

1 P. Baptiste. Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine with equal processing times. *Journal of Scheduling*, 2(6):245–252, 1999.

2 A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. *J. ACM*, 48(5):1069–1090, 2001.

3 A. Bar-Noy, S. Guha, J. Naor, and B. Schieber. Approximating the throughput of multiple machines in real-time scheduling. *SIAM J. Comput.*, 31(2):331–352, 2001.

4 P. Berman and B. DasGupta. Improvements in throughout maximization for real-time scheduling. In *32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 680–687. ACM, 2000.

5 M. Böhm, N. Megow, and J. Schlöter. Throughput scheduling with equal additive laxity. *Oper. Res. Lett.*, 50(5):463–469, 2022.

6 J. Chuzhoy, R. Ostrovsky, and Y. Rabani. Approximation algorithms for the job interval selection problem and related scheduling problems. *Math. Oper. Res.*, 31(4):730–738, 2006.

7 F. Eberle, N. Megow, and K. Schewior. Online throughput maximization on unrelated machines: Commitment is no burden. *ACM Trans. Algorithms*, 19(1):10:1–10:25, 2023.

8 L. Fleischer, M. X. Goemans, V. S. Mirrokni, and M. Sviridenko. Tight approximation algorithms for maximum separable assignment problems. *Math. Oper. Res.*, 36(3):416–431, 2011.

9 R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan. Dependent rounding and its applications to approximation algorithms. *J. ACM*, 53(3):324–360, 2006.

10 M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

11 D. Hyatt-Denesik, M. Rahgoshay, and M. R. Salavatipour. Approximations for throughput maximization. In *31st Int. Symp. on Algorithms and Computation, ISAAC 2020*, volume 181 of *LIPIcs*, pages 11:1–11:17, 2020.

12 S. Im, S. Li, and B. Moseley. Breaking the $1 - 1/e$ barrier for nonpreemptive throughput maximization. *SIAM J. Discret. Math.*, 34(3):1649–1669, 2020.

13 B. Kalyanasundaram and K. Pruhs. Eliminating migration in multi-processor scheduling. *J. Algorithms*, 38(1):2–24, 2001.

14 R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

15 A. Khan, E. Sharma, and K. V. N. Sreenivas. Approximation algorithms for generalized multidimensional knapsack. *CoRR*, abs/2102.05854, 2021. arXiv:2102.05854.

16 E. L. Lawler. Recent results in the theory of machine scheduling. In *Mathematical Programming The State of the Art, XIth Int. Symp. on Math. Programming*, pages 202–234. Springer, 1983.

17 E. L. Lawler. A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. *Ann. Oper. Res.*, 26(1–4):125–133, 1990.

18 E. L. Lawler, J. K. Lenstra, A. H. G. Rinooy Kan, and D. B. Shmoys. Sequencing and scheduling: Algorithms and complexity. In S. C. Graves, A. H. G. Rinnooy Kan, and P. H. Zipkin, editors, *Handbooks in Operations Research and Management Science, Vol 4., Logistics of Production and Inventory*, pages 445–522. North Holland, 1993.

19 K. Pruhs and G. J. Woeginger. Approximation schemes for a class of subset selection problems. *Theor. Comput. Sci.*, 382(2):151–156, 2007.

20 A. Schrijver. *Theory of linear and integer programming.* John Wiley and Sons, 1986.

21 A. Srinivasan. Distributions on level-sets with applications to approximation algorithms. In *42nd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 588–597, 2001.

A Technical details for the proof of Theorem 2

We explain how the dependent rounding method of [21, 9] is used in the proof of Theorem 2. The details in this section are a simple application of the work of Srinivasan [21] (see also [9]) and we provide them here for the sake of completeness. Let x^* be a $(\rho - \delta)$-approximate (fractional) solution of (CLP). Without loss of generality we can assume that $\sum_{C \in \mathcal{C}(i)} x^*_C = 1, \quad \forall i \in \mathcal{M}$.

Let $\bar{\mathcal{C}}(i)$ be the configurations in $\mathcal{C}(i)$ with nonzero value in x^*. We define $x^{(i)}$ to be the projection of the x^* vector to the coordinates that correspond to configurations in $\bar{\mathcal{C}}(i)$. Denote $|\bar{\mathcal{C}}(i)|$ by t_i. Srinivasan [21] defines a distribution $D(t_i, x^{(i)})$ over vectors in $\{0,1\}^{t_i}$ such that any vector X sampled from the distribution satisfies the following three properties:

(A1) *(probability preservation)* $\forall C \in \bar{\mathcal{C}}(i), \Pr[X_C = 1] = x^*_C$.

(A2) *(degree preservation)* $\Pr[|\{C \in \bar{\mathcal{C}}(i) \colon X_C = 1\}| = 1] = 1$.

(A3) *(negative correlation)* For all $S \subseteq \bar{\mathcal{C}}(i)$ we have $\Pr[(\bigwedge_{C \in S}(X_C = 0)] \leq \prod_{C \in S} \Pr[X_C = 0]$ and $\Pr[(\bigwedge_{C \in S}(X_C = 1)] \leq \prod_{C \in S} \Pr[X_C = 1]$.

The existence of the distribution is established algorithmically:

▶ **Theorem 10** ([21])**.** *Given the vector $x^{(i)}$ there is a linear-time algorithm that generates a sample from distribution $D(t_i, x^{(i)})$.*

The rounding algorithm follows. It takes as input the vector x^*.

Algorithm 1 DependentRounding.

```
For all $i \in \mathcal{M}$, do independently:
    1. Using the algorithm of Theorem 10, sample from $D(t_i; x^{(i)})$ to obtain vector
       $X^{(i)} \in \{0,1\}^{t_i}$. By Property (A2), $X^{(i)}$ has a unique entry equal to 1.
    2. Assign the configuration $C$ that corresponds to the nonzero entry of $X^{(i)}$ to
       machine $i$.
```

For every $i \in \mathcal{M}$ and $C \in \mathcal{C}(i)$ s.t. $X^{(i)} = 1$, we set $\hat{x}_C = 1$. The remaining entries of $\hat{x}$ are set to zero. The rest of the proof simply adapts the analysis of [21] for Maximum-Coverage-type problems. Since the sets $\bar{\mathcal{C}}(i)$ are pairwise disjoint we slightly abuse notation and omit the machine superscript from the vectors $X^{(i)}$. For $j \in \mathcal{J}$, let z_j be the random variable that takes value 1 if job j is assigned by Algorithm DependentRounding to at least one machine and 0 otherwise. Let $\mathcal{C}$ denote $\bigcup_{i \in \mathcal{M}} \mathcal{C}(i)$.

$$\begin{aligned} \Pr[z_j = 1] &= 1 - \Pr[\bigwedge_{C \in \mathcal{C}: C \ni j} (X_C = 0)] \\ &\geq 1 - \prod_{C \in \mathcal{C}: C \ni j} \Pr[X_C = 0] \qquad (7) \\ &= 1 - \prod_{C \in \mathcal{C}: C \ni j} (1 - x^*_C) \qquad (8) \end{aligned}$$

Inequality (7) follows from the negative correlation property (A3), and equality (8) from property (A1). Define $z^*_j := \sum_{C \in \mathcal{C}: C \ni j} x^*_C$. This is the fractional amount by which job j is scheduled, and the objective value of the solution x^* is equal to $\sum_{j \in \mathcal{J}} w_j z^*_j$. Observe that for every j, $z^*_j \leq 1$. Using calculus we obtain that the expression in (8) is greater than $(1 - 1/e) \cdot z^*_j$.

B Scheduling with job types

In this section we show how the proofs in Section 2 can be extended to accommodate job types. Only jobs with certain types can be scheduled together on each machine and the actual combination of types affects the job characteristics.

The extended problem is defined as follows. Every job j has a *type* t_j from a finite set T of types. For every machine i, we are given a set $E_i \subseteq 2^T$ that specifies the allowed combinations of types that may be processed on i in a feasible schedule. We assume from now that $\sum_{i\in\mathcal{M}}|E_i|$ is bounded by a polynomial in n and m. For machine i, let $E_i = \{e_i(1), \ldots, e_i(k_i)\}$ be the set of allowed combinations. In a feasible schedule, the set J_i assigned to machine i must meet the following two additional constraints. C(i): all jobs in J_i must have a type in $e_i(l)$ for some $e_i(l) \in E_i$. We call this type combination $e_i(l)$ *active* for machine i. C(ii): The jobs in J_i have processing times, release dates, and due dates that depend on the active type $e_i(l)$, and are denoted $p^l_{ij}, r^l_{ij}, d^l_{ij}$ respectively.

We denote by $R|\, r_{ij},\, d_{ij}$, types $|\sum_j w_j\bar{U}_j$ the problem of maximizing weighted throughput on unrelated machines under the additional constraints C(i) and C(ii). Using a ρ-approximate oracle for $1|\, r_j,\, \beta|\sum w_j\bar{U}_j$ we show how to obtain a $((1-1/e)\rho-\varepsilon)$ guarantee for $R|\, r_{ij},\, d_{ij}$,β, types $|\sum_j w_j\bar{U}_j$. We outline only the necessary modifications in the proofs of Section 2.

The definition of a configuration is now as follows. A configuration $C \in \mathcal{C}(i)$ is a schedule of a subset $J \subseteq \mathcal{J}$ of jobs on machine i such that (i) all jobs in J have types from some set $e_i(l) \in E_i$ (ii) job j in J has processing p^l_{ij} and it must be scheduled in the interval $[r^l_{ij}, d^l_{ij}]$ on i (iii) there is no unnecessary idle time (iv) the β-constraints are met.

In the proof of Lemma 1 instead of a single instance I_i for machine i we define k_i instances, one for every element of E_i. Instance $I_i(l)$ is defined on job set $J_i(l) = \{j \in \mathcal{J} \mid j \text{ has type in } e_i(l)\}$. Job j in $J_i(l)$ has processing time, release date, and due date $p^l_{ij}, r^l_{ij}, d^l_{ij}$ respectively. Running algorithm A_β on each instance $I_i(l)$, $l = 1, \ldots, k_i$, will detect whether there is a configuration C in $\mathcal{C}(i)$ with total value more than y_i/ρ.

The rounding algorithm in the proof of Theorem 2 will return on every machine a configuration C s.t. all jobs in C have a type from a set $e_i(l) \in E_i$. The rest of the proof holds as before, including the analogous extensions of Corollaries 1 and 2 to the setting with types.

C FPTAS for similarly ordered release and due dates [17]

When release and due dates are similarly ordered, Lawler ([17], Section 6) observed that there is a non-preemptive EDD (Earliest Due Date) optimal schedule, that can be computed by the following DP:

$$C_j(w) = \min\{C_{j-1}(w), \max\{r_j, C_{j-1}(w-w_j)\} + p_j\}, \tag{9}$$

where $C_j(w)$ is the minimum total scheduling time needed to achieve total weight at least w, by a feasible schedule of the first j jobs in an EDD order. If the right-hand side of (9) violates any d_i, then $C_j(w) = \infty$. While the running time of this DP is $O(n^2 w_{\max})$, it can be transformed into a FPTAS for maximum weighted throughput by rounding the job weights as in the classic knapsack problem.

The FPTAS can be adapted to work with the setting of Section 3, since there is also an optimal schedule that has all its jobs shifted as late as possible such that within each contiguous block of jobs that does not contain a due date, jobs appear in earliest release date order. The correctness of the algorithm is not affected if we construct the optimal schedule in a "towards the past" direction, starting at $d_{\max}$, and examining the jobs in a Latest Release Date (LRD) order.

For the setting of a constant number of distinct release dates (due dates), we exploit the fact that the algorithm of [17] computes the optimal solution when all jobs have a common due date (common release date).

D PTAS for GAP with a constant number of bins [15]

In the Generalized Assignment Problem (GAP), we are given m bins (machines in the terminology of [15]), with bin i having volume capacity B_i, and n jobs with weight w_j for job j, as well as processing time p_{ij} when scheduled in bin i (note that if job j cannot go to machine i, $p_{ij} = \infty$); the goal is to maximize the total weight of scheduled jobs. There are no release or due dates.

Khan et al. [15] presented a PTAS for the case of constant m. They split the jobs into "big" and small according to their processing times, by proving the following structural theorem for any feasible GAP solution:

▶ **Theorem 11** (Theorem 3 in [15]). *Let S be a feasible solution to GAP, and let $S_i \subseteq S$ be the jobs assigned to the i-th bin. Then for all $\varepsilon > 0$, there exist sets X and Y such that $|X| \leq m/\varepsilon^2$ and $w(Y) \leq \varepsilon w(S)$ and*

$$p_{ij} \leq \varepsilon \left(B_i - \sum_{j \in X \cap S_i} p_{ij} \right), \ \forall i, \ \forall j \in S_i - X - Y.$$

Having guessed the set X together with the assignment of its jobs to bins, [15] turn to the scheduling of the rest of the jobs that are small in the sense of Theorem 11. In order to recover total weight at least $(1-\varepsilon)w(S-X)$, the authors in [15] apply *resource augmentation* for processing times (Section D.1), followed by a DP-based PTAS for scheduling the jobs with their rounded processing times, and, lastly, *trimming* (Section D.2) is used in order to remove a lengthy enough set of cheap jobs so as to restore the bin volume capacities to their original values. We proceed to outline how resource augmentation and trimming are defined in [15] and under what conditions they can be used for the throughput problem.

D.1 Resource augmentation

For every bin i, define a scaling factor $\mu_i := \varepsilon B_i/n$. Also, define new bin volume capacity $B'_i := \lfloor B_i/\mu_i \rfloor + n$ and new job processing times $p'_{ij} := \lceil p_{ij}/\mu_i \rceil$. Then there is an optimal solution to maximum throughput that can be calculated by a PTAS, and the bin capacities used are $B'_j \leq (1+\varepsilon)B_j$. Obviously, this optimal solution is at least as good as the optimal solution that can be obtained with the original bin capacities. Khan et al. [15] show that if the items (jobs) have size at most εB_i after trimming the solution as in Section D.2 the bin capacities can be restored to their original values while losing only an ε-fraction of weight (profit).

Resource augmentation can be applied on a maximum throughput instance restricted to an interval I on the time axis only when there are no release or due dates contained in I.

D.2 Trimming

Khan et al. [15] designed the trimming method that follows for a knapsack instance. Suppose that in bin i, there is a schedule of a set S_i of jobs, without due dates or relase dates, s.t. $p(S_i) = \gamma B_i$, with $\gamma \leq 1 + \delta$ and with total weight $w(S_i)$. Moreover, $p_{ij} \in (0, \varepsilon B_i]$ for $j \in S_i$. Then for parameters δ, ε with $\delta \leq \varepsilon$, [15] show how to create empty space of length at least δB_i, without losing more than $((\delta + \varepsilon)/(\gamma - \varepsilon))w(S_j)$ weight, and the scheduling of the remaining jobs takes no more than B_i processing time.

This trimming technique of [15] can be extended to apply to a maximum throughput schedule with different release dates and a common due date (different due dates and a common release date). In this case, all jobs can be shifted as late (as early) as possible, to create a new schedule ending at the due date (starting at the release date) with no idle time. This can be achieved without violating any release (due) dates. We provide the details for the case of the common due date in our Lemma 3.

D.3 Proof of Lemma 3

We provide the missing details for the existence of set R. We adjust the trimming method of [15] in order to account for the existence in our setting of release dates and achieve the parameters we need in the statement of the lemma. In particular let σ' be the feasible schedule of the set $X \cup J'$ in the interval I where for every $j \in J'$, $p_j \leq \varepsilon(|I| - p(X))$. Let $B = |I| - p(X)$. We consider the set S of the subintervals of I that contain idle time or jobs from J'. Their length may vary and they may not be consecutive on the time axis as they may be intermingled with the jobs of X. Clearly the total length of these subintervals equals B. Let $k = \lfloor (1 - \varepsilon)/(2\varepsilon) \rfloor$. Put a blue marker to the leftmost endpoint of a subinterval in S. Moving to the right, sweep along the time axis within the subintervals of S only and put a red marker after length εB, a blue marker after the next length εB and so on. Therefore the markers alternate between red and blue. Proceed until you have placed $k + 1$ red markers and $k + 1$ blue markers. The last marker will be red. Since

$$(k + 1)\varepsilon B + k\varepsilon B = (\varepsilon + 2\varepsilon \lfloor (1 - \varepsilon)/(2\varepsilon) \rfloor)B \leq B$$

this is always possible. Number the markers from 0 to $2k + 1$ from left to right. Consider the set of jobs S_i, $i \in \{0, 1, \dots, k\}$ that are scheduled to a non-zero extent after blue marker $2i$ and up to red marker $2i + 1$. These sets are pairwise disjoint. Let $S_i^* = \arg\min_{i=0}^{k} w(S_i)$. It follows that $w(S_{i^*}) \leq \frac{1}{k+1} \sum_{i=0}^{k} w(S_i) \leq (2\varepsilon)/(1 - \varepsilon)w(J')$. Removing $R := S_{i^*}$ will leave empty space of εB. We have that $p(J' - R) \leq (1 - \varepsilon)B$. Define the integer $\beta \geq 0$ such that $(1 + \varepsilon)^{\beta} \leq B < (1 + \varepsilon)^{\beta + 1}$. Then $p(J' - R) \leq (1 - \varepsilon^2)(1 + \varepsilon)^{\beta}$.

Facility Location in the Sublinear Geometric Model

Morteza Monemizadeh ✉
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Abstract

In the *sublinear geometric model*, we are provided with an oracle access to a point set P of n points in a bounded discrete space $[\Delta]^2$, where $\Delta = n^{O(1)}$ is a polynomially bounded number in n. That is, we do not have direct access to the points, but we can make certain types of queries and there is an oracle that responds to our queries. The type of queries that we assume we can make in this paper, are range counting queries where ranges are axis-aligned rectangles (that are basic primitives in database [36, 11, 17], computational geometry [1, 2, 6, 5], and machine learning [35, 31, 29, 28]). The oracle then answers these queries by returning the number of points that are in queried ranges. Let Alg be an algorithm that (exactly or approximately) solves a problem $\mathcal{P}$ in the sublinear geometric model. The query complexity of Alg is measured in terms of the number of queries that Alg makes to solve $\mathcal{P}$. In this paper, we study the complexity of the (uniform) Euclidean facility location problem in the sublinear geometric model. We develop a randomized sublinear algorithm that with high probability, $(1+\epsilon)$-approximates the cost of the Euclidean facility location problem of the point set P in the sublinear geometric model using $\tilde{O}(\sqrt{n})$ range counting queries. We complement this result by showing that approximating the cost of the Euclidean facility location problem within $o(\log(n))$-factor in the sublinear geometric model using the sampling strategy that we propose for our sublinear algorithm needs $\tilde{\Omega}(n^{1/4})$ RangeCount queries. We leave it as an open problem whether such a polynomial lower bound on the number of RangeCount queries exists for any randomized sublinear algorithm that approximates the cost of the facility location problem within a constant factor.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Facility Location, Sublinear Geometric Model, Range Counting Queries

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.6

Category APPROX

1 Introduction

In the *sublinear query model*, we consider scenarios in which we do not have direct access to the underlying data, but there is an oracle or a prophet who provides us *metadata* about the data that we are interested in. An interesting real-world application of such scenarios is for retail or ride hailing companies that have millions of customers. Often these giant companies do outsourcing by hiring a party outside a company to perform temporary services to reduce costs or optimize customers buying experience. Due to privacy concerns[1], these companies do not want to provide accurate and detailed information of their customers to the outsourced companies. However, they can provide some sort of metadata (such as how many customers are in an area or a neighborhood or how often customers in a town purchase particular goods or use a service) that can help the outsourced company to perform its service.

[1] See "Threat to Security and Confidentiality" in `https://www.thebalancesmb.com/top-outsourcing-disadvantages-2533777` and "Security and Privacy Concerns" in `https://www.truppglobal.com/blog/top-10-outsourcing-problems-and-how-to-solve-them#5-security-and-privacy-concerns`

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 6; pp. 6:1–6:24
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

As a concrete example, let us consider a retail company that plans to open new stores or lockers to provide fast and responsive support for its customers. To this end, the company hires an (experienced) outsourced company to estimate the total cost of opening new stores or lockers. Due to privacy concerns, the company does not want to provide information about its customers, but it can provide aggregate data such as how many customers are in an area or a neighborhood. An interesting question is how many times these aggregate data should be provided so that the outsourced company can estimate a fairly accurate cost of opening new stores or lockers.

Motivated by these applications, we study the (uniform) Euclidean facility location in the *sublinear geometric model* which is a sublinear model suitable when the underlying data is a point set in a d-dimensional Euclidean space. In particular, we seek to design sublinear algorithms that approximate the cost of the Euclidean facility location problem when we do not have access to the underlying point set, but instead, we can make queries and there is an oracle who will respond and provide solutions to our queries. Next, we define the facility location problem formally.

▶ **Definition 1** ((uniform) Euclidean facility location)**.** *In the* (uniform) Euclidean facility location *problem, we are given a point set P of size n in a bounded discrete space $[\Delta]^2$, where $\Delta = n^{O(1)}$ is a polynomially bounded number in n and an opening cost $f > 0$, and the goal is to compute a set $F^* \subset [\Delta]^2$ of facilities that minimizes the cost function $cost_{FL}(P, F^*) = \sum_{p \in P} \operatorname{dist}(p, F^*) + f \cdot |F|$, where $dist(p, F^*) = \min_{q \in F^*} dist(p, q)$ is the Euclidean distance of p to its nearest facility in F^*. We denote the optimal facility location cost of the point set P by $OPT_{FL}(P, f)$.*

1.1 Sublinear geometric model

Let $P \subseteq [\Delta]^2$ be a point set of size n in a 2-dimensional discrete space $[\Delta]^2 = \{1, 2, 3, \cdots, \Delta\}^2$, where $\Delta = n^{O(1)}$ is a polynomially bounded number in n. For the simplicity of exposition, we assume that $\Delta = n$. This means that the space $[\Delta]^2$ is indeed, the space $[n]^2$. We also assume that every unit square (of side length one), can store at most one (weighted) point [2].

In the *sublinear geometric model*, we assume we do not have access to points of P directly, but instead, we have *query access* to points of P. That is, we can make certain queries with respect to point set P and there is an oracle that returns the solutions to our queries. The type of queries that we assume are provided in the sublinear geometric model are *range counting queries* where ranges are *axis-aligned rectangles* (that are basic primitives in database [36, 11, 17], computational geometry [1, 2, 6, 5], and machine learning [35, 31, 29, 28]). The oracle then answers these queries by returning the number of points in queried ranges.

Range counting queries. We assume that we have a query access to P where we query an axis-aligned rectangle $c \subseteq [n]^2$ and the oracle returns the number of points $n_c = |P \cap c|$ that are in rectangle c. We denote such a query by $\textsc{RangeCount}(c)$. Let $\textsc{Alg}$ be an algorithm that (exactly or approximately) solves a problem $\mathcal{P}$ in the sublinear geometric model. The query complexity of $\textsc{Alg}$ is measured in terms of the number of $\textsc{RangeCount}(c)$ queries that $\textsc{Alg}$ makes to solve $\mathcal{P}$. Assume that the input of problem $\mathcal{P}$ is a point set P of size n. If the number of $\textsc{RangeCount}(c)$ queries that we ask is $o(n)$, we say the query complexity of problem $\mathcal{P}$ is sublinear in terms of its input set.

[2] We make these explicit assumptions to simplify the notations in this paper.

1.2 Our Contribution

Here, we state our main result.

▶ **Theorem 2** (Facility location in sublinear geometric model). *Let P be a point set of size n in a discrete space $[n]^2$. Let $f > 0$ be the opening cost for the facility location problem and $0 < \epsilon \leq 1$ be the error parameter. Then, there exists a randomized sublinear algorithm that (w.h.p.) returns an $(1+\epsilon)$-estimator for the facility location cost of P in the sublinear geometric model using $\tilde{O}(\sqrt{n})$ RangeCount queries.*

We complement this result by showing that approximating the cost of the Euclidean facility location problem within $o(\log(n))$-factor in the sublinear geometric model using the sampling strategy that we propose for Theorem 2 needs $\tilde{\Omega}(n^{1/4})$ RangeCount queries. See Figure 2 in Section 4 for the illustration of the hard instance that we explain there. We leave it as an open problem whether such a polynomial lower bound on the number of RangeCount queries exists for any randomized sublinear algorithm that approximates the cost of the facility location problem within a constant factor.

▶ **Lemma 3** (Lower bound). *There exists a facility location instance for which the sublinear algorithm of Theorem 2 needs $\Omega(\frac{\sqrt[4]{n}}{\log n})$ RangeCount queries to estimate the cost of facility location within $o(\log n)$-factor in the sublinear geometric model.*

Outline of the proof of Theorem 2. Our starting point to prove this theorem is to choose a known polynomial-time approximation scheme (PTAS) for the Euclidean facility location problem in the plane. Our goal would be to simulate such a PTAS in sublinear time to obtain an estimator for the optimal cost of the facility location problem.

Known PTAS algorithms [7, 30, 15] are often based on partitioning the underlying space of a point set. Two such PTAS algorithms are known. The first one was proposed by Arora, Raghavan, and Rao [7] and later improved by Kolliopoulos and Rao [30] that partitions the space into regions and then combine solutions of small regions using the dynamic programming to obtain solutions for the bigger regions. The issue with extracting an estimator based on this PTAS is that simulating the dynamic programming in sublinear time seems to be hard.

The second PTAS for this problem was proposed by Czumaj, Lammersen, Monemizadeh, and Sohler [15] where they partition the underlying space $[n]^2$ of a point set P into a set Λ of cells in a way that they can solve the facility location instance inside each cell independent of the facility location instances of the other cells. The independency that is provided by this algorithm is a good choice for us. However, the problem with using this approach (in order to develop an estimator) in the geometric sublinear model is we do not have access to Λ.

Overall Idea: We develop a sublinear algorithm that randomly samples cells of a number of grids (of exponentially increasing granularity) that we impose on the discrete space $[n]^2$ and determines whether the sampled cells could be cells in the set Λ or not. For those sampled cells that are in fact, cells in Λ, we solve the facility location instance independently and compute a $(1+\epsilon)$-approximation of their facility location costs. We then multiply their costs with proper weights to $(1+\epsilon)$-estimate the optimal facility location cost $OPT_{FL}(P, f)$.

Next we briefly describe the space partitioning and the PTAS that are developed in [15] and then, we explain our sublinear algorithm in detail.

PTAS of [15]. Suppose we are given a point set $P \subset [n]^2$ of size n. We impose a (nested) grid set $\mathcal{G}_{\log(n)+1}, \mathcal{G}_{\log(n)}, \cdots, \mathcal{G}_1, \mathcal{G}_0$ on the space $[n]^2$ where the grid $\mathcal{G}_i$ consists of cells of side length 2^i. We then randomly shift the border lines of the grids $\mathcal{G}_{\log(n)+1}, \mathcal{G}_{\log(n)}, \cdots, \mathcal{G}_1, \mathcal{G}_0$ as follows. We choose two random real numbers $v_1, v_2 \in [0, n]$. Then, we shift every border line ℓ of every grid $\mathcal{G}_i$ using the random vector $v = (v_1, v_2)$. Observe that after this random shifting process, the point set P is enclosed in a rectangle of side length $[2n]^2$. In the sublinear geometric model, whenever we make range queries for ranges that are axis-aligned rectangles, we first shift these rectangles using v and then query them.

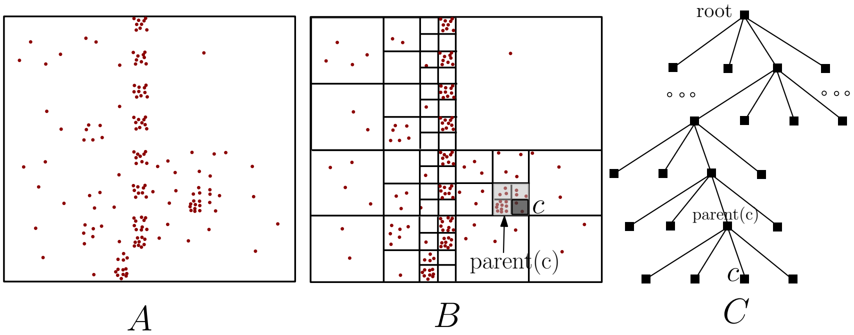


Figure 1 In the sub-figure A, a point set $P \subseteq [\Delta]^2$ is given. The sub-figure B illustrates the space partitioning Λ that is computed by the PTAS of [15]. A light cell c (dark gray square) that is in the partition set Λ and its parent *parent*(c) (gray square) are shown. In the sub-figure C, the nodes that correspond to c and *parent*(c) are also shown in the corresponding quadtree partitioning.

The PTAS [15] is based on partitioning heavy cells and detecting light cells, two notions that we define next. Let c be an arbitrary cell in a grid $\mathcal{G}_i$. We use the 3-approximation algorithm due to Jain and Vazirani [26] (who developed it for facility location) to determine if c is heavy or light. Let us call this algorithm $\textsc{Alg}^3_{FL}$. We say an arbitrary c in a grid $\mathcal{G}_i$ is a *heavy cell* if $\textsc{Alg}^3_{FL}(P \cap c)$ outputs that the facility location cost of $P \cap c$ is at least $\delta_{FL} \cdot f$ where $\delta_{FL} = O(\epsilon^{-2} \log^2 n)$. If the reported cost is less than $\delta_{FL} \cdot f$, c is a *light* cell.

Starting with the square $[2n]^2$, if a cell $c \in \mathcal{G}_i$ is heavy, we then split it into its 4 children c_1, c_2, c_3, c_4 (that are in the grid $\mathcal{G}_{i-1}$) of equal side length and recurse for those children that are heavy. At the end of this recursive algorithm, we let Λ be the set of all light cells that are constructed in this way. Interestingly, for every cell c, we can solve the facility location instance of the point subset $P \cap c$ independently. Later, we combine the solutions of these independent instances to obtain a $(1+\epsilon)$-approximate solution of the optimal facility location of P. See Figure 1 for an illustration of Λ.

Sublinear algorithm. Now, we outline our sublinear algorithm. This algorithm samples cells of the grids $\mathcal{G}_{\log(n)+1}, \mathcal{G}_{\log(n)}, \cdots, \mathcal{G}_1, \mathcal{G}_0$ randomly and determine whether the sampled cells can be in set Λ or not. If we sample enough cells from Λ uniformly at random, compute a $(1+\epsilon)$-approximation of their costs and multiply their costs with proper weights, we can approximate the optimal facility location cost $OPT_{FL}(P, f)$ within $(1+\epsilon)$-factor. However, this is not an easy task in the sublinear geometric model as we have the following challenges:

- **Query Access:** Given a cell c in a grid $\mathcal{G}_i$:
  - We need to determine if c is heavy or light using $\textsc{Alg}^3_{FL}(P \cap c)$.
  - If we know that we have sampled a cell $c \in \Lambda$, we need to compute a $(1+\epsilon)$-approximation of its cost using a PTAS algorithm, say [15].

  However, we do not have direct access to the points in $P \cap c$. We can only make range counting queries.
- **Many noisy empty cells:** In any grid $\mathcal{G}_i$ for $i \leq \frac{1}{2} \cdot \log(n)$, we have $\Omega(n)$ cells and plenty of them could be empty. Recall that $|P| = n$. Thus, a uniform random sampling may need to sample many cells to hit a cell from Λ.

We first develop a tester algorithm (so-called HeavyTester) that using $\mathrm{poly}(\epsilon^{-1} \cdot \log(n))$ RangeCount queries determines if a cell c in a grid $\mathcal{G}_i$ is a heavy or a light cell. In case that c is a light cell, the tester returns the facilities that $(1+\epsilon)$-approximates the optimal facility location cost of the cell c. This already resolves the first challenge. Next, we deal with the second challenge. We first develop a sublinear algorithm (so-called $\sqrt{n}$-Estimator) that using $\tilde{O}(\sqrt{n})$ RangeCount queries distinguishes between the following two cases:

- **Low cost instances:** The first case is if the optimal cost of P is upper-bounded by $O(\sqrt{n}f)$. If this is indeed the case, the algorithm $\sqrt{n}$-Estimator solves the facility location instance of P and returns $(1+\epsilon)$-approximate solution (not the cost) of P.
- **High cost instances:** The second case is if the optimal cost of P is $\Omega(\sqrt{n}f)$. For this case, the algorithm $\sqrt{n}$-Estimator outputs that the cost of P is $\Omega(\sqrt{n}f)$.

From now on, we assume that the optimal facility location cost of P is $\Omega(\sqrt{n}f)$.

We develop a **telescoping sampling** that samples any arbitrary cell c in a grid $\mathcal{G}_i$ with probability $\mathbf{Pr}[c] = \frac{n_{parent(c)}}{n}$, where $parent(c)$ is the parent of c in the grid $\mathcal{G}_{i+1}$ and $n_{parent(c)} = |parent(c) \cap P|$ is the number of points in $parent(c)$ (See Figure 1 for the illustration of a light cell and its parent.) The reason that we do the telescoping sampling based on $n_{parent(c)}$ not $n_c = |c \cap P|$ is that the cells in Λ are light, but their parents (and ancestors) are heavy, so heavy parents have a minimum number of points that helps us to use known concentration bounds [3, 32] to analyze the telescoping sampling.

For a grid $\mathcal{G}_i$, we implement the telescoping sampling as follows. We start with the unique cell c' in grid $\mathcal{G}_{\log(n)+1}$ and split it into 4 sub-cells c_1, c_2, c_3, c_4 and query the number of points inside each of them. Then, we sample the sub-cell c_j for $j \in [4]$ with probability $\frac{n_{c_j}}{n_{c'}}$. Suppose we sample c_1. We let $c' = c_1$ and recursively repeat the same process till we end up with the parent of a cell $c \in \mathcal{G}_i$. By the telescoping argument, a cell $c \in \mathcal{G}_i$ is sampled with probability $\mathbf{Pr}[c] = \frac{n_{parent(c)}}{n}$.

Once we sample a cell c in a grid $\mathcal{G}_i$ with probability $\mathbf{Pr}[c] = \frac{n_{parent(c)}}{n}$, we test if c is a heavy or a light cell using algorithm $\textsc{HeavyTester}(c)$. If c is heavy or both c and $parent(c)$ are light, we do not do anything. If c is a light cell and its parent is a heavy cell, the cell c must be in Λ. In this case, the tester returns set F_c of facilities that $(1+\epsilon)$-approximates the optimal facility location cost of the instance $c \cap P$. Let $\mathrm{cost}^{\epsilon}_{FL}(c, F_c, f) = \sum_{p \in P \cap c} \mathrm{dist}(p, F_c) + |F_c| \cdot f$ be the facility location cost of c with respect to facility set F_c.

Next, we would like to assign a weight to the sampled cell c that is in Λ. For that, one option would be to multiply $\mathrm{cost}^{\epsilon}_{FL}(c, F_c, f)$ by the term $\frac{n}{n_{parent(c)}}$. However, this may not be possible. The problem is $n_{parent(c)}$ might be very small and so, the weight $\frac{n}{n_{parent(c)}}$ may be $\Omega(n)$ which essentially means if we want to use known concentration bounds such as the Hoeffding bound [22], we need $\Omega(n)$ samples to approximate the optimal cost $OPT_{FL}(P, c)$.

We develop a novel sampling technique that samples (almost uniformly at random) those light cells in a grid $\mathcal{G}_i$ whose parents have roughly the same number of points. To this end, we first observe that from set Λ, we only need to consider those cells whose cost is at least

τf where $\tau = O(\frac{\epsilon}{\log(n)})$. We call these cells *significant light cells* and we let Ψ be the set of significant light cells of Λ. We show that in order to develop a $(1+\epsilon)$-estimator for facility location, we can safely ignore insignificant light cells (those with cost less than τf) of Λ.

We then partition Ψ into likelihood classes Ψ_i^j, where Ψ_i^j is the set of those significant light cells that are in the grid $\mathcal{G}_i$ and their parents have at least $(1+\epsilon)^j$ points and less than $(1+\epsilon)^{j+1}$. We consider only those likelihood classes Ψ_i^j whose size is $|\Psi_i^j| \geq \beta\sqrt{n}$ where $\beta = (\frac{\epsilon}{\log(n)})^{O(1)}$. We call these classes *contributing likelihood classes.* We show that non-contributing likelihood classes can be safely ignored.

We follow the steps below to show that if a likelihood class Ψ_i^j is contributing, we can approximate the facility location cost of it within $(1+\epsilon)$-factor.

- **Step 1:** We first show that if we sample any arbitrary cell $c \in \Psi_i^j$ with probability $\mathbf{Pr}[c] = \frac{n_{parent(c)}}{n}$, cell c is sampled almost uniformly at random from the set Ψ_i^j.
- **Step 2:** Next, we prove that with high probability, we sample at least $\text{poly}(\epsilon^{-1}\log(n))$ significant light cells from Ψ_i^j.
- **Step 3:** We finally develop an estimator that $(1+\epsilon)$-approximates the size of Ψ_i^j.

Putting everything together, we obtain an estimator that $(1+\epsilon)$-approximates the optimal cost $OPT_{FL}(P, f)$ in the sublinear geometric model using $\tilde{O}(\sqrt{n})$ range counting queries.

1.3 Related Work

Sublinear algorithms have been studied extensively for problems in metric spaces. Czumaj and Sohler [16] showed that we can $(1+\epsilon)$-approximate the weight of minimum spanning tree in metric space in time $\tilde{O}(n) = O(n\text{poly}(\epsilon^{-1}\log n))$. Badoiu, Czumaj, Indyk, and Sohler [8] showed that we can approximate the cost of facility location in metric space within constant factor in time $\tilde{O}(n)$, and Indyk [23] showed that we can return a constant factor approximation of the k-median problem in metric spaces in time $\tilde{O}(n)$. Interestingly, all sublinear algorithms that we are aware of in metric spaces have $\tilde{O}(n)$ running times. One may ask if there exists any sublinear algorithm that is truly sublinear in n?

For sparse graphs this is indeed possible. Chazelle, Rubinfeld, and Trevisan [12] in a seminal work showed that given a connected graph G (that is represented by adjacency lists) of average degree $\overline{d}$ with edge weights in the set $\{1, ..., w\}$, we can $(1+\epsilon)$-approximate the weight of the minimum spanning tree (MST) of G in time $O(\overline{d}w\epsilon^{-2}\log(\frac{\overline{d}w}{\epsilon}))$. Observe that a graph with average degree $\overline{d}$ is a sparse graph as it can have at most $O(n\overline{d})$ edges. Later, Nguyen and Onak [33] in another seminal work initiated the study of sublinear algorithms for Vertex Cover, Maximum Matching, Maximum Weight Matching, and Set Cover problems in bounded-degree graphs. As an example, for the problem of estimating the size of maximum matching, they showed how to approximate the size of the maximum matching up to an additive error ϵn in time $2^{d^{O(1/\epsilon)}}$, where d is the maximum degree of a vertex in the graph G. They obtained similar bounds for the aforementioned problems that approximate their size or cost up to an (ϵn)-additive term. Yoshida, Yamamoto, and Ito[38] and then Behnezhad [10] improve the running time (in fact, the query complexity) of Nguyen and Onak's algorithm for estimating the size of maximum matching and minimum vertex cover. As an example, Behnezhad showed that we can approximate the size of maximal matching (i.e., 2-approximate maximum matching) to within (ϵn)-additive error in time $\tilde{O}((\overline{d}+1)/\epsilon^2)$.

What do we know for Euclidean spaces? Interestingly, very few sublinear algorithms have been developed for optimization problems in Euclidean spaces. Indeed, we are aware of two sublinear algorithms for problems in Euclidean spaces. The first work is due to Czumaj, Ergün, Fortnow, Magen, Newman, Rubinfeld, and Sohler [14] who studied the problem of approximating the weight of Euclidean minimum spanning trees (EMST) in a sublinear time. The authors show that if we are provided with oracle access to basic data structures in the Euclidean space, we can estimate the weight of a EMST within $(1+\epsilon)$-factor.

In particular, they assume that we do not have access to a point set, but we can make queries and there is an oracle that answers our queries. The type of queries that they are allowed to make are *minimum bounding box* queries, *range* queries and *approximate nearest neighbor* queries. They show that in order to $(1+\epsilon)$-approximate the weight of the Euclidean minimum spanning tree in the plane (i.e., $\mathbb{R}^2$), we need to make $\tilde{O}(\sqrt{n})$ such queries.

The second sublinear algorithm that we are aware is for the minimum Euclidean bichromatic matching due to Indyk [25]. This algorithm is more of a linear-time constant factor approximation algorithm than a sublinear algorithm. Indyk assumed that we have access to points, but the main issue is solving the minimum Euclidean bichromatic matching problem using classical algorithms known for the maximum matching problem, say *Hungarian method* takes $O(n^3)$ time. (See [37]). He showed we can approximate the cost of minimum bichromatic matching within constant factor in near linear time and he showed that in time $\tilde{O}(n)$, we are able to do that. Later, Raghvendra and Agarwal [34] showed that in time $\tilde{O}(n)$ we can in fact, compute $(1+\epsilon)$-approximate Euclidean bichromatic matching.

The sublinear geometric model that we study in this paper is closely related to the dynamic geometric streaming model [24, 19] and the Massively Parallel Computations (MPC) model [27, 20]. In the the dynamic geometric streaming model [24, 19], Frahling and Sohler [19] in a groundbreaking work showed that given a (polynomially in n bounded) stream of insertions and deletions of points from an underlying space $[\Delta]^2$ (where $\Delta = n^{O(1)}$), we can compute a $(1+\epsilon)$-approximate solution for the k-median, k-means, MaxCut, maximum travelling salesperson, maximum spanning tree and average distance problems using $\text{poly}(k\epsilon^{-2}\log(n))$ space. Frahling, Indyk and Sohler [18] studied the problem of approximating the weight of the Euclidean minimum spanning tree within $(1+\epsilon)$ in the dynamic geometric streaming model. Later, Czumaj, Lammersen, Monemizadeh, and Sohler [15] developed a $(1+\epsilon)$-estimator for the cost of the Euclidean facility location using $\text{poly}(\epsilon^{-1}\log(n))$ space.

The Massively Parallel Computations (MPC) model was first introduced by Karloff, Suri, and Vassilvitskii [27] and later extended by [9, 20]. Andoni, Nikolov, Onak, Yaroslavtsev [4] studied the Euclidean minimum spanning tree and the minimum Euclidean bichromatic matching problems in the MPC model. They showed $O(1)$-round MPC $(1+\epsilon)$-algorithm for the Euclidean minimum spanning tree that uses $\tilde{O}(\sqrt{n})$ machines each one having local space $\tilde{O}(\sqrt{n})$. For the minimum Euclidean bichromatic matching problem, they showed that we can approximate the cost of minimum bichromatic matching within $(1+\epsilon)$-factor using similar number of communication rounds, number of machines, and the space per machine.

2 Preliminaries

Rounding notations. For the sake of simplicity, we assume that the logarithm of a number is always rounded down or up in an appropriate manner. As an example, if we want to use a range $[\lfloor\log(a)\rfloor, \cdots]$ or a range $[\cdots, \lceil\log(a)\rceil]$ where $a \in \mathbb{R}$, we write $[\log(a), \cdots]$ and $[\cdots, \log(a)]$ to simplify the notation, respectively.

Randomly shifted grids. Let $P \subseteq [n]^2$ be a point set of size n. We consider $\log(n)+1$ grids $\mathcal{G}_{\log(n)+1}, \mathcal{G}_{\log(n)}, \cdots, \mathcal{G}_1, \mathcal{G}_0$ where the grid $\mathcal{G}_i$ consists of cells of side length 2^i that we impose on the space $[n]^2$. We randomly shift the border lines of the grids as follows. We choose two random real numbers $v_1, v_2 \in [0, n]$ and we let $v = (v_1, v_2)$ be the random vector. Then, we shift every border line ℓ of every grid $\mathcal{G}_i$ using v. Observe that after this shift process, the point set P is enclosed in a rectangle of side length $[2n]^2$. In the sublinear geometric model, whenever we make range queries for ranges that are axis-aligned rectangles, we first shift these rectangles using v and then query them.

We let $\mathcal{G}_{\geq i} = \{\mathcal{G}_i, \mathcal{G}_{i+1}, \cdots, \mathcal{G}_{\log(n)}\}$ and $\mathcal{G}_{\leq i} = \{\mathcal{G}_i, \mathcal{G}_{i-1}, \cdots, \mathcal{G}_0\}$. Let $c \in \mathcal{G}_i$ be a cell in a grid $\mathcal{G}_i$. We denote by $c \cap P$ the subset of points that are inside the cell c or on the border lines of c. We denote the number of points in c by $n_c = |c \cap P|$. We let ℓ_c be the side length of c which is 2^i in the grid $\mathcal{G}_i$. We denote the parent of c which is in the grid $\mathcal{G}_{i+1}$ by *parent*(c). Let $c' \in \mathcal{G}_{\geq i}$ be a cell that contains c (i.e., $c \subseteq c'$). We say c' is an *ancestor* of c. We denote the ancestor of c at a grid $\mathcal{G}_{j \geq i}$ by ancestor(c, j). If $c' \in \mathcal{G}_{>i}$, we say that c' is a *proper ancestor* of c. Similarly, let $c' \in \mathcal{G}_{<i}$ be a cell that is contained in the cell c. We say c' is a *descendant* of c. We denote the descendant of c at a grid $\mathcal{G}_{j<i}$ by *descendant*(c, j).

2.1 Space partitioning and PTAS of [15]

Let $P \subseteq [2n]^2$ be a point set of size n and $f > 0$ be a positive real number. Let $0 < \epsilon \leq 1$ be the error parameter. As we explained in the introduction of this paper, Czumaj, Lammersen, Monemizadeh, and Sohler [15] show a construction for the facility location problem that partitions the space $[2n]^2$ into a set Λ of disjoint cells for which we can solve the facility location problem independently. We denote this algorithm by $\textsc{Alg}^{\epsilon}_{FL}(P)$ whose pseudocode is given below.

Algorithm 1 $\textsc{Alg}^{\epsilon}_{FL}$ - Czumaj, Lammersen, Monemizadeh, and Sohler [15].

```
Data: A point set $P \subseteq [n]^2$, an opening cost $f > 0$, and a parameter $0 < \epsilon \leq 1$.
Let $c$ be the square $[2n]^2$. Let $\Gamma = \{c\}$ and $\Lambda = \emptyset$;
while $\Gamma$ has a heavy cell $c$ do
    Let $\Gamma = \Gamma \backslash \{c\}$;
    Let $c_1, c_2, c_3, c_4$ be children of $c$ ; /* If $c \in \mathcal{G}_i$, its children are in $\mathcal{G}_{i-1}$ */
    for $c_i \in \{c_1, c_2, c_3, c_4\}$ do
        if $c_i$ is a heavy cell then
            $\Gamma = \Gamma \cup \{c_i\}$;
        else
            $\Lambda = \Lambda \cup \{c_i\}$ ;                /* $c_i$ is light */
for $c \in \Lambda$ do
    Let $F_c$ be the set of facilities returned by the PTAS [30] for the point set $P \cap c$;
    Let $\mathrm{cost}^{\epsilon}_{FL}(c, F_c, f) = \sum_{p \in P \cap c} \mathrm{dist}(p, F_c) + |F_c| \cdot f$ be the cost of $c \cap P$;
return Set $\Lambda$ and for every cell $c \in \Lambda$, set $F_c$ of opened facilities and $\mathrm{cost}^{\epsilon}_{FL}(c, F_c, f)$;
```

The PTAS [15] is based on partitioning heavy cells and detecting light cells, two notions that we define them next. Let c be an arbitrary cell in a grid $\mathcal{G}_i$. We use the 3-approximation algorithm due to Jain and Vazirani [26] to determine if c is heavy or light.

Heavy and light cells. Let c be a cell in a grid $\mathcal{G}_i$. Let $\delta_{FL} = 2^{20} \cdot (\frac{\log n}{\epsilon})^2$. Let F_c be the set of facilities returned by $\textsc{Alg}^{3}_{FL}(P \cap c)$. We say c is a *heavy cell* if $\mathrm{cost}^{3}_{FL}(c, F_c, f) = \sum_{p \in P \cap c} \mathrm{dist}(p, F_c) + |F_c| \cdot f \geq \delta_{FL} \cdot f$; otherwise it is a *light* cell.

▶ **Lemma 4** ([15]). *Let $P \subseteq [2n]^2$ be a point set of size n and $f \in \mathbb{R}^+$ be a positive real number. Let $0 < \epsilon \leq 1$ be the error parameter. For the output of $\textsc{Alg}^{\epsilon}_{FL}(P)$ we have*

- *For every cell $c \in \Lambda$, we have $cost^{\epsilon}_{FL}(c, F_c, f) \leq (1 + \epsilon) \cdot OPT_{FL}(P \cap c, f)$.*
- *$(1 - \epsilon) \cdot OPT_{FL}(P, f) \leq cost^{\epsilon}_{FL}(\Lambda, f) = \sum_{c \in \Lambda} cost^{\epsilon}_{FL}(c, F_c, f) \leq (1 + \epsilon) \cdot OPT_{FL}(P, f)$.*

3 Sublinear algorithm for facility location problem

In this section, we develop our sublinear algorithm for the facility location problem and prove Theorem 2. We break this algorithm into four steps as follows:

1. **Heavy tester:** We first develop a tester algorithm for heavy cells.
2. **Telescoping sampling:** Then, we explain the telescoping sampling.
3. **$O(\sqrt{n})$-estimator:** In the third step, we explain a $\sqrt{n}$-estimator for the Euclidean facility location problem in the sublinear geometric model.
4. **$(1+\epsilon)$-estimator:** In the final step, we describe our $(1+\epsilon)$-estimator in the sublinear geometric model and analyze it.

3.1 A tester algorithm for heavy cells

Here, we develop a tester algorithm in the sublinear geometric model that given an arbitrary cell c in a grid $\mathcal{G}_i$, makes $\text{poly}(\epsilon^{-1} \cdot \log(n))$ RangeCount queries to distinguish between the case that c is a heavy cell or a light one. The proof of the following lemma is in Appendix A.

▶ **Lemma 5** (Heavy Tester). *Let c be a cell in a grid $\mathcal{G}_i$. Then, there exists a deterministic tester algorithm that we call HeavyTester(c) so that*

- ***Testing heaviness:*** *It makes $O(\epsilon^{-8} \cdot \log^6(n))$ RangeCount queries to determine if the cell c is heavy or light.*
- ***Approximating a solution:*** *If c is a light cell in set Λ, then HeavyTester(c) returns a set F_c of facilities that $(1+\epsilon)$-approximates the optimal facility location cost of c.*

3.2 $O(\sqrt{n})$-approximate sublinear algorithm

Now, we explain an algorithm that distinguishes between the case that the facility location cost of P is at most $\sqrt{n} \cdot f$ or it is greater than $\sqrt{n} \cdot f$. For the former case, we return $(1+\epsilon)$-approximate solution and for the latter, we obtain a lower bound $\sqrt{n} \cdot f$ for the cost.

Algorithm 2 $\sqrt{n}$-Estimator.

```
Data: The discrete space $[2n]^2$, an opening cost $f > 0$, and an error parameter
      $0 < \epsilon \le 1$.
Result: A $\sqrt{n}$-estimator $\mathcal{Z}$ for the facility location cost of a point set $P \subseteq [2n]^2$.
Let $c$ be the square $[2n]^2$ and let $\Gamma$ be an empty set. Let $\mathcal{H} = \{c\}$;
At any time, let $\text{cost}^{\epsilon}_{FL}(\Gamma, f) = \sum_{c' \in \Gamma} \text{cost}^{\epsilon}_{FL}(c, F_c, f)$, where $F_c$ is the set of
  facilities that the randomized algorithm [15] $\text{Alg}^{\epsilon}_{FL}$ returns for each cell $c \in \Gamma$;
while $cost^{\epsilon}_{FL}(\Gamma, f) \le \sqrt{n} \cdot f$ and $\mathcal{H} \ne \emptyset$ do
    Take an arbitrary cell $c \in \mathcal{H}$ and delete it from $\mathcal{H}$;
    if HeavyTester$(c)$ returns that $c$ is a heavy cell then
        Let $C_c = \{c_1, c_2, c_3, c_4\}$ be the four children of $c$ in the grid $\mathcal{G}_{i-1}$;
        Let $\mathcal{H} = \mathcal{H} \cup C_c$;
    else
        Add $c$ to the set $\Gamma$;
Return $\Gamma$;
```

Lemma 6 is similar to Lemma 22 and we explain Lemma 22 in detail later.

▶ **Lemma 6.** *Let $P \subseteq [\Delta]^2$ be a point set of size n. Let $f > 0$ be the opening cost and $0 < \epsilon \leq 1$ be the error parameter. Then, the sublinear algorithm $\sqrt{n}$-*Estimator *uses $O(\epsilon^{-9} \log^7(n) \cdot \sqrt{n}))$* RangeCount *queries and distinguishes between the following cases:*

- *If the optimal facility location cost of P is $O(\sqrt{n} \cdot f)$, then this sublinear algorithm returns a set of facilities that $(1+\epsilon)$-approximates the optimal facility location cost of P.*
- *If the optimal facility location cost of P is $\Omega(\sqrt{n} \cdot f)$, then this sublinear algorithm $O(\sqrt{n})$-approximates the optimal facility location cost $OPT(P, f)$ of P. Moreover, it finds an $\Omega(\sqrt{n} \cdot f)$ lower-bound for the optimal facility location cost $OPT_{FL}(P, f)$ of P.*

3.3 Telescoping sampling

In this section we develop a sampling mechanism that samples a cell c with probability $p_c = \frac{n_c}{n}$. This will be used as a basic primitive that we later use for our $(1+\epsilon)$-estimator.

Algorithm 3 TelescopingSampling.

```
   Data: A grid $\mathcal{G}_i$
   Result: A sampled cell $c' \in \mathcal{G}_i$ such that the probability that any cell $c'' \in \mathcal{G}_i$ is $c'$ is
           $\mathbf{Pr}[c' = c''] = \frac{n_{c''}}{n}$.
1  Let $c$ be the cell in the grid $\mathcal{G}_{\log(n)}$ and let $j = \log(n)$ ;
2  while $j > i$ do
3      Split $c$ into its four sub-cells $c_1, c_2, c_3, c_4$ that are in the grid $\mathcal{G}_{j-1}$ and query the
         number of points inside each of them using the subroutine RangeCount;
4      Sample the sub-cell $c_i$ for $i \in [4]$ with probability $\frac{n_{c_i}}{n_c}$;
5      Let $c'$ be the cell that is sampled in Step 4;        // $c'$ is one of $c_1, c_2, c_3, c_4$.
6      Let $c = c'$ and $j = j - 1$;
7  Return $c$;
```

The proofs of the following lemmas are given in Appendix B.

▶ **Lemma 7.** *Assume that we invoke Subroutine* TelescopingSampling*($\mathcal{G}_i$). Then, a cell c that is returned by this subroutine is sampled with the probability that $\mathbf{Pr}[c] = \frac{n_c}{n}$.*

▶ **Lemma 8.** *Given a grid $\mathcal{G}_i$, the number of* RangeCount *queries that we make in Subroutine* TelescopingSampling*($\mathcal{G}_i$) is $O(\log n)$.*

3.4 $(1+\epsilon)$-approximate sublinear algorithm

In this section, we prove Theorem 2. We first give the pseudocode of our main sublinear algorithm. Next, we prove the correctness of our algorithm and analyze its query complexity.

3.5 Analysis

Let P be a point set of n points in discrete space $[2n]^2$. Now, for the sake of the analysis, assume that we run the following two processes.

- In the first process, for point set P, we run the quadtree construction (for the facility location problem) that we presented in Lemma 4 and it returns a set Λ of light cells.
- In the second process, given RangeCount query access to point set P, we invoke Algorithm 4 $(1+\epsilon)$-Estimator and it returns the estimator $\mathcal{Z} = \mathcal{A} + \mathcal{B}$.

Algorithm 4 $(1+\epsilon)$-Estimator.

```
Data: The discrete space $[2n]^2$, an opening cost $f > 0$, and an error parameter $0 < \epsilon$.
Result: A $(1+\epsilon)$-estimator $\mathcal{Z}$ for the facility location cost of a point set $P \subseteq [2n]^2$.
Let $\mathcal{Z} = 0$, $\mathcal{A} = 0$, and $\mathcal{B} = 0$ ;
for $i = \log(n) + 1$ down to $\frac{3}{4} \cdot \log(n)$ do
    for each cell $c \in \mathcal{G}_i$ do
        if HeavyTester$(c)$ returns that $c$ is a heavy cell then
            Let $C_c = \{c_1, c_2, c_3, c_4\}$ be the four children of $c$ in the grid $\mathcal{G}_{i-1}$;
            Invoke HeavyTester for the sub-cells $C_c$ to find out which ones are heavy;
            for each light sub-cell $c' \in C_c$ if there exists any light cell do
                Let $\mathrm{cost}^{\epsilon}_{FL}(c', F_{c'}, f)$ be the cost that the algorithm Alg$^{\epsilon}_{FL}(c')$ returns for $c$. Let $\mathcal{A} = \mathcal{A} + \mathrm{cost}^{\epsilon}_{FL}(c, F_{c'}, f)$;
for $i = \frac{3}{4} \cdot \log(n)$ down to 1 do
    Let $\mathcal{W}_{i-1} = 0$;
    Let $\mathcal{X}_{i-1}$ and $\mathcal{Y}_{i-1}$ be two vectors of length $\log_{1+\epsilon}(n)$ initialized to zero;
    for $\ell = 1$ to $z = 2^{14}\delta^2_{FL} \log^5(n)\epsilon^{-6} \cdot \sqrt{n}$ do
        Let $c$ be a cell sampled using Subroutine TelescopingSampling$(\mathcal{G}_i)$;
        if HeavyTester$(c)$ returns that $c$ is a heavy cell then
            Let $C_c = \{c_1, c_2, c_3, c_4\}$ be the four children of $c$ in the grid $\mathcal{G}_{i-1}$;
            Invoke HeavyTester for the sub-cells $C_c$ to find out which ones are heavy;
            if at least one of the sub-cells $C_c$ is a significant light cell then
                Let $j$ be the power of $(1+\epsilon)$ where $(1+\epsilon)^j \le n_c < (1+\epsilon)^{j+1}$;
                Let $C'_c$ be the sub-cells in $C_c$ that are significant light cells;
                Let $\mathcal{Y}_{i-1}[j] = \mathcal{Y}_{i-1}[j] + |C'_c|$;
                for each significant light sub-cell $c' \in C'_c$ do
                    Let $\mathrm{cost}^{\epsilon}_{FL}(c', , F_{c'}, f)$ be the cost that Algorithm Alg$^{\epsilon}_{FL}(c')$ returns for $c'$. Let $\mathcal{X}_{i-1}[j] = \mathcal{X}_{i-1}[j] + \mathrm{cost}^{\epsilon}_{FL}(c', F_{c'}, f)$;
    for $j = 1$ to $\log_{1+\epsilon}(n)$ do
        Let $w^j_i = \frac{n}{z(1+\epsilon)^j} \cdot \mathcal{Y}_{i-1}[j]$. Let $\mathcal{W}_{i-1} = \mathcal{W}_{i-1} + w^j_i \cdot \mathcal{X}_{i-1}[j]$;
    Let $\mathcal{B} = \mathcal{B} + \mathcal{W}_{i-1}$;
Return $\mathcal{Z} = \mathcal{A} + \mathcal{B}$;
```

Recall that for the quadtree construction in Lemma 4, we randomly shift the border lines of cells of the quadtree using a random vector v. In the sublinear geometric model, we randomly shift the border lines of axis-aligned rectangles (using v) that we use in RangeCount queries. Lemma 4 shows that $(1-\epsilon) \cdot OPT_{FL}(P, f) \le \sum_{c \in \Lambda} \mathrm{cost}^{\epsilon}_{FL}(c, F_c, f) \le (1+\epsilon) \cdot OPT_{FL}(P, f)$.

We further split the set Λ into two subsets $\Lambda_{\mathcal{A}}$ and $\Lambda_{\mathcal{B}}$. In particular, let $\Lambda_{\mathcal{A}}$ be the subset of light cells in Λ that are in the grids $\mathcal{G}_{\frac{3}{4} \cdot \log(n)}, \cdots, \mathcal{G}_{\log(n)+1}$. Let $\Lambda_{\mathcal{B}} = \Lambda \backslash \Lambda_{\mathcal{A}}$ be the subset of light cells in Λ that are in $\mathcal{G}_1, \cdots, \mathcal{G}_{\frac{3}{4} \cdot \log(n)-1}$. Our goal in this section is to show that with a probability greater than $1 - 1/n^5$, the following three claims are correct:

- The number of queries that Algorithm 4 $(1+\epsilon)$-Estimator makes is $\tilde{O}(\sqrt{n})$.
- The term $\mathcal{A}$ is the cost of light cells of the set $\Lambda_{\mathcal{A}}$. That is, $\mathcal{A} = \sum_{c \in \Lambda_{\mathcal{A}}} \mathrm{cost}^{\epsilon}_{FL}(c, F_c, f)$.

- The term $\mathcal{B}$ is an $(1+\epsilon)$-estimator for the cost of light cells of the set $\Lambda_{\mathcal{B}}$. That is, $(1-\epsilon)\cdot\sum_{c\in\Lambda_{\mathcal{B}}}\text{cost}^{\epsilon}_{FL}(c,F_c,f)\leq\mathcal{B}\leq(1+\epsilon)\cdot\sum_{c\in\Lambda_{\mathcal{B}}}\text{cost}^{\epsilon}_{FL}(c,F_c,f)$.

Once we prove these three claims, we have shown that $\mathcal{Z}$ is an $(1+\epsilon)$-estimator of the optimal facility location cost $OPT_{FL}(P,f)$ what proves Theorem 2.

3.5.1 The estimator term $\mathcal{A}$

The following two lemmas (whose proofs are given in Appendix C) show that (1) the number of RangeCount queries that we make to compute estimator $\mathcal{A}$ is $\tilde{O}(\sqrt{n})$, and (2) the term $\mathcal{A}$ approximates the facility location cost of light cells of set $\Lambda_{\mathcal{A}}$.

▶ **Lemma 9.** *Let $\Lambda_{\mathcal{A}}$ be the subset of light cells in Λ that are in the grids $\mathcal{G}_{\geq\frac{3}{4}\cdot\log(n)}$. Then,*

- *All light cells of $\Lambda_{\mathcal{A}}$ are detected by Algorithm 4 $(1+\epsilon)$-Estimator.*
- *To compute estimator $\mathcal{A}$, we make $O(\epsilon^{-8}\cdot\log^{6}(n)\cdot\sqrt{n})$ RangeCount queries.*

▶ **Lemma 10.** $\mathcal{A}=\sum_{c\in\Lambda_{\mathcal{A}}}cost^{\epsilon}_{FL}(c,F_c,f)$.

3.5.2 The estimator term $\mathcal{B}$

In this section, we prove that estimator $\mathcal{B}$ approximates the cost of light cells of $\Lambda_{\mathcal{B}}$ within $(1+\epsilon)$-factor. Let $c\in\Lambda$ be an arbitrary light cell in the set Λ. Suppose that c is in a grid $\mathcal{G}_i$. Then, the parent of c that we denote it by *parent*(c) is a heavy cell in grid $\mathcal{G}_{i+1}$.

▶ **Definition 11** (Significant Light Cells). *Let $\tau=\frac{\epsilon}{9\log(n)}$. We say a light cell $c\in\Lambda$ is a significant light cell if $cost^{\epsilon}_{FL}(c,F_c,f)\geq\tau f$; otherwise, we say c' is an insignificant light cell. We let $\Psi=\{c\in\Lambda:cost^{\epsilon}_{FL}(c,F_c,f)\geq\tau f\}$ be the set of light cells in Λ that are significant.*

We denote by $\Psi_i=\{c\in\Psi:c\in\mathcal{G}_i\}$ the set of significant light cells in the grid $\mathcal{G}_i$.

▶ **Definition 12** (Heavy Parents of Significant Light Cells). *We denote the set of heavy parents of significant light cells Ψ_i by $\Gamma(\Psi_i)=\{c'\in\mathcal{G}_{i+1}:\exists c\in\Psi_i$ such that c' is parent$(c)\}$.*

Note that up to 4 cells in Ψ_i can have the same heavy parent.

▶ **Definition 13** (Likelihood Classes). *We partition Ψ_i into $\epsilon^{-1}\log(n)$ likelihood classes Ψ_i^j where a cell $c\in\Psi_i^j$ if the number of points in its parent(c) is in the range $(1+\epsilon)^j\leq n_{parent(c)}<(1+\epsilon)^{j+1}$.*

We denote the set of heavy parents of significant light cells that are in the class Ψ_i^j by $\Gamma(\Psi_i^j)=\{c'\in\mathcal{G}_{i+1}:\exists c\in\Psi_i^j$ such that $c'=$ *parent*$(c)\}$.

▶ **Definition 14** (Contributing Likelihood Classes). *Let $\beta=\frac{\epsilon^2}{2\delta_{FL}\cdot\log^2(n)}$. We say a class Ψ_i^j is a contributing likelihood class if $|\Psi_i^j|\geq\beta\sqrt{n}$; otherwise, it is a non-contributing class.*

Roadmap of the proof of Theorem 2. We first prove (in Lemma 22) that the cost of significant light cells of $\Psi=\{c\in\Lambda:\text{cost}^{\epsilon}_{FL}(c,F_c,f)\geq\tau f\}$ is an $(1+\epsilon)$-approximation of $OPT_{FL}(P,f)$. Therefore, when we develop a $(1+\epsilon)$-estimator for the facility location problem, we can ignore insignificant light cells of Λ.

On the other hand, all significant light cells of Ψ are partitioned into likelihood classes Ψ_i^j. Now imagine we have the lower bound $OPT_{FL}(P,f)\geq\sqrt{n}\cdot f$ for $OPT_{FL}(P,f)$. Otherwise, the estimator $\sqrt{n}$-Estimator of Lemma 6 makes $O(\epsilon^{-9}\log^{7}(n)\cdot\sqrt{n}))$ RangeCount queries and returns a set of facilities that $(1+\epsilon)$-approximates the optimal facility location cost of P. Lemma 27 shows that we can safely ignore the contribution of the non-contributing likelihood classes Ψ_i^j. Let us fix a contributing likelihood classes Ψ_i^j.

- In Lemma 15 we show that with probability at least $1 - 2/n^5$, the estimator w_i^j in Algorithm 4 $(1+\epsilon)$-ESTIMATOR is a $(1+\epsilon)$-approximation of the size of Ψ_i^j.
- Lemma 16 shows that if we sample any arbitrary cell $c \in \Psi_i^j$ with probability $\mathbf{Pr}[c] = \frac{n_{parent(c)}}{n}$, the cell c is sampled almost uniformly at random from the set Ψ_i^j.
- Lemma 17 proves that with probability at least $1 - 1/n^{20}$, Algorithm 4 $(1+\epsilon)$-ESTIMATOR samples at least $\mathrm{poly}(\epsilon^{-1} \cdot \log(n))$ significant light cells from Ψ_i^j.

In Lemma 18, we use these three tools to show that if a likelihood class Ψ_i^j is contributing, we can approximate the facility location cost $\mathrm{cost}_{FL}^{\epsilon}(\Psi_i^j, f)$ to within $(1+\epsilon)$-factor. Putting everything together, the estimator $\mathcal{Z}$ that is computed in Algorithm 4 $(1+\epsilon)$-ESTIMATOR is an $(1+\epsilon)$-approximation of $OPT_{FL}(P, f)$ what proves Theorem 2.

The statements of Lemmas 22 and 27, and their proofs are given in Appendix D.

Approximating the number of cells of a contributing class. In this section, we show that with high probability, the estimator w_i^j in Algorithm 4 $(1+\epsilon)$-ESTIMATOR is a $(1+\epsilon)$-approximation of the size of the contributing likelihood class Ψ_i^j.

▶ **Lemma 15.** *Let Ψ_i^j be a contributing class. Then, with probability at least $1 - 2/n^5$, the estimator w_i^j in Algorithm 4 $(1+\epsilon)$-*ESTIMATOR *is a $(1+\epsilon)$-approximation of $|\Psi_i^j|$. That is,* $\mathbf{Pr}\left[(1-\epsilon) \cdot |\Psi_i^j| \le w_i^j \le (1+\epsilon) \cdot |\Psi_i^j|\right] \ge 1 - 1/n^5$.

Proof. We define z runs $\mathcal{R}_1, \cdots, \mathcal{R}_z$ where during a run $\mathcal{R}_\ell$ we sample a cell $c \in \mathcal{G}_{i+1}$. We define a random variable Y_ℓ corresponding to the number of significant light sub-cells of a heavy cell $c \in \Gamma(\Psi_i^j)$ that is sampled in the run $\mathcal{R}_\ell$. Note that a heavy cell can have $0, 1, 2, 3, 4$ significant light sub-cells. If a heavy cell c from the set $\Gamma(\Psi_i^j)$ is sampled using Subroutine TELESCOPINGSAMPLING$(\mathcal{G}_{i+1})$, the random variable Y_ℓ will be the number of significant light sub-cells of c; otherwise Y_ℓ is zero. Then, we have $\mathbf{E}[Y_\ell] = \sum_{c \in \Gamma(\Psi_i^j)} \mathbf{Pr}[c] \cdot |C'_c|$, where C'_c is the set of significant light sub-cells of a heavy cell $c \in \Gamma(\Psi_i^j)$.

Recall that the likelihood class Ψ_i^j contains those cells $c \in \Psi_i$ for which $(1+\epsilon)^j \le n_{parent(c)} < (1+\epsilon)^{j+1}$. Then, $\frac{(1+\epsilon)^j}{n} \cdot \sum_{c \in \Gamma(\Psi_i^j)} |C_c| \le \mathbf{E}[Y_\ell] \le \frac{(1+\epsilon)^{j+1}}{n} \cdot \sum_{c \in \Gamma(\Psi_i^j)} |C_c|$.

Next, we define the random variable $Y = \frac{n}{z(1+\epsilon)^j} \cdot \sum_{\ell=1}^{z} Y_\ell$ for which we have $|\Psi_i^j| = \sum_{c \in \Gamma(\Psi_i^j)} |C_c| \le \mathbf{E}[Y] \le (1+\epsilon) \cdot \sum_{c \in \Gamma(\Psi_i^j)} |C_c| = (1+\epsilon) \cdot |\Psi_i^j|$.

Since the likelihood class Ψ_i^j is contributing, Lemma 28 tells us that for $\alpha = \frac{\epsilon^3}{18\delta_{FL} \log^3(n)}$ we have $\mathrm{cost}_{FL}^{\epsilon}(\Psi_i^j, f) = \sum_{c \in \Psi_i^j} \mathrm{cost}_{FL}^{\epsilon}(c, F_c, f) \ge \alpha\sqrt{n} \cdot f$. All cells in the likelihood class Ψ_i^j are light which means that $\mathrm{cost}_{FL}^{\epsilon}(c, F_c, f) \le \delta_{FL} \cdot f$. This essentially means that $|\Psi_i^j| \ge \frac{\alpha\sqrt{n} \cdot f}{\delta_{FL} \cdot f} = \frac{\alpha}{\delta_{FL}} \cdot \sqrt{n}$. We can assume that $n \ge 2^{12}\epsilon^{-4}\log^2(n)(\frac{\delta_{FL}}{\alpha})^2$, otherwise the facility location instance of the point set P has at most $\mathrm{poly}(\epsilon^{-1}\log(n))$ points, that can be $(1+\epsilon)$-approximately solved using $\mathrm{poly}(\epsilon^{-1}\log(n))$ RANGECOUNT queries. Therefore, $\mathbf{E}[Y] \ge \frac{\alpha}{\delta_{FL}} \cdot \sqrt{n} \ge 60\epsilon^{-2}\log(n)$. Now, we use the Hoeffding bound [22] where we set $M = 4$ to obtain we have $\mathbf{Pr}\left[|w_i^j - |\Psi_i^j|| \ge \epsilon \cdot |\Psi_i^j|\right] = \mathbf{Pr}\left[|\mathbf{E}[Y] - Y|| \ge \epsilon \cdot \mathbf{E}[Y]\right] \le 2\exp(-(\frac{\epsilon^2 \cdot \mathbf{E}[Y]}{12})) \le 2/n^5$.

Thus, with probability at least $1 - 2/n^5$, the estimator w_i^j is within $(1+\epsilon)$-approximation of $|\Psi_i^j|$. This proves the lemma. ◀

Almost uniformly sampling from a contributing class. Let Ψ_i^j be a contributing likelihood class. Here we show that if we sample any arbitrary cell $c \in \Psi_i^j$ with probability $\mathbf{Pr}[c] = \frac{n_{parent(c)}}{n}$, the cell c is sampled almost uniformly at random. We later show that with high probability, Algorithm 4 $(1+\epsilon)$-ESTIMATOR samples significant light cells from each contributing likelihood class Ψ_i^j.

▶ **Lemma 16.** *Suppose we sample a cell $c \in \Psi_i^j$ with probability $\mathbf{Pr}[c] = \frac{n_{parent(c)}}{n}$. Then, the cells in Ψ_i^j are sampled almost uniformly at random. That is, the probability that we sample a cell $c \in \Psi_i^j$ is $(1-\epsilon) \cdot \frac{1}{|\Psi_i^j|} \le \mathbf{Pr}[c] \le (1+\epsilon) \cdot \frac{1}{|\Psi_i^j|}$.*

Proof. Recall that for every light cell $c \in \Psi_i^j$, we have $(1+\epsilon)^j \le n_{parent(c)} < (1+\epsilon)^{j+1}$. Thus $\frac{(1+\epsilon)^j}{n} \le \mathbf{Pr}[c] = \frac{n_{parent(c)}}{n} \le \frac{(1+\epsilon)^{j+1}}{n}$. We conclude that the probability of sampling light cells Ψ_i^j is within $(1+\epsilon)$ of $\frac{(1+\epsilon)^j}{n}$. ◀

▶ **Lemma 17.** *Let Ψ_i^j be a contributing class. Then, with probability at least $1 - 1/n^{20}$, a set of at least $x = 2^9 \delta_{FL} \epsilon^{-3} \log^2(n)$ heavy cells are sampled from the set $\Gamma(\Psi_i^j)$.*

Proof. Recall that $z = 2^{14} \delta_{FL}^2 \log^5(n) \epsilon^{-6} \cdot \sqrt{n}$. We define z runs $\mathcal{R}_1, \cdots, \mathcal{R}_z$ where during a run $\mathcal{R}_\ell$ we sample a cell $c \in \mathcal{G}_{i+1}$. Next we study the probability of sampling a heavy cell $c \in \Gamma(\Psi_i^j)$. We define an indicator random variable Y_ℓ corresponding to the run $\mathcal{R}_\ell$ which is one if a heavy cell $c \in \Gamma(\Psi_i^j)$ is sampled using Subroutine TelescopingSampling($\mathcal{G}_{i+1}$) in Line 14 (of Algorithm 4 $(1+\epsilon)$-Estimator) and zero otherwise. Then, we have $\mathbf{E}[Y_\ell] = \mathbf{Pr}[Y_\ell = 1] = \frac{1}{n} \cdot \sum_{c \in \Gamma(\Psi_i^j)} n_c$.

Based on Lemma 29, the number of points in the contributing likelihood class Ψ_i^j must be at least $n(\Psi_i^j) = \sum_{c \in \Psi_i^j} n_c \ge \frac{\alpha\sqrt{n}}{(1+\epsilon)}$, for $\alpha = \frac{\epsilon^3}{18\delta_{FL}\log^3(n)}$. Therefore, $\mathbf{E}[Y_\ell] = \frac{1}{n} \cdot \sum_{c \in \Gamma(\Psi_i^j)} n_c \ge \frac{\frac{\alpha\sqrt{n}}{(1+\epsilon)}}{n} \ge \frac{\alpha}{(1+\epsilon)} \cdot \frac{1}{\sqrt{n}}$. Next, we define the random variable $Y = \sum_{\ell=1}^{z} Y_\ell$ whose expectation is $\mathbf{E}[Y] = \sum_{\ell=1}^{z} \frac{1}{n} \cdot \sum_{c \in \Gamma(\Psi_i^j)} n_c \ge 2^{10} \delta_{FL} \epsilon^{-3} \log^2(n)$.

Using the Chernoff bound [13] and since $\delta_{FL} = 2^{20} \cdot (\frac{\log n}{\epsilon})^2$, we obtain

$$\mathbf{Pr}\left[\left|\mathbf{E}[Y] - Y\right| \ge \epsilon \cdot \mathbf{E}[Y]\right] \le 2\exp(-(\frac{\epsilon^2 \cdot 2^{10}\delta_{FL}\epsilon^{-3}\log^2(n)}{3})) \le 1/n^{20} .$$

Thus, with probability at least $1 - 1/n^{20}$, we sample at least $(1-\epsilon) \cdot 2^{10} \delta_{FL} \epsilon^{-3} \log^2(n) \ge 2^9 \delta_{FL} \epsilon^{-3} \log^2(n)$ heavy cells from the set $\Gamma(\Psi_i^j)$. Note that up to 4 cells in Λ_i^j can have the same heavy parent and they are sampled together. This proves this lemma. ◀

An unbiased estimator for contributing classes. Next, we prove that having an estimation of the size of a contributing likelihood class Ψ_i^j and assuming that we can sample cells in Ψ_i^j almost uniformly at random, we can then $(1+\epsilon)$-approximate the facility location cost of the contributing likelihood class Ψ_i^j.

▶ **Lemma 18** (Estimator for Contributing Classes). *Let Ψ_i^j be a contributing class. Suppose we can sample a cell from Ψ_i^j almost uniformly at random. That is, the probability that we sample a cell $c \in \Psi_i^j$ is $(1-\epsilon) \cdot \frac{1}{|\Psi_i^j|} \le \mathbf{Pr}[c] \le (1+\epsilon) \cdot \frac{1}{|\Psi_i^j|}$. Let S be a sampled set of $x \ge 2^9 \delta_{FL} \epsilon^{-3} \log^2(n)$ light cells $S = \{c_1, \cdots, c_x\} \subseteq \Psi_i^j$ that are sampled almost uniformly at random. Let y be an arbitrary number in the range $[\frac{|\Psi_i^j|}{(1-\epsilon)}, \frac{|\Psi_i^j|}{(1+\epsilon)}]$. Then, with probability at least $1 - 1/n^5$, we have $(1-\epsilon) \cdot \sum_{c \in \Psi_i^j} cost_{FL}^{\epsilon}(c, F_c, f) \le \frac{y}{x} \cdot \sum_{\ell=1}^{x} cost_{FL}^{\epsilon}(c_\ell, F_{c_\ell}, f) \le (1+\epsilon) \cdot \sum_{c' \in \Psi_i^j} cost_{FL}^{\epsilon}(c, F_c, f)$.*

Proof. We define x random variables $X_1, \cdots, X_x$ where X_ℓ corresponds to the cost of the light cell c_ℓ sampled from Ψ_i^j. Then, since $\mathbf{E}[X_\ell] = \sum_{c \in \Psi_i^j} \mathbf{Pr}[c] \cdot \mathrm{cost}_{FL}^{\epsilon}(c, F_c, f)$, we have $(1-\epsilon) \cdot \frac{1}{|\Psi_i^j|} \cdot \mathrm{cost}_{FL}^{\epsilon}(\Psi_i^j, f) \le \mathbf{E}[X_\ell] \le (1+\epsilon) \cdot \frac{1}{|\Psi_i^j|} \cdot \mathrm{cost}_{FL}^{\epsilon}(\Psi_i^j, f)$, where $\mathrm{cost}_{FL}^{\epsilon}(\Psi_i^j, f) = \sum_{c \in \Psi_i^j} \mathrm{cost}_{FL}^{\epsilon}(c, F_c, f)$. By the linearity of expectation for the random variable $X = \sum_{\ell=1}^{x} X_\ell$, we obtain $(1-\epsilon)^2 \cdot \mathrm{cost}_{FL}^{\epsilon}(\Psi_i^j, f) \le \frac{y}{x} \cdot \mathbf{E}[X] \le (1+\epsilon)^2 \cdot \mathrm{cost}_{FL}^{\epsilon}(\Psi_i^j, f)$.

Recall that the minimum and the maximum facility location cost of a significant light cell $c \in \Psi_i^j$ are $\tau \cdot f$ and $\delta_{FL} \cdot f$, respectively, where $\tau = \frac{\epsilon}{9\log(n)}$ and $\delta_{FL} = 2^{20} \cdot (\frac{\log n}{\epsilon})^2$. Thus, $X_\ell \leq \delta_{FL} \cdot f$. Also, $\mathbf{E}[X_\ell] \geq (1-\epsilon) \cdot \frac{1}{|\Psi_i^j|} \cdot \mathrm{cost}_{FL}^{\epsilon}(\Psi_i^j, f) \geq (1-\epsilon) \cdot \frac{1}{|\Psi_i^j|} \cdot \tau \cdot f |\Psi_i^j| = (1-\epsilon)\tau f$.

Therefore, $\mathbf{E}[X] \geq x(1-\epsilon)\tau f \geq 2^5 \delta_{FL}\epsilon^{-2}\log(n) \cdot f$. We use the Hoeffding bound [22] where we set $M = \delta_{FL} \cdot f$ and since $f > 0$ to obtain

$$\mathbf{Pr}\left[(1-\epsilon)^3 \cdot \mathrm{cost}_{FL}^{\epsilon}(\Psi_i^j, f) \leq \frac{y}{x} \cdot \mathbf{E}[X] \leq (1+\epsilon)^3 \cdot \mathrm{cost}_{FL}^{\epsilon}(\Psi_i^j, f)\right]$$
$$= \mathbf{Pr}\left[|X - \mathbf{E}[X]| \geq \epsilon \cdot \mathbf{E}[X]\right] \leq 2 \cdot \exp(-\frac{\mathbf{E}[X] \cdot \epsilon^2}{3 \cdot M}) \leq 2 \cdot \exp(-\frac{2^5 \delta_{FL}\epsilon^{-2}\log(n) \cdot f\epsilon^2}{3\delta_{FL} \cdot f}) \leq \frac{2}{n^5} \ .$$

We replace ϵ by $\epsilon/3$ to finish the proof of this lemma. ◀

Finishing Proof of Theorem 2. Now we are ready to finish the proof of Theorem 2. Lemma 18 shows that if a likelihood class Ψ_i^j is contributing, we can approximate the facility location cost $\mathrm{cost}_{FL}^{\epsilon}(\Psi_i^j, f)$ to within $(1+\epsilon)$-factor if (1) we can sample x significant light cells of Ψ_i^j almost uniformly at random, and (2) we can $(1+\epsilon)$-approximate the size $|\Psi_i^j|$.

Lemma 16 shows that if we sample any arbitrary cell $c \in \Psi_i^j$ with probability $\mathbf{Pr}[c] = \frac{n_{parent(c)}}{n}$, the cell c is sampled almost uniformly at random. Lemma 17 proves that with probability at least $1 - 1/n^{20}$, Algorithm 4 $(1+\epsilon)$-Estimator samples at least x significant light cells from each contributing likelihood class Ψ_i^j. Finally, Lemma 15, with probability at least $1 - 2/n^5$, the estimator w_i^j in Algorithm 4 $(1+\epsilon)$-Estimator is a $(1+\epsilon)$-approximation of the size of the contributing likelihood class Ψ_i^j.

Recall that in Lemma 22 we proved that the cost of significant light cells of $\Psi = \{c \in \Lambda : \mathrm{cost}_{FL}^{\epsilon}(c, F_c, f) \geq \tau f\}$ is an $(1+\epsilon)$-approximation of $OPT_{FL}(P, f)$. Therefore, we can ignore insignificant light cells of Λ. All significant light cells of Ψ are partitioned into likelihood classes Ψ_i^j. Using Lemma 27, we can safely ignore the contribution of the non-contributing likelihood classes Ψ_i^j. Putting everything together, the estimator $\mathcal{Z}$ that is computed in Algorithm 4 $(1+\epsilon)$-Estimator is an $(1+\epsilon)$-approximation of $OPT_{FL}(P, f)$.

As for the query complexity of Algorithm 4 $(1+\epsilon)$-Estimator. In Lemma 9, to compute the estimator $\mathcal{A}$ we use $O(\epsilon^{-8} \cdot \log^6(n) \cdot \sqrt{n})$ RangeCount queries. For the estimator $\mathcal{B}$, we sample $z = 2^{14}\delta_{FL}^2 \log^5(n)\epsilon^{-6} \cdot \sqrt{n}$ cells using Subroutine TelescopingSampling$(\mathcal{G}_i)$ for each grid $\mathcal{G}_{i \leq \frac{3}{4}\log(n)}$. Therefore, we invoke the tester HeavyTester for less than $5z\log(n)$ cells (z times for the sampled cells and $4z$ for their children) where each such a call makes $O(\epsilon^{-8} \cdot \log^6(n))$ RangeCount queries as is shown in Lemma 5. Therefore, in total, the query complexity of the $(1+\epsilon)$-estimator $\mathcal{Z}$ is $\tilde{O}(\sqrt{n})$. This finishes the proof of Theorem 2.

4 Hard instance for the sublinear geometric model

In this section, we prove Lemma 3. See Figure 2 for the illustration of the hard instance that we explain next.

Suppose the opening cost is $f = n^{3/4}$. In the grid $\mathcal{G}_{\frac{3}{4}\cdot\log(n)}$, we have $n^{1/2}$ cells each one having side length $n^{3/4}$. We choose a set $A \subset \mathcal{G}_{\frac{3}{4}\cdot\log(n)}$ of $n^{1/4}$ cells arbitrarily and assign $n^{3/4}$ points to each such a cell. Observe that the sparsity of the grid $\mathcal{G}_{\frac{3}{4}\cdot\log(n)}$ is $n^{1/4}$. That is in average from every $n^{1/4}$ cells in this grid, only one of them has $n^{3/4}$ points and the rest are empty. Note that every cell in A is heavy since it has at least $\frac{f}{n^{3/4}} = 1$ points.

At grids $\mathcal{G}_i$ and $\mathcal{G}_j$ for $i = \frac{\log(n)}{2}$ and $j = \frac{\log(n)}{4}$, a cell has side length $n^{1/2}$ and $n^{1/4}$. We sample a subset $B \subset A$ of $\log(n)$ cells uniformly at random. A cell $c \in B$ has a set $\mathcal{D}_c$ of $\sqrt{n}$ descendants in the grid $\mathcal{G}_{\frac{1}{2}\cdot\log(n)}$, each one having $n^{\frac{1}{4}}$ points equally distant (at distance

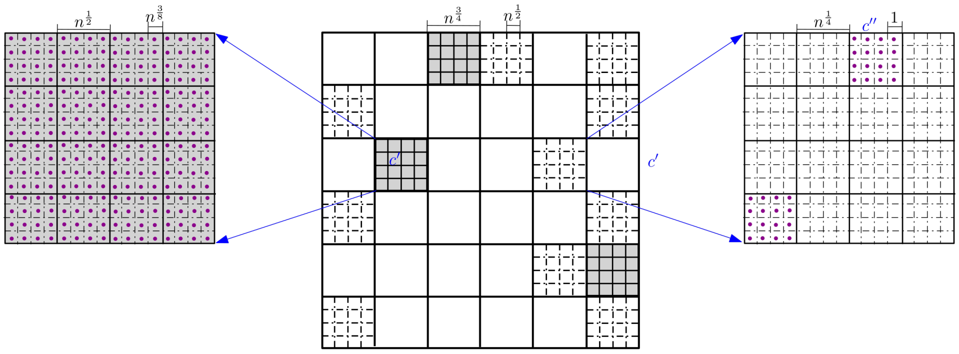


Figure 2 In this figure, cells of B are colored gray and cells in the set $A \backslash B$ are shown with a net of dash-dotted lines. The rest of cells that are shown as white cells are empty cells. The cell $c \in B$ has $\sqrt{n}$ descendants in the grid $\mathcal{G}_{\frac{1}{2}\cdot\log(n)}$ that are all heavy cells. The cell $c' \in A\backslash B$ has n descendants in the grid $\mathcal{G}_{\frac{1}{4}\cdot\log(n)}$ out of which $n^{1/4}$ of them are heavy and the rest are empty cells.

$n^{\frac{3}{8}}$) from its top, bottom, left and right points. Every cell in $\mathcal{D}_c$ has $n^{\frac{1}{4}} = \frac{f}{n^{1/2}}$ points, so it is a heavy cell in the grid $\mathcal{G}_{\frac{1}{2}\cdot\log(n)}$. In the optimal solution for the facility location problem, we must open at least one facility within distance $n^{1/2}$ of each heavy cell in the grid $\mathcal{G}_{\frac{1}{2}\cdot\log(n)}$. Thus, the optimal cost of the facility location problem for the cell set B is at least $\mathcal{L} = \log(n) \cdot \frac{n^{1/2}}{9} \cdot f$ where in the denominator we have 9 since the points inside at most 9 cells (i.e., a grid of 3×3) in the grid $\mathcal{G}_{\frac{1}{2}\cdot\log(n)}$ can be assigned to one open facility that we open in one of them, say the center one.

Each cell $c' \in A\backslash B$ has n descendants in the grid $\mathcal{G}_j$, but only $n^{1/4}$ of them that are chosen uniformly at random are heavy, i.e., has $\frac{f}{n^{1/4}} = n^{1/2}$ points and the rest are empty cells. Suppose that for each heavy descendant (in the grid $\mathcal{G}_j$) of each cell $c' \in A\backslash B$, we open at most one facility. The total cost of cells of the set $A\backslash B$ would be at most $\mathcal{U} \leq |A\backslash B| \cdot n^{1/4} \cdot \frac{f}{n^{1/4}} \cdot n^{1/4}\sqrt{2} \leq \sqrt{2}n^{1/2} \cdot f$

Now, observe that with respect to the grids $\mathcal{G}_{\geq \frac{3}{4}\cdot\log(n)}$, all the cells in the set A look exactly the same and they all have $n^{3/4}$ points each. Since $|B| = \log(n)$ and $|A| = n^{1/4}$, in expectation we need to sample $\frac{|A|}{|B|} = \frac{n^{1/4}}{\log(n)}$ cells so that we can have at least one cell from B in the sampled set; otherwise, we cannot estimate the number of cells and the cost of each cell in the set B and thus, we cannot approximate the cost of the facility location problem within a factor better than $\Omega(\frac{\mathcal{L}}{\mathcal{U}}) = \Omega(\log(n))$.

References

1 Pankaj K. Agarwal. Range searching. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry, Second Edition*, pages 809–837. Chapman and Hall/CRC, 2004. doi:10.1201/9781420035315.ch36.

2 Pankaj K. Agarwal, Edward F. Grove, T. M. Murali, and Jeffrey Scott Vitter. Binary search partitions for fat rectangles. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 482–491. IEEE Computer Society, 1996. doi:10.1109/SFCS.1996.548507.

3 N. Alon and J. Spencer. *The probabilistic method.* J. Wiley & Sons, New York, 2nd edition, 2000.

4 Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 574–583. ACM, 2014. doi:10.1145/2591796.2591805.

5 Boris Aronov, Esther Ezra, and Micha Sharir. Small-size epsilon-nets for axis-parallel rectangles and boxes. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 639–648. ACM, 2009. doi:10.1145/1536414.1536501.

6 Boris Aronov, Esther Ezra, and Micha Sharir. Small-size ϵ-nets for axis-parallel rectangles and boxes. *SIAM J. Comput.*, 39(7):3248–3282, 2010. doi:10.1137/090762968.

7 Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for euclidean k-medians and related problems. In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 106–113. ACM, 1998. doi:10.1145/276698.276718.

8 Mihai Badoiu, Artur Czumaj, Piotr Indyk, and Christian Sohler. Facility location in sublinear time. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 866–877. Springer, 2005. doi:10.1007/11523468_70.

9 Paul Beame, Paraschos Koutris, and Dan Suciu. Communication steps for parallel query processing. *J. ACM*, 64(6):40:1–40:58, 2017. doi:10.1145/3125644.

10 Soheil Behnezhad. Time-optimal sublinear algorithms for matching and vertex cover. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 873–884. IEEE, 2021. doi:10.1109/FOCS52979.2021.00089.

11 Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975. doi:10.1145/361002.361007.

12 Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM J. Comput.*, 34(6):1370–1379, 2005. doi:10.1137/S0097539702403244.

13 Herman Chernoff. A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952. doi:10.1214/aoms/1177729330.

14 Artur Czumaj, Funda Ergün, Lance Fortnow, Avner Magen, Ilan Newman, Ronitt Rubinfeld, and Christian Sohler. Approximating the weight of the euclidean minimum spanning tree in sublinear time. *SIAM J. Comput.*, 35(1):91–109, 2005. doi:10.1137/S0097539703435297.

15 Artur Czumaj, Christiane Lammersen, Morteza Monemizadeh, and Christian Sohler. $(1+\epsilon)$-approximation for facility location in data streams. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1710–1728. SIAM, 2013. doi:10.1137/1.9781611973105.123.

16 Artur Czumaj and Christian Sohler. Estimating the weight of metric minimum spanning trees in sublinear-time. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 175–183. ACM, 2004. doi:10.1145/1007352.1007386.

17 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008. URL: https://www.worldcat.org/oclc/227584184.

18 Gereon Frahling, Piotr Indyk, and Christian Sohler. Sampling in dynamic data streams and applications. In Joseph S. B. Mitchell and Günter Rote, editors, *Proceedings of the 21st ACM Symposium on Computational Geometry, Pisa, Italy, June 6-8, 2005*, pages 142–149. ACM, 2005. doi:10.1145/1064092.1064116.

19 Gereon Frahling and Christian Sohler. Coresets in dynamic geometric data streams. 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 209–217. ACM, 2005. doi:10.1145/1060590.1060622.

20 Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, searching, and simulation in the mapreduce framework. In Takao Asano, Shin-Ichi Nakano, Yoshio Okamoto, and Osamu Watanabe, editors, *Algorithms and Computation - 22nd International Symposium, ISAAC 2011, Yokohama, Japan, December 5-8, 2011. Proceedings*, volume 7074 of *Lecture Notes in Computer Science*, pages 374–383. Springer, 2011. doi:10.1007/978-3-642-25591-5_39.

21 Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 291–300. ACM, 2004. doi:10.1145/1007352.1007400.

22 Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. URL: http://www.jstor.org/stable/2282952.

23 Piotr Indyk. A sublinear time approximation scheme for clustering in metric spaces. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 154–159. IEEE Computer Society, 1999. doi:10.1109/SFFCS.1999.814587.

24 Piotr Indyk. Algorithms for dynamic geometric problems over data streams. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 373–380. ACM, 2004. doi:10.1145/1007352.1007413.

25 Piotr Indyk. A near linear time constant factor approximation for euclidean bichromatic matching (cost). 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 39–42. SIAM, 2007. URL: http://dl.acm.org/citation.cfm?id=1283383.1283388.

26 Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001. doi:10.1145/375827.375845.

27 Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for mapreduce. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 938–948. SIAM, 2010. doi:10.1137/1.9781611973075.76.

28 Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory.* MIT Press, 1994. URL: https://mitpress.mit.edu/books/introduction-computational-learning-theory.

29 Michael J. Kearns and Umesh V. Vazirani. Computational learning theory. *SIGACT News*, 26(1):43–45, 1995. doi:10.1145/203610.606411.

30 Stavros G. Kolliopoulos and Satish Rao. A nearly linear-time approximation scheme for the euclidean k-median problem. *SIAM J. Comput.*, 37(3):757–782, 2007. doi:10.1137/S0097539702404055.

31 Philip M. Long and Lei Tan. Pac learning axis-aligned rectangles with respect toproduct distributions from multiple-instance examples. *Mach. Learn.*, 30(1):7–21, January 1998. doi:10.1023/A:1007450326753.

32 Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995. doi:10.1017/cbo9780511814075.

33 Huy N. Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 327–336. IEEE Computer Society, 2008. doi:10.1109/FOCS.2008.81.

34 Sharath Raghvendra and Pankaj K. Agarwal. A near-linear time ϵ-approximation algorithm for geometric bipartite matching. *J. ACM*, 67(3):18:1–18:19, 2020. doi:10.1145/3393694.

35 Menachem Sadigurschi and Uri Stemmer. On the sample complexity of privately learning axis-aligned rectangles. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 28286–28297, 2021. URL: `https://proceedings.neurips.cc/paper/2021/hash/ee0e95249268b86ff2053bef214bfeda-Abstract.html`.

36 Srikanta Tirthapura and David P. Woodruff. Rectangle-efficient aggregation in spatial data streams. In Michael Benedikt, Markus Krötzsch, and Maurizio Lenzerini, editors, *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 283–294. ACM, 2012. `doi:10.1145/2213556.2213595`.

37 Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, 2 edition, September 2000.

38 Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. An improved constant-time approximation algorithm for maximum matchings. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 225–234. ACM, 2009. `doi:10.1145/1536414.1536447`.

A Missing proof of Subsection 3.1

Proof of Lemma 5. Testing whether a cell c is a heavy or a light cell needs to find out if c has a facility location cost of at least $\delta_{FL} f$ or lower than this quantity. To this end, we first develop a deterministic sublinear algorithm for the k-median problem that using $O(k\epsilon^{-4}\log^2(n))$ RangeCount queries returns a $(1+\epsilon)$-approximate solution. We then show how we can use this sublinear algorithm to develop the tester algorithm HeavyTester.

What is interesting is the difference between the query complexity of the k-median and the facility location problems in the sublinear geometric model. For the k-median problem, we can report a $(1+\epsilon)$-approximate solution (not the cost) in the sublinear model using $O(k\epsilon^{-4}\log^2(n))$ RangeCount queries. However, for the facility location, Lemma 3 shows that we do not hope to $o(\log n)$-approximate the cost of the facility location problem in the sublinear geometric model using $\Omega(\frac{\sqrt[4]{n}}{\log n})$ RangeCount queries.

▶ **Definition 19** (k-median problem). *In the k-*median problem*, we are given a point set $P \subset [2n]^2$ of size n in a discrete space and a number $k \in \mathbb{N}$ of centers. The goal is to return a set $C^* \subset [2n]^2$ of k centers that minimizes the cost function $cost_{k\text{-}med}(P, C^*) = \sum_{p\in P} dist(p, C^*)$, where $dist(p, C^*) = \min_{c\in C^*} dist(p, c)$ is the Euclidean distance of p to its nearest center in C^*. We denote the optimal k-median cost of the point set P by $OPT_{k\text{-}med}(P, k)$.*

▶ **Lemma 20** (Sublinear algorithm for k-median). *Let P be a point set of size n in $[2n]^2$. Let $k > 0$ be the number of centers and $0 < \epsilon \leq 1$ be the error parameter. Then, there exists a deterministic sublinear algorithm that returns a $(1+\epsilon)$-approximate solution of the k-median problem for P in the sublinear geometric model using $O(k\epsilon^{-4}\log^2(n))$ RangeCount queries.*

Assume for a moment that this lemma is correct. We next explain the tester algorithm.

Algorithm HeavyTester(c). For every choice $k \in \{1, 2, \cdots, \delta_{FL} = 2^{20} \cdot (\frac{\log n}{\epsilon})^2\}$, we run the sublinear algorithm of Lemma 20 on the input set $P \cap c$ that reports a set C_k of k centers. We then compute the k-median cost of $P \cap c$ using the center set C_k and add the opening cost kf to the k-median cost to compute the facility location cost. Among all δ_{FL} runs, we find the one that has the lowest facility location cost. If the lowest facility location cost is less than δ_{FL}, we report that the cell c is a light cell, otherwise we report it as a heavy cell. This finishes the description of the algorithm HeavyTester(c).

Query complexity of HeavyTester(c). We invoke δ_{FL} parallel runs each choice $k \in [\delta_{FL}]$. Each such a run needs $O(k\epsilon^{-4}\log^2(n))$ RangeCount queries using Lemma 20. Thus, using $O(\delta_{FL}^2 \cdot \epsilon^{-4}\log^2(n)) = O(\epsilon^{-8} \cdot \log^6(n))$ RangeCount queries, we can detect if the cell c is a heavy cell or a light one. This finishes the proof of the first part of Lemma 5.

Now we prove the second part of this lemma. Imagine the case when c is a cell in the set Λ. Recall that $c \in \Lambda$ if c is a light cell and its *parent*(c) is a heavy cell. We run the tester HeavyTester for c and *parent*(c). This tester is a deterministic algorithm and so, it correctly reports c is a light cell and *parent*(c) is a heavy cell. Lemma 4 ensures that for the correct guess of the number of facilities in c, the set F_c of facilities that the tester HeavyTester(c) returns will satisfy the following what finishes the proof of Lemma 5. $\text{cost}^{\epsilon}_{FL}(c, F_c, f) = \sum_{p\in P\cap c} \text{dist}(p, F_c) + |F_c| \cdot f \leq (1+\epsilon) \cdot OPT_{FL}(P \cap c, f)$.

Proof of Lemma 20. We prove this lemma using a modification that we do to the quadtree construction that Frahling and Sohler [19] developed for the k-median problem. The quadtree construction in [19] is based on the notions of dense and sparse cells.

Dense and sparse cells: Let P be a point set of size n in $[2n]^2$. Let $0 < \epsilon \leq 1$ be an error parameter. Let $\delta_{k\text{-}med} = 2^{20} \cdot \frac{k\log(n)}{\epsilon^3}$. We say a cell $c \in \mathcal{G}_i$ of side length 2^i is *dense* if it contains $n_c \geq \delta_{k\text{-}med} \cdot \frac{OPT_{k\text{-}med}(P,k)}{2^i}$ points; otherwise it is a *sparse* cell.

Observe that for the definition of dense and sparse cells, we assume we know the optimal k-median cost $OPT_{k\text{-}med}(P,k)$. In general, we can $(1+\epsilon)$-approximate $OPT_{k\text{-}med}(P,k)$ using $\log_{(1+\epsilon)}(4n^2) = O(\epsilon^{-1}\log(n))$ guesses. To see this, observe that in the discrete space $[2n]^2$, the maximum pairwise distance between two points is at most $4n$ and the minimum distance is 1. The set P has n points, so the ratio between the maximum and the minimum k-median costs is at most $n \times 4n = 4n^2$. We then consider $t = \log_{(1+\epsilon)}(4n^2) = O(\epsilon^{-1}\log(n))$ parallel guesses $r_0, \cdots, r_t$ for the optimal k-median cost $OPT_{k\text{-}med}(P,k)$ where $r_j = (1+\epsilon)^j$. We must have an index $j \in [t]$ for which $(1+\epsilon)^j \leq OPT_{k\text{-}med}(P,k) < (1+\epsilon)^{j+1}$.

Sublinear algorithm for k-median based on quadtree construction of [19]. We run the following algorithm for all $O(\epsilon^{-1}\log(n))$ guesses of $OPT_{k\text{-}med}(P,k)$ in parallel. Let us consider the j^{th} guess $r_j = (1+\epsilon)^j$ for $OPT_{k\text{-}med}(P,k)$. We create a run j for the j^{th} guess for which we do the following.

Let $\mathcal{K}_j$ and $\mathcal{R}_j$ be two empty sets. Given the single square $c = [2n]^2$ in the grid $\mathcal{G}_{\log(n)+1}$, we build a tree similar to the quadtree as follows. In particular, suppose that we have a cell $c \in \mathcal{G}_i$. If c is sparse, we add c to $\mathcal{K}_j$ and stop; otherwise, (i.e., if c is dense), we split it into 4 equal sub-cells c_1, c_2, c_3, c_4 (they are in the grid G_{i-1}) of the same side length and recurse for those sub-cells that are dense. We add all **non-empty sparse cells** that are constructed in this way to $\mathcal{K}_j$. If during the run j, the size of $\mathcal{K}_j$ is more than $O(k\epsilon^{-3}\log(n))$, we immediately stop that run because that run corresponds to the guess $(1+\epsilon)^j$ that is much smaller than the optimal k-median cost $OPT_{k\text{-}med}(P,k)$. At the end of this recursive procedure, for each cell $c \in \mathcal{K}_j$, we consider the center of c as the representative point r_c of c and assign the weight $w(r_c) = n_c = |P \cap c|$ to r_c. Observe that for every cell $c \in \mathcal{K}_j$, we need one RangeCount query to find out $w(r_c) = n_c = |P \cap c|$. Let $\mathcal{R}_j$ be the set of weighted representative points that we compute in this way for the run j.

Next, we need to compute the k-median cost of the representative set $\mathcal{R}_j$. To this end, we use the deterministic $(1+\epsilon)$-approximation algorithm that Har-Peled and Mazumdar [21] develop for the k-median problem. The running time of this algorithm is

$O(n + 2^{O((1+1/\epsilon)}k^{O(1)}\log^{O(1)}(n))$. We denote this algorithm by $\textsc{Alg}^{\epsilon}_{k\text{-}med}$. For the weighted representative set $\mathcal{R}_j$, we then invoke the $\textsc{Alg}^{\epsilon}_{k\text{-}med}(\mathcal{R}_j)$ that returns a set C_j of k centers and its k-median cost $\Gamma_j = \mathrm{cost}_{k\text{-}med}(\mathcal{R}_j, C_j) = \sum_{r_c \in \mathcal{R}_j} w(r_c) \cdot \mathrm{dist}(p, C_j)$, where $\mathrm{dist}(p, C_j) = \min_{c \in C_j} \mathrm{dist}(p, c)$ is the distance of p to its nearest center in C_j.

Among all parallel guesses $r_0, \cdots, r_t$ where $[t = O(\epsilon^{-1}\log(n))]$, we choose the smallest guess $r_{j^*} = (1+\epsilon)^{j^*}$ for which we have $(1+\epsilon)^{j^*} \le \Gamma_{j^*} < (1+\epsilon)^{j^*+1}$. Let $\mathcal{K}_{j^*}$ and $\mathcal{R}_{j^*}$ be the set of non-empty sparse cells and the set of representative points that correspond to the guess $(1+\epsilon)^{j^*}$, respectively. We then have the following guarantee.

▶ **Lemma 21** ([19]). *Let $P \subseteq [2n]^2$ be a point set of size n and $k \in \mathbb{N}$ be a natural number. Let $0 < \epsilon \le 1$ be the error parameter. Then:*

- *The number of non-empty sparse cells in $\mathcal{K}_{j^*}$ (or, the number of representative points in $\mathcal{R}_{j^*}$) that are found using the above construction is $|\mathcal{K}_{j^*}| = |\mathcal{R}_{j^*}| = O(k\epsilon^{-3}\log(n))$.*
- *The output of Algorithm [21] $\textsc{Alg}^{\epsilon}_{k\text{-}med}$ on input set $\mathcal{R}_{j^*}$ is a set C_{j^*} of k centers such that $(1-\epsilon)OPT_{k\text{-}med}(P, k) \le \Gamma_{j^*} = cost_{k\text{-}med}(\mathcal{R}_{j^*}, C_{j^*}) \le (1+\epsilon)OPT_{k\text{-}med}(P, k)$.*

Query Complexity of sublinear algorithm for k-median. Let us consider a run j. Recall that if during the run j, the size of $\mathcal{K}_j$ is more than $O(k\epsilon^{-3}\log(n))$, we immediately stop this run. Observe that for every cell $c \in \mathcal{K}_j$, we need one $\textsc{RangeCount}$ query to find out its weight $w(r_c) = n_c = |P \cap c|$. Thus, the number of $\textsc{RangeCount}$ queries that we make to build the set $\mathcal{K}_j$ is $O(k\epsilon^{-3}\log(n))$. In addition, observe that the algorithm $\textsc{Alg}^{\epsilon}_{k\text{-}med}(\mathcal{R}_j)$ does not make any $\textsc{RangeCount}$ query. We have $t = O(\epsilon^{-1}\log(n))$ guesses for the optimal k-median cost $OPT_{k\text{-}med}(P, k)$. Thus, the number of $\textsc{RangeCount}$ queries that we make to develop our sublinear algorithm for the k-median problem is $O(k\epsilon^{-4}\log^2(n))$. This finishes the proof of Lemma 20. ◀

◀

B Missing proofs of Subsection 3.3

Proof of Lemma 7. Let us consider the ancestors $c_{i+1}, \cdots, c_j, \cdots, c_{\log(n)}$ where cell c_j for $i+1 \le j \le \log(n)$ is the ancestor of cell c in grid $\mathcal{G}_j$. Observe that the probability that we sample the cell c is $\mathbf{Pr}[c] = \mathbf{Pr}\left[c_{\log(n)-1} | c_{\log(n)}\right] \cdot \mathbf{Pr}\left[c_{\log(n)-2} | c_{\log(n)-1}\right] \cdots \mathbf{Pr}\left[c_{i+1} | c_{i+2}\right] \cdot \mathbf{Pr}\left[c | c_{i+1}\right] = \frac{n_{c_{\log(n)-1}}}{n} \cdot \frac{n_{c_{\log(n)-2}}}{n_{c_{\log(n)-1}}} \cdots \frac{n_{c_{i+1}}}{n_{c_{i+2}}} \cdot \frac{n_c}{n_{c_{i+1}}} = \frac{n_c}{n}$. ◀

C Missing proofs of Subsection 3.5.1

Proof of Lemma 9. Recall that the estimator $\mathcal{A}$ is for the contribution of those light cells in Λ that are in the grids $\mathcal{G}_{\frac{3}{4}\cdot\log(n) \le i \le \log(n)+1}$. Let S be the set of all cells in $\mathcal{G}_{\frac{3}{4}\cdot\log(n) \le i \le \log(n)+1}$. In total, all these grids have at most $|S| \le \frac{2n^2}{n^{6/4}} + \frac{1}{4} \cdot \frac{2n^2}{n^{6/4}} + \cdots + \frac{1}{4^i} \cdot \frac{2n^2}{n^{6/4}} + \cdots + 1 \le 4\sqrt{n}$ cells. For every heavy cell $c \in S$, its sub-cells c_1, c_2, c_3, c_4 are created and tested if they are heavy or light cell. Recall that we run the $\textsc{HeavyTester}$ algorithm for those sub-cells to determine which ones are heavy or light. Using Lemma 5, to detect if a cell c is heavy or light, the tester algorithm $\textsc{HeavyTester}(c)$ uses $O(\epsilon^{-8} \cdot \log^6(n))$ $\textsc{RangeCount}$ queries.

Let us consider an empty set T in the beginning. For each heavy cell $c \in S$, we add its light sub-cells to the set T. The first claim of this lemma is proven by observing that $|T| \le 4|S| = 16\sqrt{n}$. As for the second claim, we invoke the $\textsc{HeavyTester}$ algorithm for $|S| + |T| \le 20\sqrt{n}$ cells each one uses $O(\epsilon^{-8} \cdot \log^6(n))$ $\textsc{RangeCount}$ queries. ◀

Proof of Lemma 10. Based on Lemma 5, the tester algorithm HeavyTester is a deterministic algorithm that correctly detects if an arbitrary cell c is heavy or light. Thus, Algorithm 4 $(1+\epsilon)$-Estimator detects all light cells $\Lambda_{\mathcal{A}}$. For every light cell $c \in \Lambda_{\mathcal{A}}$, Lemma 5 shows that the Tester algorithm $(1+\epsilon)$-approximates the optimal cost $OPT^{\epsilon}_{FL}(P \cap c, f)$. Thus, the term $\mathcal{A}$ is a $(1+\epsilon)$-estimator for the cost of light cells of the set $\Lambda_{\mathcal{A}}$. ◀

D Missing proofs of Subsection 3.5.2

The following lemma shows that we can safely remove insignificant light cells from the set Λ and only consider significant light cells Ψ.

▶ **Lemma 22.** *Let $\tau = \frac{\epsilon}{9\log(n)}$. Let $\Psi = \{c \in \Lambda : cost^{\epsilon}_{FL}(c, F_c, f) \geq \tau f\}$ be the set of light cells in Λ that are significant. Then,*

$$OPT_{FL}(P, f) \leq cost^{\epsilon}_{FL}(\Psi, f) = \sum_{c \in \Psi} cost^{\epsilon}_{FL}(c, F_c, f) \leq (1+\epsilon) \cdot OPT_{FL}(P, f) \ .$$

Proof. In order to prove Lemma 22, we first define charging heavy cells. We prove that at most $3\log(n)$ insignificant light cells are created during the construction of any charging heavy cells. We can assign all these insignificant light cells to the charging heavy cell whose construction creates these cells. We show that the cost of a charging heavy c is at least $\Omega(\delta_{FL} \cdot f)$. On the other hand, overall the total facility location cost of the insignificant light cells that are assigned to c is at most $\epsilon \cdot f$ that can be charged to $\mathrm{cost}^{\epsilon}_{FL}(c, F_c, f)$.

▶ **Definition 23** (Charging heavy cell). *Let c be a heavy cell having children c_1, c_2, c_3, c_4. We say c is a charging heavy cell if all children c_1, c_2, c_3, c_4 are light.*

▶ **Lemma 24.** *Let c be a heavy cell. Then, either c is a charging heavy cell or at least one of its descendants is a charging heavy cell.*

Proof. For the sake of contradiction assume that c is not a charging heavy cell. As otherwise, we have nothing to prove. Let c' be a copy of c. Let c_1, c_2, c_3, c_4 be the four children of c'. Let $C^H_{c'}$ be the set of heavy children of c'. By our assumption, $|C^H_c| > 0$. Let us pick a heavy child of c' from C^H_c, say c_1 is heavy. We let $c' = c_1$. Recursively, either all children of c' are light or at least one of the children of c' is heavy for which we recurse. This recursion repeats for $O(\log n)$ times. At the end we either find a charging heavy cell or arrive at a cell $c \in \mathcal{G}_0$ that has side length one and can store only one point p_c. If that cell c is still heavy, we can open a facility at p_c of cost f which contradicts with the fact that the cost of c must be at least $\delta_{FL} \cdot f$. Thus, the recursion will end by outputting a charging heavy cell. ◀

Let us fix a charging heavy cell c at a grid $\mathcal{G}_i$. Let us consider the quadtree construction that we explained for the facility location problem in Section 2.1. Starting from the root of the quadtree that corresponds to the square $[2n]^2$ going down to the cell c, we see the ancestors of c. At any grid $\mathcal{G}_{j>i}$, the ancestor(c, j) of c creates 4 children. At most 3 of them are insignificant light cells and one is the ancestor$(c, j-1)$ of c at grid $\mathcal{G}_{j-1}$ which is a heavy cell. Suppose we assign all the insignificant light cells that ancestor(c, j) creates to c. If there are more than one heavy cell among the children of ancestor(c, j), we can arbitrarily assign the insignificant light children of ancestor(c, j) to either of them. Let $\mathcal{O}_c$ be the set of all insignificant light cells that are assigned to c. We have the following bound for $|\mathcal{O}_c|$.

▶ **Corollary 25.** *Let c be a charging heavy cell. Let $\mathcal{O}_c$ be the set of all insignificant light cells that are assigned to c. Then, $|\mathcal{O}_c| \leq 3\log(n)$.*

▶ **Corollary 26.** *Let c be a charging heavy cell. Let $\mathcal{O}_c$ be the set of all insignificant light cells that are assigned to c. Then, the total facility location cost of insignificant light cells $\mathcal{O}_c$ is $cost^{\epsilon}_{FL}(\mathcal{O}_c, f) = \sum_{c' \in \mathcal{O}_c} cost^{\epsilon}_{FL}(c', F_{c'}, f) \leq \frac{\epsilon}{3} \cdot f$.*

Now we finish the proof of Lemma 22. Recall that the charging heavy cell c has cost $\text{cost}^{\epsilon}_{FL}(c, F_c, f) \geq \delta_{FL} \cdot f$. On the other hand, the four children $C_c = \{c_1, c_2, c_3, c_4\}$ of c are light that are in the set Λ as we have seen in Lemma 4. This lemma shows that for each cell $c' \in C_c$ we must have $\text{cost}^{\epsilon}_{FL}(c', F_c, f) \leq (1+\epsilon) \cdot OPT_{FL}(P \cap c', f)$. However, $\sum_{c' \in C_c} \text{cost}^{\epsilon}_{FL}(c', F_c, f) \geq \frac{\delta_{FL} \cdot f}{4}$. Otherwise, we can upper bound $\text{cost}^{\epsilon}_{FL}(c, F_c, f)$ by $\sum_{c' \in C_c} \text{cost}^{\epsilon}_{FL}(c', F_c, f) \leq \frac{\delta_{FL} \cdot f}{4}$ which essentially means that c is light cell contradicting our assumption that c is a charging heavy cell. Thus, $\frac{\delta_{FL} \cdot f}{4(1+\epsilon)} \leq \frac{\sum_{c' \in C_c} \text{cost}^{\epsilon}_{FL}(c', F_c, f)}{(1+\epsilon)} \leq \min(\sum_{c' \in C_c} OPT_{FL}(P \cap c', f), OPT_{FL}(P \cap c', f))$. Using Corollary 26, for the set $\mathcal{O}_c$ of insignificant light cells that are assigned to c we have $\text{cost}^{\epsilon}_{FL}(\mathcal{O}_c, f) \leq \frac{\epsilon}{3} \cdot f$ which is less than ϵ-fraction of the optimal facility location cost $OPT_{FL}(c, f)$ of c. Therefore, the overall facility location cost of insignificant light cells in Λ is at most $\epsilon \cdot OPT_{FL}(P, f)$. Thus, for set $\Psi = \{c \in \Lambda : \text{cost}^{\epsilon}_{FL}(c, F_c, f) \geq \tau f\}$ of significant light cells we have $OPT_{FL}(P, f) \leq \text{cost}^{\epsilon}_{FL}(\Psi, f) = \sum_{c \in \Psi} \text{cost}^{\epsilon}_{FL}(c, F_c, f) \leq (1+\epsilon) \cdot OPT_{FL}(P, f)$. ◀

Next lemma shows that the cost of non-contributing classes Ψ^j_i is only $\frac{\epsilon}{2}$-fraction of the optimal facility location cost of point set P. Thus, when we develop a sublinear algorithm that $(1+\epsilon)$-approximates $OPT_{FL}(P, f)$, we can ignore the cost of non-contributing classes.

▶ **Lemma 27.** *Suppose we know that $OPT_{FL}(P, f) \geq \sqrt{n} \cdot f$. Let Ψ^j_i be a likelihood class that is non-contributing, i.e., $|\Psi^j_i| < \beta\sqrt{n}$, where $\beta = \frac{\epsilon^2}{2\delta_{FL} \cdot \log^2(n)}$. We can safely ignore the contribution of the class Ψ^j_i toward the optimal facility location cost of the point set P.*

Proof. Recall that the class Ψ^j_i consists of only light cells and that a cell c is light if $\text{cost}^{\epsilon}_{FL}(c, F_c, f) < \delta_{FL} \cdot f$. Recall that we test whether a cell is light or heavy using the tester algorithm HeavyTester of Lemma 5. Since $|\Psi^j_i| < \beta\sqrt{n}$, for $\beta = \frac{\epsilon^2}{2\delta_{FL} \cdot \log^2(n)}$, the overall cost of the class Ψ^j_i is upper-bounded by $\text{cost}^{\epsilon}_{FL}(\Psi^j_i, f) = \sum_{c \in \Psi^j_i} \text{cost}^{\epsilon}_{FL}(c, F_c, f) \leq \frac{\epsilon^2}{2\delta_{FL} \cdot \log^2(n)} \cdot \sqrt{n} \cdot \delta_{FL} \cdot f \leq \frac{\epsilon^2}{2\log^2 n} \cdot OPT_{FL}(P, f)$.

Note that based on Definition 13, we can have at most $\epsilon^{-1} \cdot \log(n)$ classes for every grid $\mathcal{G}_i$ and we have at $\log(n) + 1$ grids. Thus, the total cost of non-contributing classes is at most $\sum_{i=1}^{\log(n)+1} \sum_{j=1}^{\epsilon^{-1} \cdot \log(n)} \text{cost}^{\epsilon}_{FL}(\Psi^j_i, f) \leq \frac{\epsilon}{2} \cdot OPT_{FL}(P, f)$. Thus we can safely ignore the contribution of these grids toward the optimal facility location cost of the point set P. ◀

We first find a lower bound for the cost of contributing classes. Next, we obtain a lower bound for the minimum number of points in a contributing likelihood class.

▶ **Lemma 28.** *Let $\alpha = \frac{\epsilon^3}{18\delta_{FL} \log^3(n)}$ and $\beta = \frac{\epsilon^2}{2\delta_{FL} \cdot \log^2(n)}$. Let Ψ^j_i be a likelihood class that is contributing, i.e., $|\Psi^j_i| \geq \beta\sqrt{n}$. Then, $cost^{\epsilon}_{FL}(\Psi^j_i, f) = \sum_{c \in \Psi^j_i} cost^{\epsilon}_{FL}(c, F_c, f) \geq \alpha\sqrt{n} \cdot f$.*

Proof. A class Ψ^j_i is contributing if $|\Psi^j_i| \geq \beta\sqrt{n}$. Moreover, every class Ψ^j_i consists of light cells that are significant. A light cell $c \in \Lambda$ is significant if $\text{cost}^{\epsilon}_{FL}(c, F_c, f) \geq \tau f$, where $\tau = \frac{\epsilon}{9\log(n)}$. Thus, we have the following lower bound for the facility location cost of the class Ψ^j_i: $\text{cost}^{\epsilon}_{FL}(\Psi^j_i, f) = \sum_{c' \in \Psi^j_i} \text{cost}^{\epsilon}_{FL}(c', f) \geq |\Psi^j_i| \cdot \tau \cdot f \geq \frac{\epsilon^3}{18\delta_{FL} \log^3(n)} \cdot \sqrt{n} \cdot f$. ◀

Next, using the lower bound that we obtained for the cost of contributing classes, we find a lower-bound on the number of points in every contributing class. This will help us to prove that can sample cells of contributing classes almost uniformly at random.

▶ **Proposition 29.** *For a contributing likelihood class Ψ_i^j we have $n(\Psi_i^j) = \sum_{c \in \Psi_i^j} n_c \geq \frac{\alpha\sqrt{n}}{(1+\epsilon)}$.*

Proof. Let us consider a significant light cell $c \in \Psi_i^j$. Since Algorithm [15] $\textsc{Alg}_{FL}^{\epsilon}$ is a $(1+\epsilon)$-approximation algorithm for the facility location problem, we have $\text{cost}_{FL}^{\epsilon}(c', f) \leq (1+\epsilon) \cdot OPT_{FL}(P \cap c, f)$. This essentially means that $OPT_{FL}(P \cap c, f) \geq \frac{\text{cost}_{FL}^{\epsilon}(c',f)}{(1+\epsilon)} \geq \frac{\tau}{(1+\epsilon)} \cdot f$.

Now we take the sum over the cells of Ψ_i^j, apply Lemma 28, and let $\alpha = \frac{\epsilon^3}{18\delta_{FL}\log^3(n)}$ to obtain $\sum_{c \in \Psi_i^j} OPT_{FL}(P \cap c, f) \geq \sum_{c \in \Psi_i^j} \frac{\text{cost}_{FL}^{\epsilon}(c',f)}{(1+\epsilon)} \geq |\Psi_i^j| \cdot \frac{\tau}{(1+\epsilon)} \cdot f \geq \frac{\alpha\sqrt{n}}{(1+\epsilon)} \cdot f$.

We claim that $n(\Psi_i^j) = \sum_{c \in \Psi_i^j} n_c \geq \frac{\alpha\sqrt{n}}{(1+\epsilon)}$. As for the contradiction, suppose this claim is not correct. That is assume that $n(\Psi_i^j) < \frac{\alpha\sqrt{n}}{(1+\epsilon)}$. Then, imagine that we open one facility for each point in Ψ_i^j. Since we assume that $n(\Psi_i^j) \leq \frac{\alpha\sqrt{n}}{(1+\epsilon)}$, the overall opening cost (in fact, the facility location cost since the connection cost is zero) of Ψ_i^j is less than $\frac{\alpha\sqrt{n}}{(1+\epsilon)} \cdot f$ which can not be the case. Thus, overall the cells in Ψ_i^j must have at least $\frac{\alpha\sqrt{n}}{(1+\epsilon)}$ points. ◀

Bicriteria Approximation Algorithms for Priority Matroid Median

Tanvi Bajpai ✉
Department of Computer Science, University of Illinois, Urbana-Champaign, Urbana, IL, USA

Chandra Chekuri ✉
Department of Computer Science, University of Illinois, Urbana-Champaign, Urbana, IL, USA

Abstract

Fairness considerations have motivated new clustering problems and algorithms in recent years. In this paper we consider the Priority Matroid Median problem which generalizes the Priority k-Median problem that has recently been studied. The input consists of a set of facilities $\mathcal{F}$ and a set of clients $\mathcal{C}$ that lie in a metric space $(\mathcal{F} \cup \mathcal{C}, d)$, and a matroid $\mathcal{M} = (\mathcal{F}, \mathcal{I})$ over the facilities. In addition, each client j has a specified radius $r_j \geq 0$ and each facility $i \in \mathcal{F}$ has an opening cost $f_i > 0$. The goal is to choose a subset $S \subseteq \mathcal{F}$ of facilities to minimize $\sum_{i \in \mathcal{F}} f_i + \sum_{j \in \mathcal{C}} d(j, S)$ subject to two constraints: (i) S is an independent set in $\mathcal{M}$ (that is $S \in \mathcal{I}$) and (ii) for each client j, its distance to an open facility is at most r_j (that is, $d(j, S) \leq r_j$). For this problem we describe the first bicriteria (c_1, c_2) approximations for fixed constants c_1, c_2: the radius constraints of the clients are violated by at most a factor of c_1 and the objective cost is at most c_2 times the optimum cost. We also improve the previously known bicriteria approximation for the uniform radius setting ($r_j := L \; \forall j \in \mathcal{C}$).

2012 ACM Subject Classification Theory of computation → Facility location and clustering

Keywords and phrases k-median, fair clustering, matroid

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.7

Category APPROX

Related Version *Full Version*: https://arxiv.org/abs/2210.01888

Funding *Tanvi Bajpai*: Supported in part by NSF grant CCF-1910149.
Chandra Chekuri: Supported in part by NSF grants CCF-1910149 and CCF-1907937.

1 Introduction

Clustering and facility-location problems are widely studied in areas such as machine learning, operations research, and algorithm design. Among these, center-based clustering problems in metric spaces form a central topic and will be our focus. The input for these problems is a set of clients $\mathcal{C}$ and a set of facilities $\mathcal{F}$ from a metric space $(\mathcal{F} \cup \mathcal{C}, d)$. The goal is to select a subset of facilities $S \subseteq \mathcal{F}$ to open, subject to various constraints, so as to minimize an objective that depends on the distances of the clients to the chosen centers; we use $d(j, S)$ to denote the quantity $\min_{i \in S} d(j, i)$ which is the distance from j to S. Typical objectives are of the form $(\sum_{j \in \mathcal{C}} d(j, S)^p)^{1/p}$ for some parameter p (the ℓ_p norm of the distances). When the constraint on facilities is that at most k can be chosen (that is, $|S| \leq k$), we obtain several standard and well-studied problems such as k-Center ($p = \infty$), k-Median ($p = 1$), and k-Means ($p = 2$) problems. These problems are extensively studied from many perspectives [15, 25, 10, 2, 5, 19, 16]. These are also well-studied in the geometric setting when $\mathcal{F}$ is the continuous space $\mathbb{R}^\ell$ for some finite dimension ℓ. In this paper we restrict our attention to the discrete setting, and in particular, to the median objective ($p = 1$).

The Matroid Median problem is a generalization of the k-Median clustering problem. Here, the cardinality constraint k on S is replaced by specifying a matroid $\mathcal{M} = (\mathcal{F}, \mathcal{I})$ on the facility set $\mathcal{F}$ and requiring that $S \in \mathcal{I}$ (we refer a reader unfamiliar with matroids to

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 7; pp. 7:1–7:22
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Section 2 formal definitions and details). The k-Median clustering problem can be written as an instance of Matroid Median where $\mathcal{M}$ is the uniform matroid of rank k. The Matroid Median problem was first introduced by Krishnaswamy et al. [21] as a generalization of k-Median and Red-Blue Median [14].

Motivated by the versatility of the Matroid Median problem, and several other considerations that we will discuss shortly, in this paper we study the Priority Matroid Median problem (PMatMed). Formally, in PMatMed we are given a set of clients $\mathcal{C}$ and facilities $\mathcal{F}$ from a metric space $(\mathcal{F} \cup \mathcal{C}, d)$ where each facility $i \in \mathcal{F}$ has a facility opening cost f_i, and each client $j \in \mathcal{C}$ has a radius value r_j. We are also given a matroid $\mathcal{M} = (\mathcal{F}, \mathcal{I})$ over the facilities. The goal is to select a subset of facilities S that is an independent set of the matroid $\mathcal{M}$ where the objective $\sum_{j \in \mathcal{C}} d(j, S) + \sum_{i \in S} f_i$ (i.e. the *cost* induced by selected facilities) is minimized, and the *radius* constraint $d(j, S) \leq r_j$ is satisfied for all clients $j \in \mathcal{C}$.

Most of the center-based clustering problems are NP-Hard even in very restricted settings. We focus on polynomial-time approximation algorithms which have an extensive history in center-based clustering. Moreover, due to the nature of the constraints in PMatMed, we can only obtain bicriteria approximation guarantees that violate both the objective and the radius constraints. An (α, β)-approximation algorithm for PMatMed is a polynomial-time algorithm that either correctly states that no feasible solution is possible or outputs a set of facilities $S \in \mathcal{I}$ (hence satisfies the matroid constraint) such that (i) $d(j, S) \leq \alpha r_j$ for all clients $j \in \mathcal{C}$ and (ii) the cost objective value of S is at most $\beta \cdot OPT$ where OPT is the cost of an optimum solution.

1.1 Motivation, Applications to Fair Clustering, and Related Work

Our study of PMatMed is motivated, at a high-level, by two considerations. First, there has been past work that combines the k-Median objective with that of the k-Center objective. Alamdari and Shmoys [3] considered the k-Median problem with the additional constraint that each client is served within a given uniform radius L and obtained a $(4, 8)$-approximation. Their work is partially motivated by the ordered median problem [24, 4, 8]. Kamiyama [18] studied a generalization of this uniform radius requirement on clients to the setting of Matroid Median and derived a $(11, 16)$-approximation algorithm. Note that this is a special case of PMatMed where $r_j = L$ for each j. We call this the UniPMatMed problem.

Another motivation for studying PMatMed is the recent interest in *fair clustering* in the broader context of algorithmic fairness. The goal is to capture and address social concerns in applications that rely on clustering procedures and algorithms. Various notions of fair clustering have been proposed. Chierchetti et al. [11] formulated the Fair k-Center problem: clients belong to one or more groups based on various attributes. The objective is to return a clustering of points where each chosen center services a representative number of clients from *every* group. This notion has since been classified as one that seeks to achieve *group* fairness. Several other group fair clustering problems have since been introduced and studied [7, 20, 1, 13]. Subsequently, clustering formulations that aimed to encapsulate *individual* fairness were explored which seek to ensure that each individual is treated fairly. One such formulation was introduced by Jung et al. [17]. This formulation is related to the well-studied k-Center clustering and is the following. Given n points in a metric space representing users, and an integer k, find a set of k centers S such that $d(j, S)$ is at most r_j where r_j denotes the smallest radius around j that contains n/k points. Such a clustering is fair to individual users since no user will be forced to travel outside their *neighborhood.* Jung et al. [17] showed that the problem is NP-Hard and described a simple greedy algorithm that finds k centers S such that $d(j, S) \leq 2r_j$ for all j. Jung et al.'s model can be related to an earlier model of Plesník who considered the Weighted k-Center problem [25]. In Plesník's version, each user j specifies an *arbitrary* radius $r_j > 0$ and the goal is to find k centers S to serve each user within their radius requirement. Plesník showed that a simple variant of a well-known algorithm for k-Center due to Hochbaum and Shmoys [15] yields a 2-approximation. Plesník's problem has been relabeled as the Priority k-Center problem in recent work [6].

Priority clustering. The model of Jung et al. motivated several variations and generalizations of the Priority k-Center problem. Bajpai et al. [6] defined, and provided constant factor approximations, for Priority k-Supplier (where facilities and clients are considered to be disjoint sets), as well as Priority Matroid and Knapsack Center, where facilities are subject to matroid and knapsack constraints, respectively. Mahabadi and Vakilian [23] explored and developed approximation algorithms for Priority k-Median and Priority k-Means problems; their motivation was to combine the individual fairness requirements in terms of radii proposed by Jung et al., with the traditional objectives of clustering. They obtained bicriteria approximation algorithms via local-search. The approximation bounds were later improved via LP-based techniques. Chakrabarty and Negahbani [9] obtained an $(8, 8)$-approximation for Priority k-Median and a $(8, 16)$-approximation for Priority k-Means. Vakilian and Yalcner [28] further improved these results via a nice black box reduction of Priority k-Median to the Matroid Median problem! Via their reduction they obtained $(3, 7.081 + \epsilon)$-approximation for the Priority k-Median problem (relying on the algorithm for Matroid Median from [22]). They extended the algorithmic ideas from Matroid Median to handle ℓ_p norm objectives and were thus able to derive algorithms for Priority k-Means as well. The advantage of the reduction to Matroid Median is the guarantee of 3 on the radius dilation. This is optimal even for the k-Supplier problem [15].

1.2 Results and Technical Contribution

In this paper, we define the PMatMed problem and derive the first (c_1, c_2)-bicriteria approximation algorithms where c_1, c_2 are both constants. There are different trade-offs between c_1 and c_2 that we can achieve. Since PMatMed simultaneously generalizes k-Supplier and Matroid Median, the best c_1 we can hope for is 3, and the best c_2 that we can hope for is ≈ 8, which comes from current algorithms for Matroid Median [22, 27]. We prove the following theorem which captures two results, one optimizing for the radius guarantee, and the other for the cost guarantee.

▶ **Theorem 1.** *There is a $(21, 12)$-approximation algorithm for the Priority Matroid Median Problem. There is also a $(36, 8)$-approximation algorithm.*

As we previously mentioned, [28], via their black box reduction to Matroid Median achieve a $(3, \alpha)$ approximation for Priority k-Median where α is the best approximation for Matroid Median. We conjecture that there is a $(3, O(1))$-approximation for PMatMed. This is interesting and open even for the special case with uniform radii under partition matroid constraint.

Our second set of results are for UniPMatMed. Recall that [18] obtained a $(11, 16)$-approximation for this problem. We prove the following theorem that strictly dominates the bound from [18]. In addition, we show that a tighter radius guarantee is achievable.

▶ **Theorem 2.** *There is a $(9, 8)$-approximation algorithm for the Uniform Priority Matroid Median Problem. For any fixed $\epsilon > 0$ there is a $(5 + 8\epsilon, 4 + \frac{2}{\epsilon})$-approximation.*

▶ Remark 3. We believe that we can extend the ideas from this paper to obtain bicriteria approximation algorithms for Priority Matroid objectives that involve the ℓ_p norm of distances (Priority Matroid Median is when $\ell_p := 1$). Such an approximation algorithm would result in a radius factor dependent on p. [28] already showed that Matroid Median can be extended to the p-norm objective.

Now, we give a brief overview of our technical approach. The reader may wonder about the reduction of Priority k-Median to Matroid Median [28]. Can we make use of this for PMatMed? Indeed one can employ the same reduction, however, the resulting instance is no longer an instance of Matroid Median but an instance of Matroid Intersection Median which is inapproximable [27]. The reduction works in the special case of Priority k-Median since the intersection of a matroid with a cardinality constraint yields another matroid. We therefore address PMatMed directly. Our approximation algorithms are based on a natural LP relaxation. It is not surprising that we need to build upon the techniques for Matroid Median since it is a special case. We build extensively on the LP-based 8-approximation for Matroid Median given by Swamy [27] which improved the first constant factor approximation algorithm of Krishnaswamy et al. [21]. Although the Matroid Median approximation has been improved to 7.081 [22], the approach in [22] seems more difficult to adapt to PMatMed.

Our main technical contribution is to handle the non-uniform radii constraints imposed in PMatMed in the overall approach for Matroid Median. We note that the rounding algorithms for Matroid Median are quite complex, and involve several non-trivial stages: filtering, finding half integral solutions via an auxiliary polytope, and finally rounding to an integral solution via matroid intersection [21, 27, 22]. Kamiyama adapted the ideas in [21] to UniPMatMed and his work involves four stages of reassigments that are difficult to follow. The non-uniform radii case introduces additional complexity. We explain the differences between the uniform radii case and the non-uniform radii case briefly. The LP relaxation opens fractional facilities and assigns each client j to fractionally open facilities. In the LP for PMatMed we write a natural constraint that j cannot be assigned to any facility i where $d(i, j) > r_j$. Let $\bar{C}_j$ denote the distance paid by j in the LP solution. The preceding constraint ensures that $\bar{C}_j \leq r_j$. For UniPMatMed, $r_j = L$ for all $j \in \mathcal{C}$. LP-based approximation algorithms for k-Median use filtering and other rounding steps by sorting clients in increasing order of $\bar{C}_j$ values since they are directly relevant to the objective. When one considers uniform radius constraint, one can still effectively work with $\bar{C}_j$ values since we have $\bar{C}_j \leq L$ for all j. However, when clients have non-uniform radii we can have the following situation; there can be clients j and k such that $\bar{C}_j \ll \bar{C}_k$ but $r_j \gg r_k$. Thus the radius requirements may not correspond to the fractional distances paid in the LP.

We handle the above mentioned complexity via two careful adaptations to Matroid Median rounding. One of these changes occurs in the second stage of Matroid Median rounding, where we construct a half-integral solution using an auxiliary polytope. We must take care to ensure that the half-integral solution constructed in this stage is one that will not violate the radius requirements for clients. To do so, we create additional constraints in the auxiliary polytope. These constraints ensure the half-integral solution satisfies certain properties that are crucial to obtain a constant factor radius guarantee.

The second change occurs in the first filtering stage and plays a role not only for adapting Matroid Median to PMatMed, but also for each of our other results. We first provide an abstract way to describe the filtering stage that allows us to specificy the order in which points are considered, and the distances each point can travel to be reassigned. For our first PMatMed result, the ordering and distances are based on both $\bar{C}_j$ and r_j. For UniPMatMed, we slightly alter the ordering and distances (using the above observations and some ideas from [18]). Our remaining results will also involve changes to the filtering stage. This seems to indicate that filtering plays a large role in the cost and radius trade-off.

Organization. In Section 2, we discuss preliminaries. In particular, we provide definitions and relevant information regarding matroids, define PMatMed and provide its LP relaxation, and discuss the generalized filtering procedure we will use in our algorithm. In Section 3 we present our algorithm for PMatMed and show that it can be used to obtain $(21, 12)$-approximate solutions for instances of PMatMed. In Section 4, we describe how to modify our algorithm for PMatMed to obtain a $(9, 8)$-approximate solution for instances of UniPMatMed, and the remaining results. We also provide some details for the remaining results, and $(36, 8)$-approximate solutions for instances of PMatMed. We defer the proofs of our results to the Appendix.

2 Preliminaries

2.1 Matroids, Matroid Intersection and Polyhedral Results

We assume some basic knowledge about matroids, but provide a few relevant definitions for sake of completeness; we refer the reader to [26] for more details. A matroid $\mathcal{M} = (S, \mathcal{I})$ consists of a finite ground set S and a collection of *independent* sets $\mathcal{I} \subseteq 2^S$ that satisfy the following axioms: (i) $\emptyset \in \mathcal{I}$ (non-emptiness of $\mathcal{I}$) (ii) $A \in \mathcal{I}$ and $B \subset A$ implies $B \in \mathcal{I}$ (downward closure) and (iii) $A, B \in \mathcal{I}$ with $|A| < |B|$ implies there is $i \in B \setminus A$ such that $A \cup \{i\} \in \mathcal{I}$ (exchange property). The rank function of a matroid, $r_{\mathcal{M}} : 2^S \to \mathbb{Z}^{\geq 0}$ assigns to each $X \subseteq S$ the cardinality of a maximum independent subset in X. It is known that $r_{\mathcal{M}}$ is a monotone submodular function. The matroid polytope for a matroid $\mathcal{M}$, denoted by $\mathcal{P}_{\mathcal{M}}$ is the convex hull of the characteristic vectors of the independent sets of $\mathcal{M}$. This can be characterized via the rank function:

$$\mathcal{P}_{\mathcal{M}} = \{v \in \mathbb{R}^S \mid \forall X \subseteq S : \ v(X) \leq r_M(X) \text{ and } \forall e \in S : \ v(e) \geq 0\}.$$

Assuming an independence oracle[1] or a rank function oracle for $\mathcal{M}$, one can optimize and separate over $\mathcal{P}_{\mathcal{M}}$ in polynomial time. A *partition matroid* $\mathcal{M} = (S, \mathcal{I})$ is a special type of matroid that is defined via a partition $S_1, S_2, \ldots, S_h$ of S and non-negative integers $k_1, \ldots, k_h$. A set $X \subseteq S$ is independent, that is $X \in \mathcal{I}$, iff $|X \cap S_i| \leq k_i$ for $1 \leq i \leq h$. A simple partition matroid is one in which $k_i = 1$ for each i.

Given two matroids $\mathcal{M} = (S, \mathcal{I}_1)$ and $\mathcal{N} = (S, \mathcal{I}_2)$, on the same ground set, their intersection is defined as $\mathcal{M} \cap \mathcal{N} := (S, \mathcal{I}_1 \cap \mathcal{I}_2)$ consisting of sets that are independent in both $\mathcal{M}$ and $\mathcal{N}$. Computing a maximum weight independent set in the intersection can be done efficiently. The convex hull of the characteristic vectors of the independent sets of $\mathcal{M} \cap \mathcal{N}$, denoted by $\mathcal{P}_{\mathcal{M},\mathcal{N}}$, is simply the intersection of $\mathcal{P}_{\mathcal{M}}$ and $\mathcal{P}_{\mathcal{N}}$! That is

$$\mathcal{P}_{M,N} = \{v \in \mathbb{R}_+^S \mid \forall X \subseteq S : \ v(X) \leq r_M(X), v(X) \leq r_{\mathcal{N}}(X)\}.$$

Thus, one can optimize over $\mathcal{P}_{\mathcal{M},\mathcal{N}}$ if one has independence or rank oracles for $\mathcal{M}$ and $\mathcal{N}$. We will need these results later in the paper. See [26] for these classical results.

The input matroid $\mathcal{M}$ for Priority Matroid Median has ground set $\mathcal{F}$ i.e. the set of facilities. Thus, an integer point of the polytope $v^* \in \mathcal{P}_{\mathcal{M}}$ will represent a subset of facilities that is an independent set of the matroid $\mathcal{M}$.

[1] An independence oracle returns whether $A \in \mathcal{I}$ for a given $A \subseteq S$.

2.2 Priority Matroid Median

We provide below a more general definition of Priority Matroid Median that includes a notion of client *demands.*

▶ **Definition 4** (PMatMed). *The input is a set of facilities $\mathcal{F}$ and clients $\mathcal{C}$ from a metric space $(\mathcal{F} \cup \mathcal{C}, d)$. Each $i \in \mathcal{F}$ has an opening cost $f_i \geq 0$. Each client $j \in \mathcal{C}$ has a radius value, $r_j \geq 0$ and a demand value $a_j \geq 0$. We are also given a matroid $\mathcal{M} = (\mathcal{F}, \mathcal{I})$. The goal is to choose a set $S \in \mathcal{I}$ to minimize $\sum_{i \in S} f_i + \sum_{j \in \mathcal{C}} a_j d(j, S))$ with the constraint that $d(j, S) \leq r_j$ for each $j \in \mathcal{C}$.*

A PMatMed instance $\mathscr{I}$ is the tuple $(\mathcal{F}, \mathcal{C}, d, \mathbf{f}, \mathbf{r}, \mathbf{a}, \mathcal{M})$, where $\mathbf{f} \in \mathbb{R}^{\mathcal{F}}$ and $\mathbf{r}, \mathbf{a} \in \mathbb{R}^{\mathcal{C}}$.

2.3 LP relaxation for PMatMed

Our algorithm is based on an LP relaxation for a PMatMed instance $\mathscr{I} = (\mathcal{F}, \mathcal{C}, d, \mathbf{f}, \mathbf{r}, \mathbf{a}, \mathcal{M})$ that we describe next. We use i to index facilities in $\mathcal{F}$, j to index clients in $\mathcal{C}$. Recall that $r_{\mathcal{M}}$ denotes the rank function of the matroid $\mathcal{M}$. The y_i variables denote the fractional amount a facility i is open, while the x_{ij} variables indicate the fractional amount a client j is assigned to facility i.

$$\min \quad \sum_{i \in \mathcal{F}} f_i y_i + \sum_j \sum_i a_j d(i,j) x_{ij} \tag{1a}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{F}} x_{ij} \geq 1 \qquad \forall j \in \mathcal{C} \tag{1b}$$

$$x_{ij} \leq y_i \qquad \forall i \in \mathcal{F}, j \in \mathcal{C} \tag{1c}$$

$$x_{ij} = 0 \qquad \forall i \in \mathcal{F}, j \in \mathcal{C} : d(i,j) > r_j \tag{1d}$$

$$y \in \mathcal{P}_{\mathcal{M}} \tag{1e}$$

$$x_{ij}, y_i \geq 0 \qquad \forall i \in \mathcal{F}, j \in C \tag{1f}$$

Constraint 1b states that each client must be fully assigned to facilities, and constraint 1c ensures that these facilities have indeed been opened enough to service clients. For integral y, constraint 1e mandates that the facilities come from an independent set of the matroid $\mathcal{M}$. Finally, constraint 1d ensures that no client is assigned to a center that is farther than its radius value.

We make a few basic observations about the LP relaxation. We assume that it is feasible for otherwise the algorithm can terminate reporting that there is no feasible integral solution. Indeed, the LP is solvable in polynomial time via the rank oracle for $\mathcal{M}$. First, some notation. For $X \subseteq \mathcal{F}$, we let $y(X)$ denote $\sum_{i \in X} y_i$. For client j and radius parameter R we let $B(j, R)$ denote the set $\{i \in \mathcal{F} \mid d(i,j) \leq R\}$ of facilities within R of j. Constraints 1b and 1d ensure the following simple fact.

▶ **Fact 5.** *Let (x, y) be a feasible solution to the PMatMed LP. Then $y(B(j, r_j)) \geq 1$ holds $\forall j \in \mathcal{C}$.*

Let $COST(x, y)$ denote the cost of the LP using solution (x, y). Going forward, we will assume that we are working with an optimum fractional solution to the LP relaxation for the given instance.

▶ Remark 6. We say that y is feasible if $y \in \mathcal{P}_{\mathcal{M}}$ and $y(B(j, r_j)) \geq 1$ for all $j \in C$. Given feasible y, a corresponding x satisfying the constraints and minimizing $COST(x, y)$ is determined by solving a min-cost assignment problem for each client $j \in C$ separately.

2.4 Filtering

Filtering is a standard step in several approximation algorithms for clustering and facility location wherein one identifies a subset of well-separated and representative clients. Each client is assigned to a chosen representative. In priority median problems there are two criteria that dictate the filtering process. One is the radius upper bound r_j for the client j. The other is the LP distance $\bar{C}_j = \sum_i d(i,j)x(i,j)$ paid by the client which is part of the objective. Balancing these two criteria is important. To facilitate different scenarios later we develop a slightly abstract filtering process. Building on a procedure introduced in [15, 25], Filter takes in the metric and demands from a PMatMed instance $\mathscr{I} = (\mathcal{F}, \mathcal{C}, d, \mathbf{f}, \mathbf{r}, \mathbf{a}, \mathcal{M})$, as well as functions $\phi, \lambda : \mathcal{C} \rightarrow \mathbb{R}_+$ that satisfy the following condition.

▶ **Definition 7** (compatibility). *Functions $\phi, \lambda : \mathcal{C} \rightarrow \mathbb{R}_+$ are compatible if for any ordering of clients $j_1, j_2, \ldots, j_n$ where $\phi(j_1) \leq \phi(j_2) \leq \ldots \leq \phi(j_n)$, it is the case that $\lambda(j_1) \leq \lambda(j_2) \leq \ldots \leq \lambda(j_n)$.*

▶ Remark 8. This condition trivially holds when ϕ and λ are identical. The filtering stages of many clustering approximation algorithms [6, 17, 9] utilize equal ϕ and λ functions. We use both identical and non-identical settings for ϕ and λ in this paper.

The function ϕ encodes an ordering of clients, while λ represents a client's coverage distance. Filter chooses cluster centers in order of increasing ϕ values, and then "covers" any remaining client k that is within distance $2 \cdot \lambda(k)$ from the newly added center j. The demand from the covered points is transferred to the center that first covered them. The new demand variables a' represent the aggregated demand for the chosen centers. Filter returns the set of cluster centers, the clusters assigned to each cluster center, and new demand assignments for all clients.

Algorithm 1 Filter.

```
Require: Metric $(\mathcal{F} \cup \mathcal{C}, d)$, demands $\mathbf{a}$, compatible functions $\phi, \lambda : \mathcal{C} \rightarrow \mathbb{R}_{>0}$
$U \leftarrow \mathcal{C}$                                        ▷ The set of uncovered clients
$C \leftarrow \emptyset$                                        ▷ The set of cluster centers
$\forall j \in \mathcal{C}$ set $a'_j := 0$                     ▷ Initialize new demand variables
while $U \neq \emptyset$ do
    $j \leftarrow \arg\min_{j \in U} \phi(j)$
    $C \leftarrow C \cup \{j\}$
    $D(j) \leftarrow \{k \in U : d(j,k) \leq 2 \cdot \lambda(k)\}$    ▷ Note: $D(j)$ includes $j$ itself
    $a'_j = \sum_{k \in D(j)} a_k$                             ▷ Accumulate all demands of $D(j)$ to $j$
    $U \leftarrow U \backslash D(j)$
end while
Return cluster centers $C$, $\{D(j) : j \in C\}$, updated demands $\mathbf{a}' \in \mathbb{R}^{\mathcal{C}}$
```

The resulting cluster centers $C \subseteq \mathcal{C}$, and the sets of clients relocated to each cluster center $\{D(j) \mid j \in C\}$ form a partition of the client set $\mathcal{C}$. When the given ϕ and λ are compatible, the returned clusters satisfy certain desirable properties, described in the following facts which are relatively easy to see, and standard in the literature. For this reason we omit formal proofs.

▶ **Fact 9.** *The following statements hold for the output of Filter: (a) $\forall j, j' \in C, d(j,j') > 2\max\{\lambda(j), \lambda(j')\}$. (b) $\{B(j, \lambda(j)) \mid j \in C\}$ are mutually disjoint. (c) $\{D(j) \mid j \in C\}$ partitions $\mathcal{C}$. (d) $\forall j \in C, \forall k \in D(j), \phi(j) \leq \phi(k)$ and $\lambda(j) \leq \lambda(k)$. (e) $\forall j \in C, \forall k \in D(j), d(j,k) \leq 2 \cdot \lambda(k)$*

Choosing ϕ and λ. As we remarked, the two criteria that influence the filtering process are r_j and $\bar{C}_j$. For the algorithm in Section 3 we choose $\phi(j) = \lambda(j) = \min\{r_j, 2\bar{C}_j\}$. There are other valid settings of compatible ϕ and λ that can be used in the filtering stage. Different settings of ϕ and λ will result in different approximation factors for cost and radius. We elaborate on this further in Section 4.

3 A $(21, 12)$-approximation for Priority Matroid Median

Our algorithm will follow the overall structure of the LP-based procedure used for approximating Matroid Median from [27], but will contain a few key alterations that allow us to be mindful of the radius objective of PMatMed. Stage 1 of our algorithm involves filtering the client set to construct an updated instance $\mathscr{I}'$ using the cluster centers and updated demands. We will show that a solution to $\mathscr{I}'$ can be converted to a solution for $\mathscr{I}$ while only incurring a small increase to the cost and radius. The focus then shifts to constructing a solution for $\mathscr{I}'$. In Stage 2, we obtain a half-integral solution for the LP-relaxation for $\mathscr{I}'$ by working with an auxiliary polytope. In Stage 3, this half-integral solution is converted to an integral solution for $\mathscr{I}'$. This is done via a reduction to matroid intersection. Finally, we will show that this solution yields a $(21, 12)$-approximation for the original instance $\mathscr{I}$. Algorithm 2 is given as a summary of the various stages of our algorithm. The omitted proofs from this section can be found in Appendix A.

Algorithm 2 Overview of bi-criteria approximation algorithm for PMatMed.

```
Input: PMatMed instance $\mathscr{I} = (\mathcal{F}, \mathcal{C}, d, \mathbf{f}, \mathbf{r}, \mathbf{a}, \mathcal{M})$.
Output: $(\alpha, \beta)$-approximate solution for $\mathscr{I}$.
0: Solve $LP$ for $\mathscr{I}$ and let $(x, y)$ denote the optimal fractional solution. Use $(x, y)$ and
   radius values $\mathbf{r}$ to help set $\phi$ and $\lambda$.
1: Stage 1 - Run Filter$((\mathcal{F} \cup \mathcal{C}, d), \mathbf{a}, \phi, \lambda)$ which returns cluster centers $C$, and updated
   client demands $\mathbf{a}'$. Create an updated instance $\mathscr{I}' = (\mathcal{F}, C, d, \mathbf{f}, \mathbf{r}, \mathbf{a}', \mathcal{M})$ (Section 3.1).
2: Stage 2 - Construct a half-integral solution $(\hat{x}, \hat{y})$ for $\mathscr{I}'$ by setting up a polytope $\mathcal{Q}$
   with half-integral extreme points (Section 3.2).
3: Stage 3 - Convert the half-integral solution to an integral solution $(\tilde{x}, \tilde{y})$ for $\mathscr{I}'$ by
   setting up an instance of matroid intersection between the input matroid $\mathcal{M}$, and a
   partition matroid $\mathcal{N}$ constructed with respect to the half-integral solution (Section 3.3).
4: Convert the integral solution for $\mathscr{I}'$ to one for $\mathscr{I}$ (Lemma 10).
```

3.1 Stage 1: Filtering Clients

In this stage, we create a new instance of PMatMed from the initial one by using the Filter process described in Section 2.4. Recall that Filter will return a set of cluster centers $C \subseteq \mathcal{C}$, and collections of clients that are relocated to each cluster center $\{D(j) \mid j \in C\}$. Filter also returns a set of updated demands for all clients, $\mathbf{a}'$. Now, using C and $\mathbf{a}'$, we construct a new instance of PMatMed $\mathscr{I}' = (\mathcal{F}, C, d, \mathbf{f}, \mathbf{r}, \mathbf{a}', \mathcal{M})$. Here, we overload notation and take $\mathbf{r}$ and $\mathbf{a}'$ to denote the vector of radius values and demands, respectively, for cluster centers (i.e. $\mathbf{r}, \mathbf{a}' \in \mathbb{R}^C$). Notice that we do not lose any information by restricting $\mathbf{a}'$ to C, since the updated demands for relocated points are set to 0. Furthermore, we will reconcile the radius objective for relocated points in the final solution at the end of the section.

The solution (x, y) for instance $\mathscr{I}$, when restricted to C, will still be a feasible solution for the LP for $\mathscr{I}'$, since the new LP is made up of a subset of constraints from the original LP. For updated instance $\mathscr{I}'$, we denote the cost of the LP solution (x, y) by $COST'(x, y)$.

$$COST'(x,y) = \sum_{i\in\mathcal{F}} f_i y_i + \sum_{j\in C} a'_j \sum_{i\in\mathcal{F}} d(i,j)x_{ij} = \sum_{i\in\mathcal{F}} f_i y_i + \sum_{j\in C} a'_j \bar{C}_j$$

The next lemma shows that an integral solution to $\mathscr{I}'$ can be translated to an integer solution for $\mathscr{I}$ by incurring a small additive increase to the cost objective. In subsequent sections we will address how the translated solution also ensures that all clients are served within a constant factor of their radius constraint.

▶ **Lemma 10.** *The following is true of $\mathscr{I}'$: (a) $COST'(x, y) \leq 2 \cdot COST(x, y)$. (b) Any integer solution (x', y') for $\mathscr{I}'$ can be converted to an integer solution for $\mathscr{I}$ that incurs an additional cost of at most $4 \cdot COST(x, y)$.*

The following lemma follows directly from Fact 9.

▶ **Lemma 11.** *Let $k \in \mathcal{C}$ be assigned to $j \in C$ after* Filter *(i.e. $k \in D(j)$). Then, $d(j, k) \leq 2\lambda(k) \leq 2r_k$.*

3.2 Stage 2: Constructing Half-Integral Solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$

In the second stage the goal is to construct a half-integral solution to $\mathscr{I}'$. This means that each cluster center/client $j \in C$ will connect to at most two facilities. This is accomplished by constructing a specific polytope $\mathcal{Q}$ with only facility variables, and a proxy objective that also has only facility variables and arguing about the properties of $\mathcal{Q}$ and the objective function.

To describe $\mathcal{Q}$, we define, for each client $j \in C$, several facility sets that will play an important role. Let $F_j = \{i \in \mathcal{F} \mid d(i, j) = \min_{k\in C} d(i, k)\}$ denote the set of facilities i for which j is the closest client in C (ties are broken arbitrarily). Let $F'_j = \{i \in F_j \mid d(i, j) \leq \lambda(j)\} \subseteq F_j$. Let $\gamma_j := \min_{i\notin F_j} d(i, j)$ denote the distance between client $j \in C$ and the closest facility i not included in F_j. In other words, i in the definition of γ_j is the closest facility to j that has some other closest cluster center $j' \in C$ such that $j \neq j'$. Using γ_j, let $G_j = \{i \in F_j \mid d(i, j) \leq \gamma_j\}$. Finally, let ρ_j be the smallest distance such that $y(B(j, \rho_j)) \geq 1$, and $B_j := B(j, \rho_j)$.[2] See Figure 1.

We summarize some basic properties of the defined sets below.

▶ **Fact 12.** *The following hold for all $j \in C$: (a) If $j' \neq j$, $F_j \cap F_{j'} = \emptyset$; (b) F_j contains all the facilities i such that $d(i, j) \leq \lambda(j)$; (c) $\gamma_j > \lambda(j)$; (d) $F'_j \subseteq G_j$; (e) $\rho_j \leq r_j$, (f) $\sum_{i\in F'_j} x_{ij} \geq 1/2$ and when $\lambda(j) = r_j$, $\sum_{i\in F'_j} x_{ij} = 1$;*

Proof. (a) follows from definition of F_j, (b), (c), (d) follow from Fact 9(b) and definitions. (e) follows from the LP constraint. We now prove (f). If $\lambda(j) = r_j$, $F'_j = \{i \mid d(i, j) \leq r_j\}$, and by LP constraint $\sum_{i\in F'_j} x_{ij} = 1$. Otherwise $\lambda(j) = 2\bar{C}_j < r_j$. Note that $\bar{C}_j = \sum_i d(i, j)x_{ij}$. By averaging argument (Markov's inequality) we have $\sum_{i:d(i,j)\leq 2\bar{C}_j} x_{ij} \geq 1/2$. This gives the desired claim since $F'_j = \{i \mid d(i, j) \leq \lambda(j)\}$. ◀

[2] Note that though it may be the case that $y(B(j, \rho_j)) > 1$, we can split facilities and define B_j as the points of $B(j, \rho_j)$ such that $y(B_j) = 1$.

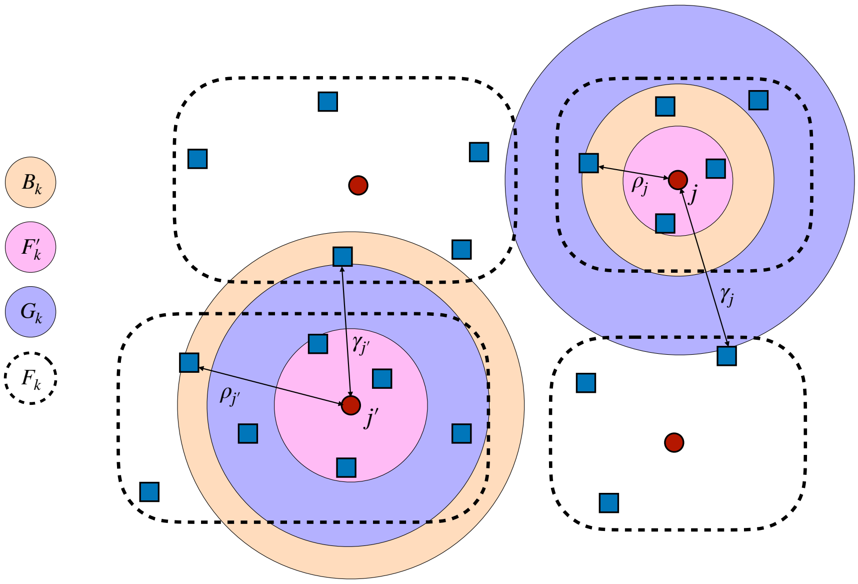


Figure 1 The F, F', G, and B sets for points $j \in C_s$ and $j' \in C_b$. Observe that for j, $\rho_j \leq \gamma_j$, hence $B_j \subseteq G_j$.

At this point in the algorithm, in a departure from the Matroid Median algorithm of [27], we need to be mindful of two cases. If $\rho_j \leq \gamma_j$, in order to satisfy the radius requirements of PMatMed, it is important to open one facility within radius ρ_j of j. If it is the case that $\rho_j > \gamma_j$, it is not necessary to do so. To distinguish these two cases, we partition C into $C_s = \{j \in C \mid \rho_j \leq \gamma_j\}$, and $C_b = \{j \in C \mid \rho_j > \gamma_j\}$. For $j \in C_s$, it should be clear that $B_j \subseteq G_j$. Using these sets, we define a polytope $\mathcal{Q}$ with facility variables $v_i, i \in \mathcal{F}$ as follows. It consists of the matroid constraints induced by $\mathcal{M}$ and a second set of constraints induced by C and C_s as defined above. In particular, we require that all points j in C has at least $1/2$ value assigned cumulatively to facilities within their F'_j balls. We require points of C_s to have exactly 1 assigned to facilities within B_j.

$$\mathcal{Q} = \Big\{ v \in \mathbb{R}^{\mathcal{F}}_{\geq 0} \;\mid\; \forall S \subseteq \mathcal{F} : v(S) \leq r_{\mathcal{M}}(S), \;\; \forall j \in C : \;\; v(F'_j) \geq 1/2 \text{ and } v(G_j) \leq 1, \\ \forall j \in C_s : v(B_j) = 1 \Big\}$$

▶ **Lemma 13.** *The extreme points of the polytope $\mathcal{Q}$, if non-empty, are half-integral.*

The proof of the preceding lemma is similar to those in previous works on Matroid Median [21, 27]. We give a proof (found in Appendix A) for the sake of completeness since the polytope we define is slightly different due to the separation of clients in C into C_s and C_b in order to enforce an additional constraint.

We will now define a vector y' that lies in $\mathcal{Q}$ which will prove that it is non-empty. Further, we also define a linear objective function $T(\cdot)$ over vectors in $\mathcal{Q}$ to serve as a proxy for the cost. Following the analysis for the improved bound in [27], we set up $T(\cdot)$ with some slack so that the slack can be exploited in the analysis of the next step in the algorithm.

We define $y' \in \mathbb{R}^{\mathcal{F}}_{\geq 0}$ as follows. For all $j \in C$ and $i \in G_j$, set $y'_i = x_{ij} \leq y_i$. For a facility $i \notin \cup_j G_j$ set $y'_i = 0$. From this definition it should be clear that $y'(G_j) \leq 1$ for all $j \in C$, since $\sum_{i \in G_j} x_{ij} \leq 1$. Also, from Fact 12, $y'(F'_j) \geq 1/2$. For $j \in C_s$, it will be the case that $y'(B_j) = 1$ since $\sum_{i \in B_j} x_{ij} = 1$; we also know that for these points, $y'(G_j) = y'(B_j)$.

To build up to the definition of T, we first state the following lemma, which we will prove in the proof of Lemma 17 (Appendix A).

▶ **Lemma 14.** *Consider some $j \in C$, and let i and j' be the facility and cluster used to define γ_j (i.e. $\gamma_j = d(i,j)$) where $i \in F_{j'}$ for some $j' \neq j$. For every $i' \in F'_{j'}$, $d(i',j) \leq 3\gamma_j$.*

Keeping the preceding lemma in mind, we can use as proxy for j's per-unit-demand cost a function written in terms of the facility vector v. When $y'(G_j) = 1$, the cost for j can be bounded by $\sum_{i\in G_j} d(i,j)y'_i \leq \bar{C}_j$. When $y'(G_j) < 1$, the preceding lemma indicates that we can upper bound the cost of the solution by $\sum_{i\in G_j} d(i,j)y'_i + 3\gamma_j(1 - y'(G_j)) \leq 3 \cdot \bar{C}_j$. Using these two bounds, we define $T(\cdot)$ for $v \in \mathcal{Q}$ as follows:

$$T(v) = \sum_{i\in\mathcal{F}} f_i v_i + \sum_{j\in C} a'_j \Big(2 \sum_{i\in G_j} d(i,j)v_i + 4\gamma_j(1 - v(G_j))\Big)$$

For v such that $v(F'_j) \geq 0.5$ and $v(G_j) \leq 1$ for all $j \in C$, the term $a'_j(2\sum_{i\in G_j} d(i,j)y'_i + 4\gamma_j(1-y'(G_j)))$ will upper bound j's assignment cost with respect to v via Lemma 14. When $v(G_j) = v(B_j) = 1$ for $j \in C_s$, j's assignment cost will be at most $a'_j(2\sum_{i\in B_j} d(i,j)v_i)$. Indeed $T(v)$ is an *overestimate* and we will use this in the next step.

We find an optimum half-integral solution $\hat{y}$ to $\mathcal{Q}$ with objective $T(v)$. It follows that $T(\hat{y}) \leq T(y')$. Now, we construct a half-integral solution $(\hat{x}, \hat{y})$ from $\hat{y} \in \mathcal{Q}$: For each cluster center $j \in C$, if $\hat{y}(G_j) = 1$, set $\sigma(j) = j$. Otherwise, set $\sigma(j) = \arg\min_{j'\in C:j'\neq j} d(j,j')$. Now, the *primary facility* for each cluster center is the closest facility $i \in \mathcal{F}$ such that $\hat{y}_i > 0$ (this will always be located in F'_j), is denoted by $i_1(j)$, and thus $\hat{x}_{i_1(j)j} = \hat{y}_{i_1(j)}$. A cluster center's *secondary* facility, denoted by $i_2(j)$, is the next option of facility for j to use, when it cannot be completely serviced by its primary facility. If $\hat{y}_{i_1(j)} = 1$, then j does not need a secondary facility, since $i_1(j)$ has been completely opened, and will remain completely opened. When $\hat{y}_{i_1(j)} < 1$ and $\hat{y}(G_j) = 1$, then set $i_2(j)$ to be the second closest partially opened facility to j (where $\hat{y}_{i_2(j)} > 0$). Otherwise, when $\hat{y}_{i_1(j)} < 1$ and $\hat{y}(G_j) < 1$, we now set $i_2(j) = i_1(\sigma(j))$ and $\hat{x}_{i_1(j)} = \hat{x}_{i_2(j)} = 1/2$. Note that if $j \in C_s$ then $\hat{y}(B_j) = 1$ which implies that j's primary and secondary facilities are both in B_j and $\sigma(j) = j$. The following two claims are easy to see.

▷ Claim 15. For all $j \in C$, $d(j, \sigma(j)) \leq 2\gamma_j$.

▷ Claim 16. For all $j \in C_s$, $\hat{y}(G_j) = 1$. If $\hat{y}(G_j) < 1$, it must be the case that $j \in C_b$.

By Fact 9(b), each j will have a unique primary facility that is at least partially opened in F'_j. For points $j \in C_s$, their secondary facility must be in B_j. However, for points in $j \in C_b$, $i_2(j)$ might not be in G_j or even F_j. As per Lemma 14, we know that j will be able to find a partially open facility to be serviced by that is within distance $3\gamma_j < 3\rho_j$. In the following lemma, we derive our bound for the cost of $(\hat{x}, \hat{y})$.

▶ **Lemma 17.** *$COST'(\hat{x}, \hat{y}) \leq T(\hat{y}) \leq T(y') \leq 4 \cdot COST'(x,y) \leq 8 \cdot COST(x,y)$.*

Before moving on to the final stage of the algorithm, we prove a few lemmas that will be relevant for our analysis of the radius dilation of the final solution. Lemma 18 allows us to relate the radius of cluster center j to that of a client k in the original instance that was relocated to j. We need such a lemma because even though we know that $\phi(j) = \min\{r_j, 2\bar{C}_j\}$ and $\phi(j) \leq \phi(k)$ for all $k \in D(j)$, we cannot assume that $r_j \leq r_k$.

▶ **Lemma 18.** *Suppose client $k \in \mathcal{C}$ is relocated to $j \in C$ after filtering ($k \in D(j)$). Then $\rho_j \leq 3r_k$.*

Proof. Note that $y(B(k, r_k)) \geq 1$ via the LP constraint. We have $d(j,k) \leq 2\lambda(k) \leq 2r_k$ since $\lambda(k) = \min\{r_k, 2\bar{C}_k\}$. Via triangle inequality, $B(k, r_k) \subseteq B(j, 3r_k)$. Thus $\rho_j \leq 3r_k$. ◀

Lemma 14 and Lemma 18 imply Lemma 19, which bounds the distance between relocated points and the primary and secondary facilities of the cluster center they are relocated to.

▶ **Lemma 19.** *Let $k \in \mathcal{C}$ and $k \in D(j)$ for a cluster center $j \in C$. Then, $d(j, i_1(j)) \leq \lambda(j) \leq \lambda(k) \leq r_k$. When $j \in C_s$, $d(j, i_2(j)) \leq \rho_j \leq 3r_k$. When $j \in C_b$, $d(j, i_2(j)) \leq d(j, \sigma(j)) + d(i_1(\sigma(j)), \sigma(j)) \leq 3\gamma_j \leq 3\rho_j \leq 9r_k$.*

▶ Remark 20. Notice that the $v(B_j) = 1$ constraint imposed for points $j \in C_s$ ultimately did not effect the cost analysis in Lemma 17. That is, we did not need to draw a distinction between points in C_s and points in C_b in order to obtain $COST'(\hat{x}, \hat{y}) \leq 4 \cdot COST'(x, y)$. The purpose of defining sets C_s and C_b and imposing an additional constraint for points in C_s is to ensure certain radius guarantees. In particular, Lemma 19 would not hold if the constraint $v(B_j) = 1$ for $j \in C_s$ was not enforced in $\mathcal{Q}$.

3.3 Stage 3: Converting to an Integral Solution

The procedure to convert the half-integral $(\hat{x}, \hat{y})$ to an integral solution involves setting up a matroid intersection instance consisting of the input matroid $\mathcal{M}$ and a partition matroid that is constructed using the primary and secondary facilities from $(\hat{x}, \hat{y})$ after another clustering step. The solution to this instance will be used to construct an integral solution $(\tilde{x}, \tilde{y})$ to $\mathscr{I}'$.

For $j \in C$ set $\hat{C}_j = (d(i_1(j), j) + d(j, \sigma(j)) + d(i_2(j), \sigma(j)))/2$. In cases where j has no secondary facility, let $i_2(j) = i_1(j)$. For each $j \in C$, define $S_j = \{i \mid \hat{x}_{ij} > 0\} = \{i_1(j), i_2(j)\}$. S_j has either one or two facilities. In addition, the following holds and will be relevant later.

▷ Claim 21. When $S_j \cap S_{j'} \neq \emptyset$, one of three cases can occur. (i) $S_j \cap S_{j'} = \{i_1(j), i_2(j)\}$, in which case $\sigma(j) = j'$ and $\sigma(j') = j$; (ii) $S_j \cap S_{j'} = \{i_1(j)\}$, and thus $\sigma(j') = j$ and $\sigma(j) \neq j'$ (a symmetric case occurs when switching j and j'); (iii) $S_j \cap S_{j'} = \{i_2(j)\}$ where $i_2(j) = i_2(j')$, hence $\sigma(j) = \sigma(j') = p$ and $p \neq j, j'$.

We construct a partition matroid $\mathcal{N}$ on ground set $\mathcal{F}$ via another clustering process to create a set $C' \subseteq C$. Repeat the following two steps until no clients in C are left to consider: (1) Pick $j \in C$ with the smallest $\hat{C}_j$ value and add j to the set C' then (2) remove every $j' \in C$ where $S_j \cap S_{j'} \neq \emptyset$, and have j be the center of j' (denoted by $\mathsf{ctr}(j') = j$). It is easy to see that the sets $S_j, j \in C'$ are mutually disjoint. Thus, a partition of $\mathcal{F}$ is induced by $\{S_j \mid j \in C'\}$, and the set $\mathcal{F} \setminus \cup_{j \in C'} S_j$. Set the capacity for each set of this partition to 1.

Now we consider the polytope that is intersection of the matroid polytopes of $\mathcal{M}$ and $\mathcal{N}$:

$$\mathcal{R} = \{z \in \mathbb{R}_+^{\mathcal{F}} \mid \forall S \subseteq \mathcal{F} : \ z(S) \leq r(S), \quad \forall j \in C' : \ z(S_j) \leq 1\}$$

The polytope $\mathcal{R}$ has integral extreme points via the classical result of Edmonds [12, 26].

The goal now is to figure out the set of facilities to open by optimizing a relevant objective over $\mathcal{R}$. First, we define a vector $\hat{y}' \in \mathbb{R}_+^{\mathcal{F}}$: if $i \in S_j$ for some $j \in C'$ we set $\hat{y}'_i = \hat{x}_{ij} \leq \hat{y}_i$, otherwise we set $\hat{y}'_i = \hat{y}_i$. Observe that $\hat{y}'$ is feasible for $\mathcal{R}$ and shows that $\mathcal{R}$ is not empty.

We now define a linear function $H(\cdot)$ over vectors in $\mathcal{R}$. We will optimize $H(\cdot)$ over $\mathcal{R}$ to obtain an integral extreme point $\tilde{y}$ and we will analyze its cost via $\hat{y}'$. For $z \in \mathbb{R}_+^{\mathcal{F}}$, define $H(z)$ as follows.

$$H(z) = \sum_i f_i z_i + \sum_{j \in C} L_j(z), \text{ where}$$

$$L_j(z) = \begin{cases} \sum_{i\in S_{\mathsf{ctr}(j)}} a'_j d(i,j) z_i & i_1(j) \in S_{\mathsf{ctr}(j)} \\ \sum_{i\in S_{\mathsf{ctr}(j)}} a'_j \Big(d(j,\sigma(j)) + d(\sigma(j),i)\Big) z_i & \\ \qquad +a'_j\Big(d(i_1(j),j) - d(j,\sigma(j)) - d(i_1(\sigma(j)),\sigma(j))\Big) z_{i_1(j)} & \text{otherwise} \end{cases}$$

Let $\tilde{y} \in \mathcal{R}$ be an integer extreme point such that $H(\tilde{y}) \leq H(\hat{y}')$. We use this to define an integral solution $(\tilde{x}, \tilde{y})$ to the modified instance by assigning each $j \in C'$ to the facility opened from S_j i.e. the facility $i \in S_j$ such that $\tilde{y}_i = 1$. For each $j' \in C \setminus C'$, assign j' to either $i_1(j')$ if it is open or the facility opened from $S_{\mathsf{ctr}(j')}$. $L_j(\tilde{y})$ serves as a proxy and upper bound for j's assignment cost. When $i_1(j) \notin S_{\mathsf{ctr}(j)}$, the second term of $L_j(\tilde{y})$ will adjust the distance j pays depending on whether $i_1(j)$ is opened or not. This adjustment is not needed when $i_1(j) \in S_j$ or when $i_1(j) \notin S_j$ is not opened, since in this case j must be assigned to the center opened from $S_{\mathsf{ctr}(j)}$. The following lemmas will show how the cost of $(\tilde{x}, \tilde{y})$ can be bounded by that of the half-integral solution $(\hat{x}, \hat{y})$ from the previous stage.

▶ **Lemma 22.** *$COST'(\tilde{x}, \tilde{y})$ is at most $H(\tilde{y}) \leq H(\hat{y}')$.*

▶ **Lemma 23.** *$H(\hat{y}') \leq T(\hat{y})$.*

▶ Remark 24. We do not lose a factor in the cost when converting the half-integral solution to an integral solution because the analysis in Stage 2 "overpays" for the half-integral solution. We follow the approach from [27].

3.4 Cost and Radius Analysis for PMatMed

Lemmas 10, 17, 22, and 23 together imply the following bound on the cost of $(\tilde{x}, \tilde{y})$ for instance $\mathscr{I}$ with respect to the cost of the LP solution (x, y).

▶ **Theorem 25.** *$COST(\tilde{x}, \tilde{y}) \leq 12 \cdot COST(x, y)$.*

Proof. $COST'(\tilde{x}, \tilde{y})$ will be at most $T(\hat{y})$ (Lemmas 22 and 23), and $T(\hat{y})$ is at most $4 \cdot COST'(x, y) \leq 8 \cdot COST(x, y)$ (Lemma 17). Hence, $(\tilde{x}, \tilde{y})$ will give a solution to $\mathscr{I}'$ of cost at most $8 \cdot COST(x, y)$. Lemma 10 tells us that translating an integer solution for $\mathscr{I}'$ to an integer solution for $\mathscr{I}$ will incur an additional cost of at most $4 \cdot COST(x, y)$. All together, $COST(\tilde{x}, \tilde{y}) \leq COST'(\tilde{x}, \tilde{y}) + 4 \cdot COST(x, y) \leq 8 \cdot COST(x, y) + 4 \cdot COST(x, y) = 12 \cdot COST(x, y)$. ◀

To complete our analysis of the radius approximation factor, we must determine how far points will be made to travel once the final centers are chosen. In Lemma 19 we guaranteed that each cluster center j will not travel farther than $3\rho_j$ to reach its secondary facility. However, in this final stage, we are assigning some cluster centers to others, and cannot guarantee that their primary or secondary facility will be opened. We can still show that even if a cluster center j from C_s gets assigned to a cluster center ℓ from C_b (i.e. that $\mathsf{ctr}(j) = \ell$), j will still only travel a constant factor outside of ρ_j. Consequently, using Lemma 18 we can show that each client $k \in \mathcal{C}$ will travel only a constant factor times its radius value r_k.

▶ **Lemma 26.** *Let $k \in \mathcal{C}$, where $k \in D(j)$ for $j \in C$. The final solution will open a facility i such that $d(i, j) \leq 19 r_k$.*

Proof. There are several cases to consider but most of them are simple. We provide the analysis for the case that gives the 19 factor, and other notable cases.

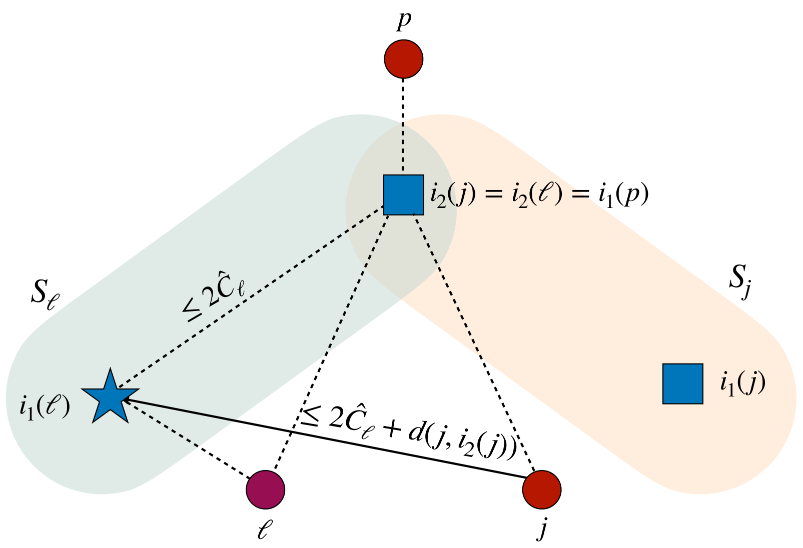

p
$i_2(j) = i_2(\ell) = i_1(p)$
S_ℓ
S_j
$\leq 2\hat{C}_\ell$
$i_1(\ell)$
$i_1(j)$
$\leq 2\hat{C}_\ell + d(j, i_2(j))$
ℓ
j

Figure 2 The farthest a point $j \in C$ will be from an opened center occurs when $\mathsf{ctr}(j) = \ell$, $\sigma(j) = \sigma(\ell) = p$, and $i_1(\ell)$ is opened.

If $j \in C'$, then either $i_1(j)$ or $i_2(j)$ will be opened in the final solution. Lemma 19 indicates that j will be assigned to a center that is at most $9r_k$ away. If $j \notin C'$, it must be the case that $\mathsf{ctr}(j) = \ell$ where $S_\ell \cap S_j \neq \emptyset$, and $\hat{C}_\ell \leq \hat{C}_j$. We claim that $\hat{C}_j \leq \frac{1}{2}(d(i_1(j), j) + d(i_2(j), j)) \leq \frac{1}{2}(r_k + 9r_k) = 5r_k$ where we used Lemma 19 to bound $d(i_1(j), j)$ and $d(i_2(j), j)$).

The farthest that j would have to travel occurs when j and ℓ share secondary facilities, and ℓ's primary facility is opened (see Figure 2). More precisely, this is when $S_\ell \cap S_j = \{i_2(\ell)\} = \{i_2(j)\}$ and $\sigma(\ell) = \sigma(j) = p$ where p is not j or ℓ, and $i_1(\ell)$ is opened at the end of Stage 3. In this case, we have

$$\begin{aligned} d(i_1(\ell), j) &\leq d(i_1(\ell), i_2(\ell)) + d(i_2(\ell), j) \leq d(i_1(\ell), \ell) + d(i_2(\ell), \ell) + d(i_2(j), j) \\ &= 2\hat{C}_\ell + d(i_2(j), j) \leq 2\hat{C}_j + d(i_2(j), j) \leq 10r_k + 9r_k = 19r_k. \end{aligned}$$ ◀

▶ Remark 27. Notice that in the last step of our proof for Lemma 26, we bound the distance $d(i_1(\ell), j)$ by $d(i_1, j) + 2d(i_2(j), j)$, where $d(i_2(j), j) \leq 9r_k$. Hence, the majority of the distance that j is traveling, according to our analysis, is due to the distance between j and its secondary facility. If we could guarantee that cluster center j has a reasonably close secondary facility, we could improve this radius factor. We will explore this further in Section 4.3.

Using Lemmas 11 and 26, we have the following radius bound for the output of our algorithm.

▶ **Theorem 28.** *Let S be the output of the aforementioned approximation algorithm. For all $k \in \mathcal{C}$, $d(k, S) \leq 21r_k$.*

Theorems 25 and 28 together give us Theorem 1.

4 Exploring Cost and Radius Trade-offs

In this section, we will outline the remaining results for PMatMed and UniPMatMed. The algorithms for these results are nearly identical to the one in the previous section. The only change is in setting ϕ and λ (in the first step of Algorithm 2). These results suggest that filtering plays a non-trivial role in the cost and radius trade-offs, and that further improvements may be possible if one finds effective ways to filter points.

We begin with the $(9, 8)$-approximate solution for UniPMatMed. As we discussed in the introduction, having uniform radii allows us to rely only on $\bar{C}_j$ values obtained from the LP solution. We will then discuss how we can extend this approach to the non-uniform case to obtain a $(36, 8)$-approximate solution for PMatMed. Finally, we show how to further tighten the radius guarantee for UniPMatMed.

4.1 $(9, 8)$-approximation for UniPMatMed

First, observe that instances of UniPMatMed can be written as instances of PMatMed, where each $r_j := L$ for all $j \in \mathcal{C}$. As such, our algorithm to obtain a $(9, 8)$-approximation for UniPMatMed is the following: Run Algorithm 2 on UniPMatMed instance $\mathscr{J}$, but in Line 0, set $\phi(j) := \bar{C}_j$ and $\lambda(j) := \min\{r_j, 2\bar{C}_j\}$.

Notice that these assignments of ϕ and λ satisfy compatibility (Definition 7) only when r_j's are uniform. We explain this in more detail in Appendix B, and defer the analysis and proofs for this result to that section.

4.2 $(36, 8)$-approximation for PMatMed

Building off the result for UniPMatMed, which is able to optimize for the cost by changing $\phi(j)$, we show how to obtain a $(36, 8)$-approximate solution for PMatMed. To do so, we will keep the same setting for $\phi(j) := \bar{C}_j$ as UniPMatMed, but will instead choose a λ that is compatible for non-uniform radii. Our algorithm is as follows: Run Algorithm 2 on PMatMed instance $\mathscr{I}$, but in Line 0, set $\phi(j) := \bar{C}_j$ and $\lambda(j) := 2\bar{C}_j$.

Clearly, ϕ and λ are compatible. Furthermore, notice that this setting of ϕ is identical to that of our algorithm of UniPMatMed. Since cost analysis for the filtering stage only uses ϕ (and not λ), our analysis for cost will be identical to that of our analysis of UniPMatMed, therefore we will have a cost guarantee of 8.

To analyze the radius guarantee, notice that while λ does not explicitly use radius values, PMatMed LP has a constraint that ensures $\forall j \in \mathcal{C}$ $\bar{C}_j \leq r_j$. Therefore, $\lambda(j) = 2\bar{C}_j \leq 2r_j$. Our initial setting of λ ($\lambda(j) := \min\{r_j, 2\bar{C}_j\}$) made it so $\lambda(j) \leq r_j$. Hence, the new setting of λ will worsen the radius guarantee of the final solution. The analysis for this result can be found in Appendix C.

4.3 Tighter radius guarantee for UniPMatMed

In the previous result for UniPMatMed, we set $\phi(j) := \bar{C}_j$ and $\lambda := \min\{L, 2\bar{C}_j\}$. In the second result for PMatMed, we showed how setting $\lambda(j) := 2L$ would increase the radius guarantee. Thus, in order to tighten the radius guarantee for UniPMatMed, we will again change $\lambda(j)$, but this time in a way that will allow points to have tighter radius bounds.

To build up to our new setting for λ, we first partition points in the original client set into points that have relatively small, or *tiny* $\bar{C}_j$ values, $\mathcal{C}_T = \{j \in \mathcal{C} \mid \bar{C}_j \leq \epsilon L\}$ and points that have *large* $\bar{C}_j$ values, $\mathcal{C}_L = \{j \in \mathcal{C} \mid \bar{C}_j > \epsilon L\}$. Now, our algorithm is as follows: Run Algorithm 2 on PMatMed instance $\mathscr{I}$, but in Line 0, set $\phi(j) := \bar{C}_j$ and $\lambda(j)$ as defined below.

$$\lambda(j) = \begin{cases} 2\bar{C}_j & j \in \mathcal{C}_T \\ L & j \in \mathcal{C}_L \end{cases}$$

Note that ϕ and λ will satisfy compatibility. Furthermore, this setting of λ improves the radius bound of Section 4.1 since it forces cluster centers from $\mathcal{C}_L$ to open their own primary and secondary facilities (i.e. they will force $\sigma(j) = j$ for cluster centers j that are from $\mathcal{C}_L$). Any point j such that $\sigma(j) \neq j$ will be from $\mathcal{C}_T$, and furthermore $\sigma(j) \in \mathcal{C}_T$ for all of these points. Therefore, we are decreasing the distance between any point and its secondary facility, for both points in $\mathcal{C}_L$ and $\mathcal{C}_T$. As noted in Remark 27, this will help reduce the radius guarantee.

To achieve the $(5+8\epsilon, 4+2/\epsilon)$-approximation result, we also make a change to Section 2.4 of Filter. Details about this change, as well as the full analysis for this result can be found in the full version of this paper.

References

1 Mohsen Abbasi, Aditya Bhaskara, and Suresh Venkatasubramanian. Fair clustering via equitable group representations. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 504–514, 2021.

2 Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. *SIAM Journal on Computing*, 49(4):FOCS17–97–FOCS17–156, 2020. doi:10.1137/18M1171321.

3 Soroush Alamdari and David B. Shmoys. A bicriteria approximation algorithm for the k-center and k-median problems. In *WAOA*, 2017.

4 Ali Aouad and Danny Segev. The ordered k-median problem: surrogate models and approximation algorithms. *Mathematical Programming*, 177(1):55–83, 2019.

5 David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics.

6 Tanvi Bajpai, Deeparnab Chakrabarty, Chandra Chekuri, and Maryam Negahbani. Revisiting priority *k*-center: Fairness and outliers. *arXiv preprint*, 2021. arXiv:2103.03337.

7 Sayan Bandyapadhyay, Tanmay Inamdar, Shreyas Pai, and Kasturi Varadarajan. A Constant Approximation for Colorful k-Center. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms (ESA 2019)*, volume 144 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ESA.2019.12.

8 Jarosław Byrka, Krzysztof Sornat, and Joachim Spoerhase. Constant-factor approximation for ordered k-median. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 620–631, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3188745.3188930.

9 Deeparnab Chakrabarty and Maryam Negahbani. Better algorithms for individually fair *k*-clustering, 2021. arXiv:2106.12150.

10 Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65(1):129–149, 2002. doi:10.1006/jcss.2002.1882.

11 Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. Fair clustering through fairlets. *Advances in Neural Information Processing Systems*, 30, 2017.

12 Jack Edmonds. Submodular functions, matroids, and certain polyhedra. In *Combinatorial Optimization—Eureka, You Shrink!*, pages 11–26. Springer, 2003.

13 Mehrdad Ghadiri, Samira Samadi, and Santosh Vempala. Socially fair k-means clustering. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, pages 438–448, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3442188.3445906.

14 MohammadTaghi Hajiaghayi, Rohit Khandekar, and Guy Kortsarz. Budgeted red-blue median and its generalizations. In *European Symposium on Algorithms*, pages 314–325. Springer, 2010.

15 Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k-center problem. *Math. Oper. Res.*, 10(2):180–184, May 1985. doi:10.1287/moor.10.2.180.

16 Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *J. ACM*, 48(2):274–296, March 2001. doi:10.1145/375827.375845.

17 Christopher Jung, Sampath Kannan, and Neil Lutz. A center in your neighborhood: Fairness in facility location. *arXiv preprint*, 2019. arXiv:1908.09041.

18 Naoyuki Kamiyama. The distance-constrained matroid median problem. *Algorithmica*, 82(7):2087–2106, July 2020. doi:10.1007/s00453-020-00688-5.

19 T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, 2002. doi:10.1109/TPAMI.2002.1017616.

20 Matthäus Kleindessner, Pranjal Awasthi, and Jamie Morgenstern. Fair k-center clustering for data summarization. In *International Conference on Machine Learning*, pages 3448–3457. PMLR, 2019.

21 Ravishankar Krishnaswamy, Amit Kumar, Viswanath Nagarajan, Yogish Sabharwal, and Barna Saha. The matroid median problem. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 1117–1130, USA, 2011. Society for Industrial and Applied Mathematics.

22 Ravishankar Krishnaswamy, Shi Li, and Sai Sandeep. Constant approximation for k-median and k-means with outliers via iterative rounding. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 646–659, 2018.

23 Sepideh Mahabadi and Ali Vakilian. Individual fairness for k-clustering, 2020. arXiv:2002.06742.

24 Stefan Nickel and Justo Puerto. *Location theory: a unified approach*. Springer Science & Business Media, 2006.

25 J. Plesník. A heuristic for the p-center problems in graphs. *Discrete Applied Mathematics*, 17(3):263–268, 1987. doi:10.1016/0166-218X(87)90029-1.

26 Alexander Schrijver et al. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

27 Chaitanya Swamy. Improved approximation algorithms for matroid and knapsack median problems and applications. *ACM Trans. Algorithms*, 12(4), August 2016. doi:10.1145/2963170.

28 Ali Vakilian and Mustafa Yalçıner. Improved approximation algorithms for individually fair clustering, 2021. doi:10.48550/arXiv.2106.14043.

A Omitted Proofs

Proof of Lemma 10. We first prove that $COST'(x,y) \leq 2 \cdot COST(x,y)$. The fractional facility opening cost, $\sum_i f_i y_i$ is identical in both. The difference in the client connection cost is because the demands of clients in $\mathcal{C} \setminus C$ are relocated. Consider a client $k \in \mathcal{C} \setminus C$ that is relocated to its cluster center $j \in C$ (thus $k \in D(j)$). In $COST(x,y)$ client k pays $a_k \bar{C}_k$. In $COST'(x,y)$, the demand of k is moved to j and pays $a_k \bar{C}_j$. Thus, it suffices to prove that $\bar{C}_j \leq 2\bar{C}_k$. From Fact 9, $\phi(j) \leq \phi(k) \leq 2\bar{C}_k$. LP constraints 1d and 1c of the LP for $\mathscr{I}$ ensures that $\bar{C}_j \leq r_j$ for all $j \in \mathcal{C}$. Hence, if $\bar{C}_j > 2\bar{C}_k$ we would have $\phi(j) = \min\{r_j, 2\bar{C}_j\} > 2\bar{C}_k$ which would be a contradiction to $\phi(j) \leq \phi(k)$. This shows that $\bar{C}_j \leq 2\bar{C}_k$.

Now we consider the second part. From Fact 9, $d(j,k) \leq 2\lambda(k) \leq 2(2 \cdot \bar{C}_k)$. Suppose the cost of an integer solution to $\mathscr{I}'$ is α. We keep the same facilities for $\mathscr{I}$ and account for the increase in connection cost when considering the original client locations. Consider a client $k \in \mathcal{C} \setminus C$ that is relocated to center $j \in C$. If j connects to i in the integer solution for $\mathscr{I}'$, k can connect to i in the solution to $\mathscr{I}$, and its per unit connection cost increases by at most $d(j,k) \leq 4\bar{C}_k$. Thus the total increase in the connection cost when comparing to α is upper bounded by $\sum_{j \in C} \sum_{k \in D(j)} a_k \cdot 4\bar{C}_k \leq 4 \cdot COST(x,y)$. ◀

Proof of Lemma 13. Suppose $\mathcal{Q}$ is non-empty and v^* is any extreme point. Then v^* is the unique solution of a linear system $Av = b$ where A is a subset of the inequalities of $\mathcal{Q}$ with A having full row and column rank (in particular the rows of A are linearly independent vectors). A can be partitioned into A_1 and A_2 where A_1 is a subset of the inequalities coming from the matroid $\mathcal{M}$ (of the form $v(S) = r_{\mathcal{M}}(S)$), while A_2 is a subset of the remaining inequalities. Via the submodularity of the matroid rank function, it is known that one can choose A_1 such that the rows of A_1 correspond to a laminar family of subsets of $\mathcal{F}$ [26]. We observe that the non-matroidal system of inequalities in $\mathcal{Q}$ correspond to a laminar family of sets over $\mathcal{F}$: (a) the sets $G_j, j \in C$ are disjoint and $F'_j \subseteq G_j$ for each j and (b) for $j \in C_s$, we have $B_j \subseteq G_j$. See Figure 1.

Thus the rows of the matrix of A come from two laminar families of sets over $\mathcal{F}$, and it is known that such a matrix is totally uniodular [26]. Thus $v^* = A^{-1}b$ where A^{-1} is an integer matrix, and b is half-integral which implies that v^* is half-integral. ◀

Proof of Lemma 17. We first show that $T(y') \leq 4 \cdot COST'(x,y)$ (we already have $T(\hat{y}) \leq T(y')$). We know that $COST'(x,y)$ can be expressed as $\sum_i f_i y_i + \sum_j a'_j \cdot \bar{C}_j$. For any $j \in C$, observe that $\bar{C}_j = \sum_{i \in G_j} d(i,j)x_{ij} + \sum_{i \notin G_j} d(i,j)x_{ij}$ and hence $\bar{C}_j \geq \sum_{i \in G_j} d(i,j)x_{ij} + \gamma_j \sum_{i \notin G_j} x_{ij}$.

$$
\begin{aligned}
T(y') &\leq \sum_i f_i y_i + \sum_j a'_j \Big(2 \sum_{i \in G_j} d(i,j)x_{ij} + 4\gamma_j \Big(1 - \sum_{i \in G_j} x_{ij}\Big)\Big) \\
&\leq \sum_i f_i y_i + 4 \sum_j a'_j \cdot \bar{C}_j \leq 4 \cdot COST'(x,y)
\end{aligned}
$$

Next, we upper bound $COST'(\hat{x},\hat{y})$ by $T(\hat{y})$. It suffices to focus on the assignment cost. Consider $j \in C_s$. Its primary and secondary facilities are in B_j and it is easy to see that its connection cost is precisely $\sum_{i \in B_j} d(i,j)\hat{x}_{ij}$. Now consider $j \in C_b$. Recall that when $\hat{y}(G_j) = 1$, the total assignment cost of j is at most $\sum_{i \in G_j} d(i,j)\hat{y}_i$. When $\hat{y}(G_j) < 1$, j connects to primary facility in F'_j and a secondary facility. The second nearest facility will not be in its G_j ball, i.e. $i_2(j) \notin F_j$. Let $j' \neq j$ be client that defines γ_j. Via Lemma 14, we have $d(i_2(j), j) \leq 3\gamma_j$. Assuming this, when $\hat{y}(G_j) < 1$, the total assignment cost of j is at most $\sum_{i \in G_j} d(i,j)\hat{y}_i + 3\gamma_j(1 - \hat{y}(G_j))$. Based on these assignment cost upper bounds we see that $COST'(\hat{x},\hat{y}) \leq T(\hat{y})$.

Now we prove Lemma 14. From Fact 9 we have $2\max\{\lambda(j), \lambda(j')\} \leq d(j,j')$. Via triangle inequality $d(j,j') \leq d(j,i) + d(i,j') \leq 2\gamma_j$. Thus $2\lambda(j') \leq 2\gamma_j$ which implies that $\lambda(j') \leq \gamma_j$. Recall that $F'_{j'}$, from its definition, is contained in a ball of radius $\lambda(j')$ around j'. Thus, for any facility $i' \in F'_{j'}$, $d(i',j') \leq \lambda(j') \leq \gamma_j$, Therefore, $d(i',j) \leq d(j,j') + d(j',i') \leq 3\gamma_j$. This gives us the lemma.

Finally, using Lemma 10, we know that $COST'(x,y) \leq 2 \cdot COST(x,y)$, hence $4 \cdot COST'(x,y) \leq 8 \cdot COST(x,y)$. ◀

Proof of Lemma 22. Since the facility costs of $(\tilde{x}, \tilde{y})$ will remain as they are in $H(\tilde{y})$, it suffices to show that for all $j \in C$, the assignment cost of j is at most $L_j(\tilde{y})$. When $j \in C'$, $\mathsf{ctr}(j) = j$ and the assignment cost of j will be exactly $L_j(\tilde{y})$.

Now we consider two possibilities for $j' \in C \setminus C'$. Let $\mathsf{ctr}(j') = j$. If j' gets assigned to a center from S_j, there are two possible cases for the value of $L_{j'}(\tilde{y})$. If $i_1(j') \in S_j$ then the assignment cost for j' is exactly $L_{j'}(\tilde{y})$. Otherwise, $i_1(j') \notin S_j$ and $\tilde{y}_{i_1(j')} = 0$. In this case $L_{j'}(\tilde{y}) = \sum_{i \in S_j} a'_{j'}(d(j', \sigma(j)) + d(i, \sigma(j)))\tilde{y}_i$. By triangle inequality, $d(i, j') \leq d(i, \sigma(j')) + d(j, \sigma(j'))$, therefore the assignment cost of j' is at most $L_{j'}(\tilde{y})$.

If j' is assigned to a center that is not from S_j, it is because $\tilde{y}_{i_1(j')} = 1$ and $i_1(j') \notin S_j$. Here, the assignment cost of j' is $a'_{j'}d(i_1(j'), j')$. Let $i \in S_j$ be such that $\tilde{y}_i = 1$. The value of $L_{j'}(\tilde{y})$ is therefore

$$\begin{aligned}L_{j'}(\tilde{y}) &= a'_j\Big(d(j', \sigma(j')) + d(i, \sigma(j')) + d(i_1(j'), j) - d(j', \sigma(j')) - d(i_1(\sigma(j)), \sigma(j))\Big)\\ &= a'_j\Big(d(i, \sigma(j')) + d(i, i_1(j')) - d(i_1(\sigma(j)), \sigma(j))\Big)\end{aligned}$$

Since $i \in S_j$ cannot be closer to $\sigma(j')$ than the primary facility of $\sigma(j')$, we know that $d(i_1(\sigma(j)), \sigma(j)) \leq d(i, \sigma(j'))$. Thus, the assignment cost of j' is at most $L_{j'}(\tilde{y})$. ◀

Proof of Lemma 23. For notational ease, let $Q_j(\hat{y}) := 2\sum_{i \in G_j} d(i,j)\hat{y}_i + 4\gamma_j(1 - \sum_{i \in G_j} \hat{y}_i)$. Thus, $T(\hat{y}) = \sum_i f_i \hat{y}_i + \sum_{j \in C} a'_j Q_j(\hat{y})$. As in the proof of the previous lemma, we focus on just the assignment costs of clients, since clearly $\sum_i f_i \hat{y}'_i \leq \sum_i f_i \hat{y}_i$. Specifically, we will show that $L_j(\hat{y}') \leq a'_j Q_j(\hat{y})$ for all $j \in C$. For the remainder of the proof, we omit the term a'_j from both sides of this inequality, since it remains fixed throughout our analysis.

First, we show $\hat{C}_j \leq Q_j(\hat{y})$ for all $j \in C$. Recall that j has no secondary facility when $\hat{y}_{i_1(j)} = 1$, in which case $i_2(j) = i_1(j)$. When $\hat{y}(G_j) = 1$, $\sigma(j) = j$ and the primary and secondary facilities of j are the only facilities in G_j where $\hat{y}_i > 0$. Since $\hat{y}$ is half integral, we get $\hat{C}_j = (d(i_1(j), j) + d(i_2(j), j))/2 = \sum_{i \in G_j} d(i,j)\hat{y}_i \leq Q_j(\hat{y})$. When $\hat{y}(G_j) = 1/2$, $\sigma(j) = \ell \neq j$ and $i_2(j) = i_1(\ell)$. In this case $\hat{C}_j = (d(i_1(j), j) + d(j, \ell) + d(i_1(\ell), \ell))/2$. Using Claim 15 and definitions, $d(j, \ell) + d(\ell, i_1(\ell)) \leq 3\gamma_j$. Therefore $\hat{C}_j \leq \sum_{i \in G_j} d(i,j)\hat{y}_i + 3\gamma_j(1 - \hat{y}(G_j)) \leq Q_j(\hat{y})$. To prove $L_j(\hat{y}') \leq a'_j Q_j(\hat{y})$ we consider several cases.

1. $j \in C'$: we have $\mathsf{ctr}(j) = j$ and $i_1(j) \in S_j$.

$$\begin{aligned}L_j(\hat{y}') = \sum_{i \in S_j} d(i,j)\hat{y}'_i \leq \sum_{i \in S_j} d(i,j)\hat{y}_i &= \frac{1}{2}(\Big(d(i_1(j), j) + d(i_2(j), j)\Big)\\ &\leq \frac{1}{2}\Big(d(i_1(j), j) + d(i_1(j), \sigma(j)) + d(i_2(j), \sigma(j))\Big) \quad \text{(via triangle ineq.)}\\ &= \hat{C}_j \leq Q_j(\hat{y}).\end{aligned}$$

2. $j' \in C \setminus C'$. Let $\mathsf{ctr}(j') = j$. We have $\hat{C}_j \leq \hat{C}_{j'}$.
   a. $i_1(j') \in S_j$. Then $i_2(j) = i_1(j')$ hence $\sigma(j) = j'$.

$$\begin{aligned}L_{j'}(\hat{y}') &= \frac{1}{2}(d(i_1(j), j') + d(i_2(j), j'))\\ &\leq \frac{1}{2}(d(i_1(j), j) + d(j, j') + d(i_2(j), j')) \quad \text{(via triangle ineq.)}\\ &= \hat{C}_j \leq \hat{C}_{j'} \leq Q_{j'}(\hat{y}).\end{aligned}$$

   b. $i_1(j') \notin S_j$: Then $S_j \cap S_{j'}$ is either $\{i_1(j)\}$ or $\{i_2(j)\}$ (Claim 21). In both cases, $\sigma(j') = \ell \neq j'$ and therefore $\hat{y}(G_{j'}) = \hat{y}_{i_1(j')} = 1/2$. Hence

$$L_{j'}(\hat{y}') = \frac{1}{2} \cdot \Big(2d(j', \ell) + d(i_1(j), \ell) + d(i_2(j), \ell) + d(i_1(j'), j') - d(j', \ell) - d(i_1(\ell), \ell)\Big)$$

i. When $S_j \cap S_{j'} = \{i_1(j)\}$, $i_1(j) = i_2(j')$ thus $\ell = j$. Using the fact that $d(i_2(j), j) \leq 2\hat{C}_j - d(i_1(j), j)$, we have

$$\begin{aligned} L_{j'}(\hat{y}') &= \frac{1}{2}\Big(2d(j', j) + d(i_1(j), j) + d(i_2(j), j) + d(i_1(j'), j') - d(j', j) - d(i_1(j), j)\Big) \\ &= \frac{1}{2}\Big(d(j, j') + d(i_2(j), j) + d(i_1(j'), j')\Big) \\ &\leq \frac{1}{2}\Big(d(j, j') + 2\hat{C}_j - d(i_1(j), j) + d(i_1(j'), j')\Big) \\ &\leq \frac{1}{2}\Big(d(j, j') + 2\hat{C}_{j'} - d(i_1(j), j) + d(i_1(j'), j')\Big) \\ &= d(j, j') + d(i_1(j'), j'). \end{aligned}$$

ii. When $S_j \cap S_{j'} = \{i_2(j)\}$, $i_2(j) = i_2(j') = i_1(\ell)$ and so $\ell \neq j, j'$ and $\sigma(j) = \sigma(j') = \ell$. Since $2\hat{C}_j \leq 2\hat{C}_{j'}$, $d(i_1(j), j) + d(j, \ell) \leq d(i_1(j'), j') + d(j', \ell)$. Therefore,

$$\begin{aligned} L_{j'}(\hat{y}') &= \frac{1}{2}\Big(d(j', \ell) + d(i_1(j), \ell) + d(i_1(j'), j')\Big) \\ &\leq \frac{1}{2}\Big(d(j', \ell) + d(i_1(j), j) + d(j, \ell) + d(i_1(j'), j')\Big) \quad \text{(via triangle ineq.)} \\ &\leq \frac{1}{2}\Big(d(j', \ell) + d(i_1(j'), j') + d(j', \ell) + d(i_1(j'), j')\Big) \\ &\leq d(i_1(j'), j') + d(j', \ell). \end{aligned}$$

Thus, in both cases we have

$$\begin{aligned} L_{j'}(\hat{y}') &\leq d(i_1(j'), j') + d(j', \ell) \\ &\leq d(i_1(j'), j') + 2\gamma_{j'} \quad \text{(via Claim 15)} \\ &\leq 2\sum_{i \in G_{j'}} d(i, j')\hat{y}_i + 4\gamma_{j'}(1 - \sum_{i \in G_{j'}} \hat{y}_i) = Q_{j'}(\hat{y}) \quad \text{(since } \hat{y}(G_{j'}) = 1/2). \end{aligned}$$

This finishes the case analysis and the proof. ◀

B Uniform Priority Matroid Median

The UniPMatMed problem is a special case of the PMatMed problem in which all clients have the same radius value L. An instance $\mathscr{I}$ of the UniPMatMed problem can be described using the tuple $(\mathcal{F}, \mathcal{C}, d, \mathbf{f}, L, \mathbf{a}, \mathcal{M})$. We will abuse notation and interpret L as not only a single radius value, but also as a vector from $\mathbb{R}^{\mathcal{C}}$ where each entry is L; this will allow us to use our algorithm for PMatMed on instances of UniPMatMed.

In this section we show how we can take advantage of the uniform radius requirement to improve upon the $(21, 12)$-approximation for PMatMed. In particular, since we have $\bar{C}_j \leq L$ for all $j \in \mathcal{C}$, we can pick points in filtering in order of their $\bar{C}_j$ values and set $\phi(j) := \bar{C}_j$ for Filter. This setting of ϕ will be compatible with the setting of $\lambda(j) := \min\{L, 2\bar{C}_j\}$. Furthermore, Filter with these ϕ and λ functions is identical to the filtering step in Kamiyama's algorithm [18]. Notice that these same settings for PMatMed, i.e. $\phi(j) := \bar{C}_j$ and $\lambda := \min\{r_j, 2\bar{C}_j\}$, are not necessarily compatible. The uniform radius constraint also help us to derive tighter bounds throughout the radius analysis of the PMatMed algorithm.

Using the above observations, our algorithm for UniPMatMed is the following: Run Algorithm 2 on UniPMatMed instance $\mathscr{I}$, but in Line 0, set $\phi(j) := \bar{C}_j$ and $\lambda(j) := \min\{r_j, 2\bar{C}_j\}$. Thus, the only change in the algorithm is the filtering step. We argue that this algorithm yields a better approximation algorithm for UniPMatMed.

▶ **Theorem 29** (Theorem 2a). *There is a $(9, 8)$-approximation algorithm for UniPMatMed.*

B.1 Cost and Radius Analysis for UniPMatMed

Since our algorithm for UniPMatMed only slightly differs from the one in Section 3, we omit several proofs that would be identical. The only change to the cost analysis occurs in the filtering stage (Section 3.1). In particular, we can derive a tighter bound than in Lemma 10. This ultimately leads to the improved cost bound, shown in Theorem 31.

▶ **Lemma 30.** *The following is true of $\mathscr{J}'$. (a) $COST'(x,y) \leq COST(x,y)$. (b) Any integer solution (x',y') for $\mathscr{J}'$ can be converted to an integer solution for $\mathscr{J}$ that incurs an additional cost of at most $4 \cdot COST(x,y)$.*

Recall that $(\tilde{x},\tilde{y})$ is the final integeral solution output by the algorithm.

▶ **Theorem 31.** *$COST(\tilde{x},\tilde{y}) \leq 8 \cdot COST(x,y)$.*

We now analyze the radius guarantee and outline the changes in the analysis. First, we have the following lemma in place of Lemma 11 which also follows directly from Fact 9.

▶ **Lemma 32.** *Let $k \in \mathcal{C}$ be assigned to $j \in C$ after using the Filtering procedure (i.e. $k \in D(j)$). Then, $d(j,k) \leq 2\lambda(k) \leq 2L$.*

Since all radius values are equal, we do not need Lemma 18 to relate the radius values of different clients. We do need to update Lemma 19 and Lemma 26. These updated lemmas are given below.

▶ **Lemma 33.** *Let $k \in \mathcal{C}$ and $k \in D(j)$ for a cluster center $j \in C$. Then (a) $d(j,i_1(j)) \leq \lambda(j) \leq \lambda(k) \leq L$ and (b) when $j \in C_s$ $d(j,i_2(j)) \leq \rho_j \leq L$ and (c) when $j \in C_b$ $d(j,i_2(j)) \leq d(j,\sigma(j)) + d(i_1(\sigma(j)),\sigma(j)) \leq 3\gamma_j \leq 3\rho_j \leq 3L$.*

The reasoning for the preceding lemma is the same as Lemma 19, except L is used in place of r_j and r_k values. This results in the following update to Lemma 26

▶ **Lemma 34.** *Let $j \in C$. The final solution will open a facility i such that $d(i,j) \leq 7L$.*

Lemma 32 and Lemma 34 give us the following improved radius bound for the solution output by the algorithm. This, along with Theorem 31, proves Theorem 29.

▶ **Theorem 35.** *Let S be the output of the aforementioned approximation algorithm for UniPMatMed. For all $k \in \mathcal{C}$, $d(k,S) \leq 9L$.*

C Analysis for $(36,8)$-approximation for PMatMed

In this section we show how to obtain a $(36,8)$-approximate solution for PMatMed. Our algorithm is as follows: Run Algorithm 2 on PMatMed instance $\mathscr{I}$, but in Line 0, set $\phi(j) := \bar{C}_j$ and $\lambda(j) := 2\bar{C}_j$. Clearly, ϕ and λ are compatible. Furthermore, notice that this setting of ϕ is identical to that of our algorithm of UniPMatMed. Since cost analysis for the filtering stage of UniPMatMed only uses ϕ (and not λ), Lemma 30 and Theorem 31 hold in this case as well. This is the reason why the cost factor guarantee will be 8.

Though our setting for λ does not use radius values, from the PMatMed LP constraint, $\forall j \in \mathcal{C}$, $\bar{C}_j \leq r_j$ holds. Therefore, $\lambda(j) = 2\bar{C}_j \leq 2r_j$. Previous settings of λ (where $\lambda(j) := \min\{r_j, 2\bar{C}_j\}$) made it so $\lambda(j) \leq r_j$. Thus the new setting of λ can lead to a weakening of the radius guarantee. First, we formalize the above observation in Fact 36 which we will use to update the radius analysis of Section 3.

▶ **Fact 36.** *The following holds after* Filter *when $\phi(j) := \bar{C}_j$ and $\lambda(j) := 2\bar{C}_j$: (a) $\bar{C}_j \leq r_j$, and hence $\lambda(j) = 2\bar{C}_j \leq 2r_j$, (b) $\forall k \in D(j) \;\; d(j,k) \leq 2\lambda(k) \leq 4C_k \leq 4r_k$.*

The following updated lemmas now hold in place of their counterparts from Section 3. The proofs for these results are identical to those from Section 3 up to certain bounds that change due to the above fact and the subsequent lemmas. These changes occur whenever definitions of ϕ and λ are used in the analysis, and the following lemmas will be invoked in place of their counterparts from Section 3.

▶ **Lemma 37** (Updated Lemma 11). *Let $k \in \mathcal{C}$ be assigned to $j \in C$ after using the Filtering procedure (i.e. $k \in D(j)$). Then, $d(j,k) \leq 2\lambda(k) \leq 4r_k$.*

▶ **Lemma 38** (Updated Lemma 18). *For some $k \in \mathcal{C}$, where $k \in D(j)$, $\rho_j \leq 5r_k$.*

▶ **Lemma 39** (Updated Lemma 19). *Let $k \in \mathcal{C}$ where $k \in D(j)$ for $j \in C$. (a) $d(j, i_1(j)) \leq \lambda(j) \leq \lambda(k) \leq 2r_k$, (b) when $j \in C_s$, $d(j, i_2(j)) \leq \rho_j \leq 5r_k$, and (c) when $j \in C_b$, $d(j, i_2(j)) \leq d(j, \sigma(j)) + d(i_1(\sigma(j)), \sigma(j)) \leq 3\gamma_j \leq 3\rho_j \leq 15r_k$.*

▶ **Lemma 40** (Updated Lemma 26). *Let $k \in \mathcal{C}$, where $k \in D(j)$ for $j \in C$. The final solution will open a facility i such that $d(i,j) \leq 32r_k$.*

Finally, using Lemma 37 Lemma 40, along with Theorem 31, we get the following result.

▶ **Theorem 41** (Theorem 1(b)). *There is a $(36, 8)$-approximation algorithm for Priority Matroid Median.*

Approximation Algorithms for Directed Weighted Spanners

Elena Grigorescu
Department of Computer Science, Purdue University, West Lafayette, IN, USA

Nithish Kumar
Department of Computer Science, Purdue University, West Lafayette, IN, USA

Young-San Lin[1]
Melbourne Business School, Australia

Abstract

In the *pairwise weighted spanner* problem, the input consists of a weighted directed graph on n vertices, where each edge is assigned both a *cost* and a *length*. Furthermore, we are given k terminal vertex pairs and a distance constraint for each pair. The goal is to find a minimum-cost subgraph in which the distance constraints are satisfied.

We study the weighted spanner problem, in which the edges have positive *integral* lengths of magnitudes that are *polynomial* in n, while the costs are *arbitrary* non-negative rational numbers. Our results include the following in the classical offline setting:

- An $\tilde{O}(n^{4/5+\varepsilon})$-approximation algorithm for the weighted pairwise spanner problem. When the edges have unit costs and lengths, the best previous algorithm gives an $\tilde{O}(n^{3/5+\varepsilon})$-approximation, due to Chlamtáč, Dinitz, Kortsarz, and Laekhanukit (Transactions on Algorithms, 2020).
- An $\tilde{O}(n^{1/2+\varepsilon})$-approximation algorithm for the weighted spanner problem when the terminal pairs consist of *all* vertex pairs and the distances must be preserved *exactly*. When the edges have unit costs and arbitrary positive lengths, the best previous algorithm gives an $\tilde{O}(n^{1/2})$-approximation for the all-pair spanner problem, due to Berman, Bhattacharyya, Makarychev, Raskhodnikova, and Yaroslavtsev (Information and Computation, 2013).

We also prove the first results for the weighted spanners in the *online* setting. Our results include the following:

- An $\tilde{O}(k^{1/2+\varepsilon})$-competitive algorithm for the online weighted pairwise spanner problem. The state-of-the-art results are an $\tilde{O}(n^{4/5})$-competitive algorithm when edges have unit costs and arbitrary positive lengths, and a $\min\{\tilde{O}(k^{1/2+\varepsilon}), \tilde{O}(n^{2/3+\varepsilon})\}$-competitive algorithm when edges have unit costs and lengths, due to Grigorescu, Lin, and Quanrud (APPROX, 2021).
- An $\tilde{O}(k^{\varepsilon})$-competitive algorithm for the online weighted single-source (or single-sink) spanner problem. Without distance constraints, this problem is equivalent to the online directed Steiner tree problem. The best previous algorithm for online directed Steiner trees is an $\tilde{O}(k^{\varepsilon})$-competitive algorithm, due to Chakrabarty, Ene, Krishnaswamy, and Panigrahi (SICOMP, 2018).

Our online results also imply efficient approximation algorithms for the corresponding offline problems. To the best of our knowledge, these are the first approximation (online) polynomial-time algorithms with sublinear approximation (competitive) ratios for the weighted spanner problems.

2012 ACM Subject Classification Theory of computation → Online algorithms; Theory of computation → Routing and network design problems; Theory of computation → Rounding techniques

Keywords and phrases directed weighted spanners, linear programming, junction tree

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.8

Category APPROX

Related Version *Full Version*: https://arxiv.org/abs/2307.02774 [32]

[1] Corresponding author.

Funding *Elena Grigorescu*: Supported in part by NSF CCF-1910659, NSF CCF-1910411, and NSF CCF-2228814.
Nithish Kumar: Supported in part by NSF CCF-1910411 and NSF CCF-2228814.

Acknowledgements We thank several anonymous reviewers for their valuable comments and suggestions that improved the quality of the writeup.

1 Introduction

We study a multi-commodity problem in directed graphs, which we call the *pairwise weighted spanner* problem. In this problem, we are given a directed simple graph $G = (V, E)$ with n vertices, and a set of k terminal pairs $D \subseteq V \times V$. Each edge $e \in E$ is associated with a *cost* given by the function $c : E \to \mathbb{R}_{\geq 0}$ and a *length* given by the function $\ell : E \to \mathbb{R}_{\geq 0}$. We say that the graph has unit lengths if $\ell(e) = 1$ (respectively, unit costs if $c(e) = 1$) for all $e \in E$. Each pair $(s, t) \in D$ is associated with a target distance given by a function $Dist : D \to \mathbb{R}_{\geq 0}$. Let $H = (V(H), E(H))$ be a subgraph of G and $d_H(s, t)$ denote the *distance* from s to t in H, i.e., the total length of a shortest $s \rightsquigarrow t$ path of edges in $E(H)$. The cost of H is $\sum_{e \in E(H)} c(e)$. The goal is to find a *minimum-cost subgraph* H of G such that the distance from s to t is at most $Dist(s, t)$, namely, $d_H(s, t) \leq Dist(s, t)$ for each $(s, t) \in D$.

The pairwise weighted spanner problem captures many network connectivity problems and is motivated by common scenarios, such as constructing an electricity or an internet network, which requires not only cost minimization but also a delivery time tolerance for the demands. Each edge is thus associated with two "weights" in this setting: the cost and the delivery time. This general formulation has been studied under many variants: when the edges have general lengths and unit costs, one may ask for sparse subgraphs that *exactly* maintain pairwise distances, i.e., *distance preservers*, or for sparse subgraphs that *approximately* maintain pairwise distances, i.e., *spanners*; when the edges have general costs and unit lengths, one may ask for cheap subgraphs that maintain pairwise connectivity, i.e., *Steiner forests*. Spanners and distance preservers are well-studied objects, which have found applicability in domains such as distributed computation [7,49], data structures [4,55], routing schemes [20,47,50,52], approximate shortest paths [8,24,25], distance oracles [8,17,48], and property testing [6,11]. Similarly, Steiner forests have been studied in the context of multicommodity network design [30,34], mechanism design and games [16,42,43,53], computational biology [40,51], and computational geometry [9,13].

A slightly more special case of the pairwise weighted spanner problem was originally proposed by Kortsarz [44] and Elkin and Peleg [27], where it was called the *weighted s-spanner* problem. The precise goal in [27,44] is to find a minimum-cost subgraph that connects *all* the pairs of vertices in G, and each $Dist(s, t) = s \cdot d_G(s, t)$ for some integer s called the *stretch* of the spanner. The work of [27] presents a comprehensive list of inapproximability results for different variants of sparse s-spanners. Even in the special case where edges have unit costs (i.e., the directed s-spanner problem defined below), the problem is hard to approximate within a factor of $O(2^{\log^{1-\varepsilon} n})$ unless $NP \subseteq DTIME(n^{\text{polylog}\, n})$.

In the case when the edges have unit costs, the weighted s-spanner problem is called the *directed s-spanner* problem. For low-stretch spanners, when $s = 2$, there is a tight $\Theta(\log n)$-approximate algorithm [26,44]; with unit lengths and costs, when $s = 3, 4$, there are $\tilde{O}(n^{1/3})$-approximation algorithms [10,23]. For $s > 4$ with general lengths, the best known approximation is $\tilde{O}(n^{1/2})$ [10].

The *pairwise spanner* problem considers graphs with unit edge costs, D can be any subset of $V \times V$, and the target distances are general. The state-of-the-art is $\tilde{O}(n^{4/5})$-approximate for general lengths [33] and $\min\{\tilde{O}(k^{1/2+\varepsilon}), \tilde{O}(n^{3/5+\varepsilon})\}$-approximation for unit lengths [19,33].

When the target distances are infinite and the edges have unit lengths, the pairwise weighted spanner problem captures the *directed Steiner forest* problem. For the directed Steiner forest problem, there is an $\min\{\tilde{O}(k^{1/2+\varepsilon}), \tilde{O}(n^{2/3+\varepsilon})\}$-approximate algorithm for general costs [10, 18] and an $\tilde{O}(n^{4/7})$-approximate algorithm for unit costs [1].

1.1 Our contributions

1.1.1 Pairwise weighted spanners

To the best of our knowledge, none of the variants studied in the literature gives efficient sublinear-factor approximation algorithms for the pairwise weighted spanner problem, even in the case of unit edge length. Our main result for pairwise weighted spanners is stated as follows and proved in Section 2.

▶ **Definition 1** (Pairwise Weighted Spanner).

Instance: *A directed graph $G = (V, E)$ with n vertices and edge costs $c : E \to \mathbb{Q}_{\geq 0}$, edge lengths $\ell : E \to \{1, 2, 3, ..., \mathsf{poly}(n)\}$, and a set $D \subseteq V \times V$ of vertex pairs and their corresponding pairwise distance bounds $Dist : D \to \mathbb{Q}_{\geq 0}$ (where $Dist(s, t) \geq d_G(s, t)$) for every terminal pair $(s, t) \in D$.*

Objective: *Find a min-cost subgraph H of G such that $d_H(s, t) \leq Dist(s, t)$ for all $(s, t) \in D$.*

▶ **Theorem 2.** *For any constant $\varepsilon > 0$, there is a polynomial-time randomized algorithm for Pairwise Weighted Spanner with approximation ratio $\tilde{O}(n^{4/5+\varepsilon})$, which succeeds in resolving all pairs in D with high probability.*[2]

The Pairwise Weighted Spanner problem is equivalent to the problem of finding a minimum-cost Steiner forest under pairwise distance constraints, and hence our result is the first polynomial-time $o(n)$-approximate algorithm for the directed Steiner forests with distance constraints. This problem is hard to approximate within a factor of $O(2^{\log^{1-\varepsilon} n})$ unless $NP \subseteq DTIME(n^{\text{polylog}\, n})$ even for the special case when all vertex pairs are required to be connected and the stretch $s \geq 5$ [44].

1.1.2 All-pair weighted distance preservers

When the target distances are the distances in the given graph G, the spanner problem captures the *distance preserver* problem. When edges have unit costs, there exists a distance preserver of cost $O(n)$ if the number of the source vertices is $O(n^{1/3})$ [12]. When edges have unit costs and lengths, the state-of-the-art result is $\tilde{O}(n^{3/5+\varepsilon})$-approximate [19]. We consider the case where the terminal set consists of *all* vertex pairs and the subgraph is required to preserve the distances of all vertex pairs. This problem is called All-pair Weighted Distance Preserver. We prove the following in Section 3.

▶ **Definition 3** (All-pair Weighted Distance Preserver).

Instance: *A directed graph $G = (V, E)$ with edge costs $c : E \to \mathbb{Q}_{\geq 0}$, edge lengths $\ell : E \to \{1, 2, 3, ..., \mathsf{poly}(n)\}$.*

Objective: *Find a min-cost subgraph H of G such that $d_H(s, t) = d_G(s, t)$, for all $(s, t) \in V \times V$.*

[2] Throughout our discussion, when we say high probability we mean probability at least $1 - 1/n$.

▶ **Theorem 4.** *For any constant $\varepsilon > 0$, there is a polynomial-time randomized algorithm for* All-pair Weighted Distance Preserver *with approximation ratio $\tilde{O}(n^{1/2+\varepsilon})$, which succeeds in resolving all pairs in $V \times V$ with high probability.*

Beside distance preservers, there are other previous special-case results for the all-pair weighted spanner problem. When edges have unit costs, the state-of-the-art is an $\tilde{O}(n^{1/2})$-approximation algorithm [10]. When there are no distance constraints, this problem is termed the *minimum strongly connected subgraph* problem and is equivalent to the all-pair Steiner forest problem. This problem is NP-hard and does not admit a polynomial-time approximation scheme if $NP \neq P$ [39]. The best algorithm is a 3/2-approximation [54].

1.1.3 Online weighted spanners

Next, we turn to online weighted spanners. In the online problem, the directed graph, the edge lengths, and the edge costs are given offline. The vertex pairs and the corresponding target distances arrive one at a time, in an online fashion, at each time stamp. The algorithm must irrevocably select edges at each time stamp and the goal is to minimize the cost, subject to the target distance constraints. We call this problem the Online Pairwise Weighted Spanner problem. For notation convenience, the vertex pair (s_i, t_i) denotes the i-th pair that arrives online.

▶ **Definition 5** (Online Pairwise Weighted Spanner).

Instance: *A directed graph $G = (V, E)$ with edge costs $c : E \to \mathbb{Q}_{\geq 0}$, edge lengths $\ell : E \to \{1, 2, 3, ..., \mathsf{poly}(n)\}$, and vertex pairs $D = \{(s_i, t_i) \in V \times V \mid i \in [k]\}$ (k is unknown) with their corresponding pairwise distance bounds $Dist(s_i, t_i) \in \mathbb{Q}_{\geq 0}$ (where $Dist(s_i, t_i) \geq d_G(s_i, t_i)$) arrive online one at a time.*

Objective: *Upon the arrival of (s_i, t_i) with $Dist(s_i, t_i)$, construct a min-cost subgraph H of G such that $d_H(s_i, t_i) \leq Dist(s_i, t_i)$ by irrevocably adding edges from E.*

The performance of an online algorithm is measured by its *competitive ratio*, namely the ratio between the cost of the online solution and that of an optimal offline solution. With unit edge costs, the best algorithm is $\tilde{O}(n^{4/5})$-competitive; with unit edge costs and lengths, the state-of-the-art is $\min\{\tilde{O}(k^{1/2+\varepsilon}), \tilde{O}(n^{2/3+\varepsilon})\}$-competitive [33]. Our result for Online Pairwise Weighted Spanner is stated as follows and proved in Section 4.

▶ **Theorem 6.** *For any constant $\varepsilon > 0$, there is a polynomial-time randomized online algorithm for* Online Pairwise Weighted Spanner *with competitive ratio $\tilde{O}(k^{1/2+\varepsilon})$, which succeeds in resolving all pairs in D with high probability.*

In a special case of Pairwise Weighted Spanner where the source vertex $s \in V$ is fixed, we call this problem Single-source Weighted Spanner. Without distance constraints, this problem is equivalent to the directed Steiner tree problem.[3] The best algorithm for the directed Steiner tree problem is $O(k^{\varepsilon})$-approximate [15].

▶ **Definition 7** (Single-source Weighted Spanner).

Instance: *A directed graph $G = (V, E)$ with edge costs $c : E \to \mathbb{Q}_{\geq 0}$, edge lengths $\ell : E \to \{1, 2, 3, ..., \mathsf{poly}(n)\}$, and a set $D \subseteq \{s\} \times V$ of vertex pairs and their corresponding pairwise distance bounds $Dist : D \to \mathbb{Q}_{\geq 0}$ (where $Dist(s, t) \geq d_G(s, t)$) for every terminal pair $(s, t) \in D$).*

Objective: *Find a min-cost subgraph H of G such that $d_H(s, t) \leq Dist(s, t)$ for all $(s, t) \in D$.*

[3] Throughout the paper, the term *without distance constraints* means that the target distances are infinity. This is equivalent to the connectivity problem.

When $D \subseteq \{s\} \times V$, a single-source weighted spanner connects s to the sinks. We say that s is the *root* of the single-source weighted spanner and the single-source weighted spanner is *rooted at* s. The definition for a single-sink weighted spanner where the terminal pairs share the same sink is defined similarly.

The online version of Single-source Weighted Spanner is termed Online Single-source Weighted Spanner. For notation convenience, the vertex pair (s, t_i) denotes the i-th pair that arrives online.

▶ **Definition 8** (Online Single-source Weighted Spanner).
Instance: *A directed graph $G = (V, E)$ with edge costs $c : E \to \mathbb{Q}_{\geq 0}$, edge lengths $\ell : E \to \{1, 2, 3, ..., \mathsf{poly}(n)\}$, and vertex pairs $D = \{(s, t_i) \mid t_i \in V, i \in [k]\}$ (k is unknown) with their corresponding pairwise distance bounds $Dist(s, t_i) \in \mathbb{Q}_{\geq 0}$ (where $Dist(s, t_i) \geq d_G(s, t_i)$) arrive online one at a time.*
Objective: *Upon the arrival of (s, t_i) with $Dist(s, t_i)$, construct a min-cost subgraph H of G such that $d_H(s, t_i) \leq Dist(s, t_i)$ by irrevocably selecting edges from E.*

The state-of-the-art result for online directed Steiner trees is $\tilde{O}(k^{\varepsilon})$-competitive implied by a more general online buy-at-bulk network design framework [14]. Our result is stated as follows and proved in Section 4.

▶ **Theorem 9.** *For any constant $\varepsilon > 0$, there is a polynomial-time randomized online algorithm for Online Single-source Weighted Spanner with approximation ratio $\tilde{O}(k^{\varepsilon})$, which succeeds in resolving all pairs in D with high probability.*

Our online framework essentially generalizes the online Steiner forest problem by allowing distance constraints when edge lengths are positive integers of magnitude $\mathsf{poly}(n)$. We note that the online algorithms also imply efficient algorithms for the corresponding offline problems with the same approximation ratios.

1.1.4 Summary

We summarize our main results for weighted spanners in Table 1 by listing the approximation (competitive) ratios and contrast them with the corresponding known approximation (competitive) ratios. We note that offline $\tilde{O}(k^{1/2+\varepsilon})$-approximate Pairwise Weighted Spanner and offline $\tilde{O}(k^{\varepsilon})$-approximate Single-source Weighted Spanner can be obtained by our online algorithms.

1.2 High-level technical overview

Most of the literature on approximation algorithms for *offline* spanner problems [10, 11, 19, 21, 28, 33] partition the terminal pairs into two types: thin or thick. A pair $(s, t) \in D$ is *thin* if the graph $G^{s,t}$ induced by feasible $s \rightsquigarrow t$ paths has a small number of vertices, and *thick* otherwise. To connect each thick terminal pair (s, t), it is sufficient to sample vertices from the graph G to hit $G^{s,t}$, and then add shortest-path in-and-out-arborescences rooted at the sampled vertices. To connect each thin terminal pair (s, t), one uses a flow-based linear program (LP) and then rounds the solution.

1.2.1 Pairwise Weighted Spanners

For this problem, the goal is to approximately minimize the total cost while maintaining the required distances between terminal pairs, so it turns out that the approach for directed Steiner forests [10, 28] is more amenable to this formulation. The approach for directed

Table 1 Summary of the approximation and competitive ratios. Here, n refers to the number of vertices and k refers to the number of terminal pairs. All edge lengths are positive integers in $\mathsf{poly}(n)$ and all edge costs are non-negative rational numbers. We note that Pairwise Weighted Spanner without distance constraints is equivalent to the directed Steiner forest problem. The all-pair weighted spanner problem without distance constraints is equivalent to the all-pair Steiner forest problem or the minimum strongly connected subgraph problem.

Problem	Our Results	Previous Results
Pairwise Weighted Spanner	$\tilde{O}(n^{4/5+\varepsilon})$ (Thm 2) $\tilde{O}(k^{1/2+\varepsilon})$ (Thm 6)	$\tilde{O}(n^{4/5})$ (unit edge costs) [33] $\tilde{O}(n^{3/5+\varepsilon})$ (unit edge costs and lengths) [19] $\min\{\tilde{O}(k^{1/2+\varepsilon}), \tilde{O}(n^{2/3+\varepsilon})\}$ (without distance constraints) [10, 18] $\tilde{O}(n^{4/7+\varepsilon})$ (unit edge costs and lengths, without distance constraints) [1]
All-pair Weighted Spanner	$\tilde{O}(n^{1/2+\varepsilon})$ (distance preservers, Thm 4)	$\tilde{O}(n^{1/2})$ (unit edge costs) [10] 3/2 (without distance constraints) [54]
Online Pairwise Weighted Spanner	$\tilde{O}(k^{1/2+\varepsilon})$ (Thm 6)	$\tilde{O}(n^{4/5})$ (unit edge costs) [33] $\min\{\tilde{O}(k^{1/2+\varepsilon}), \tilde{O}(n^{2/3+\varepsilon})\}$ (unit edge costs and lengths) [33] $\tilde{O}(k^{1/2+\varepsilon})$ (without distance constraints) [14]
Single-Source Weighted Spanner	$\tilde{O}(k^{\varepsilon})$ (also holds for online, Thm 9)	$O(k^{\varepsilon})$ (without distance constraints) [15] $\tilde{O}(k^{\varepsilon})$ (online, without distance constraints) [14]

Steiner forests [10, 28] is slightly different, namely, thick pairs are connected by adding cheap paths that contain at least one sampled vertex. In the Steiner forest algorithms, there are usually three cases for the terminal pairs: 1) pairs that are thick and have low-cost connecting paths, 2) the majority of the remaining pairs have high-cost connecting paths, and 3) the majority of the remaining pairs have low-cost connecting paths.

With distance constraints, we have to modify the analysis for *all* three cases. The actual implementation of the strategy requires several new ideas and it significantly departs from the analysis of [10, 28] in several aspects, as we describe below.

In our first case, we cannot simply add cheap paths because they might violate the distance requirement. Instead, we add *feasible* cheap paths that satisfy the distance requirements, in order to connect the terminal pairs. For this purpose, we use the restricted shortest path FPTAS from [37, 45] as our subroutine.

The remaining two cases are resolved by using an iterative greedy algorithm based on a *density* argument. In each iteration, the greedy algorithm constructs a partial solution $E' \subseteq E$ with low density. We define the density of E' to be the ratio of the total edge cost of E' to the number of pairs connected by E'. Iteratively adding low-density partial solutions leads to a global solution of approximately minimum cost.

In the second case, [10,28] use the low-density *junction trees* (the union of an in-arboresence and an out-arboresence rooted at the same vertex) from [18] in order to connect pairs with high-cost paths. However, the junction tree approximation in [18] cannot handle the distance constraints in our setting. Fortunately, with slight modifications, the junction tree approximation from [19] can be made to handle our distance requirements.

In the third case, [10, 28] formulate an LP where each edge has an indicator variable, then round the LP solution, and argue that with high probability, one can obtain a low-density partial solution that connects the terminal pairs with cheap paths. Two challenges arise in our setting. First, the LP formulation is different from the one in [10, 28] because we have to handle both distance and cost requirements. We resolve these constraints by using a different separation oracle from the previous literature [37, 45], namely we use the FPTAS for the *resource-constrained shortest path* problem from [38] (see Section A.1 for more details). Secondly, in order to round the LP solution, we can no longer use the analysis in [10]. This is because the LP rounding scheme uses a union bound that depends on the number of the minimal subset of edges whose removal disconnects the terminal pairs (i.e., anti-spanners). Since we have to handle both lengths and costs in the LP constraints, we consider all possible subsets of edges and this is sufficient to achieve the $\tilde{O}(n^{4/5+\varepsilon})$-approximation.

1.2.2 All-pair Weighted Distance Preservers

For this problem, the solution takes advantage of the requirement that we have to *exactly* preserve the *all-pair* distances. It turns out that the strategy for the spanner problems [10, 11, 19, 21, 33] is more amenable. Recall that terminal pairs are either thin or thick.

To settle thick terminal pairs, most of the previous work that considers graphs with unit edge cost samples vertices from the graph G to hit $G^{s,t}$, and then adds shortest-path in-and-out-arborescences rooted at the sampled vertices. Note that the cost of an in-arborescence or an out-arborescence is always $n-1$. However, with edge costs, it is not clear how the *cheapest* shortest-path in-and-out arborescences can be obtained. Instead, we add cheap *single-source and single-sink weighted distance preservers* rooted at the sampled vertices. This approach requires using the algorithm for Online Single-source Weighted Spanner described in details in Section 4. The key observation is that the terminal pairs of any single-source (single-sink) weighted distance preserver is a subset of all vertex pairs. This implies that any approximately optimal single-source (single-sink) weighted distance preserver must be cheap compared to the cost of the optimal all-pair weighted distance preserver.

The approach that settles the thin pairs closely follows the algorithm in [10], which rounds a fractional solution of the LP for all-pair spanners. Different from Pairwise Weighted Spanner, we only have to handle the lengths in the LP constraints, so the analysis follows [10] and we can get a better approximation ratio. Ultimately, the costs for settling thick and thin pairs are both at most an $\tilde{O}(n^{1/2+\varepsilon})$ factor of the optimal solution.

1.2.3 Online Weighted Spanners

The main challenge for the online pairwise weighted spanner problem is that the standard greedy approach, which iteratively extracts low-density greedy solutions partially connecting terminal pairs, is no longer applicable. Another challenge is to handle the distance constraints for the terminal pairs that arrive online. Fortunately, the online spanner framework from [33] already adapts both the approach introduced in [14], which constructs a *collection of junction trees* in an online fashion, and the approach of [19], which judiciously handles distance constraints when edges have unit lengths. Our online results are obtained by extending the framework of [33] from graphs with unit edge costs to general edge costs.

1.3 Organization

In Section 2, we present the $\tilde{O}(n^{4/5+\varepsilon})$-approximation algorithm for Pairwise Weighted Spanner. In Section 3, we present the $\tilde{O}(n^{1/2+\varepsilon})$-approximation algorithm for All-pair Weighted Distance Preserver. In Section 4, we present the $\tilde{O}(k^{1/2+\varepsilon})$-competitive algorithm for Online Pairwise Weighted Spanner and the $\tilde{O}(k^{\varepsilon})$-competitive algorithm for Online Single-source Weighted Spanner. We refer the reader to Appendix A for a detailed description of related work and Appendix C for the concluding remarks.

2 Pairwise Weighted Spanners

In this section, we prove Theorem 2. For ease of presentation, we assume that we have a guess for the cost of the optimal solution - OPT for that instance as in [10, 28]. Let τ denote the value of our guess. We set $\tau_0 = \min_{e \in E}\{c(e) \mid c(e) > 0\}$; then we carry out multiple iterations of our overall procedure setting τ to be equal to an element in $\{(\tau_0, 2 \cdot \tau_0, 4 \cdot \tau_0, \ldots, 2^i \cdot \tau, \ldots, \sum_{e \in E} c(e))\}$ for those iterations. Finally, we take the cheapest spanner from across all these iterations. Thus, it is sufficient to give the approximation guarantee for the iteration when $\mathsf{OPT} \leq \tau \leq 2 \cdot \mathsf{OPT}$. We can obtain this guess in $O(\log(\sum_{e \in E} c(e)/\tau_0))$ iterations, which is polynomial in the input size.

We define some notions commonly used in the spanner and Steiner forest literature. Fix some parameters $\beta = n^{3/5}$ and $L = \tau / n^{4/5}$. We say that a path $p(s,t)$ that connects a specific terminal pair (s,t) is *feasible* if $\sum_{e \in p(s,t)} \ell(e) \leq \mathit{Dist}(s,t)$. We say that $p(s,t)$ is *cheap* if the $\sum_{e \in p(s,t)} c(e) \leq L$. We say that a terminal pair $(s,t) \in D$ is *thick* if the *local graph* $G^{s,t} = (V^{s,t}, E^{s,t})$ induced by the vertices on feasible $s \leadsto t$ paths of cost at most L has at least n/β vertices; we say it is *thin* otherwise. We note that the definitions of thick and thin pairs are slightly different from how they are defined in [10, 28]. We say that a set $E' \subseteq E$ settles (or resolves) a pair $(s,t) \in D$ if the subgraph (V, E') contains a feasible $s \leadsto t$ path.

2.1 Resolving thick pairs

Let $S = \{s \mid \exists t : (s,t) \in D\}$ and $T = \{t \mid \exists s : (s,t) \in D\}$. We first settle the thick pairs with high probability. We do this by sampling some vertices using Algorithm 1 and then adding some incoming paths and outgoing paths from the samples to the vertices in S and T respectively using Algorithm 2. We ensure that any path we build is both feasible and cheap and we do that with the help of a black box for the Restricted Shortest Path problem.

Algorithm 1 Sample($G(V, E)$).

```
$R \leftarrow \phi$, $k \leftarrow 3\beta \ln n$.
Sample $k$ vertices independently and uniformly at random and store them in the set $R$.
return $R$.
```

▷ Claim 10. Algorithm 1 selects a set of samples R such that with high probability any given thick pair (s,t) has at least one vertex from its local graph in R.

In Algorithm 2, we call Algorithm 1 to get a set of samples R. For each $u \in R, s \in S, t \in T$, we try to add a shortest $s \leadsto u$ path and a shortest $u \leadsto v$ path each of cost at most L.

For this purpose, we use the restricted shortest path (bi-criteria path) problem from [45].

▶ **Definition 11** (Restricted Shortest Path).

Instance: *A directed graph $G = (V, E)$, edge lengths $\ell : E \to \mathbb{R}_{\geq 0}$, edge costs $c : E \to \mathbb{R}_{\geq 0}$, a vertex pair $(s,t) \in V \times V$, and a threshold $T \in \mathbb{R}_{>0}$.*

Objective: *Find a minimum cost $s \leadsto t$ path P such that $\sum_{e \in P} \ell(e) \leq T$.*

The following lemma from [37, 45] gives an FPTAS for Restricted Shortest Path.

▶ **Lemma 12** ([37, 45]). *There exists an FPTAS for Restricted Shortest Path that gives a $(1 + \varepsilon, 1)$ approximation, i.e., the path ensures that $\sum_{e \in P} \ell(e) \leq T$ and has a cost at most $1 + \varepsilon$ times the optimal.*

Using Lemma 12 as our black box, we binary search for a path of length between $\min_{e \in E}\{\ell(e)\}$ and $n \cdot \max_{e \in E}\{\ell(e)\}$ that will give us a cheap $s \rightsquigarrow u$ path. Since the edge lengths and thus the path lengths are all integers, this is possible in $O(\log(n \cdot \max_{e \in E}\{\ell(e)\}))$ iterations which is polynomial in the input size. It is possible that we never find an $s \rightsquigarrow u$ path of cost less than L, in which case we just ignore this (s, u) pair. We then do the same for all the (u, t) pairs. See the full details in Algorithm 2.

Algorithm 2 Thick pairs resolver $(G(V, E), \{\ell(e), c(e)\}_{e \in E})$.

```
R ← ϕ, G′ ← ϕ.
R ← Sample(G(V, E)).
for u ∈ R do
    for s ∈ S do
        Use RSP to find the shortest s ⇝ u path of cost at most L · (1 + ε) and add it to G′.
for u ∈ R do
    for t ∈ T do
        Use RSP to find the shortest u ⇝ t path of cost at most L · (1 + ε) and add it to G′.
return G′
```

▶ **Lemma 13.** *With high probability, the set of edges returned by Algorithm 2 resolves all thick pairs in D with a total cost $\tilde{O}(n^{4/5} \cdot \tau)$. Moreover, Algorithm 2 runs in polynomial time.*

Proof. If some $u \in R$ was originally in the local graph $G^{s,t}$, then Algorithm 2 would have added at least one $s \rightsquigarrow u \rightsquigarrow t$ path from $G^{s,t}$ that is feasible and has cost less than $2L(1+\varepsilon)$. This is because if u was in the local graph of (s, t), then there exists an $s \rightsquigarrow u$ path of cost less than L of some length $\ell \in [n \cdot \max_{e \in E}\{\ell(e)\}]$. Since we binary search over the possible values for ℓ and take the lowest possible one, we will find such a path with distance at most the minimum length of an $s \rightsquigarrow t$ path that is cheap. Note that we could have a smaller distance because we use a larger bound for cost for our path in comparison to the local graph. Using Claim 10, such a cheap and feasible path will exist with a high probability for all $(s, t) \in D$ that are thick for some samples $u \in R$.

Now we analyze the cost of Algorithm 2. The cost from all the edges we add in Algorithm 2 would be $O(n \cdot k \cdot L)$. This is because we pick k samples and each of them needs to add an incoming and outgoing path of cost $L(1 + \varepsilon)$ to at most n vertices. Plugging in the values for k and L, we can see that the total cost would be $\tilde{O}(n^{4/5} \cdot \tau)$. In addition, note that Algorithm 2 will run in polynomial time because our binary search only needs to search the integers in $[\min_{e \in E}\{\ell(e)\}, n \cdot \max_{e \in E}\{\ell(e)\}]$.[4] ◀

2.2 Resolving thin pairs

Now we focus on the thin pairs after removing the settled thick pairs from the set D. The *density* of a set of edges E' is the ratio of the total cost of E' to the number of pairs settled by E'. We first describe how to efficiently construct a subset K of edges with density $\tilde{O}(n^{4/5+\varepsilon})\tau/|D|$. Then we iteratively find edge sets with that density, remove the pairs, and repeat until we resolve all thin pairs. This gives a total cost of $\tilde{O}(n^{4/5+\varepsilon} \cdot \tau)$. We construct K by building two other sets K_1 and K_2 and picking the smaller density of them. Let H be

[4] Note that Algorithm 2 will still run in polynomial-time if we use an exhaustive search instead of a binary search since the edge lengths are in $\mathsf{poly}(n)$.

an optimal solution with cost τ. Let C be the set of demand pairs for which the minimum cost of a feasible $s \rightsquigarrow t$ path in H is at least L (note that the local graph for these pairs would be empty). We have two cases: 1) $|D|/2 \leq |C| \leq |D|$ and 2) $0 \leq |C| < |D|/2$.

2.2.1 When $|D|/2 \leq |C| \leq |D|$

We will use the notion of *junction tree* as a black box for resolving this case. Informally, junction trees are trees that satisfy significant demand at low cost. They have a root node r, a collection of paths into r, and paths out of r to satisfy some of this demand. In our case, we also need these paths to be cheap and short. The following is a formal definition of a junction tree variant that fits the needs of our problem.

▶ **Definition 14** (*Distance-preserving Weighted Junction Tree*)**.** *Let $G = (V, E)$ be a directed graph with edge lengths $\ell : E \to \mathbb{R}_{\geq 0}$, edge costs $c : E \to \mathbb{R}_{\geq 0}$, and a set $D \subseteq V \times V$ of ordered pairs and their corresponding pairwise distance bounds $Dist : D \to \mathbb{R}$ (where $Dist(s,t) \geq d_G(s,t)$ for every terminal pair $(s,t) \in D$), and a root $r \in V$. We define* distance-preserving weighted junction tree *to be a subgraph H of G that is a union of an in-arboresences and an out-arboresences both rooted at r containing an $s \rightsquigarrow t$ path going through the root r of length at most $Dist(s,t)$, for one or more $(s,t) \in D$.*

The *density* of a junction tree is defined as the ratio of the sum of costs of all edges in the junction tree to the number of pairs settled by the junction tree.

▷ Claim 15. If $|D|/2 \leq |C| \leq |D|$, then there exists a distance-preserving weighted junction tree of density $O\left(n^{4/5} \cdot \tau/|D|\right)$.

Proof. Let H be an optimal solution subgraph of G that connects all the costly thin pairs. Take the paths in H connecting the pairs in C. The sum of the costs of all such paths is at least $|C|L$. Now, let μ be the maximum number of these paths that any edge in G belongs to. The sum of the costs of the paths is at most $\mu \cdot \tau$ and thus there must exist an edge belonging to $\mu \geq |C|L/\tau$ paths. Pick such an arbitrary edge and call it the *heavy-enough edge*, and call its source as the *heavy-enough vertex*, denoted h_v. Now, consider a tree made by adding feasible paths from $s \in S$ to h_v and h_v to $t \in T$ that satisfies at least μ pairs. We do not add an edge if it is not in H. This ensures that the cost of this tree is less than τ. This tree would connect at least μ pairs, and thus it would have a density at most $\tau/\mu = \tau^2/(|C|L)$.

If $L = \tau/n^{4/5}$, then $\tau^2/(|C|L) = n^{4/5} \cdot \tau/|C|$. If $|D| > |C| > |D|/2$, we have $n^{4/5} \cdot \tau/|C| = O\left(n^{4/5} \cdot \tau/|D|\right)$. We have proved the existence of a junction tree of the required density. ◁

The following lemma is essentially from [19] (Theorem 5.1). But the small yet important modifications that we need are not covered in [19]. We refer the reader to the full version [32] for the complete proof.

▶ **Lemma 16.** *For any constant $\varepsilon > 0$, there is a polynomial-time approximation algorithm for the minimum density distance-preserving junction tree as long as the edge lengths are integral and polynomial in n. In other words, there is a polynomial time algorithm which, given a weighted directed n vertex graph $G = (V, E)$ where each edge $e \in E$ has a cost $c(e) \in \mathbb{R}_{\geq 0}$ and integral length $\ell(e) \in \{1, 2, ..., \mathsf{poly}(n)\}$, terminal pairs $P \subseteq V \times V$, and distance bounds $Dist : P \to \mathbb{N}$ (where $Dist(s,t) \geq d_G(s,t)$) for every terminal pair $(s,t) \in P$, approximates the following problem to within an $O(n^{\varepsilon})$ factor:*

- *Find a non-empty set of edges $F \subseteq E$ minimizing the ratio:*

$$\min_{r \in V} \frac{\sum_{e \in F} c(e)}{|\{(s,t) \in P | d_{F,r}(s,t) \leq Dist(s,t)\}|} \tag{1}$$

where $d_{F,r}(s,t)$ is the length of the shortest path using edges in F which connects s to t while going through r (if such a path exists).

▶ **Lemma 17.** *When $|D|/2 \leq |C| \leq |D|$, we can get a set of edges K_1 that has density at most $\tilde{O}(n^{4/5+\varepsilon} \cdot \tau/|D|)$*

Proof. From Claim 15, there exists a distance-preserving weighted junction tree of density at most $O(n^{4/5} \cdot \tau/|D|)$. We use Lemma 16 to get a distance-preserving weighted junction tree with density at most $\tilde{O}(n^{4/5+\varepsilon} \cdot \tau/|D|)$ and store the edges returned by it in K_1. ◀

2.2.2 When $0 \leq |C| < |D|/2$

To handle this case, we build a linear program (LP) that fits our problem's requirements, solve it approximately with the help of a separation oracle, and finally round it to get a set of edges with density $\tilde{O}(n^{4/5+\varepsilon}) \cdot \tau/|D|$. The linear program is quite similar to the one used in [10, 28], but it has a subtle distinction that significantly changes the tools and proof techniques we have to use. We will be referring to [28] quite frequently in this section because [10] does not directly present a way to solve the LP (it relies on [28] for this).

2.2.2.1 Building and solving the linear program

We will need the following definition in order to set up a relevant LP. For $(s,t) \in D$, let $\Pi(s,t)$ be the set of all *feasible* $s \rightsquigarrow t$ paths of cost at most L, and let $\Pi = \cup_{(s,t)\in D}\Pi(s,t)$. Each edge e has a capacity x_e, each path $p \in \Pi$ carries f_p units of flow, and $y_{s,t}$ is the total flow through all paths from s to t. Define a linear program as follows:

$$\begin{aligned}
\min \quad & \sum_{e \in E} c(e) \cdot x_e & & \\
\text{subject to} \quad & \sum_{(s,t)\in D} y_{s,t} \geq \frac{|D|}{2}, & & \\
& \sum_{\Pi(s,t) \ni p \ni e} f_p \leq x_e & & \forall (s,t) \in D, e \in E, \\
& \sum_{p \in \Pi(s,t)} f_p = y_{s,t} & & \forall (s,t) \in D, \\
& 0 \leq y_{s,t}, f_p, x_e \leq 1 & & \forall (s,t) \in D, p \in \Pi, e \in E.
\end{aligned} \tag{2}$$

LP (2) tries to connect at least $|D|/2$ pairs from D using paths of cost at most L while minimizing the total cost of the used edges. It is almost identical to the corresponding LP in [10, 28] for Steiner forests, except that we consider only *feasible* paths that are cheaper than L, while they consider all paths that are cheaper than L. The crucial part of our approach is the use of an FPTAS for the Restricted Shortest Path problem, which served as an approximate separation oracle for the dual program. The proof of the following lemma is provided in Appendix B.1.

▶ **Lemma 18.** *Let OPT be the optimal value of an instance of* Pairwise Weighted Spanner. *Then, the optimal value of LP (2) corresponding to that instance is at most OPT. In addition, a solution for LP (2) of value at most $(1+\varepsilon) \cdot OPT$ can be found in polynomial time.*

2.2.2.2 Rounding our solution

Now we need to round the solution of LP (2) appropriately to decide which edges we need to include in our final solution. The overall structure of our rounding procedure is similar to that of [10], but there are some important differences in the proof techniques we use here because the nature of our problem prevents us from using some of the techniques used by [10]. Let $\{\hat{x}_e\} \cup \{\hat{y}_{s,t}\}$ be a feasible approximate solution to LP (2). Let K_2 be the set of edges obtained by running Algorithm 3 on $\{\hat{x}_e\}$.

Algorithm 3 Thin pair rounding [LP rounding] (x_e).

```
$E'' \leftarrow \phi$ .
for $e \in E$ do
    Add $e$ to $K_2$ with probability $\min\{n^{4/5} \ln n \cdot x_e, 1\}$;
return $E''$
```

The following lemma is an adaptation of Claim 2.3 from [10].

▷ Claim 19. Let $A \subseteq E$. If Algorithm 3 receives a fractional vector $\{\hat{x}_e\}$ with nonnegative entries satisfying $\sum_{e \in A} \hat{x}_e \geq 2/5$, the probability that it outputs a set E'' disjoint from A is at most $\exp(-2n^{4/5} \cdot \ln n/5)$.

Proof. If A contains an edge e which has $\hat{x}_e \geq 1/(n^{4/5} \ln n)$, then e is definitely included in E''. Otherwise, the probability that no edge in A is included in E'' is

$$\prod_{e \in A} (1 - n^{4/5} \ln n \cdot \hat{x}_e) \leq \exp\left(-\sum_{e \in A} n^{4/5} \ln n \cdot \hat{x}_e\right) \leq \exp\left(-\frac{2}{5} n^{4/5} \ln n\right). \quad \triangleleft$$

Let us now define anti-spanners which serve as a useful tool to analyze the rounding algorithm for our LP. Our definition of anti-spanners is slightly different from Definition 2.4 in [10] to account for the fact we also have distance constraints.

▶ **Definition 20.** *A set $A \subseteq E$ is an anti-spanner for a terminal pair $(s,t) \in E$ if $(V, E \setminus A)$ contains no feasible path from s to t of cost at most L. If no proper subset of anti-spanner A for (s,t) is an anti-spanner for (s,t), then A is minimal. The set of all minimal anti-spanners for all thin edges is denoted by $\mathcal{A}$.*

The following lemma is an analogue of Claim 2.5 from [10].

▶ **Lemma 21.** *Let $\mathcal{A}$ be the set of all minimal anti-spanners for thin pairs. Then $|\mathcal{A}|$ is upper-bounded by $|D| \cdot 2^{(n/\beta)^2/2}$.*

Proof. Let $PS(s,t)$ be the power set of all edges in the local graph for a given thin pair (s,t). Since (s,t) is a thin pair we have at most n/β vertices and $(n/\beta)^2/2$ edges in the local graph, therefore $|PS(s,t)| \leq 2^{(n/\beta)^2/2}$ for any (s,t) that is a thin pair. Now, every anti-spanner for a specific demand pair $(s,t) \in D$ is a set of edges and therefore corresponds to an element in $PS(s,t)$. Let $PS_{\text{thin}} = \bigcup_{(s,t)} PS(s,t)$ where $(s,t) \in D$ are thin pairs. Every anti-spanner for a thin pair is a set of edges and therefore corresponds to an element in PS_{thin}. We have $|\mathcal{A}| \leq |PS_{\text{thin}}| \leq |D| \cdot 2^{(n/\beta)^2/2}$ which proves the lemma. ◀

The rest of this discussion is quite similar to [10] although the exact constants and the expressions involved are different because of the result in Lemma 21. Lemma 22 is similar to Lemma 5.2 from [10].

▶ **Lemma 22.** *With high probability set K_2 settles every thin pair (s,t) with $\hat{y}_{s,t} \geq 2/5$.*

Proof. For every thin pair $(s,t) \in D$ with $\hat{y}_{s,t} \geq 2/5$, if A is an anti-spanner for (s,t) then $\sum_{e\in A} \hat{x}_e \geq \sum_{P\in\Pi(s,t)} \hat{f}_p \geq 2/5$, where $\hat{f}_p$ is the value of the variable f_p in LP (2) that corresponds to the solution $\{\hat{x}_e\} \cup \{\hat{y}_{s,t}\}$.

By Claim 19, the probability that A is disjoint from K_2 is at most $\exp(-2n^{4/5} \cdot \ln n/5)$. Further using Lemma 21, we can bound the number of minimal anti-spanners for thin pairs and then if we apply union bound, we have the probability that K_2 is disjoint from any anti spanner for a thin pair is at most $\exp\left(-2n^{4/5} \cdot \ln n/5\right) \cdot |D| \cdot 2^{(n/\beta)^2/2}$. In the worst case, $|D|$ is n^2. Recall that $\beta = n^{3/5}$, we have $(n/\beta)^2 = n^{4/5}$, so

$$\exp\left(-\frac{2}{5} \cdot n^{4/5} \cdot \ln n + \ln\left(n^2 \cdot 2^{n^{4/5}/2}\right)\right) = \exp\left(-\Theta(n^{4/5} \ln n)\right).$$

Thus we have shown that the probability K_2 is disjoint from any anti-spanner for a thin pair is exponentially small when $\hat{y}_{s,t} \geq 2/5$. ◀

▶ **Lemma 23.** *When $0 \leq |C| < |D|/2$, with high probability, the density of K_2 is at most*

$$\tilde{O}(n^{4/5} \cdot \tau/|D|).$$

Proof. Firstly notice that the expected cost of K_2 would be at most $n^{4/5} \ln n \cdot \tau$. We also point out that the number of pairs $(s,t) \in D$ for which $\hat{y}_{s,t} < 2/5$ is at most $5|D|/6$ because otherwise the amount of flow between all pairs is strictly less than $|D|/2$ which violates a constraint of LP (2). Since with high probability all pairs for which $\hat{y}_{s,t} \geq 2/5$ are satisfied, this means that the expected density of K_2 is at most

$$\frac{n^{4/5} \ln n \cdot \tau}{|D|/6} = \frac{6n^{4/5} \ln n \cdot \tau}{|D|} = \frac{\tilde{O}(n^{4/5} \cdot \tau)}{|D|}.$$ ◀

Now we are ready to prove Theorem 2.

Proof of Theorem 2. Using Lemma 13 we can resolve all thick pairs with high probability with cost at most $\tilde{O}(n^{4/5+\varepsilon})$. Then, we can make two sets of edges K_1 and K_2 using a distance-preserving weighted junction tree and by rounding the approximate solution to LP (2) respectively. By Lemmas 17 and 23, we can see that at least one of them will have a density at most $\tilde{O}(n^{4/5} \cdot \tau/|D|)$. If we take the cheaper among them and keep iterating we can resolve all thin pairs with a high probability and with cost at most $\tilde{O}(n^{4/5+\varepsilon})$. ◀

3 All-pair Weighted Distance Preservers

In this section, we prove Theorem 4. Our proof structure for this subsection is very similar to that of [10] except for our use of single sink and single source spanners.

As in Section 2, we assume that we have a guess for the cost of the optimal solution - OPT for the given instance of All-pair Weighted Distance Preserver. Let τ denote the value of our guess. Let us set $\beta = n^{1/2}$. We say that a terminal pair $(s,t) \in D$ is *thick* if the *local graph* $G^{s,t} = (V^{s,t}, E^{s,t})$ induced by the vertices on feasible paths from s to t has at least n/β vertices; we say it is *thin* otherwise. We note that the definitions of thick and thin pairs are slightly different from how they are defined in Section 2 as we only care about the feasibility of a path, not its cost. We say that a set $E' \subseteq E$ settles (or resolves) a pair $(s,t) \in D$ if the subgraph (V, E') contains a feasible path from s to t.

3.1 Thick pairs

We first resolve the thick pairs by randomly sampling vertices and building single-source and single-sink spanners from the samples using Theorem 9. We then resolve thin pairs by building a linear program, solving it, and rounding as in [10]. As mentioned earlier, our definition of thick and thin pairs is different in this section when compared to Section 2, and this allows us to use a much simpler proof (although one that will be effective only in the case of weighted distance preservers as opposed to the more general weigthed spanners).

Algorithm 4 Thick pairs resolver - Distance preserver $(G(V, E), \{\ell(e), c(e)\}_{e\in E})$.

```
$R \leftarrow \phi$, $G' \leftarrow \phi$.
for $i = 1$ to $\beta \ln n$ do
    $v \leftarrow$ a uniformly random element of $V$.
    $S_v^{source} \leftarrow$ a single-source distance preserver rooted at $v$ with $D = \{v\} \times V$ .
    $S_v^{sink} \leftarrow$ a single-sink distance preserver rooted $v$ with $D = \{v\} \times V$.
    $G' \leftarrow G' \cup S_v^{source} \cup S_v^{sink}$, $R \leftarrow R \cup \{v\}$.
return $G'$
```

▶ **Lemma 24.** *Algorithm 4 resolves all thick pairs for* All-pair Weighted Distance Preserver *with high probability and cost* $\tilde{O}(n^{\varepsilon} \cdot \beta \cdot OPT) = \tilde{O}(n^{1/2+\varepsilon} \cdot OPT)$.

Proof. Let $\mathsf{OPT}(S_v^{source})$ be the optimal costs of a single-source distance preserver rooted at v with $D = \{v\} \times V$ and $\mathsf{OPT}(S_v^{sink})$ be the optimal costs of a single-sink distance preserver rooted at v with $D = \{v\} \times V$.

This theorem also gives an $\tilde{O}(k^{\delta})$-approximation for the offline problem Single-source Weighted Spanner for any constant $\delta > 0$. Single-sink distance preservers can be obtained by simply reversing the edges. The number of terminal pairs $k = \Theta(n^2)$. By setting $\delta = \varepsilon/2$ and the target distances to the exact distances in G for all vertex pairs, we observe that the cost due to one sample in Algorithm 4 is at most $\tilde{O}(n^{\varepsilon}(\mathsf{OPT}(S_v^{source}) + \mathsf{OPT}(S_v^{sink})))$. Note that a distance preserver for all pairs also serves as a distance preserver for any subset of the pairs and thus we have for any $v \in V$, $\mathsf{OPT}(S_v^{sink}) \le \mathsf{OPT}$ and $\mathsf{OPT}(S_v^{source}) \le \mathsf{OPT}$. Thus, using Theorem 9, the cost of the G' returned by Algorithm 4 is at most $|R| \cdot \tilde{O}(n^{\varepsilon} \cdot \mathsf{OPT}) \le \tilde{O}(n^{\varepsilon} \cdot \beta \cdot \mathsf{OPT})$.

Using a hitting set argument very similar to Claim 10, we can see that with high probability, there is at least one sample v such that there is a $s \leadsto v \leadsto t$ path for every $(s, t) \in V \times V$ where $d_{G'}(s, v) + d_{G'}(v, t) = d_G(s, t)$.

The single-sink distance preserver gives us a $s \leadsto v$ path of length $d_G(s, v)$ and the single-source distance preserver gives us a $v \leadsto t$ path of length $d_G(v, t)$. Thus, thick pairs are resolved with high probability by the edges in G'. ◀

3.2 Thin pairs

To resolve thin pairs, we start by redefining anti-spanners by ignoring the path costs in Definition 20.

▶ **Definition 25.** *A set $A \subseteq E$ is an anti-spanner for a demand pair $(s, t) \in E$ if $(V, E \setminus A)$ contains no feasible path from s to t. If no proper subset of anti-spanner A for (s, t) is an anti-spanner for (s, t), then A is minimal. The set of all minimal anti-spanners for all thin edges is denoted by $\mathcal{A}$.*

Consider the following LP which is a slightly modified version of a similar LP from [10].

$$\min \sum_{e\in E} c(e)\cdot x_e \quad \text{subject to} \sum_{e\in A} x_e \geq 1 \quad \forall A \in \mathcal{A} \text{ and } x_e \geq 0 \quad \forall e \in E. \tag{3}$$

Let OPT denote the optimal solution to the LP. We can obtain this in a way identical to [10] as we only change the objective (which does not affect the separation oracle). Now, if $\{\hat{x_e}\}$ denotes the vector of x_e's in the solution to LP (3), then add every edge $e \in E$ to G' with probability $\min\{\sqrt{n}\cdot \ln n \cdot \hat{x_e}, 1\}$. We now state the following claim from [10].

▷ Claim 26. Given a feasible solution to LP (3), the rounding procedure produces a set of edges E'' that settles all thin pairs with high probability and has size at most $2\mathsf{OPT}\cdot\sqrt{n}\cdot\ln n$.

▶ **Theorem 4.** *For any constant $\varepsilon > 0$, there is a polynomial-time randomized algorithm for* All-pair Weighted Distance Preserver *with approximation ratio $\tilde{O}(n^{1/2+\varepsilon})$, which succeeds in resolving all pairs in $V \times V$ with high probability.*

Proof of Theorem 4. Using Lemma 24 we can resolve all thick pairs with high probability with cost at most $\tilde{O}(n^{1/2+\varepsilon}\cdot\mathsf{OPT})$ by running Algorithm 4. Then, using Claim 26, we can solve and round LP (3) to resolve the thin pairs with high probability and cost $\tilde{O}(\mathsf{OPT}\cdot\sqrt{n})$. ◀

4 Online Weighted Spanners

This section is dedicated to proving Theorems 6 and 9. The proof outline is as follows.

1. We first show that there exists an α-approximate solution consisting of distance-preserving weighted junction trees (see Definition 14). Here, $\alpha = O(\sqrt{k})$ for Pairwise Weighted Spanner and $\alpha = 1$ for Single-source Weighted Spanner.
2. We slightly modify the online algorithm from [33] to find an online solution consisting of distance-preserving weighted junction trees by losing a factor of $\tilde{O}(k^{\varepsilon})$.

The main difference between the online approach and the offline approach in Section 2 is that we cannot greedily remove partial solutions to settle the terminal pairs in the online setting. Instead, we construct a distance-preserving weighted junction tree solution in an online fashion.

▶ **Definition 27.** *A* distance-preserving weighted junction tree solution *is a collection of distance-preserving weighted junction trees rooted at different vertices, that satisfies all the terminal distance constraints.*

We construct a distance-preserving weighted junction tree solution online and compare the online objective with the optimal distance-preserving weighted junction tree solution with objective value OPT_{junc}. The following theorem is essentially from [33] for the case when the edges have unit costs and lengths. However, the slight yet important modifications that we need when edges have arbitrary positive costs and integral lengths in $\mathsf{poly}(n)$ are not covered in [33]. We refer the reader to the full version [32] for the complete proof.

▶ **Theorem 28.** *For any constant $\varepsilon > 0$, there exists a polynomial-time randomized online algorithm for* Online Pairwise Weighted Spanner *that constructs a distance-preserving weighted junction tree solution online with a cost at most $\tilde{O}(k^{\varepsilon})OPT_{junc}$ with high probability.*

With this theorem, we are ready to prove Theorems 6 and 9.

Proof for Theorems 6 and 9. Let OPT denote the cost of the optimal solution and α denote the ratio between OPT_{junc} and OPT. It suffices to show that $\alpha = O(\sqrt{k})$ for Pairwise Weighted Spanner and $\alpha = 1$ for Single-source Weighted Spanner because Theorem 28 implies the existence of an $\tilde{O}(\alpha k^{\varepsilon})$-competitive online algorithm.

To show that $\alpha = 1$ for Single-source Weighted Spanner, let H be an optimal solution. We observe that H itself is a distance-preserving weighted junction tree rooted at the source s that is connected to all the k sinks, so $\alpha = 1$.

To show that $\alpha = O(\sqrt{k})$ for Pairwise Weighted Spanner, we use a density argument via a greedy procedure which implies an $O(\sqrt{k})$-approximate distance-preserving weighted junction tree solution. We recall the density notion in Section 2.2. The density of a distance-preserving weighted junction tree is its cost divided by the number of terminal pairs that it connects within the required distances.

Intuitively, we are interested in finding low-density distance-preserving weighted junction trees. We show that there always exists a distance-preserving weighted junction tree with density at most a $\sqrt{k}$ factor of the optimal density. The proof of Lemma 29 closely follows the one for the directed Steiner network problem in [18] and pairwise spanners [33] by considering whether there is a *heavy* vertex that lies in $s_i \rightsquigarrow t_i$ paths for distinct i or there is a simple path with low density. The case analysis also holds when there is a distance constraint for each (s_i, t_i). We refer the reader to Appendix B.2 for the complete proof.

▶ **Lemma 29.** *There exists a distance-preserving weighted junction tree J with density at most $\mathsf{OPT}/\sqrt{k}$.*

Consider the procedure that finds a minimum density distance-preserving weighted junction tree in each iteration, and continues on the remaining disconnected terminal pairs. Suppose there are t iterations, and after iteration $j \in [t]$, there are n_j disconnected terminal pairs. Let $n_0 = k$ and $n_t = 0$. After each iteration, the minimum cost for connecting the remaining terminal pairs in the remaining graph is at most OPT, so the total cost of this procedure is upper-bounded by

$$\sum_{j=1}^{t} \frac{(n_{j-1} - n_j)\mathsf{OPT}}{\sqrt{n_{j-1}}} \leq \sum_{i=1}^{k} \frac{\mathsf{OPT}}{\sqrt{i}} \leq \int_{1}^{k+1} \frac{\mathsf{OPT}}{\sqrt{x}} dx = 2\mathsf{OPT}(\sqrt{k+1} - 1) = O(\sqrt{k})\mathsf{OPT}$$

where the first inequality uses the upper bound by considering the worst case when only one terminal pair is removed in each iteration of the procedure. ◀

References

1 Amir Abboud and Greg Bodwin. Reachability preservers: New extremal bounds and approximation algorithms. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1865–1883. SIAM, 2018.

2 Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen Kobourov, and Richard Spence. Graph spanners: A tutorial review. *Computer Science Review*, 37:100253, 2020.

3 Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. A general approach to online network optimization problems. *ACM Transactions on Algorithms (TALG)*, 2(4):640–660, 2006.

4 Noga Alon and Baruch Schieber. Optimal preprocessing for answering on-line product queries. Technical report, Tel-Aviv University, 1987.

5 Spyridon Antonakopoulos. Approximating directed buy-at-bulk network design. In *International Workshop on Approximation and Online Algorithms*, pages 13–24. Springer, 2010.

6 Pranjal Awasthi, Madhav Jha, Marco Molinaro, and Sofya Raskhodnikova. Testing lipschitz functions on hypergrid domains. *Algorithmica*, 74(3):1055–1081, 2016.

7 Baruch Awerbuch. Communication-time trade-offs in network synchronization. In *Proceedings of the 4th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 272–276, New York, NY, USA, 1985.

8 Surender Baswana and Telikepalli Kavitha. Faster algorithms for all-pairs approximate shortest paths in undirected graphs. *SIAM Journal on Computing*, 39(7):2865–2896, 2010.

9 MohammadHossein Bateni and MohammadTaghi Hajiaghayi. Euclidean prize-collecting steiner forest. *Algorithmica*, 62(3-4):906–929, 2012.

10 Piotr Berman, Arnab Bhattacharyya, Konstantin Makarychev, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Approximation algorithms for spanner problems and directed steiner forest. *Information and Computation*, 222:93–107, 2013.

11 Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P Woodruff. Transitive-closure spanners. *SIAM Journal on Computing*, 41(6):1380–1425, 2012.

12 Greg Bodwin. New results on linear size distance preservers. *SIAM Journal on Computing*, 50(2):662–673, 2021.

13 Glencora Borradaile, Philip N Klein, and Claire Mathieu. A polynomial-time approximation scheme for euclidean steiner forest. *ACM Transactions on Algorithms (TALG)*, 11(3):1–20, 2015.

14 Deeparnab Chakrabarty, Alina Ene, Ravishankar Krishnaswamy, and Debmalya Panigrahi. Online buy-at-bulk network design. *SIAM Journal on Computing*, 47(4):1505–1528, 2018.

15 Moses Charikar, Chandra Chekuri, To-Yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation algorithms for directed steiner problems. *Journal of Algorithms*, 33(1):73–91, 1999.

16 Shuchi Chawla, Tim Roughgarden, and Mukund Sundararajan. Optimal cost-sharing mechanisms for steiner forest problems. In *International Workshop on Internet and Network Economics*, pages 112–123. Springer, 2006.

17 Shiri Chechik. Approximate distance oracles with improved bounds. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–10. ACM, 2015.

18 Chandra Chekuri, Guy Even, Anupam Gupta, and Danny Segev. Set connectivity problems in undirected graphs and the directed steiner network problem. *ACM Transactions on Algorithms (TALG)*, 7(2):1–17, 2011.

19 Eden Chlamtáč, Michael Dinitz, Guy Kortsarz, and Bundit Laekhanukit. Approximating spanners and directed steiner forest: Upper and lower bounds. *ACM Transactions on Algorithms (TALG)*, 16(3):1–31, 2020.

20 Lenore Cowen and Christopher G. Wagner. Compact roundtrip routing in directed networks. *Journal on Algorithms*, 50(1):79–95, 2004.

21 Michael Dinitz and Robert Krauthgamer. Directed spanners via flow-based linear programs. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 323–332, 2011.

22 Michael Dinitz and Robert Krauthgamer. Fault-tolerant spanners: better and simpler. In *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 169–178, 2011.

23 Michael Dinitz and Zeyu Zhang. Approximating low-stretch spanners. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 821–840. SIAM, 2016.

24 Dorit Dor, Shay Halperin, and Uri Zwick. All-pairs almost shortest paths. *SIAM Journal on Computing*, 29(5):1740–1759, 2000.

25 Michael Elkin. Computing almost shortest paths. *ACM Transactions on Algorithms (TALG)*, 1(2):283–323, 2005.

26 Michael Elkin and David Peleg. The client-server 2-spanner problem with applications to network design. In Francesc Comellas, Josep Fàbrega, and Pierre Fraigniaud, editors, *SIROCCO 8, Proceedings of the 8th International Colloquium on Structural Information and Communication Complexity, Vall de Núria, Girona-Barcelona, Catalonia, Spain, 27-29 June, 2001*, volume 8 of *Proceedings in Informatics*, pages 117–132. Carleton Scientific, 2001.

27 Michael Elkin and David Peleg. The hardness of approximating spanner problems. *Theory of Computing Systems*, 41(4):691–729, 2007.

28 Moran Feldman, Guy Kortsarz, and Zeev Nutov. Improved approximation algorithms for directed steiner forest. *Journal of Computer and System Sciences*, 78(1):279–292, 2012.

29 Arnold Filtser. Hop-constrained metric embeddings and their applications. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 492–503. IEEE, 2021.

30 Lisa Fleischer, Jochen Könemann, Stefano Leonardi, and Guido Schäfer. Simple cost sharing schemes for multicommodity rent-or-buy and stochastic steiner tree. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 663–670, 2006.

31 Fedor V Fomin, Petr A Golovach, William Lochet, Pranabendu Misra, Saket Saurabh, and Roohani Sharma. Parameterized complexity of directed spanner problems. *Algorithmica*, 84(8):2292–2308, 2022.

32 Elena Grigorescu, Nithish Kumar, and Young-San Lin. Approximation algorithms for directed weighted spanners, 2023. `arXiv:2307.02774`.

33 Elena Grigorescu, Young-San Lin, and Kent Quanrud. Online directed spanners and steiner forests. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

34 Anupam Gupta, Amit Kumar, Martin Pál, and Tim Roughgarden. Approximation via cost-sharing: a simple approximation algorithm for the multicommodity rent-or-buy problem. In *44th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2003. Proceedings.*, pages 606–615. IEEE, 2003.

35 Bernhard Haeupler, D Ellis Hershkowitz, and Goran Zuzic. Tree embeddings for hop-constrained network design. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 356–369, 2021.

36 Mohammad Taghi Hajiaghayi, Guy Kortsarz, and Mohammad R Salavatipour. Approximating buy-at-bulk and shallow-light k-steiner trees. *Algorithmica*, 53:89–103, 2009.

37 Refael Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations research*, 17(1):36–42, 1992.

38 Markó Horváth and Tamás Kis. Multi-criteria approximation schemes for the resource constrained shortest path problem. *Optimization Letters*, 12(3):475–483, 2018.

39 Samir Khuller, Balaji Raghavachari, and Neal Young. Approximating the minimum equivalent digraph. *SIAM Journal on Computing*, 24(4):859–872, 1995.

40 Vikram Khurana, Jian Peng, Chee Yeun Chung, Pavan K Auluck, Saranna Fanning, Daniel F Tardiff, Theresa Bartels, Martina Koeva, Stephen W Eichhorn, Hadar Benyamini, et al. Genome-scale networks link neurodegenerative disease genes to α-synuclein through specific molecular pathways. *Cell systems*, 4(2):157–170, 2017.

41 Shimon Kogan and Merav Parter. Having hope in hops: New spanners, preservers and lower bounds for hopsets. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 766–777. IEEE, 2022.

42 Jochen Könemann, Stefano Leonardi, Guido Schäfer, and Stefan van Zwam. From primal-dual to cost shares and back: a stronger lp relaxation for the steiner forest problem. In *International Colloquium on Automata, Languages, and Programming*, pages 930–942. Springer, 2005.

43 Jochen Könemann, Stefano Leonardi, Guido Schäfer, and Stefan HM van Zwam. A group-strategyproof cost sharing mechanism for the steiner forest game. *SIAM Journal on Computing*, 37(5):1319–1341, 2008.

44 Guy Kortsarz. On the hardness of approximating spanners. *Algorithmica*, 30:432–450, 2001.

45 Dean H Lorenz and Danny Raz. A simple efficient approximation scheme for the restricted shortest path problem. *Operations Research Letters*, 28(5):213–219, 2001.

46 Madhav V Marathe, Ravi Sundaram, SS Ravi, Daniel J Rosenkrantz, and Harry B Hunt III. Bicriteria network design problems. *Journal of algorithms*, 28(1):142–171, 1998.

47 Jakub Pachocki, Liam Roditty, Aaron Sidford, Roei Tov, and Virginia Vassilevska Williams. Approximating cycles in directed graphs: Fast algorithms for girth and roundtrip spanners. In Artur Czumaj, editor, *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 1374–1392. SIAM, 2018.

48 Mihai Patrascu and Liam Roditty. Distance oracles beyond the Thorup-Zwick bound. *SIAM Journal on Computing*, 43(1):300–311, 2014.

49 David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989. `doi:10.1002/jgt.3190130114`.

50 David Peleg and Jeffrey D. Ullman. An optimal synchronizer for the hypercube. *SIAM Journal on Computing*, 18(4):740–747, 1989.

51 Leila Pirhaji, Pamela Milani, Mathias Leidl, Timothy Curran, Julian Avila-Pacheco, Clary B Clish, Forest M White, Alan Saghatelian, and Ernest Fraenkel. Revealing disease-associated pathways by network integration of untargeted metabolomics. *Nature methods*, 13(9):770–776, 2016.

52 Liam Roditty, Mikkel Thorup, and Uri Zwick. Roundtrip spanners and roundtrip routing in directed graphs. *ACM Transactions on Algorithms (TALG)*, 4(3):29:1–29:17, 2008.

53 Tim Roughgarden and Mukund Sundararajan. Optimal efficiency guarantees for network design mechanisms. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 469–483. Springer, 2007.

54 Adrian Vetta. Approximating the minimum strongly connected subgraph via a matching lower bound. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 417–426, 2001.

55 Andrew Chi-Chih Yao. Space-time tradeoff for answering range queries (extended abstract). In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing (STOC)*, 1982.

56 Alexander Zelikovsky. A series of approximation algorithms for the acyclic directed steiner tree problem. *Algorithmica*, 18(1):99–110, 1997.

A Related work

A.1 Resource-constrained Shortest Path

In the resource-constrained shortest path problem [38] for directed networks, each edge is associated with r non-negative weights. Each type of weight $i \in [r-1]$ is associated with a budget. The r-th weight denotes the cost of the edge. The goal is to find a minimum-cost path that connects the single source to the single sink without violating the $r-1$ budgets. The results of [38] show that when r is a constant, there exists an FPTAS that finds a path with a cost at most the same as the feasible minimum-cost path by violating each budget by a factor of $1+\varepsilon$. When $r=2$, this problem is equivalent to the restricted shortest path problem [37, 45], which has been used extensively in the LP formulations for spanners and directed Steiner forests [10, 19, 21, 28, 33]. For our purpose, $r=3$ because the LP formulation implicitly considers whether there exists a feasible path between terminal pairs whose cost exceeds a given threshold.

A.2 Undirected Bi-criteria Network Design

A general class of undirected bi-criteria network problems was introduced by [46]. A more related problem to ours is the undirected Steiner tree problem. The goal is to connect a subset of vertices to a specified root vertex. In the bi-criteria problem, the distance from the root to a target vertex is required to be at most the given global threshold. [46] presented a bi-criteria algorithm for undirected Steiner trees that is $O(\log n)$-approximate and violates the distance constraints by a factor of $O(\log n)$. Following [46], [36] extends this result to a more general buy-at-bulk bi-criteria network design problem, where the objective is $\mathsf{polylog}(n)$-approximate and violates the distance constraints by a factor of $\mathsf{polylog}(n)$.

Recently, [29, 35] studied the tree embedding technique used for undirected network connectivity problems with *hop* constraints. For a positively weighted graph with a global parameter $h \in [n]$, the *hop distance* between vertices u and v is the minimum weight among the u-v-paths using at most h edges. Under the assumption that the ratio between the maximum edge weight and the minimum edge weight is $\mathsf{poly}(n)$, the tree embedding technique allows a $\mathsf{polylog}(n)$-approximation by relaxing the hop distance within a $\mathsf{polylog}(n)$ factor for a rich class of undirected network connectivity problems.

A.3 Other related directed network problems

The more related directed network problems are variants of spanners and Steiner problems, including directed Steiner trees [15, 56], directed Steiner network [18], fault-tolerance spanners [21, 22], and parameterized complexity analysis for directed s-spanners [31]. For a comprehensive account of the vast literature, we refer the reader to the excellent survey for spanners [2].

There is an extensive list of other related directed network problems, including distance preservers [12, 19], approximate distance preservers [41], reachability preservers [1], and buy-at-bulk network design [5]. One direction along this line of research is to study the extremal bounds for the optimal subgraph in terms of the input parameters, instead of comparing the costs of the approximate and optimal solution [1, 12, 41]. Another direction is to consider the online problem where terminal pairs arrive online and the goal is to irrevocably select edges so that the cost of the network is approximately minimized [3, 14, 33].

B Missing Proofs in Section 2

B.1 Proof for Lemma 18

▶ **Lemma 18.** *Let OPT be the optimal value of an instance of* Pairwise Weighted Spanner*. Then, the optimal value of LP* (2) *corresponding to that instance is at most OPT. In addition, a solution for LP* (2) *of value at most* $(1+\varepsilon)\cdot \mathsf{OPT}$ *can be found in polynomial time.*

Proof. We recall LP (2):

$$\begin{aligned}
\min \quad & \sum_{e \in E} c(e)\cdot x_e \\
\text{subject to} \quad & \sum_{(s,t)\in D} y_{s,t} \geq \frac{|D|}{2}, \\
& \sum_{\Pi(s,t)\ni p \ni e} f_p \leq x_e && \forall (s,t)\in D, e \in E, \\
& \sum_{p \in \Pi(s,t)} f_p = y_{s,t} && \forall (s,t) \in D, \\
& 0 \leq y_{s,t}, f_p, x_e \leq 1 && \forall (s,t)\in D, p \in \Pi, e \in E.
\end{aligned} \tag{2}$$

Let us consider the dual of LP (2).

$$\begin{aligned}
\max \quad & \sum_{e\in E} x_e + \sum_{(s,t)\in D} y_{s,t} - W \cdot \frac{|D|}{2} && && \text{(4a)}\\
\text{subject to} \quad & \sum_{(s,t)\in D} z_{(s,t),e} + c(e) \le x_e && \forall e \in E, && \text{(4b)}\\
& y_{s,t} + w_{s,t} \ge W && \forall (s,t) \in D, && \text{(4c)}\\
& w_{s,t} \le \sum_{e\in p} z_{(s,t),e} && \forall (s,t) \in D, p \in \Pi(s,t), && \text{(4d)}\\
& W, x_e, y_{s,t}, z_{(s,t),e} \ge 0 && \forall (s,t) \in D, e \in E. && \text{(4e)}
\end{aligned}$$

Our dual is slightly different from the dual in [10, 28]. Constraints in (4b), (4c), and (4e) are identical, but constraints in (4d) are slightly different because the set of paths Π we consider are different from the set of paths considered in [28]. As in [28], we could find violating constraints for (4b), (4c), and (4e) in polynomial time. The only constraints that require care are the constraints in (4d), which may be exponentially many.

When we consider a single (s,t) pair, [28] pointed out that their variant of constraints in (4d) are equivalent to Restricted Shortest Path which is NP-hard, see [37, 45]. [28] then uses an FPTAS [37, 45] for Restricted Shortest Path as an approximate separation oracle for those constraints. But we need a different separation oracle because the set of paths Π allowed in our LP have two restrictions (as opposed to [28] which has only one) in addition to an objective. We now define the Resource-constrained Shortest Path problem that is presented in [38].

▶ **Definition 30** (Resource-constrained Shortest Path (k-RCSP)).

Instance: *A directed graph $G = (V, E)$, with edge costs $c : E \to \mathbb{Q}_{\ge 0}$, and a pair (s,t). For each edge $e \in E$, we have a vector $r_e = (r_{1,e}, r_{2,e}, \ldots, r_{k,e})$ of size k where each $r_{i,e} \in \mathbb{Q}_{\ge 0}$ $\forall i \in [k]$.*

Objective: *Find a minimum cost $s \rightsquigarrow t$ path P such that $\sum_{e\in P} r_{i,e} \le R_i$, $\forall i \in [k]$.*

▷ Claim 31. 2-RCSP acts as a separation oracle for those constraints in equation (4d) that correspond to a specific $(s,t) \in D$.

Proof. We can use one of the resource constraints in 2-RCSP for ensuring that the distance constraints for (s,t) are satisfied and use the other resource constraint to ensure that $w_{s,t} > \sum_{e\in P} z_{(s,t),e}$. In other words, we use one resource to model the edge lengths and another to model the dual variable $z_{\{s,t\},e}$. We can now try to find a minimum cost $s \rightsquigarrow t$ path in this instance of 2-RCSP where costs for 2-RCSP are equivalent to the costs in our instance of Pairwise Weighted Spanner. If the minimum cost obtained when we meet these constraints is less than L, then we have a violating constraint and if not we do not have one. ◁

The Resource-constrained shortest path problem is NP-hard [38]. So, we instead get a separation oracle for an approximate variant of LP (4). Now, given resource constraints $R_1, R_2, \ldots, R_k$ for the Resource-constrained shortest path problem, let OPT_{RCSP} be the cost of the minimum cost $s \rightsquigarrow t$ path that satisfies the resource constraints. An $(1; 1+\varepsilon, \ldots, 1+\varepsilon)$-approximation scheme finds an $s \rightsquigarrow t$ path whose cost is at most OPT_{RCSP}, but the resource constraints are satisfied up to a factor of $1+\varepsilon$ for that path.

▶ **Lemma 32** (RCSP – [38]). *If k is a constant then there exists a fully polynomial time $(1; 1+\varepsilon, \dots, 1+\varepsilon)$-approximation scheme for the k-RCSP that runs in time polynomial in input size and $1/\varepsilon$.*

We have to be careful in our usage of Lemma 32. The FPTAS for the Restricted Shortest Path problem from [45] cleanly serves the requirements of [28] as it is a $(1+\varepsilon; 1)$ approximation and it can strictly satisfy the constraints. [28] then uses these constraints to ensure that the weight requirements are strictly met. But the FPTAS given by [38] does not strictly satisfy the constraints since we need both the length and weight constraints to be satisfied strictly.

We overcome this obstacle by re-purposing the objective to handle the edge weights and by carefully ensuring that any error in the path length caused by using the $1+\varepsilon$ approximation from [38] does not make us use an incorrect path. To ensure that we do not select an incorrect path, it is sufficient to ensure that the potential error from [38] is less than any error that is possible in our given input graph. Since the edge lengths are positive integers, observe that for any two $s \rightsquigarrow t$ paths with different lengths, the length difference is at least one. In addition, the path lengths are at most $n \cdot \max_{e\in E}\{\ell(e)\}$. Since all edge lengths are integral and of magnitude $\mathsf{poly}(n)$, it is sufficient to have $\varepsilon \le 1/(n \cdot \max_{e\in E}\{\ell(e)\})$. Thus, we can fix ε such that $1/\varepsilon = O(n \cdot \mathsf{poly}(n))$ to ensure that the running time will remain polynomial in input size and strictly satisfy the distance constraints.

Now, we take an approximate version of LP (4) which is the following LP

$$\begin{aligned}
\max \quad & \sum_{e\in E} x_e + \sum_{s,t} y_{s,t} - W \cdot \frac{|D|}{2} && \\
\text{subject to} \quad & \sum_{s,t} z_{(s,t),e} + c(e) \le x_e && \forall e \in E, \\
& y_{s,t} + w_{s,t} \ge W && \forall (s,t) \in D, \\
& (1+\varepsilon) \cdot w_{s,t} \le \sum_{e\in p} z_{(s,t),e} && \forall (s,t) \in D, p \in \Pi(s,t), \\
& W, x_e, y_{s,t}, z_{(s,t),e} \ge 0 && \forall (s,t) \in D, e \in E.
\end{aligned} \tag{5}$$

We can exactly solve LP (5) using [38] and thus we can also exactly solve the dual of LP (5) which would be:

$$\begin{aligned}
\min \quad & \sum_{e\in E} c(e) \cdot x_e && \\
\text{subject to} \quad & \sum_{(s,t)\in D} y_{s,t} \ge \frac{|D|}{2}, && \\
& \sum_{\Pi(s,t)\ni P\ni e} f_p \le x_e \cdot (1+\varepsilon) && \forall (s,t) \in D, e \in E, \\
& \sum_{P\in\Pi(s,t)} f_p = y_{s,t} && \forall (s,t) \in D, \\
& 0 \le y_{s,t}, f_p, x_e \le 1 && \forall (s,t) \in D, p \in \Pi, e \in E.
\end{aligned} \tag{6}$$

Let $\mathsf{OPT}(\varepsilon)$ and OPT be the optimal values to (6) and (2) respectively. Observe that $\mathsf{OPT}(\varepsilon) \le \mathsf{OPT}$ because the constraints in LP (6) are slacker than the constraints in (2) and both these LPs are minimization LPs. Also note that if $\hat{x}(\varepsilon)$ is a feasible solution to (6), then by replacing the value of every variable x_e in $\hat{x}(\varepsilon)$ by $\min(1, x_e \cdot (1+\varepsilon))$, we get a new solution $\hat{x}$ which is a feasible solution to (2). The value of the optimal solution then is at most $(1+\varepsilon) \cdot \mathsf{OPT}(\varepsilon) \le (1+\varepsilon) \cdot \mathsf{OPT}$. ◀

B.2 Missing proof for Lemma 29

▶ **Lemma 29.** *There exists a distance-preserving weighted junction tree J with density at most $\mathsf{OPT}/\sqrt{k}$.*

Proof. Let G^* (a subgraph of G) be the optimal pairwise weighted spanner solution with cost OPT. The proof proceeds by considering the following two cases: 1) there exists a vertex $r \in V$ that belongs to at least $\sqrt{k}$ $s_i \rightsquigarrow t_i$ paths of distance at most $Dist(s_i, t_i)$ in G^* for distinct i, and 2) there is no such vertex $r \in V$.

For the first case, we consider the union of the $s_i \rightsquigarrow t_i$ paths in G^*, each of distance at most $Dist(s_i, t_i)$, that passes through r. This subgraph in G^* contains an in-arborescence and an out-arborescence both rooted at r, whose union forms a distance-preserving weighted junction tree. This distance-preserving weighted junction tree has cost at most OPT and connects at least $\sqrt{k}$ terminal pairs, so its density is at most $\mathsf{OPT}/\sqrt{k}$.

For the second case, each vertex $r \in V$ appears in at most $\sqrt{k}$ $s_i \rightsquigarrow t_i$ paths in G^*. More specifically, each edge $e \in E$ also appears in at most $\sqrt{k}$ $s_i \rightsquigarrow t_i$ paths in G'. By creating $\sqrt{k}$ copies of each edge, all terminal pairs can be connected by edge-disjoint paths. Since the overall duplicate cost is at most $\sqrt{k} \cdot \mathsf{OPT}$, at least one of these paths has cost at most $\sqrt{k} \cdot \mathsf{OPT}/k$. This path constitutes a distance-preserving weighted junction tree whose density is at most $\mathsf{OPT}/\sqrt{k}$. ◀

C Conclusion

In this paper, we presented algorithms for a variant of directed spanners that could also handle costs on edges, in addition to the more standard setting of edge lengths. The proof strategy for Theorem 2 follows a high-level structure that is similar to other results for directed Steiner forests, but involves significant obstacles in each part of the proof due to the addition of distance constraints. We overcome these obstacles by using the proper approaches. For example, the Resource-constrained Shortest Path problem from [38] is carefully adapted for our specifics. We also needed to carefully adapt many other parts of the proof, such as the analysis of our junction-tree approximation and our rounding algorithm for the LPs, to fit the addition of distance constraints.

We also present online algorithms for Online Pairwise Weighted Spanner and Online Single-source Weighted Spanner. We use our result for Online Single-source Weighted Spanner to solve a special case of Pairwise Weighted Spanner, namely, All-pair Weighted Distance Preserver, and obtain a significantly better approximation for that case.

We propose the following directions for future work:

- Is it possible to get a better analysis for the rounding algorithm for Theorem 2 as in [10]? This should improve the overall approximation factor for Pairwise Weighted Spanner in Theorem 2.
- Is there a hardness bound for Pairwise Weighted Spanner that is greater than the existing hardness bounds for Steiner forests and unit-cost spanners?
- Is there a better approximation factor for all-pair weighted spanners, i.e., an instance of Pairwise Weighted Spanner where $D = V \times V$?
- Can we get a result for pairwise weighted distance preserver that is better than using the Pairwise Weighted Spanner results in Theorem 2?

Approximation Algorithms and Lower Bounds for Graph Burning

Matej Lieskovský
Faculty of Mathematics and Physics, Computer Science Institute of Charles University, Prague, Czech Republic

Jiří Sgall
Faculty of Mathematics and Physics, Computer Science Institute of Charles University, Prague, Czech Republic

Andreas Emil Feldmann
Department of Computer Science, University of Sheffield, UK

Abstract

Graph Burning models information spreading in a given graph as a process such that in each step one node is infected (informed) and also the infection spreads to all neighbors of previously infected nodes. Formally, given a graph $G = (V, E)$, possibly with edge lengths, the burning number $b(G)$ is the minimum number g such that there exist nodes $v_0, \ldots, v_{g-1} \in V$ satisfying the property that for each $u \in V$ there exists $i \in \{0, \ldots, g-1\}$ so that the distance between u and v_i is at most i.

We present a randomized 2.314-approximation algorithm for computing the burning number of a general graph, even with arbitrary edge lengths. We complement this by an approximation lower bound of 2 for the case of equal length edges, and a lower bound of 4/3 for the case when edges are restricted to have length 1.

This improves on the previous 3-approximation algorithm and an APX-hardness result.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Approximation algorithms analysis

Keywords and phrases Graph Algorithms, approximation Algorithms, randomized Algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.9

Category APPROX

Funding *Matej Lieskovský*: Partially supported by GAUK project 234723, and GA ČR project 19-27871X.
Jiří Sgall: Partially supported by GA ČR project 19-27871X.

Acknowledgements We are grateful to anonymous referees for detailed reviews and helpful comments.

1 Introduction

Graph Burning was introduced by Bonato et al. [4] as a model of information spreading and, more specifically, as a model of contagion or influence in social networks. The idea is that "infecting" several nodes in the network and spreading information from them may reach the whole network very fast. Formally, it is defined as the following process, where *burned* nodes in a graph represent the infected part of the network.

At time $t = 0$, no node of the graph is burned. At time $t = 1$, we choose a node and burn it. At each time step $t > 1$, all neighbors of already burned nodes are also burned, and we may choose another node to burn. The process stops when all nodes are burned. We call the sequence of chosen nodes a burning schedule of the graph. The burning number $b(G)$ of a graph G is the minimum number of steps needed for all nodes of G to be burned.

For a burning schedule of length ℓ, the node selected at time t ensures that all nodes within distance $\ell - t$ from it will be burned. We can thus reformulate the Graph Burning problem as looking for the minimal g such that all nodes of the graph can be covered by some placement of g balls with unique radii from $g - 1$ to 0. Placing a ball with radius r at node v thus ensures the coverage of all nodes within the ball and represents selecting node v to be set on fire at time $g - r$ in the original formulation of the problem. This formulation also works naturally in the presence of arbitrary edge lengths.

The ball formulation emphasizes the relation of Graph Burning to the k-Center problem, where we try to cover the graph by k balls of equal radius, with the objective of minimizing the radius. In the non-uniform variant of the k-Center problem [7], different balls (centers) can have different radii, and in particular, the variant with a constant number of different radii is well-studied [11, 10]. Graph Burning can be viewed as an extreme version of the Non-Uniform k-Center problem, where neither the number of centers nor the number of distinct radii are fixed. This setting is challenging both for the design of efficient algorithms and for proving lower bounds. Thus studying Graph Burning can help to develop new techniques and insights that may be useful even for variants of k-Center and/or other facility location problems.

1.1 State of the art

Adapting a well-known greedy 2-approximation algorithm for the k-Center problem, Bonato and Kamali [5] gave a 3-approximation algorithm for Graph Burning. In fact, the approximation ratio of the algorithm is $3 - 2/b(G)$, which is a slight improvement if $b(G)$ is small; see also Appendix 3 for a review of this algorithm. Another $(3 - 2/b(G))$-approximation algorithm for arbitrary graphs was reported in [9]. In [5] it was stated as an open problem to find an improved algorithm, but no $(3 - \varepsilon)$-approximation algorithm for general graphs was known before our work.

In previous work approximation algorithms and approximation schemes were developed for trees [5] and other special graph classes [6], we refer to survey [3] for further references. This stream of research is now subsumed by [15], where it is shown that for graphs of small treewidth, a PTAS exists. The parameterized complexity of Graph Burning was studied in [12, 13, 14].

On the hardness side, it is known that computing the burning number of a graph is NP-complete even on simple classes of graphs such as trees or even disjoint unions of paths with unit edge lengths [2]. Answering an open question from [5], Mondal et al. [17, 18] have shown that Graph Burning is APX-hard, using a complex reduction from vertex cover in cubic graphs, but did not give an explicit lower bound on the approximation ratio.

1.2 Our results

In this paper, we present a randomized $(\alpha + \varepsilon)$-approximation algorithm for the Graph Burning problem on graphs with arbitrary edge lengths, where $\alpha = 2e^2/(e^2 - 1) < 2.314$ and $\varepsilon > 0$ is arbitrarily small. This is the first improvement of the previous easy 3-approximation algorithm based on k-Center results.

It has been speculated in [5] that an improved approximation algorithm will need to use a better lower bound on the burning number, presumably combinatorial. We do not use this line of attack. Instead, we use the power of randomization and analyze the following procedure: we greedily process the yet uncovered nodes and, on each such node, we put a center with radius chosen uniformly from a range $0, \dots, R$ for a carefully chosen R. This process is not easy to analyze, as the choices of nodes to be covered depend on the previous random choices. Nevertheless, we are able to show that it has a martingale property and use this to prove that the process succeeds with high probability.

On the hardness side, we give simple and explicit lower bounds on the approximation ratio for Graph Burning. We first prove a lower bound of 2 on the approximation ratio for graphs with edge lengths, which is close to our upper bound. This is an important insight since the methods we use in our algorithms but also other current approaches work even in the presence of arbitrary edge lengths, so they are all subject to this lower bound. We note that the lower bound works even if all the edge lengths are equal. We also prove a lower bound of 4/3 for graphs where all edges have length 1.

In [5] the authors note that known hardness results for related problems such as the k-Center or Dominating Set problems do not apply to Graph Burning. Indeed, they cannot be applied directly, as in approximations for Graph Burning we implicitly use both more centers and larger radii of the balls. Lower bounds for this kind of bicriteria approximations are naturally harder to show, which explains the slow progress on the inapproximability side.

We use a somewhat indirect reduction from the Dominating Set problem. For edges of length 1 the reduction is more complicated, as we need to subdivide the edges of the Dominating Set instance. As a consequence, when converting the dominating set to a Graph Burning solution, we have to use larger radii to cover the new vertices, which decreases the inapproximability bound. When other edge lengths are allowed, the subdivided edges can be replaced by long edges, which leads to the improved lower bound of 2. This is close to the performance of our algorithm.

Closing the remaining gap is an interesting open problem. We conjecture that the optimal approximation ratio is 2 at least for the version with arbitrary edge lengths.

1.3 Paper overview

In Section 2 we introduce some notations and conventions. In Section 3, we review the 3-approximation algorithm in our framework, which amounts to using the same radius for all centers. In Sections 4 and 5 we gradually introduce the building blocks of our algorithm. We examine the possibility of using a constant number of different radii, which leads to a deterministic algorithm for a logarithmic number of centers (Section 4). Then we examine the possibility of randomly choosing among two radii in Section 5. This leads to an approximation ratio below 3 and allows us to gently introduce the necessary probabilistic machinery, including the martingale approach. In Section 6 we apply all the ingredients and present the final 2.314-approximation algorithm. Finally, in Section 7 we present our lower bounds.

2 Preliminaries

We start with a formal definition of Graph Burning.

▶ **Definition 2.1.** *An instance of* Graph Burning *is an undirected graph $G = (V, E)$ with non-negative edge lengths. The* distance $d(u, v)$ *is defined as the length of the shortest path from u to v. We let $n = |V|$.*

A center *is a pair $c = (v, r)$ with $v \in V$ and r a non-negative integer; v denotes the node where the center is placed and r denotes the radius. Given a center c we denote its radius by $r(c)$. A solution of a Graph Burning instance is a set of centers $\{(v_0, 0), (v_1, 1), \dots, (v_{g-1}, g-1)\}$. It is feasible if for any node $w \in V$ there exists $i < g$ such that $d(w, v_i) \leq i$. The objective of the Graph Burning problem is to minimize g over all feasible solutions. The minimum g is called the* burning number *of G and is denoted $b(G)$.*

Conventions for the algorithms. When running our algorithms, we will be building up a set of centers ALG. We allow the set ALG not to use all the radii between 0 and the maximal radius. At the end of the algorithm, if needed, balls with the remaining radii can be placed arbitrarily completing a feasible solution without changing the maximal radius.

The set $\{(v_0, 0), \ldots, (v_{n-1}, n-1)\}$ is a *trivial* valid solution for any ordering of $V = \{v_0, \ldots, v_{n-1}\}$. Thus $b(G) \leq n$ even with arbitrary edge lengths. We assume that $V \neq \emptyset$ and thus $b(G) \geq 1$. For most of our algorithms, we will assume that the algorithm is given an integer g, presumably the burning number, and its goal is to find a feasible solution of size at most σg for some approximation ratio σ, or possibly fail if g is smaller than $b(G)$. We then try all n possible values of g in our final algorithm.

Notations for the optimum. In the analysis of our algorithms, we fix some optimal solution OPT using centers with unique radii from 0 to $b(G) - 1$. Furthermore, we assign each node $w \in V$ to a unique center $c_{\text{OPT}}(w) = (v, r) \in \text{OPT}$ that is covering it (i.e., $d(w, v) \leq r$), thus creating a partition of the set of nodes of the graph. For every $c_{\text{OPT}} \in \text{OPT}$ we denote the set of nodes assigned to it as $V(c_{\text{OPT}})$.

Good centers. For any center c_{OPT} in the optimal solution, if the computed solution ALG contains a center $c_{\text{ALG}} = (v, r)$ where $r \geq 2r(c_{\text{OPT}})$ and $v \in V(c_{\text{OPT}})$, then all nodes in $V(c_{\text{OPT}})$ are covered by c_{ALG}. If such a c_{ALG} exists, we say that c_{OPT} is made *good* by c_{ALG}.

▶ **Definition 2.2.** *A center $c_{\text{OPT}} \in \text{OPT}$ is* good *if there exists a $c_{\text{ALG}} = (v, r) \in \text{ALG}$ with $r \geq 2r(c_{\text{OPT}})$ and $v \in V(c_{\text{OPT}})$.*

Note that c_{OPT} being good is not a necessary condition for all of $V(c_{\text{OPT}})$ to be covered – a well-placed center of radius $r < 2r(c_{\text{OPT}})$, multiple smaller centers, and/or centers located outside of $V(c_{\text{OPT}})$ might cover all of $V(c_{\text{OPT}})$.

We design our algorithms so that the (expected) number of good centers increases in each step and rely on the fact that a solution ALG is feasible whenever it makes all centers of the optimum good. While it may happen that not all centers of OPT become good, eventually all nodes end up covered as the number of good centers is increasing and it cannot increase beyond the number of centers in the optimal solution.

Classes of centers. Most of our algorithms divide the radii at their disposal using a series of thresholds. The thresholds are given as a list $q_1, q_2, q_3, \ldots$ of increasing numbers (not necessarily integers). For a center $c_{\text{OPT}} \in \text{OPT}$, we define its *class* as the smallest i such that using q_i for $v \in V(c_{\text{OPT}})$ ensures that c_{OPT} becomes good:

▶ **Definition 2.3.** *For a center $c_{\text{OPT}} \in \text{OPT}$ we refer to the minimal i such that the threshold q_i is larger than or equal to $2r(c_{\text{OPT}})$ as the* class *of c_{OPT}.*

Procedure for selecting a radius. Whenever our algorithms need to use a radius to add a center to ALG, they call the procedure $\textsc{Select}(q)$, which takes a number q (typically a threshold) as its input. It then returns the minimal integer radius that has not yet been used by the algorithm and is at least q.

This procedure guarantees that our algorithms always use centers of distinct radii, at the cost of possibly increasing the largest radius and thus the size of the solution. In the analysis, we then need to prove that the maximal radius used is (with high probability) small enough.

Notations. We use $\exp(x)$ to denote e^x. We also use the convention that $x/yz = x/(yz)$.

3 Deterministic 3-approximation algorithm and overall approach

We revisit the 3-approximation algorithm from [4]. See Algorithm 1 for our formalization; recall that we assume that the algorithm is given g. We use a single threshold $q_1 = 2(g-1)$. This means that all centers c_{OPT} are class 1, as the optimum uses radii up to $g-1$, while our algorithm will use radii $2g-2$, $2g-1$ and so on.

Algorithm 1 The deterministic 3-approximation algorithm.

```
ALG ← ∅; $q_1 = 2g - 2$
while an uncovered node exists do
    Select an arbitrary uncovered node $v$
    $r = \textsc{Select}(q_1)$
    Add center $(v, r)$ to ALG and update the set of uncovered nodes
return ALG
```

Observe that whenever we select an uncovered node v, it is covered by some class 1 center $c_{\text{OPT}}(v)$ that is not good. Placing a center c_{ALG} with radius at least q_1 at v is guaranteed to make $c_{\text{OPT}}(v)$ good. Thus, once we place g centers c_{ALG} with radii at least q_1, all centers of OPT will be good and the algorithm finishes. Note that all nodes might be covered without all centers of OPT being good, making the algorithm finish sooner. Either way, we are guaranteed to never select a radius greater than $q_1 + g - 1 = 3g - 3$, making this a 3-approximation algorithm. More precisely, the algorithm uses $3g-2$ centers, as opposed to optimal $b(G) \leq g$. So the approximation ratio becomes $3 - 2/b(G)$, as mentioned in the introduction.

4 Deterministic algorithm for small burning number

Let us now explore an approach that uses z thresholds equally spaced between 0 and $2(g-1)$, for some constant z. This means, disregarding rounding, that the optimum centers are divided into z classes, where the class i has g/z centers of radii between $(i-1)g/z$ and ig/z. The algorithm will aim to cover the centers of class i using thresholds approximately $2ig/z$, each for g/z nodes. As a result the largest radius can be bounded by $2g + g/z$.

Similar to before, this algorithm will choose an uncovered node in each iteration as a center for the approximate solution. This would work easily if we knew in advance the class of the center c_{OPT} covering the chosen node, so that we could use the corresponding threshold in the algorithm. Instead, we try all the possible sequences of thresholds, using each threshold for at most $\lceil g/z \rceil$ nodes, matching the number of optimal centers of each class. The analysis is somewhat subtle, as the node to be processed depends on the previous choices of the algorithm, so the sequence of thresholds used in the optimum needs to be constructed based on the nodes chosen by the algorithm.

Formalizing this idea with some attention to rounding and also the fact that the optimum can use fewer than g centers, we obtain an algorithm that finds a Graph Burning solution with objective at most $(2 + 1/z)g$ in time $\mathcal{O}\left(z^g \text{poly}(n)\right)$ for any given integer z and $g \geq b(G)$; see Algorithm 2.

▶ **Theorem 4.1.** *If $g \geq b(G)$ then Algorithm 2 runs in $\mathcal{O}\left(z^g \text{poly}(n)\right)$-time and achieves an approximation ratio of $(2 + 1/z)$.*

For the full proof of Theorem 4.1 see Appendix A. (For $z \geq g$, the proof actually gives ratio $2 - 1/g$. We omit that in the statement, as in the relevant case z is small.)

Algorithm 2 Deterministic algorithm for small g.

```
if $z > g$ then $z \leftarrow g$
Order $V$ arbitrarily
$q_i \leftarrow 2gi/z - 2$ for $i \in \{1, \dots, z\}$
Let $\mathbb{S}$ be a set of all possible sequences $S = s_0, s_1, \dots, s_{g-1}$ with $s_t \in \{1, \dots, z\}$
  such that each value is used at most $\lceil g/z \rceil$ times
for $S \in \mathbb{S}$ do
    ALG $\leftarrow \emptyset$
    for $t \leftarrow 0, 1, \dots, g-1$ do
        Let $v$ be the first uncovered node; $r \leftarrow$ SELECT$(q_{s_t})$
        Add center $(v, r)$ to ALG and update the set of uncovered nodes
        if all nodes are covered then return ALG
return Fail
```

The running time is polynomial if $g \in \mathcal{O}(\log n)$ and $z \in \mathcal{O}(1)$, thus we get a deterministic $(2 + 1/z)$-approximation algorithm in this case. We will later use this result in our final algorithm with $z = 4$, ensuring an approximation ratio of $2 + 1/4 < 2.314$ for small $b(G)$.

5 Randomized approach with two classes of centers

As a warmup, in this section we introduce the main ideas of our techniques by considering a simpler setup. In particular, let us only use two thresholds $g - 1$ and $2(g - 1)$, and choose a radius randomly among the two values of the threshold; see Algorithm 3.

Algorithm 3 The randomized algorithm with two classes.

```
ALG $\leftarrow \emptyset$; $q_i \leftarrow i(g-1)$ for $i \in \{1, 2\}$
while an uncovered node exists do
    Select an arbitrary uncovered node $v$
    Let $k_v \in \{1, 2\}$ be chosen uniformly at random, independently from other choices
    $r \leftarrow$ SELECT$(q_{k_v})$
    Add center $(v, r)$ to ALG and update the set of uncovered nodes
return ALG
```

Note that, as the radii start with 0, the optimum has $\lceil g/2 \rceil$ class 1 centers and $\lfloor g/2 \rfloor$ class 2 centers. Observe that a class 1 center c_{OPT} becomes good whenever we select an uncovered node from $V(c_{\text{OPT}})$. A class 2 center c_{OPT} becomes good if we select an uncovered node from $V(c_{\text{OPT}})$ and then randomly decide to use threshold q_2. If we were to make all centers of OPT good, we would use in expectation one radius per class 1 center and two radii per class 2 center, evenly distributed between threshold q_1 and threshold q_2. Since each class of centers of OPT contains roughly $0.5g$ centers, we expect at most $1.5g$ total radii being needed. Of these, we expect half, i.e., $0.75g$, to be at least the threshold $q_2 \leq 2g$. This indicates that roughly a 2.75-approximation algorithm should be the result.

Using a detailed analysis it is indeed possible to prove that Algorithm 3 is a $(2.75 + \varepsilon)$-approximation algorithm, improving on the previously known 3-approximation. For simplicity and to introduce the key ideas for our main algorithm, we will however only prove that this is a 2.9-approximation algorithm for g divisible by 10 (to avoid rounding issues), and then move on to give our 2.314-approximation in the following section. We show that the following holds with high probability:

- less than $1.6g$ centers c_{ALG} are placed by Algorithm 3, and
- less than $0.9g$ centers c_{ALG} with radius at least q_2 are placed, conditioned on the previous event (i.e., that the algorithm uses less than $1.6g$ centers).

5.1 Total number of centers

We will use martingales to analyze the algorithm; see the textbooks [1, 16] for reference. For convenience, we use a slight modification of the one-sided Azuma inequality for supermartingales. This follows easily, as any supermartingale can be converted to a martingale by shifting (increasing) the random variables appropriately.

▶ **Definition 5.1.** *A sequence of random variables $X_0, X_1, X_2, \ldots$ is a supermartingale if and only if both $|\mathbb{E}[X_t]| < \infty$ and $\mathbb{E}[X_{t+1} \mid X_0, \ldots, X_t] \le X_t$ for all t.*

▶ **Theorem 5.2** (Azuma Inequality). *If $X_0, X_1, X_2, \ldots$ is a supermartingale with $|X_t - X_{t-1}| \le \delta$ for all t, the following inequality holds:*

$$\Pr[X_t - X_0 \ge \Delta] \le \exp\left(\frac{-\Delta^2}{2\delta^2 t}\right).$$

We will track the progress of our algorithm by defining a potential describing the difference between the algorithm's true and expected behaviour. Formally, let C_t be a random variable denoting the set of centers of OPT made good by the first t centers c_{ALG} placed by Algorithm 3. Let us define $f(c_{\text{OPT}})$ as the class of c_{OPT}. Finally, let the potential be

$$\Phi_t = t - \sum_{c_{\text{OPT}} \in C_t} f(c_{\text{OPT}}).$$

The value $f(c_{\text{OPT}})$ is chosen so that a center c_{OPT} is expected to become good after the algorithm selects $f(c_{\text{OPT}})$ centers from $V(c_{\text{OPT}})$. Thus, at a given moment, the sum in the potential denotes the number of centers the algorithm was expected to select to make all $c_{\text{OPT}} \in C_t$ good. If t, the actual number of centers of the algorithm, is smaller than the sum, the potential is negative and the algorithm is progressing better than expected. If t is smaller than the sum, the potential is positive and the algorithm is progressing worse than expected. Our goal is to show that the potential is unlikely to be positive and large.

The random choice of a threshold influences whether the current $c_{\text{OPT}}(v)$ enters C_t. Besides that, the random choices also possibly change the set of covered nodes, influencing the later choices of the algorithm. Thus the changes of the potential in different steps are not independent (even if the choices of thresholds are) and we need to analyze the process as a martingale.

Observe that $\Phi_0, \Phi_1, \ldots$ is a sequence of random variables with $\Phi_0 = 0$. When we select node v as the $(t+1)$-st node to cover, only $c_{\text{OPT}} = c_{\text{OPT}}(v)$ can become good. If center c_{OPT} is class 1, then $f(c_{\text{OPT}}) = 1$ and we make it good with probability 1, which guarantees $\Phi_{t+1} = \Phi_t$. If center c_{OPT} is class 2, $f(c_{\text{OPT}}) = 2$. We make it good with probability at least $1/2$ and in this case $\Phi_{t+1} = \Phi_t - 1$. Otherwise $\Phi_{t+1} = \Phi_t + 1$. Thus $\mathbb{E}[\Phi_{t+1} \mid \Phi_0, \ldots, \Phi_t] \le \Phi_t$ and $\Phi_0, \Phi_1, \ldots$ is a supermartingale with $\Phi_0 = 0$ and $\delta = 1$.

We have $\sum_{c_{\text{OPT}} \in \text{OPT}} f(c_{\text{OPT}}) \le 1.5g$, corresponding to the fact that the expected number of radii needed for the algorithm is $1.5g$. Thus, if the algorithm does not stop before placing $1.6g$ centers, we have $\Phi_{1.6g} = 1.6g - \sum_{c_{\text{OPT}} \in C_{1.6g}} f(c_{\text{OPT}}) \ge 0.1g$. We use Azuma's inequality to show that the probability of this happening is small, keeping in mind that $\Phi_0 = 0$ and $\delta = 1$:

$$\Pr[\Phi_{1.6g} \ge 0.1g] \le \exp\left(\frac{-0.01g^2}{3.2g}\right) = \exp\left(\frac{-g}{320}\right).$$

5.2 Number of large centers

To show that less than $0.9g$ centers c_{ALG} with radius at least q_2 are placed among the first $1.6g$ centers chosen by the algorithm, we use the standard Chernoff bound, see the textbooks [1, 16].

▶ **Theorem 5.3** (Chernoff bound). *Suppose $X_1, \dots, X_J \in \{0,1\}$ are independent random variables. Let $X = \sum X_j$ and $\mu = \mathbb{E}[X] = \sum \mathbb{E}[X_j]$. Then for any $\varepsilon \in (0,1]$,*

$$\Pr[X \geq (1+\varepsilon)\mu] \leq \exp\left(\frac{-\varepsilon^2\mu}{3}\right).$$

Suppose that the algorithm chooses J centers for some $J \leq 1.6g$. Let X_j, $j = 1, \dots, J$, be a 0-1 indicator variable equal to 1 if we use a threshold q_2 when choosing the j-th center in the algorithm. Since every X_j is equal to 1 with probability 1/2 we have $\mu = \mathbb{E}[\sum X_j] = J/2 \leq 0.8g$. By Chernoff's bound, the probability of using threshold 2 more than $0.9g$ times is at most

$$\Pr[X \geq (1+\tfrac{1}{8})\mu] \leq \exp\left(\frac{-(\frac{1}{8})^2\mu}{3}\right) \leq \exp\left(\frac{-g}{320}\right).$$

5.3 Final bound

▶ **Theorem 5.4.** *The probability that Algorithm 3 uses a radius larger than or equal to $2.9g$ is at most $2\exp(-g/320)$.*

Proof. Using the previous two estimates, the probability that the algorithm uses more than $1.6g$ centers in total or more than $0.9g$ times threshold q_2 is bounded by $2\exp(-g/320)$.

If none of these events happen, we distinguish two cases. Either the algorithm uses all the radii between q_1 and q_2; then we use the fact that the total number of centers is at most $1.6g$ and thus the largest radius is at most $q_1 + 1.6g < 2.6g$. Otherwise, radii q_2 and larger are used only when threshold q_2 is selected, which happens at most $0.9g$ times and thus the largest radius is less than $q_2 + 0.9g < 2.9g$. ◀

6 Main randomized algorithm

The performance of Algorithm 3 can be further improved by using more thresholds. Ultimately, this leads to using a range of integers as thresholds. In particular, for a given g we use all integers up to approximately αg where α is the desired approximation ratio given by the following definition.

▶ **Definition 6.1.** *We set $\alpha = 2e^2/(e^2-1) < 2.314$.*

Let us present and analyze Algorithm 4 which is the key part of the final algorithm. For any $\varepsilon > 0$, we shall show that its approximation ratio is at most $(1+\varepsilon)^4\alpha$. We assume that $\varepsilon < 1$ (as otherwise we could use the known 3-approximation), $\alpha g < |V|$ (as otherwise a trivial solution is an α-approximation), $g \geq b(G)$ (as we will later iterate over all g) and $g \geq 100/\varepsilon^3$ (as we will use Algorithm 2 otherwise).

6.1 Expected number of centers

Next we analyze the expected number of centers that the algorithm uses. This is the key part that determines the approximation ratio and our choice of α. In particular, we set α so that the expected number of centers used is αg. On one hand, the largest radius is

Algorithm 4 The randomized algorithm for large g.

```
$\textsc{alg} \leftarrow \emptyset$; $R \leftarrow (1+\varepsilon)^2\alpha g$
while an uncovered node exists do
    Select an arbitrary uncovered node $v$
    Let $k_v \in \{0, \ldots, \lfloor R \rfloor\}$, be chosen uniformly and independently at random
    $r \leftarrow \textsc{Select}(k_v)$
    Add center $(v, r)$ to $\textsc{alg}$ and update the set of uncovered nodes
return $\textsc{alg}$
```

necessarily at least the number of centers used, on the other hand, once the range of radii exceeds the expected number of centers by a small factor, we will be able to show that with high probability the largest radius used is not much larger than the expected number of centers. This eventually gives an approximation factor arbitrarily close to the chosen α.

Similar to Section 5, the value $f(c_{\textsc{opt}})$ is chosen so that a center $c_{\textsc{opt}}$ is expected to become good after the algorithm selects $f(c_{\textsc{opt}})$ centers from $V(c_{\textsc{opt}})$.

▶ **Definition 6.2.** *For any center $c_{\textsc{opt}}$ of the optimum, define*

$$f(c_{\textsc{opt}}) = \frac{\alpha g}{\alpha g - 2r(c_{\textsc{opt}})} = \frac{1}{1 - \frac{2r(c_{\textsc{opt}})}{\alpha g}} .$$

▶ **Lemma 6.3.** *For any given center $c_{\textsc{opt}}$, whenever Algorithm 4 selects a node in $V(c_{\textsc{opt}})$, the probability that $c_{\textsc{opt}}$ is made good is at least $1/f(c_{\textsc{opt}})$. The expected number of nodes in $V(c_{\textsc{opt}})$ selected by Algorithm 4 is bounded by $\mathbb{E}\left[|\textsc{alg} \cap V(c_{\textsc{opt}})|\right] \leq f(c_{\textsc{opt}})$.*

Proof. Every time the algorithm selects a node in $V(c_{\textsc{opt}})$, the number of possible values for uniformly random k_v is $1 + \lfloor R \rfloor \geq R \geq \alpha g$. Thus the algorithm selects a radius $r' \geq 2r(c_{\textsc{opt}})$ with probability at least $p = 1 - 2r(c_{\textsc{opt}})/\alpha g = 1/f(c_{\textsc{opt}})$ and this makes $c_{\textsc{opt}}$ good. Once $c_{\textsc{opt}}$ is made good, no more nodes in $V(c_{\textsc{opt}})$ are selected. Since the choices of k_v are independent, we get $\mathbb{E}\left[|\textsc{alg} \cap V(c_{\textsc{opt}})|\right] \leq 1/p = f(c_{\textsc{opt}})$. ◀

▶ **Lemma 6.4.** *The expected total number of radii used by Algorithm 4 is at most αg.*

Proof. Each node of G is in $V(c_{\textsc{opt}})$ for exactly one $c_{\textsc{opt}}$. Thus, by Lemma 6.3 and linearity of expectation,

$$\mathbb{E}\left[|\textsc{alg}|\right] = \mathbb{E}\left[\sum_{c_{\textsc{opt}} \in \textsc{opt}} |\textsc{alg} \cap V(c_{\textsc{opt}})|\right] = \sum_{c_{\textsc{opt}} \in \textsc{opt}} \mathbb{E}\left[|\textsc{alg} \cap V(c_{\textsc{opt}})|\right] \leq \sum_{c_{\textsc{opt}} \in \textsc{opt}} f(c_{\textsc{opt}}) .$$

We have, using $\alpha = 2e^2/(e^2-1)$ in the penultimate step,

$$\sum_{c_{\textsc{opt}} \in \textsc{opt}} f(c_{\textsc{opt}}) = \sum_{r=0}^{g-1} \frac{\alpha g}{\alpha g - 2r} < \int_0^g \frac{\alpha g}{\alpha g - 2r} dr = \left[-\frac{\alpha g}{2} \ln(\alpha g - 2r)\right]_0^g$$
$$= \frac{\alpha g}{2} \ln\left(\frac{\alpha g}{\alpha g - 2g}\right) = \frac{\alpha g}{2} \ln\left(\frac{\alpha}{\alpha - 2}\right) = \frac{\alpha g}{2} \ln\left(e^2\right) = \alpha g . \quad ◀$$

6.2 High-probability bound

Next, we need to prove that there is a low probability that Algorithm 4 will use significantly more radii than expected. Similar to Section 5 we define a potential Φ_t comparing the number of the algorithm's centers to the expected progress and use martingale bounds to show that the potential is unlikely to be large.

▶ **Definition 6.5.** *Let C_t be a random variable denoting the set of centers of* OPT *made good by the first t centers c_{ALG} placed by the algorithm and*

$$\Phi_t = t - \sum_{c_{\text{OPT}} \in C_t} f(c_{\text{OPT}})\,.$$

Observe that $\Phi_0 = 0$ and that $\Phi_0, \Phi_1, \ldots$ is a sequence of random variables, since C_t depends on the random choices of the values k_v in Algorithm 4. We now prove that the potential is a supermartingale and thus we can use Azuma's inequality to show that Algorithm 4 is unlikely to use more than $(1+\varepsilon)\alpha g$ radii.

▶ **Lemma 6.6.** *$\Phi_0, \Phi_1, \ldots$ is a supermartingale with $|\Phi_t - \Phi_{t-1}| \le 7$ for all t.*

Proof. Let v be the t-th selected node. Since v is not covered when it is selected, we have $c_{\text{OPT}}(v) \notin C_{t-1}$. Only $c_{\text{OPT}}(v)$ can be made good by a center c_{ALG} placed at v and thus Φ_t is equal to either $\Phi_{t-1} + 1 - f(c_{\text{OPT}}(v))$ or $\Phi_{t-1} + 1$. By Lemma 6.3, the probability p of center $c_{\text{OPT}}(v)$ being made good is at least $1/f(c_{\text{OPT}}(v))$, making $\mathbb{E}[\Phi_t | \Phi_0, \ldots, \Phi_{t-1}] = p(\Phi_{t-1} + 1 - f(c_{\text{OPT}}(v)) + (1-p)(\Phi_{t-1}+1) = \Phi_{t-1} + 1 - pf(c_{\text{OPT}}(v)) \le \Phi_{t-1}$ and thus $\Phi_0, \Phi_1, \ldots$ is a supermartingale.

As $\alpha = 2e^2/(e^2-1) > 16/7$ and $r(c_{\text{OPT}}) < g$, we get $f(c_{\text{OPT}}) < \alpha g/(\alpha g - 2g) < 8$. Thus $\Phi_t - \Phi_{t-1}$ is at least $1 - f(c_{\text{OPT}}) \ge -7$ and at most 1, meaning that $|\Phi_t - \Phi_{t-1}| \le 7$ for all t. ◀

▶ **Lemma 6.7.** *The probability that Algorithm 4 uses at least $(1+\varepsilon)\alpha g$ radii is at most $1/e$.*

Proof. We know that $\Phi_0 = 0$. By Lemma 6.4 we know that $\sum f(c_{\text{OPT}}) \le \alpha g$. Thus, if the process has not finished before using $T = \lceil (1+\varepsilon)\alpha g \rceil$ radii, the value of Φ_T is at least $\varepsilon \alpha g$. By Lemma 6.6 we know that $\Phi_0, \Phi_1, \ldots$ is a supermartingale with differences at most 7. From Azuma's inequality (Theorem 5.2) for $X_t = \Phi_t$, $\Delta = \varepsilon\alpha g$, $T = \lceil (1+\varepsilon)\alpha g \rceil$ and $\delta = 7$, using also the bounds $g \ge 100/\varepsilon^3$ and $\varepsilon < 1$ we get

$$\Pr[\Phi_T \ge \varepsilon \alpha g] \le \exp\left(\frac{-(\varepsilon\alpha g)^2}{2 \cdot 7^2 \cdot \lceil (1+\varepsilon)\alpha g \rceil}\right) < \exp\left(\frac{-\varepsilon^2 g}{100}\right) < \frac{1}{e}\,.$$ ◀

6.3 Bounding the maximum radius

The only possibility for Algorithm 4 to use a radius larger than R is due to the procedure Select in which case the larger radii are used one by one in increasing order. In particular, to use a radius larger than $R + b$, the algorithm must, for some x, generate at least $b + x$ values k_v that are at least $R - x$ and thus cause Select to pick a radius larger than R at least b times. Using Chernoff bounds, we now show that this event is unlikely for $b = \varepsilon R$.

▶ **Lemma 6.8.** *Conditioned on the event that Algorithm 4 uses no more than $(1+\varepsilon)\alpha g$ radii, Algorithm 4 uses a radius of value at least $(1+\varepsilon)^3\alpha g$ with probability at most $1/100$.*

Proof. Recall that $R = (1+\varepsilon)^2\alpha g$ and let $b = \varepsilon R = (1+\varepsilon)^2\varepsilon\alpha g$. Therefore, $(1+\varepsilon)^3\alpha g = (1+\varepsilon)R = R + b$. We proceed to show that the probability of Algorithm 4 using radius $\lceil R + b \rceil$ is small.

Let r_{min} be a random variable denoting the minimal radius such that all radii from $\{r_{min}, \ldots, \lceil R+b \rceil\}$ are used by Algorithm 4; if $\lceil R + b \rceil$ is not used, r_{min} is undefined. Algorithm 4 uses a radius at least $R + b$ if and only if r_{min} is defined.

Informally, the proof idea is as follows. We partition this event according to the value of r_{min}, grouping its possible values into blocks of size b (disregarding rounding) so that block i contains values in $(R - ib, R - (i-1)b]$ for $i \in \{1, \dots, \lceil 1/\varepsilon \rceil\}$. If r_{min} is in block 1, i.e., larger than $R - b$, it must be the case that k_v was chosen larger than $R - b$ at least b times. In general, for block i, k_v must have been chosen to be larger than $R - ib$ at least ib times. This probability decreases geometrically with i; this follows from Chernoff bounds, using also the conditioning in the lemma, which implies that the number of chosen radii is smaller than R by a constant factor $(1 + \varepsilon)$.

For a full version of the proof, please see Appendix B. ◀

6.4 Final algorithm

We combine the preceding algorithms to get the final Algorithm 5.

Algorithm 5 The final algorithm.

```
Input: Graph $G(V, E)$ and real number $\bar{\varepsilon} > 0$
Output: A burning schedule for $G$ of length at most $(\alpha + \bar{\varepsilon})b(G)$
$\varepsilon \leftarrow (1 + \bar{\varepsilon}/\alpha)^{1/4} - 1$; $\ell \leftarrow 100/\varepsilon^3$          ▷ Note that $\varepsilon \in \Theta(\bar{\varepsilon})$
for $g \leftarrow 1, \dots, \lfloor |V|/\alpha \rfloor$ do
    if $g \leq \ell$ then
        Use Algorithm 2 with $z = 4$ to get ALG or Fail
        if ALG was found then return ALG
    else
        Use Algorithm 4 to get ALG
        if ALG uses no radius greater than $(1 + \varepsilon)^3 \alpha g - 1$ then return ALG
return a trivial solution $\{(v_0, 0), \dots, (v_{n-1}, n-1)\}$ for $V = \{v_0, \dots, v_{n-1}\}$
```

▶ **Theorem 6.9.** *Algorithm 5 is an $(\alpha + \bar{\varepsilon})$-approximation algorithm for any given $\bar{\varepsilon} > 0$ and runs in time $2^{\mathcal{O}(1/\bar{\varepsilon}^3)}\text{poly}(n)$.*

Proof. By Theorem 4.1, Algorithm 2 is guaranteed to succeed if run with $g \geq b(G)$ and it never uses a radius greater than $2.25g - 1$. Thus, if $b(G) \leq \ell$, Algorithm 5 always finds a 2.25-approximation.

If $b(G) > \ell$, either Algorithm 2 succeeds with $g < b(G)$, or we run Algorithm 4 while increasing g. By Lemma 6.7, once $g \geq b(G)$, Algorithm 4 uses more than $(1 + \varepsilon)\alpha g$ radii with probability at most $1/e$. By Lemma 6.8, if Algorithm 4 uses at most $(1 + \varepsilon)\alpha g$ radii, it uses a radius greater than $(1 + \varepsilon)^3 \alpha g - 1$ with probability at most $1/100$. Thus it succeeds with probability at least $1/2$. This probability is independent and increasing for each successive iteration, ensuring that the expected number of failures is at most 2. Thus the expected maximum radius is at most $(1 + \varepsilon)^3 \alpha (b(G) + 2) - 1$. We get $(1 + \varepsilon)^3 \alpha (b(G) + 2) = (1+\varepsilon)^3 \alpha b(G) + 2(1+\varepsilon)^3 \alpha \leq (1+\varepsilon)^3 \alpha g + 50 \leq (1+\varepsilon)^4 \alpha b(G) - 1$, using the value of α for the first inequality and for the last one also the case condition $b(G) > \ell$ and $\ell \geq 100/\varepsilon \geq 51/\varepsilon(1+\varepsilon)^3 \alpha$. By the choice of ε in the algorithm, the approximation ratio is at most $(1 + \varepsilon)^4 \alpha = \alpha + \bar{\varepsilon}$.

Since Algorithm 5 iterating over the possible values of g multiplies the time complexity by at most n, the running time is dominated by Algorithm 2, and thus it is bounded by $2^{\mathcal{O}(\ell)}\text{poly}(n)$. Our choice of ε satisfies $\varepsilon = \Theta(\bar{\varepsilon})$ for a small $\bar{\varepsilon} > 0$ (using the fact that $(1 + \varepsilon)^4 \approx 1 + 4\varepsilon$), thus we get $\ell = 100/\varepsilon^3 = \mathcal{O}\left(1/\bar{\varepsilon}^3\right)$ and the bound in the theorem follows. ◀

We note that, instead of using Algorithm 2 for $g \leq \ell$, we can find an optimal solution by exhaustive search in time $n^{\mathcal{O}(g)}$, which is still polynomial. We prefer to use Algorithm 2 to achieve FPT-type running time dependency on ε.

7 Lower bounds for Graph Burning

In this section, we first prove a lower bound of 2 for the approximation ratio of the Graph Burning problem on a graph with edge lengths. We also prove a lower bound of 4/3 for graphs with unit edge lengths.

Both results rely on a reduction from the Dominating Set problem defined as follows.

▶ **Definition 7.1.** *A* dominating set *of an undirected graph $G = (V, E)$ is a set $D \subseteq V$ such that each vertex of G either is in D or has a neighbor in D. In an instance of the Dominating Set problem, we are given an undirected graph and our task is to find a dominating set of the lowest possible size.*

The hardness of approximation of Dominating Set is equivalent to that of Set Cover. After a long line of research in PCP theory, the ultimate result is the following one.

▶ **Theorem 7.2** (Dinur, Steurer [8])**.** *For any $\varepsilon > 0$, there exists no $(1-\varepsilon)\ln n$-approximation algorithm for Dominating Set or Set Cover problems unless $P = NP$.*

In the first reduction, we modify a Dominating Set instance by setting the length of all edges to k for some integer k. We use the fact that in the burning schedule, the centers with radii in $[k, 2k)$ cover exactly their neighbors (due to the edge lengths). Thus a dominating set of size t implies a burning schedule of length $t + k$ in the graph with edge lengths, and a graph burning schedule with length at most $2k$ implies a dominating set of size at most $2k$. By considering a carefully chosen k, this together with a $(2 - \varepsilon)$-approximation algorithm for Graph Burning gives an $\mathcal{O}(1)$-approximation algorithm for Dominating Set, contradicting Theorem 7.2.

Technically, we try all values of k and stop at the smallest k that yields a Graph Burning solution of size at most $2k$ and thus a Dominating Set approximation. The number of possible values of k is bounded by $|V|$. Testing only the chosen important values of k used in the proof (i.e., approximately $k = 2t/\varepsilon$ for testing size t Dominating Set) would be sufficient and slightly more efficient, but we prefer to keep the algorithm simple.

▶ **Theorem 7.3.** *For any constant $\varepsilon > 0$, there exists no $(2 - \varepsilon)$-approximation algorithm for the Graph Burning problem with arbitrary edge lengths unless $P = NP$. This holds even for the case when restricted to Graph Burning instances with all edge lengths equal.*

Proof. For a contradiction, assume that we have a $(2-\varepsilon)$-approximation algorithm for Graph Burning with arbitrary but equal edge lengths. Let $\sigma = \lceil 1/\varepsilon \rceil$. We give a reduction that results in an approximation algorithm for the Dominating Set problem with an approximation ratio of at most $4\sigma \in \mathcal{O}(1)$.

We use the reduction described above; see Algorithm 6 for a formal description.

First, we claim that the output D is always a dominating set. Indeed, if we stop at a feasible solution with $g \leq 2k$, any $w \in G$ is covered by some center (v_i, i) for $i < 2k$, i.e., the distance between w and v_i is less than $2k$. Since the lengths of the edges are set to k, this means that v_i is a neighbor of or equal to w. In the fallback case, $D = V$ is a dominating set.

Now assume that G has a dominating set $\{w_0, \ldots, w_{t-1}\}$ of size t. We claim that for any given k, the burning number of G with edge lengths is at most $k + t$. Indeed, the set of centers $\{(w_i, k + i) \mid i = 0, \ldots, t - 1\}$ augmented by arbitrary centers with radii smaller than k is a required feasible solution.

Algorithm 6 Reduction using edge lengths.

```
Input: Dominating Set instance $G = (V, E)$
for $k \leftarrow 1, \dots, \lfloor |V|/2 \rfloor$ do
    Set the length of each edge to $k$
    Run the $(2 - \varepsilon)$-approximation algorithm for Graph Burning on $G$,
        obtaining a feasible solution $\{(v_0, 0), (v_1, 1), \dots, (v_{g-1}, g - 1)\}$ for some $g$.
    if $g \le 2k$ then return the set $D = \{v_0, v_1, \dots, v_{g-1}\}$.
return the set $D = V$.
```

If $t < |V|/4\sigma$, consider $k = 2\sigma t \le |V|/2$. The choice of k implies that $\varepsilon k \ge 2t$ as $\sigma = \lceil 1/\varepsilon \rceil$. The assumed $(2 - \varepsilon)$-approximation algorithm for Graph Burning finds a feasible solution of size at most $(2 - \varepsilon)(k + t)$, which is bounded by $(2 - \varepsilon)(k + t) < 2k - \varepsilon k + 2t \le 2k$. Thus the algorithm stops for some $k' \le k$ with a dominating set D of size at most $2k = 4\sigma t$.

If $t \ge |V|/4\sigma$, the reduction trivially gives a 4σ-approximation. Overall we have obtained a 4σ-approximation for Dominating Set.

By Theorem 7.2, no $\mathcal{O}(1)$-approximation algorithm for the Dominating Set problem exists, unless $P = NP$. This yields the contradiction and the theorem. ◀

We can modify the previous reduction so that, instead of edges of length k, we subdivide them into paths of unit-length edges. As a consequence, when converting the dominating set to a Graph Burning solution, we have to use larger radii to cover the new vertices, which degrades the inapproximability bound. Also, when converting a burning schedule to a dominating set, we cannot use the subdivision points as elements of the dominating set. Instead, we substitute any subdivision point with the two closest original vertices, i.e., the endpoints of the subdivided edge.

▶ **Theorem 7.4.** *For any constant $\varepsilon > 0$, there exists no $(4/3 - \varepsilon)$-approximation algorithm for the Graph Burning problem with unit edge lengths, unless $P = NP$.*

Proof. For a contradiction, assume that we have a $(4/3 - \varepsilon)$-approximation algorithm for Graph Burning with unit-length edges. Let $\sigma = \lceil 1/\varepsilon \rceil$. We give a reduction that results in an algorithm for Dominating Set with an approximation ratio of at most $8\sigma \in \mathcal{O}(1)$. The reduction follows the outline given above; see Algorithm 7 for a formal description.

Graph G' is of polynomial size, and thus the whole reduction is polynomial. Furthermore, if $u, w \in V$ are vertices of the input graph G, then the distance between u and w in G' is a multiple of $2k$. In particular, if u and w are not equal, their distance in G' is equal to $2k$ if u and w are neighbors in G, and their distance is at least $4k$ otherwise.

We claim that the output D is always a dominating set. Indeed, if we stop at a feasible solution with $g \le 4k$, any $w \in V$ is covered by some center (v_i, i) for $i < 4k$, i.e., the distance between w and v_i is less than $4k$. The construction of D_i implies that the shortest path between w and v_i contains one of $u \in D_i$. Thus the distance between u and w in G' is less than $4k$ and u and w are neighbors in G. It follows that D is a dominating set. In the fallback case, $D = V$ is a dominating set as well.

Now assume that G has a dominating set $\{w_0, \dots, w_{t-1}\}$ of size t. We claim that for any given k, the burning number of G' is at most $3k + t$. Indeed, the set of centers $\{(w_i, 3k + i) \mid i = 0, \dots, t - 1\}$ augmented by $3k$ arbitrary centers with radii smaller than $3k$ is a required feasible solution. Each vertex v in G' has distance at most k to the closest vertex in G, which in turn has distance at most $2k$ to a vertex in the dominating set. Thus the balls of radius at least $3k$ at the vertices of the dominating set cover the whole G'.

Algorithm 7 Reduction for unit edge lengths.

```
Input: Dominating Set instance $G = (V, E)$
for $k = 1, \dots, |V|$ do
    Create $G'$ from $G$ by replacing each edge of $G$ by a path of $2k$ new edges
        with $2k - 1$ new internal vertices.
    Run the $(4/3 - \varepsilon)$-approximation algorithm for Graph Burning on $G'$,
        obtaining a feasible solution $\{(v_0, 0), (v_1, 1), \dots, (v_{g-1}, g - 1)\}$ for some $g$.
    if $g \leq 4k$ then
        for $i = 0, 1, \dots, g - 1$ do
            if $v_i \in V$ then                ▷ i.e., $v_i$ is a vertex of the input graph $G$
                $D_i \leftarrow \{v_i\}$
            else                ▷ i.e., $v_i$ is a new internal vertex on an added path
                $D_i \leftarrow \{u, u'\}$ where $u, u' \in V$ are such that $v_i$ is a new vertex
                    on the path in $G'$ that replaced the edge $uu'$ of $G$.
        return the set $D = D_0 \cup D_1 \cup \dots \cup D_{g-1}$.
return the set $D = V$.
```

If $t < |V|/\sigma$, consider $k = \sigma t \leq |V|$. The choice of k implies that $t \leq \varepsilon k$ as $\sigma = \lceil 1/\varepsilon \rceil$. The assumed $(4/3 - \varepsilon)$-approximation algorithm for Graph Burning finds a feasible solution of size at most $(4/3 - \varepsilon)(3k + t)$. We have

$$\left(\frac{4}{3} - \varepsilon\right)(3k + t) \leq 4k - 3\varepsilon k + \frac{4}{3}t \leq 4k - 3t + \frac{4}{3}t < 4k\,.$$

Thus the algorithm stops for some $k' \leq k$ with a dominating set D of size at most $2 \cdot 4k = 8\sigma t$, since each set D_i contains at most two vertices for each center v_i of the burning schedule.

If $t \geq |V|/\sigma$, the reduction trivially gives an σ-approximation. Altogether we have obtained an 8σ-approximation algorithm for Dominating Set.

By Theorem 7.2, no $\mathcal{O}(1)$-approximation algorithm for the dominating set problem exists, unless $P = NP$. This yields the contradiction and the theorem. ◀

References

1 Noga Alon and Joel H. Spencer. *The Probabilistic Method, Third Edition*. Wiley, 2008.

2 Stéphane Bessy, Anthony Bonato, Jeannette Janssen, Dieter Rautenbach, and Elham Roshanbin. Burning a graph is hard. *Discrete Applied Mathematics*, 232:73–87, 2017. doi:10.1016/j.dam.2017.07.016.

3 Anthony Bonato. A survey of graph burning. *Contributions Discret. Math.*, 16(1):185–197, 2021. URL: https://cdm.ucalgary.ca/article/view/71194.

4 Anthony Bonato, Jeannette Janssen, and Elham Roshanbin. How to burn a graph. *Internet Mathematics*, 12, July 2015. doi:10.1080/15427951.2015.1103339.

5 Anthony Bonato and Shahin Kamali. Approximation algorithms for graph burning. In T. V. Gopal and Junzo Watada, editors, *Theory and Applications of Models of Computation - 15th Annual Conference, TAMC 2019, Kitakyushu, Japan, April 13-16, 2019, Proceedings*, volume 11436 of *Lecture Notes in Computer Science*, pages 74–92. Springer, 2019. doi:10.1007/978-3-030-14812-6_6.

6 Anthony Bonato and Thomas Lidbetter. Bounds on the burning numbers of spiders and path-forests. *Theor. Comput. Sci.*, 794:12–19, 2019. doi:10.1016/j.tcs.2018.05.035.

7 Deeparnab Chakrabarty, Prachi Goyal, and Ravishankar Krishnaswamy. The non-uniform k-center problem. *ACM Trans. Algorithms*, 16(4), June 2020. doi:10.1145/3392720.

8 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.

9 Jesús García-Díaz, Julio César Pérez-Sansalvador, Lil María Xibai Rodríguez-Henríquez, and José Alejandro Cornejo-Acosta. Burning graphs through farthest-first traversal. *IEEE Access*, 10:30395–30404, 2022. doi:10.1109/ACCESS.2022.3159695.

10 Tanmay Inamdar and Kasturi R. Varadarajan. Non-uniform k-center and greedy clustering. In Artur Czumaj and Qin Xin, editors, *18th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT*, volume 227 of *LIPIcs*, pages 28:1–28:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SWAT.2022.28.

11 Xinrui Jia, Lars Rohwedder, Kshiteej Sheth, and Ola Svensson. Towards non-uniform k-center with constant types of radii. In Karl Bringmann and Timothy Chan, editors, *5th Symposium on Simplicity in Algorithms, SOSA@SODA 2022*, pages 228–237. SIAM, 2022. doi:10.1137/1.9781611977066.16.

12 Shahin Kamali, Avery Miller, and Kenny Zhang. Burning two worlds. In Alexander Chatzigeorgiou, Riccardo Dondi, Herodotos Herodotou, Christos A. Kapoutsis, Yannis Manolopoulos, George A. Papadopoulos, and Florian Sikora, editors, *SOFSEM 2020: Theory and Practice of Computer Science - 46th International Conference on Current Trends in Theory and Practice of Informatics, SOFSEM 2020, Limassol, Cyprus, January 20-24, 2020, Proceedings*, volume 12011 of *Lecture Notes in Computer Science*, pages 113–124. Springer, 2020. doi:10.1007/978-3-030-38919-2_10.

13 Anjeneya Swami Kare and I. Vinod Reddy. Parameterized algorithms for graph burning problem. In Charles J. Colbourn, Roberto Grossi, and Nadia Pisanti, editors, *Combinatorial Algorithms - 30th International Workshop, IWOCA 2019, Pisa, Italy, July 23-25, 2019, Proceedings*, volume 11638 of *Lecture Notes in Computer Science*, pages 304–314. Springer, 2019. doi:10.1007/978-3-030-25005-8_25.

14 Yasuaki Kobayashi and Yota Otachi. Parameterized complexity of graph burning. *Algorithmica*, 84(8):2379–2393, 2022. doi:10.1007/s00453-022-00962-8.

15 Matej Lieskovský and Jirí Sgall. Graph burning and non-uniform k-centers for small treewidth. In *Approximation and Online Algorithms - 20th International Workshop, WAOA 2022, Potsdam, Germany, September 8-9, 2022, Proceedings*, volume 13538 of *Lecture Notes in Computer Science*, pages 20–35. Springer, 2022. doi:10.1007/978-3-031-18367-6_2.

16 Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005. doi:10.1017/CBO9780511813603.

17 Debajyoti Mondal, N. Parthiban, V. Kavitha, and Indra Rajasingh. APX-hardness and approximation for the k-burning number problem. In Ryuhei Uehara, Seok-Hee Hong, and Subhas C. Nandy, editors, *WALCOM: Algorithms and Computation - 15th International Conference and Workshops, WALCOM 2021, Yangon, Myanmar, February 28 - March 2, 2021, Proceedings*, volume 12635 of *Lecture Notes in Computer Science*, pages 272–283. Springer, 2021. doi:10.1007/978-3-030-68211-8_22.

18 Debajyoti Mondal, Angelin Jemima Rajasingh, N. Parthiban, and Indra Rajasingh. APX-hardness and approximation for the k-burning number problem. *Theor. Comput. Sci.*, 932:21–30, 2022. doi:10.1016/j.tcs.2022.08.001.

A Proof of Theorem 4.1

We first prove the following lemma.

▶ **Lemma A.1.** *If $g \geq b(G)$ then Algorithm 2 always returns a solution.*

Proof. Let $v(S, t)$ be the t-th selected node while the algorithm is using sequence S. Observe that $v(S, t)$ only depends on the initial segment $s_1, \ldots, s_{t-1}$ of S. With this in mind, we show in the next paragraph that there exists $S^* \in \mathbb{S}$ such that s^*_t of S^* is the class of $c_{\text{OPT}}(v(S^*, t))$.

Formally, for $t = 1, 2, \ldots$ we define s_t^* as the class of $c_{\text{OPT}}(v(\bar{S}, t))$ for an arbitrary extension $\bar{S} \in \mathbb{S}$ of $s_1^*, \ldots, s_{t-1}^*$. Such an $\bar{S} \in \mathbb{S}$ exists, as the optimum solution OPT contains at most g centers with no more than $\lceil g/z \rceil$ centers of any given class. If $v(\bar{S}, t)$ is undefined (i.e., all nodes are covered at this point), we choose $s_t^* = \bar{s}_t$. As $v(\bar{S}, t)$ and thus s_t^* depends only on $s_1^*, \ldots, s_{t-1}^*$, which is the initial segment of $\bar{S}$, eventually S^* is the sequence of the classes of centers $c_{\text{OPT}}(v(S^*, t))$.

It follows that using this S^* in the loop makes all centers good, thus all nodes are covered and the algorithm stops with a valid solution. ◀

▶ **Theorem A.2** (Theorem 4.1). *If $g \geq b(G)$ then Algorithm 2 runs in $\mathcal{O}(z^g \text{poly}(n))$-time and achieves an approximation ratio of $(2 + 1/z)$.*

Proof. Lemma A.1 implies that the algorithm returns a solution covering all nodes. We first bound the maximal radius used and thus the approximation ratio. We distinguish two cases.

If $z \geq g$, the algorithm uses $z = g$. Each threshold is used at most once, making the maximal used radius at most $q_z = 2g - 2$, giving a solution with $2g - 1$ many centers, and thus approximation ratio of $2 - 1/g$.

If $z < g$ there are at least $\lfloor 2g/z \rfloor$ radii between any two consecutive thresholds as there are at least $\lfloor 2g/z \rfloor$ integers in $[q_i, q_{i+1})$. We use the same threshold at most $\lceil g/z \rceil$ times by the definition of $\mathbb{S}$. We have $\lceil g/z \rceil \leq \lfloor 2g/z \rfloor$, as $\lceil x \rceil \leq \lfloor 2x \rfloor$ for any $x \geq 1$, while we use the same threshold for at most $\lceil g/z \rceil$ nodes by the definition of $\mathbb{S}$. Thus the radii produced by using a given threshold q_i do not exceed the following threshold q_{i+1}. Furthermore no more than $\lceil g/z \rceil$ nodes use the last threshold $q_z = 2g - 2$, making the maximal used radius at most $2g - 2 + \lceil g/z \rceil \leq 2g + g/z - 1$. Thus the objective value is at most $2g + g/z$ and the ratio $2 + 1/z$.

The runtime bound follows since the number of sequences of length g with numbers from $\{1, \ldots, z\}$ is z^g, and this gives an upper bound on the size of $\mathbb{S}$. ◀

B Proof of Lemma 6.8

▶ **Lemma B.1** (Lemma 6.8). *Conditioned on the event that Algorithm 4 uses no more than $(1+\varepsilon)\alpha g$ radii, Algorithm 4 uses a radius of value at least $(1+\varepsilon)^3\alpha g$ with probability at most $1/100$.*

Proof. Recall that $R = (1+\varepsilon)^2\alpha g$ and let $b = \varepsilon R = (1+\varepsilon)^2\varepsilon\alpha g$. Therefore, $(1+\varepsilon)^3\alpha g = (1+\varepsilon)R = R + b$. We proceed to show that the probability of Algorithm 4 using radius $\lceil R + b \rceil$ is small.

Let r_{min} be a random variable denoting the minimal radius such that all radii from $\{r_{min}, \ldots, \lceil R + b \rceil\}$ are used by Algorithm 4; if $\lceil R + b \rceil$ is not used, r_{min} is undefined. Algorithm 4 uses a radius at least $R + b$ if and only if r_{min} is defined.

Informally, the proof idea is as follows. We partition this event according to the value of r_{min}, grouping its possible values into blocks of size b (disregarding rounding) so that block i contains values in $(R - ib, R - (i-1)b]$ for $i \in \{1, \ldots, \lceil 1/\varepsilon \rceil\}$. If r_{min} is in block 1, i.e., larger than $R - b$, it must be the case that k_v was chosen larger than $R - b$ at least b times. In general, for block i, k_v must have been chosen to be larger than $R - ib$ at least ib times. This probability decreases geometrically with i; this follows from Chernoff bounds, using also the conditioning in the lemma, which implies that the number of chosen radii is smaller than R by a constant factor $(1 + \varepsilon)$.

For $i \in \{1, \ldots, \lceil 1/\varepsilon \rceil\}$, we define A_i as the event that r_{min} is defined and i is the smallest integer such that $R - ib < r_{min}$. Exactly one of the events A_i occurs whenever r_{min} is defined. The event A_i implies that the algorithm randomly generated k_v greater than $R - ib$

at least ib times, as by definition of r_{min}, choosing $k_v < r_{min}$ does not allow SELECT to generate a radius equal to or larger than r_{min} and at the same time all at least ib radii between $r_{min} \leq R - (i-1)b$ and $\lceil R + b \rceil$ are used.

By the conditioning in the lemma, we assume that the algorithm uses no more than $(1+\varepsilon)\alpha g$ radii. Let p be the probability that upon processing v, the algorithm generates k_v larger than $R - ib$. We have $p \leq ib/R$, as out of $\lfloor R \rfloor + 1 > R$ options for k_v, there are at most ib values larger than $R - ib$. Consider X which is a sum of $J = \lfloor (1+\varepsilon)\alpha g \rfloor$ independent indicator random variables, each equal to 1 with probability ib/R. We have $Pr[A_i] \leq Pr[X \geq ib]$, as X overestimates the number of k_v larger than $R - ib$ in two ways: First, we possibly increase the number of trials and second, we increase the probability of the indicator variable to be 1 from p to ib/R.

Let $\mu = \mathbb{E}[X]$. We have $\mu = \lfloor (1+\varepsilon)\alpha g \rfloor \cdot ib/R \leq (1+\varepsilon)\alpha g ib/R = ib/(1+\varepsilon)$. As $g \geq 100/\varepsilon^3 \geq 100$, the rounding error is small, namely we have $\mu \geq \frac{99}{100}(1+\varepsilon)\alpha g \cdot ib/R = \frac{99}{100}\varepsilon(1+\varepsilon)i\alpha g$, using also $b = \varepsilon R$. Now we use Chernoff bound and get

$$\begin{aligned}\Pr[A_i] &\leq \Pr[X \geq ib] \leq \Pr[X \geq (1+\varepsilon)\mu] \leq \exp\left(\frac{-\varepsilon^2\mu}{3}\right) \leq \exp\left(\frac{-\frac{99}{100}\varepsilon^3(1+\varepsilon)i\alpha g}{3}\right) \\ &\leq \exp\left(-\frac{3}{4}\varepsilon^3 ig\right) \leq \exp(-75i)\ ,\end{aligned}$$

where we used $\frac{99}{100}(1+\varepsilon)\alpha/3 > 3/4$. We obtain that the probability of using radius at least $R + b$ is at most

$$\Pr\left[\bigcup_i A_i\right] \leq \sum_{i=1}^{\infty} \exp(-75i) \leq 1/100\,. \qquad \blacktriangleleft$$

The (Im)possibility of Simple Search-To-Decision Reductions for Approximation Problems

Alexander Golovnev ✉
Georgetown University, Washington, D. C., USA

Siyao Guo ✉
Shanghai Frontiers Science Center of Artificial Intelligence and Deep Learning,
NYU Shanghai, China

Spencer Peters ✉
Cornell University, Ithaca, NY, USA

Noah Stephens-Davidowitz ✉
Cornell University, Ithaca, NY, USA

Abstract

We study the question of when an approximate search optimization problem is harder than the associated decision problem. Specifically, we study a natural and quite general model of black-box search-to-decision reductions, which we call *branch-and-bound reductions* (in analogy with branch-and-bound algorithms). In this model, an algorithm attempts to minimize (or maximize) a function $f : D \to \mathbb{R}_{\geq 0}$ by making oracle queries to $h_f : \mathcal{S} \to \mathbb{R}_{\geq 0}$ satisfying

$$\min_{x \in S} f(x) \leq h_f(S) \leq \gamma \cdot \min_{x \in S} f(x) \tag{1}$$

for some $\gamma \geq 1$ and any subset S in some allowed class of subsets $\mathcal{S}$ of the domain D. (When the goal is to maximize f, h_f instead yields an approximation to the maximal value of f over S.) We show tight upper and lower bounds on the number of queries q needed to find *even a γ'-approximate minimizer* (or maximizer) for quite large γ' in a number of interesting settings, as follows.

- For arbitrary functions $f : \{0,1\}^n \to \mathbb{R}_{\geq 0}$, where $\mathcal{S}$ contains all subsets of the domain, we show that no branch-and-bound reduction can achieve $\gamma' \lesssim \gamma^{n/\log q}$, while a simple greedy approach achieves essentially $\gamma^{n/\log q}$.
- For a large class of MAX-CSPs, where $\mathcal{S} := \{S_w\}$ contains each set of assignments to the variables induced by a partial assignment w, we show that no branch-and-bound reduction can do significantly better than essentially a random guess, even when the oracle h_f guarantees an approximation factor of $\gamma \approx 1 + \sqrt{\log(q)/n}$.
- For the Traveling Salesperson Problem (TSP), where $\mathcal{S} := \{S_p\}$ contains each set of tours extending a path p, we show that no branch-and-bound reduction can achieve $\gamma' \lesssim (\gamma - 1)n/\log q$. We also prove a nearly matching upper bound in our model.

These results show an oracle model in which approximate search and decision are strongly separated. (In particular, our result for TSP can be viewed as a negative answer to a question posed by Bellare and Goldwasser (SIAM J. Comput. 1994), though only in an oracle model.) We also note two alternative interpretations of our results. First, if we view h_f as a data structure, then our results unconditionally rule out black-box search-to-decision reductions for certain data structure problems. Second, if we view h_f as an efficiently computable heuristic, then our results show that any reasonably efficient branch-and-bound *algorithm* requires more guarantees from its heuristic than simply Eq. (1).

Behind our results is a "useless oracle lemma," which allows us to argue that under certain conditions the oracle h_f is "useless," and which might be of independent interest. See also the full version [7].

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases search-to-decision reductions, oracles, constraint satisfaction, traveling salesman, discrete optimization

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 10; pp. 10:1–10:20
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.10

Category APPROX

Related Version *Full Version*: https://eccc.weizmann.ac.il/report/2021/141/

Funding *Siyao Guo*: Supported by National Natural Science Foundation of China Grant No. 62102260, Shanghai Municipal Education Commission (SMEC) Grant No. 0920000169, NYTP Grant No. 20121201 and NYU Shanghai Boost Fund.
Spencer Peters: Supported in part by the NSF under Grant No. CCF-2122230.
Noah Stephens-Davidowitz: Supported in part by the NSF under Grant No. CCF-2122230.

1 Introduction

We study *branch-and-bound search-to-decision* reductions for approximate optimization problems. These are algorithms that attempt to find an approximate minimizer (or maximizer) of a function $f : D \to \mathbb{R}_{\geq 0}$ by making oracle queries to an oracle h_f that yields an approximation to $\min_{x \in S} f(x)$ (or to $\max_{x \in S} f(x)$) for certain subsets $S \subseteq D$.[1] In the introduction, we will typically assume that $D = \{0, 1\}^n$.

1.1 Background and motivation

To explain our motivation, we first recall that any *exact* optimization problem comes in two flavors. The *decision* problem asks us to compute $\min_{x \in \{0,1\}^n} f(x)$ (or to compute the maximum) for some input objective function f.[2] The *search* problem asks us to find an $x \in \{0, 1\}^n$ that optimizes $f(x)$. In practice, for *many* problems of interest, search and decision seem to be more-or-less equally hard, in the sense that the fastest known algorithm for the decision problem effectively already solves the search problem. In other words, often it seems that the best way to determine whether something exists is to look for it. (Of course, this is certainly not always the case!)

One can try to formally explain this phenomenon by showing a *search-to-decision reduction*. In other words, to show that search and decision are essentially equivalent, we can prove that we can efficiently solve the search problem using an oracle for the decision problem.

Indeed, there is a well known and very elegant greedy approach to search-to-decision reductions that works for many *exact* optimization problems. Specifically, to minimize some function $f : \{0, 1\}^n \to \mathbb{R}_{\geq 0}$, one can define $f_{x_1} : \{0, 1\}^{n-1} \to \mathbb{R}_{\geq 0}$ for a bit $x_1 \in \{0, 1\}$ to be the function $f_{x_1}(x') := f(x_1, x')$, i.e., the function f with its first input bit set to x_1. One can then use a decision optimization oracle to find $V_{x_1} := \min_{x' \in \{0,1\}^{n-1}} f_{x_1}(x')$. (Here, we are assuming for simplicity that the decision problem is expressive enough to represent f_{x_1}. More generally, one can define f_0 and f_1 to be restrictions of f onto some sets S_0, S_1 that partition the domain.) Notice that $V_{x_1} \leq V_{1-x_1}$ if and only if there is an x that minimizes f

[1] The name "branch-and-bound reduction" comes from an analogy with *branch-and-bound algorithms* for optimization problems, which are ubiquitous in both theory and practice. (See, e.g., [14] and the references therein.) In the branch-and-bound literature, one typically uses some efficiently computable heuristic h_f to estimate $\min_{x \in S} f(x)$ in order to find an (exact or approximate) minimizer of f. See Section 1.4.

[2] Formally, to make this a true decision problem, we should include a threshold r in the input, and the problem should be to determine whether $\min_{x \in \{0,1\}^n} f(x) \leq r$. However, these two problems are essentially equivalent, as a simple binary-search-based reduction shows. We therefore ignore such subtleties.

and whose first bit is x_1. We can then repeat the process on f_{x_1} with x_1 chosen such that V_{x_1} is minimal, finding a second bit $x_2 \in \{0,1\}$ such that there exists a minimizer with first bit x_1 and second bit x_2, etc. Eventually, we will have found all of the bits of a minimizer x of f.[3]

This elegant and natural idea has many applications. Perhaps most importantly, this shows that if the decision problem of computing the minimal value of a function over subsets of the domain is solvable in time $T_D(n)$, then the search problem of *finding* a minimizer is solvable in time $T_S(n) \le O(n) \cdot T_D(n)$, so that the search and decision variants of this problem are equivalent in quite a strong sense. Indeed, this reduction even shows a type of "instance-wise equivalence," in the sense that it shows that the search problem of minimizing any *specific* function f is essentially equivalent to the problem of computing the minimal value of simple restrictions of f. Slight variants of this simple greedy reduction work for many *exact* optimization problems of interest, so that one can conclude that search and decision are essentially equivalent (and even instance-wise equivalent) for *many* important optimization problems. (However, Bellare and Goldwasser showed that this is not always the case by proving that there exist search problems that are not solvable in polynomial time but whose decision problems *are* solvable in polynomial time [2] (assuming a certain reasonable complexity-theoretic conjecture).)

It is then natural to ask what happens when we move to *approximate* optimization problems. Indeed, Feige first raised the question of whether *approximation* can be harder than *estimation* [5], and Feige and Jozeph later showed that there exist problems for which approximation is harder than estimation if and only if $\mathsf{FP} \neq \mathsf{TFNP}$ [6]. Here, we have adopted Feige's terminology, in which the task of *finding* $x \in \{0,1\}^n$ such that $f(x) \le \gamma \min_{x \in \{0,1\}^n} f(x)$ is called an *approximation problem*, while the task of determining $y \ge 0$ such that $\min_{x \in \{0,1\}^n} f(x) \le y \le \gamma \min_{x \in \{0,1\}^n} f(x)$ is called an *estimation problem*. In other words, approximation is the approximate search problem, and estimation is the approximate decision problem.

In fact, if the estimation problem is NP-complete for some approximation factor γ, then there is a certain rather weak sense in which approximation is provably equivalent to estimation (where both problems have the same approximation factor γ). For example, since such problems are polynomial-time equivalent to various exact optimization problems for which the above search-to-decision reduction works (e.g., MAX-3-SAT), one can simply combine reductions to and from such an exact optimization problem together with the search-to-decision reduction described above to reduce approximation to estimation for any NP-complete approximate optimization problem. However, this reduction is much less satisfying than the "greedy" search-to-decision reduction described above because, e.g., it can increase the size of the input instance by an arbitrary polynomial and will not in general preserve properties of the input instance (i.e., it will not prove instance-wise equivalence).[4]

[3] For a more concrete example, consider the maximization problem MAX-3-SAT. In this example, the reduction iteratively finds an assignment $(x_1, \dots, x_n) \in \{0,1\}^n$ that maximizes the number of satisfied clauses one bit at a time, using a decision MAX-3-SAT oracle. Specifically, it uses its decision oracle to determine x_1 such that such an assignment exists with first bit x_1, then to determine x_2 such that such an assignment exists with first two bits (x_1, x_2), etc.

[4] For example, the greedy reduction described above implies that $T_S(n) \le O(n) \cdot T_D(n)$, where $T_S(n)$ and $T_D(n)$ are "the fastest possible running times" for exact search and decision optimization problems. In contrast, the above NP-completeness-based reduction only guarantees that $T_{A,\gamma}(n) \le T_{E,\gamma}(n^C)$ where $T_{A,\gamma}(n)$ and $T_{E,\gamma}(n)$ are "the fastest possible running times" for approximation and estimation respectively for some fixed optimization problem for which γ-estimation is NP-complete (each with approximation factor γ), and C is an arbitrarily large constant. Since $T_{E,\gamma}(n)$ is superpolynomial in n (unless $\mathsf{P} = \mathsf{NP}$), this is not a very useful conclusion. E.g., it could be the case that $T_{E,\gamma}(n) = 2^n$, while $T_{A,\gamma}(n) = 2^{n^{100}}$. The fundamental issue is that this NP-completeness-based reduction can increase the instance size n by an arbitrary polynomial.

What we would really like is a *simple* reduction from approximation to estimation, that is, a reduction roughly like the "greedy" search-to-decision reduction for exact optimization described above. Such a reduction would ideally not increase the size n of the input instance. And, it would be even better if such a reduction only called its oracle on "subinstances" of the input instance, again in analogy with the simple greedy reduction. We might even be willing to sacrifice in the approximation factor in exchange for this simplicity – reducing γ'-approximation to γ-estimation for some $\gamma' > \gamma$.

Indeed, we originally arrived at this question in the context of lattice problems, in which preserving the size and structure of the input instance is often even more important than preserving the approximation factor. (See, e.g., [13, 15, 16].)

1.2 Our model

Our definition of branch-and-bound reductions can be viewed as one way of formalizing such "simple" reductions. Specifically, our model captures reductions that work via black-box access to an estimation oracle for "subinstances," i.e., an oracle h_f that estimates the optimal value of f up to a factor of γ over a subset S of the domain D.

In more detail, for an unknown objective function $f : D \to \mathbb{R}_{\geq 0}$ over a domain D (from some family of objective functions), such (possibly randomized) reductions have access to an oracle $h_f : \mathcal{S} \to \mathbb{R}_{\geq 0}$, where $\mathcal{S} \subseteq 2^D$ is a collection of subsets of the domain D. We assume that the oracle h_f satisfies

$$\min_{x \in S} f(x) \leq h_f(S) \leq \gamma \cdot \min_{x \in S} f(x)$$

for some not-too-large $\gamma \geq 1$, i.e., that h_f is a *γ-approximate estimation oracle* that solves the estimation problem on restrictions of f to various subsets. The goal of such a reduction is to find an explicit x such that $f(x) \leq \gamma' \min_{x \in D} f(x)$ for some not-too-large γ'.

We measure the running time of the reduction entirely in terms of the number of *queries* q made to this oracle. This makes our model quite strong in some sense. In particular, the lower bounds that we describe below are lower bounds on the number of queries, and therefore even rule out algorithms that perform a bounded number q of queries but otherwise perform arbitrary unbounded computation. (To be clear, queries may even be adaptive, i.e., the result of a previous query may be used to decide what to query next.) And, we study the best achievable approximation factor γ' for the problem of optimizing f that is achievable in this model, as a function of the number of queries q and the estimation approximation factor γ.

We stress that the *only* information that these reductions have about the function f comes from the oracle h_f. In particular, these reductions do *not* take as input a description of the function f. (Formally, the input to the reduction is actually empty.) This can be viewed as a weakness of our model. But, it does capture a wide class of reductions, and even more importantly, it is precisely this choice that will allow us to prove such strong *unconditional* lower bounds on the approximation factor γ' that is achievable after a certain number of oracle queries (without, e.g., requiring us to prove that $\mathsf{FP} \neq \mathsf{TFNP}$ along the way). Indeed, if we gave our reductions direct access to the input, then our model would actually be *stronger* than the standard model(!), and we would therefore have little hope of proving unconditional lower bounds in such a model.

We call reductions in our model *branch-and-bound reductions* in analogy with the paradigm of branch-and-bound algorithms. (See Section 1.4.) E.g., the simplest branch-and-bound reductions use a greedy approach like the one described above. In other words, they find

$x^* \in \{0,1\}^n$ that approximately minimizes $f : \{0,1\}^n \to \mathbb{R}_{\geq 0}$ as follows. The reduction first *branches*, i.e., it chooses subsets $S_1, \ldots, S_k \subseteq \{0,1\}^n$ that partition the input space $\{0,1\}^n$. It then uses h_f to compute values $h_f(S_1), \ldots, h_f(S_k)$ such that $h_f(S_i) \approx \min_{x \in S_i} f(x)$. It then chooses an i such that $h_f(S_i)$ is minimal and repeats the procedure on S_i – partitioning S_i into subsets $T_1, \ldots, T_k \subseteq S_i$, computing estimates $h_f(T_j)$, selecting T_j that minimizes this estimate, and so on. Eventually, the reduction is left with a single element $x^* \in \{0,1\}^n$, and we hope that $f(x^*)$ is relatively close to the optimal value $\min_{x \in \{0,1\}^n} f(x)$. It is not hard to see that this greedy reduction makes at most $q \leq k(n/\log k + 1)$ queries and outputs x^* with

$$f(x^*) \leq \gamma^{n/\log k+1} \min_{x \in \{0,1\}^n} f(x) \approx \gamma^{n/\log q} \min_{x \in \{0,1\}^n} f(x) \; .$$

(See Proposition 10 for a formal statement and proof.)

Even this quite large approximation factor $\gamma' \approx \gamma^{n/\log q}$ has proven to be quite useful in some rather specific contexts (e.g., [9, 16]). But, we of course would like to know if we can do better. I.e., is there a branch-and-bound reductions that makes at most q queries but achieves an approximation factor significantly better than $\gamma^{n/\log q}$? For example, a significant improvement to this approximation factor would resolve an important open problem in lattice-based cryptography [16]. (Indeed, this work originally arose from an effort to improve [16].)

1.3 Our results

Our first main result shows that the greedy reduction described above is essentially optimal in general. That is, there exist(s a distribution over) functions $f : \{0,1\}^n \to \mathbb{R}_{\geq 0}$ and a γ-approximate estimation oracle $h_f : 2^{\{0,1\}^n} \to \mathbb{R}_{\geq 0}$ such that no algorithm making significantly fewer than q queries to h_f can find $x^* \in \{0,1\}^n$ with $f(x^*) \ll \gamma^{n/\log q} \min_{x \in \{0,1\}^n} f(x)$. We use the notation $y \leftarrow \mathcal{A}^{h_f}()$ for sampling from the output distribution of the oracle algorithm $\mathcal{A}^{h_f}$, which formally takes no input.

▶ **Theorem 1** (Lower bound for arbitrary functions f. See Section 4). *For every $\gamma \geq 1$ and positive integer ℓ, there exists a distribution over functions $f : \{0,1\}^n \to \mathbb{R}_{\geq 0}$ and a γ-approximate estimation oracle $h_f : 2^{\{0,1\}^n} \to \mathbb{R}_{\geq 0}$ for f such that for any oracle algorithm $\mathcal{A}$ making at most q queries to h_f,*

$$\Pr[x^* \leftarrow \mathcal{A}^{h_f}() \;:\; f(x^*) \leq \gamma' \min_{x \in \{0,1\}^n} f(x)] \leq \varepsilon$$

where $\gamma' \approx \gamma^{n/\ell}$ and $\varepsilon \approx q2^{-\ell}$.

This shows that there is no branch-and-bound search-to-decision reduction that performs significantly better than the greedy reduction for *all* functions – even if it has an estimation oracle that works for arbitrary subsets S of the domain. So, if we want to do better than the generic greedy approach, we must place some restrictions on our objective function f. (In particular, this shows that we cannot improve upon the search-to-decision reductions in [9, 16] without using some specific properties of the relevant lattice-based objective functions f.)

We therefore turn our attention to specific classes of functions that have additional structure. Specifically, we study branch-and-bound search-to-decision reductions for Max-Constraint Satisfaction Problems (Max-CSPs), and the Traveling Salesperson Problem (TSP). These problems are natural in this context in part because both of these problems are often solved using branch-and-bound techniques in practice, for finding either exact solutions or approximate solutions. See, e.g., [3] for discussion of branch-and-bound algorithms for Max-CSPs (specifically MAX-k-SAT), and [4] for discussion of branch-and-bound algorithms for TSP.

In the case of Max-CSPs, we show a strong lower bound for any "reasonable" set of constraints $\mathcal{F}$. (See Section 6.) In this introduction, we discuss only the application of our more general result to MAX-3-SAT for simplicity.

Our result holds for *partial assignment queries.* That is, our oracle h_I for an instance I takes as input a partial assignment w to the n input variables and outputs a γ-approximation to the maximal number of constraints in the instance I satisfied by any assignment v that matches w.[5] This notion of a partial assignment arises naturally in this context (e.g., in practical branch-and-bound algorithms for MAX–CSP).

Before we state our result, we note that it is trivial to find an assignment to a MAX-3-SAT instance that satisfies roughly a $7/8 - o(1)$ fraction of the constraints. Indeed, a random assignment suffices with high probability. So, we can easily output an assignment that satisfies at least $7/8 - o(1)$ times as many clauses as an optimal assignment (with high probability, using an algorithm that is actually independent of the instance). Our lower bound shows that a branch-and-bound reduction cannot achieve an approximation factor of better than $7/8 + o(1)$, even with a very good estimation oracle with approximation factor $\beta = 1 - o(1)$. (Here, since we have switched from minimization to maximization, we have switched to oracles h satisfying

$$\beta \max_{x \in S} f(x) \le h_f(x) \le \max_{x \in S} f(x)$$

for some $\beta \le 1$.) In other words, no branch-and-bound reduction performs significantly better than the algorithm that simply outputs a uniformly random assignment, even if the estimation oracle is extremely powerful.

▶ **Theorem 2** (Lower bound for MAX-3-SAT. See Section 6 for a more general result.). *There exists a distribution over* MAX-3-SAT *instances I such that for every $\beta < 1$, there is a β-approximate estimation oracle h_I such that for any oracle algorithm $\mathcal{A}$ making at most q queries to h_I,*

$$\Pr[v^* \leftarrow \mathcal{A}^{h_f}() \ : \ \mathsf{SAT}_I(v^*) \ge \delta \max_{v \in \{0,1\}^n} \mathsf{SAT}_I(v)] \le \varepsilon$$

where $\delta = 7/8 + o(1)$ and $\varepsilon \approx q2^{-(1-\beta)^2 n}$.

Finally, we study (non-metric) TSP. Here, we consider a natural class of oracles that work for *subtour* queries. That is, they take as input a path $(v_1, \ldots, v_k)$ of (distinct) vertices in the input graph, and they output a γ-approximation to the minimal value of a tour that contains the path $(v_1, \ldots, v_k)$. This naturally captures branch-and-bound reductions that, e.g., "build a path one edge at a time." We prove the following lower bound.

▶ **Theorem 3** (Lower bound for TSP. See Section 5.). *For every $\gamma > 1$ and positive integer $\ell \ll n$, there exists a distribution G of TSP instances and a γ-approximate estimation oracle h_G such that for any oracle algorithm $\mathcal{A}$ making at most q oracle queries,*

$$\Pr[c^* \leftarrow \mathcal{A}^{h_G}() \ : \ w_G(c^*) \le \gamma' \cdot \min_c w_G(c)] \le \varepsilon \ ,$$

where $\gamma' \approx (\gamma - 1)n/\ell$, $\varepsilon \approx qe^{-\ell}$, and $w_G(c)$ is the weight of the tour c in G.

[5] To formally represent this in our model, consider the function $f : \{0,1\}^n \to \mathbb{R}_{\ge 0}$ such that $f(v)$ is the number of clauses satisfied by an assignment v. Then, h_f takes as input subsets $S_w \subseteq \{0,1\}^n$ consisting of all assignments v that match a partial assignment w. E.g., $\{v = (v_1, \ldots, v_n) \in \{0,1\}^n \ : \ v_1 = 0, v_{82} = 1\}$. Of course, it is far more natural to simply consider an oracle h_I that takes w as input directly.

We note in passing that this lower bound can be viewed as a partial answer to a question posed by Bellare and Goldwasser [2], who asked whether a search-to-decision reduction was possible for approximate TSP, and particularly one that preserves the approximation factor. Theorem 3 rules out a large class of such reductions, even those achieving a significantly worse approximation factor than what was considered in [2]. But, we only rule this out in an oracle model.

Our lower bound for TSP is in some sense quite strong. For example, it shows that a polynomial-time branch-and-bound reduction cannot achieve an approximation factor $\gamma' < o(n/\log n)$, even with an estimation oracle that achieves a constant approximation factor $\gamma = O(1)$. However, it is not immediately obvious whether there is a nearly matching upper bound. E.g., the natural greedy approach achieves an approximation factor of roughly $\gamma' = \gamma^{n/k}$ using roughly n^k queries, which is quite far from our lower bound. Perhaps our lower bound can be improved?

As it happens, there *is* a nearly matching upper bound (specifically, a reduction that uses roughly n^t queries and achieves an approximation factor of roughly $\gamma n/t$). So, our lower bound is essentially tight in our model. But, the reduction achieving this upper bound is inefficient and therefore rather unsatisfying. In other words, while the reduction uses relatively few oracle queries, it performs additional computation that requires superpolynomial time, as described in Theorem 14.[6]

1.4 Other interpretations of our model

Above, we have presented our model in terms of branch-and-bound reductions from approximation problems to associated estimation problems. However, we note that there are at least two different interpretations of our model (and results) that are perhaps just as interesting.

Branch-and-bound algorithms

We of course named our model of branch-and-bound reductions in analogy with branch-and-bound *algorithms.* Such algorithms are ubiquitous in the study of optimization problems. (See, e.g., [14] and the references therein.) To view our model in terms of branch-and-bound algorithms, one simply needs to view the estimation oracle h_f as some efficiently computable *heuristic.* In other words, a branch-and-bound algorithm works by using some heuristic h_f to estimate the optimal value of f on various subsets until it has found an approximate optimizer of f.

Though branch-and-bound algorithms are very natural, they seem to be quite difficult to understand from a theoretical perspective. E.g., Nemhauser said "I have always wanted to prove a lower bound about the behavior of branch and bound, but I never could" [11]. (See also the third proposed research direction in [14].) Of course, part of the difficulty in proving such lower bounds is simply coming up with a model that is sufficiently strong to capture a wide class of branch-and-bound algorithms but weak enough to allow for provable lower bounds.

[6] The reduction first calls the oracle on all n^k subpaths $V = (v_1, \ldots, v_k)$ of length k. It then finds (in superpolynomial time) a way to combine subpaths $V_1, \ldots, V_{n/k}$ together into a tour while minimizing $h_G(V_1) + \cdots + h_G(V_{n/k})$. It is not hard to show that such a reduction achieves an approximation factor of essentially $\gamma n/k$.

One can view our results as progress towards that goal. In particular, we prove lower bounds for branch-and-bound algorithms with arbitrary estimation heuristics h_f. That is, any branch-and-bound algorithm that beats our lower bounds must use some property of the heuristic h_f that is stronger than the approximation guarantee

$$\min_{x \in S} f(x) \leq h_f(S) \leq \gamma \min_{x \in S} f(x) .$$

Approximation vs. estimation for data structure problems

Yet another interpretation of our results is in terms of *data structures* for optimization problems. For this interpretation, we focus on the problem of optimizing an arbitrary function with arbitrary subset queries, as this is most natural in this context.

Specifically, consider the following very natural and simple data structure problem. We are first given as input a function $f : \{0,1\}^n \to \mathbb{R}_{\geq 0}$ (or, equivalently, a list of 2^n numbers) and allowed arbitrary time to preprocess it into some data structure H (with some constraint on the size of H). Then, we receive as input some subset $S \subseteq \{0,1\}^n$. In the estimation version of this problem, our goal is to estimate $\min_{x \in S} f(x)$ using as few queries to H as possible (i.e., reading as few bits or words from H as possible). In the approximation version, our goal is to find an $x^* \in S$ such that $f(x^*)$ is as small as possible, relative to $\min_{x \in S} f(x)$ (again, using as few queries to H as possible).

It is then natural to ask whether there is a *black-box data-structure reduction* from approximation to estimation in this setting. Here, a black-box reduction means an algorithm that solves the approximation problem using *only* the estimates obtained from the data structure for the estimation problem – i.e., a branch-and-bound algorithm. Notice in particular that, in the setting of data structures, the number of queries made by such a reduction is the natural complexity measure.

Our lower bound on branch-and-bound reductions extends immediately to black-box data-structure reductions as well. Specifically, no black-box data-structure reduction making q estimate queries can achieve an approximation factor significantly better than $\gamma' \approx \gamma^{\log |S| / \log q}$.

1.5 Our techniques

Our main technical tool is a "useless oracle lemma." The idea is that "an oracle cannot be useful if most of its answers are predictable." While this lemma is quite general, we focus below on how it applies to our setting. (Very similar ideas are used in a different context in the literature on submodular optimization. See, e.g., [18, 17].)

Suppose that we can construct a distribution over functions f together with a γ-approximate estimation oracle $h_f : \mathcal{S} \to \mathbb{R}_{\geq 0}$ such that for any fixed query $S \in \mathcal{S}$, $h_f(S) = g(S)$ with high probability over f, where g is some *fixed* function that does not depend on f. E.g., our hard instance for arbitrary optimization has this property with $g(S) \approx \gamma^{n - \log |S|}$. Then, the useless oracle lemma tells us that "an algorithm with oracle access to h_f is essentially no more powerful than an algorithm with no access to any oracle that depends on f at all." The specific statement is of course quantitative and depends on the number of oracle queries and the probability that $h_f(S) \neq g(S)$. (See Section 3. We do not claim that this idea is original, though we do not know of prior work using it.)

With this tool in hand, our goal becomes to construct distributions of functions f together with oracles h_f such that (1) the oracle is "useless" in the sense described above; and (2) for every fixed $x \in \{0,1\}^n$, $f(x) \geq \gamma' \min_{x' \in \{0,1\}^n} f(x')$ with high probability over f (i.e., there is no way to choose an x independently of f such that $f(x)$ is nearly optimal). This boils down to finding a distribution over functions f such that (1) the random variable $\min_{x\in S} f(x)$ is highly concentrated; and (2) this quantity tends to be much larger as $|S|$ becomes smaller. In particular, concentration of $\min_{x\in S} f(x)$ around some value $g(S)$ (independent of f) implies that we can set $h_f(S) \approx g(S)$. And, if $g(S)$ increases rapidly as $|S|$ becomes smaller, then the optimal value of f, $\min_{x\in\{0,1\}^n} f(x) \approx g(\{0,1\}^n)$ will in particular be much smaller than $g(\{x'\})$ for any fixed $x' \in \{0,1\}^n$, causing our final approximation factor to be large.

Below, we briefly describe the distributions that we use and some of the intuition behind them.

Arbitrary functions

For our lower bound against optimizing arbitrary functions (Theorem 1), we sample each value $f(x)$ independently (but not uniformly) from a distribution over the set $\{1, \gamma, \gamma^2 \ldots, \gamma^{n/\ell}\}$, where 2^ℓ is the number of queries made by the algorithm. We choose our distribution so that any subset $S \subseteq \{0,1\}^n$ of size roughly $2^{\ell k}$ (1) contains at least one element with value either $\gamma^{n/k-1}$ or $\gamma^{n/k}$ with probability significantly larger than $1 - 2^{-\ell}$; and (2) does not contain an element with value less than $\gamma^{n/k-2}$, except with probability significantly less than $2^{-\ell}$. We can then set $h_f(S) = \gamma^{n/k}$, and we see that with high probability "any 2^ℓ queries made by the algorithm will reveal no information about the function f with high probability."

TSP

For the Traveling Salesperson Problem, we sample the weight of each edge independently to be either 1 or essentially γn, with equal probabilities. We then use the theory of random graphs to argue that with high probability the value of an optimal tour containing any subpath of length ℓ is highly concentrated around a value of roughly $\gamma \ell n/2 + n - \ell/2$.

CSPs

For ("reasonable") k-CSPs, we construct "a random instance with a random planted satisfying assignment." That is, we first sample a uniformly random assignment $P \sim [c]^n$.[7] We then repeatedly sample a k-tuple $T = (t_1, \ldots, t_k) \in [n]^k$ and add to our CSP a random constraint on the variables $x_{t_1}, \ldots, x_{t_k}$ that is satisfied by our planted assignment $P_{t_1}, \ldots, P_{t_k}$. We do this $m = O(n)$ times to generate our CSP. (For simplicity, we are ignoring the possibility that there might not exist any such constraint. In Section 6, we handle this carefully.)

Then, for any partial assignment w, we study $\rho_w(P)$, which is the maximum over all assignments $v \in [c]^n$ extending w of the probability that v satisfies a constraint sampled as above. It is easy to see via the Chernoff-Hoeffding bound that, for a fixed planted assignment P, the maximum over all such v of the actual number of satisfied constraints in our instance will be heavily concentrated around $\rho_w(P) \cdot m$. The difficult step in our analysis is then to prove that $\rho_w(P)$ is itself highly concentrated around its expectation $\mathbb{E}_P[\rho_w(P)]$. This follows from a careful application of McDiarmid's inequality.

[7] Throughout this paper, for $x \in \mathbb{N}$, $[x]$ denotes the set $\{1, 2, \ldots, x\}$.

2 Preliminaries

Given an event E and a random variable X, we will write $\Pr[E \mid X]$ for the random variable whose value in case $X = x$ is $\Pr[E \mid X = x]$. We will need the following concentration inequalities.

▶ **Lemma 4** (Chernoff-Hoeffding bound [8]). *Suppose $X_1, \dots, X_n$ are independent random variables, and let $X := \sum_{i=1}^n X_i$. Then for any $0 < \delta < 1$,*

$$\Pr\left[|X - \mathbb{E}[X]| \ge \delta\, \mathbb{E}[X]\right] \le 2\exp(-\delta^2\, \mathbb{E}[X]/3).$$

▶ **Lemma 5** (McDiarmid's inequality [12]). *Suppose $X_1, \dots, X_n$ are independent random variables, $c_1, \dots, c_n$ are real numbers, and $f : \mathbb{R}^n \to \mathbb{R}$ is a function with the property that for all $i \in [n]$ and $x_i, x_i' \in \mathbb{R}$, $|f(x_1, \dots, x_i, \dots, x_n) - f(x_1, \dots, x_i', \dots, x_n)| \le c_i$. Then for all $t > 0$,*

$$\Pr\left[|f(X_1, \dots, X_n) - \mathbb{E}[f(X_1, \dots, X_n)]| \ge t\right] \le 2\exp\left(\frac{-t^2}{2\sum_{i=1}^n c_i^2}\right).$$

3 Useless Oracles

We now present the technical tool that we will use for all of our lower bounds, which we call the useless oracle lemma. (The authors do not claim that this result is original, though they do not know of any prior work using a similar idea.)

To understand the lemma and its relation to branch-and-bound algorithms, suppose that we sample the objective function $f : D \to \mathbb{R}_{\ge 0}$ that our algorithm is meant to optimize from some distribution, and then choose our heuristic h_f according to f. And, recall that such an algorithm's "only access to this objective function f" is via oracle access to this heuristic h_f, which satisfies $\min_{x \in S} f(x) \le h_f(S) \le \gamma \min_{x \in S} f(x)$ for every S in some collection of subsets $\mathcal{S}$ of the domain D.

Now, imagine that instead of giving our algorithm oracle access to h_f, we give it oracle access to some *other* function g, which is fixed (i.e., not a random variable) and therefore independent of our choice of f. Notice that this is a rather cruel thing to do, since now our algorithm cannot possibly hope to glean any information about f from its oracle queries. So, at least intuitively, it should be quite easy to prove lower bounds against such algorithms.

The useless oracle lemma provides conditions that allow us to effectively replace the oracle $\mathcal{O} = h_f$ with the "useless" oracle g. In particular, it says that if (1) h_f and g have the property that $\Pr[h_f(S) = g(S)]$ is large for all *fixed* S, and (2) our algorithm does not make too many oracle queries; then the output of our algorithm is nearly the same whether it is given access to h_f or to g.

▶ **Lemma 6** (The useless oracle lemma). *Let $g : \mathcal{S} \to R$ be a* fixed *function, and let $\mathcal{O} : \mathcal{S} \to R$ be a distribution over oracles such that for all $S \in \mathcal{S}$,*

$$\Pr[\mathcal{O}(S) \ne g(S)] \le p\,.$$

Then, for any oracle algorithm $\mathcal{A}^{\mathcal{O}}$ making at most q queries to $\mathcal{O}$, the statistical distance between $\mathcal{A}^{\mathcal{O}}()$ and $\mathcal{A}^{g}()$ is at most qp.

Proof. Let $S_1, \ldots, S_q \in \mathcal{S}$ be the sequence of oracle queries made by $\mathcal{A}^{\mathcal{O}}$. Notice that the S_i are random variables, and that S_i might depend on $\mathcal{O}(S_j)$ for $j < i$ (which is what makes the lemma non-trivial). Let E_i be the event that there exists a $j \le i$ such that $\mathcal{O}(S_j) \neq g(S_j)$. It suffices to prove that $\Pr[E_q] \le qp$ (because the two distributions are identical if we condition on $\neg E_q$).

Notice that

$$\Pr[E_q] = \Pr[E_{q-1}] + \Pr[\mathcal{O}(S_q) \neq g(S_q) \text{ and } \neg E_{q-1}] \; .$$

Let $S'_1, \ldots, S'_q$ be the sequence of oracle queries made by $\mathcal{A}^g$ (as opposed to $\mathcal{A}^{\mathcal{O}}$), and notice that S'_i is independent of $\mathcal{O}$ by definition. In particular, this implies that $\Pr[\mathcal{O}(S'_i) \neq g(S'_i)] \le p$. Therefore,

$$\Pr[\mathcal{O}(S_q) \neq g(S_q) \text{ and } \neg E_{q-1}] = \Pr[\mathcal{O}(S'_q) \neq g(S'_q) \text{ and } \neg E_{q-1}] \le \Pr[\mathcal{O}(S'_q) \neq g(S'_q)] \le p \; .$$

It follows that $\Pr[E_q] \le p + \Pr[E_{q-1}]$, which implies the result when combined with the trivial fact that $\Pr[E_1] \le p$. ◀

▶ **Corollary 7.** *Let $f \sim \mathcal{D}$ be sampled over some distribution of functions $f : D \to \mathbb{R}_{\ge 0}$, and let $\mathcal{S} \subseteq 2^D$ be a collection of subsets of D such that $D \in \mathcal{S}$ and $\{x\} \in \mathcal{S}$ for all $x \in D$. Suppose that there exists some fixed function $g : \mathcal{S} \to \mathbb{R}_{\ge 0}$ such that*

$$\Pr[g(S)/\gamma \le \min_{x \in S} f(x) \le g(S)] \ge 1 - p$$

for all $S \subseteq \mathcal{S}$ and some $p > 0$, $\gamma \ge 1$.

Then, there exists a γ-approximate heuristic $h_f : \mathcal{S} \to \mathbb{R}_{\ge 0}$ for f such that for any algorithm $\mathcal{A}$ making at most q oracle queries to h_f,

$$\Pr[x^* \leftarrow \mathcal{A}^{h_f}(), \; f(x^*) < \gamma' \min_{x \in D} f(x)] \le (q+2)p \; ,$$

where

$$\gamma' := \gamma^{-1} \cdot \min_{x \in D} \frac{g(\{x\})}{g(D)} \; .$$

Proof. We define

$$h_f(S) := \begin{cases} g(S) & g(S)/\gamma \le \min_{x \in S} f(x) \le g(S) \\ \min_{x \in S} f(x) & \text{otherwise.} \end{cases}$$

In other words, $h_f(S)$ is a γ-approximate heuristic that agrees with $g(S)$ whenever it is possible for a γ-approximate heuristic to do so (and when this is not possible, it simply outputs the exact minimal value).

Notice that by assumption

$$\Pr_{f \sim D}[h_f(S) \neq g(S)] \le p \; .$$

We may therefore apply the useless oracle lemma with $\mathcal{O} = h_f$ to show that $\mathcal{A}^{h_f}()$ is within statistical distance qp of $\mathcal{A}^g()$. So, it suffices to show that

$$\Pr_{f \sim \mathcal{D}}[x^* \leftarrow \mathcal{A}^g(), \; f(x^*) \le \gamma' \min_{x \in D} f(x)] \le 2p \; .$$

Indeed, since g is independent of f (as it is a fixed function), we have that for any $r \geq 0$,

$$\Pr_{f \sim \mathcal{D}}[x^* \leftarrow \mathcal{A}^g(),\ f(x^*) \leq r] = \sum_{x \in D} \Pr[x^* \leftarrow \mathcal{A}^g(),\ x^* = x] \cdot \Pr_{f \sim \mathcal{D}}[f(x) \leq r] \leq \max_{x \in D} \Pr_{f \sim \mathcal{D}}[f(x) \leq r] \ .$$

By assumption, for any $x \in D$,

$$\Pr[f(x) < \min_{x' \in D} g(\{x'\})/\gamma] \leq p \ ,$$

and similarly,

$$\Pr[\min_{x' \in D} f(x') > g(D)] \leq p \ .$$

Therefore, for any $x \in D$,

$$\Pr[f(x) < \gamma' \min_{x' \in D} f(x')] \leq \Pr[f(x) < \min_{x' \in D} g(\{x'\})/\gamma] + \Pr[\min_{x' \in D} f(x') > g(D)] \leq 2p \ ,$$

and the result follows. ◀

4 A Generic Optimization Problem

▶ **Theorem 8.** *For any n, any $\gamma \geq 1$, and any integer $1 \leq \alpha \leq n$, there exists a distribution over functions $f : \{0,1\}^n \to \mathbb{R}_{\geq 0}$ such that for any integer $1 \leq k \leq \alpha$ and any $S \subseteq \{0,1\}^n$ with $2^{(k-1)n/\alpha} \leq |S| \leq 2^{kn/\alpha}$,*

$$\Pr_f[\gamma^{\alpha-k+1} \leq \min_{x \in S} f(x) \leq \gamma^{\alpha-k+2}] \geq 1 - 4(n/\alpha)2^{-n/\alpha} \ .$$

Proof. For each $x \in \{0,1\}^n$ and integer $1 \leq i \leq \alpha$, we set $f(x) = \gamma^i$ independently with probability p_i, where $p_i := \delta \cdot 2^{in/\alpha - n}$ for $i = 1, \ldots, \alpha - 1$ and $p_\alpha := 1 - p_1 - p_2 - \cdots - p_{\alpha-1}$, with $\delta := n/\alpha \cdot 2^{-n/\alpha} < 1$.

Notice that

$$p_\alpha = 1 - \delta \cdot \sum_{i=1}^{\alpha-1} 2^{in/\alpha - n} \geq 1 - \delta \ .$$

In particular, this is non-negative, so that this is in fact a valid probability distribution. We have

$$\Pr_f[\gamma^{\alpha-k+1} \leq \min_{x \in S} f(x) \leq \gamma^{\alpha-k+2}] = 1 - \Pr[\min_{x \in S} f(x) < \gamma^{\alpha-k+1}] - \Pr[\min_{x \in S} f(x) > \gamma^{\alpha-k+2}] \ .$$

We wish to argue that each of the probabilities on the right-hand side are smaller than, say, 2δ. First, we see that, for the non-trivial case $k > 2$,

$$\Pr[\min_{x \in S} f(x) > \gamma^{\alpha-k+2}] \leq (1 - p_{\alpha-k+2})^{|S|} \leq \left(e^{-p_{\alpha-k+2}}\right)^{2^{(k-1)n/\alpha}} = e^{-\delta 2^{n/\alpha}} \leq \delta \ ,$$

where the second inequality uses the fact that $1 - x \leq e^{-x}$. Second,

$$\Pr[\min_{x \in S} f(x) < \gamma^{\alpha-k+1}] \leq |S| \cdot \Pr[f(x) \leq \gamma^{\alpha-k}] \leq 2^{kn/\alpha} \cdot \delta \cdot \sum_{i=1}^{\alpha-k} 2^{in/\alpha - n} \leq 2\delta \ ,$$

as needed. ◀

▶ **Corollary 9.** *For any integer n, any $1 \le \ell \le n/3$, and any $\gamma \ge 1$, there exists a distribution of functions $f : \{0,1\}^n \to \mathbb{R}_{\ge 0}$ and a γ-approximate heuristic $h_f : 2^{\{0,1\}^n} \to \mathbb{R}_{\ge 0}$ for f such that for any algorithm $\mathcal{A}^{h_f}$ making at most q oracle queries to h_f,*

$$\Pr[x^* \leftarrow \mathcal{A}^{h_f}(),\ f(x) < \gamma' \cdot \min_{x \in \{0,1\}^n} f(x)] \le (q+2) \cdot \ell 2^{-\ell+3}$$

where $\gamma' := \gamma^{\lfloor n/\ell \rfloor - 1}$.

Proof. We apply Corollary 7 to the choice of f from Theorem 8, with $g(S) := \gamma^{\alpha-k+2}$ for the unique integer $1 \le k \le \alpha$ satisfying $2^{(k-1)n/\alpha} \le |S| < 2^{kn/\alpha}$ where $\alpha := \lfloor n/\ell \rfloor$ and $g(\{0,1\}^n) := \gamma$. Notice in particular that $g(\{x\})/g(\{0,1\}^n) = \gamma^\alpha$. ◀

The following proposition shows that the lower bound of Corollary 9 is essentially tight.

▶ **Proposition 10.** *For all integers $n \ge \ell \ge 1$, there exists an algorithm $\mathcal{A}^h$ such that for all functions $f : \{0,1\}^n \to \mathbb{R}_{\ge 0}$, all $\gamma \ge 1$, and all γ-approximate heuristics $h : 2^{\{0,1\}^n} \to \mathbb{R}_{\ge 0}$ for f, $\mathcal{A}^h$ makes at most $\lceil n/\ell \rceil \cdot 2^\ell$ queries to h, and outputs x^* with $f(x^*) \le \gamma^{\lceil n/\ell \rceil} \min_{x \in S} f(x)$.*

Proof. The algorithm $\mathcal{A}^h$ is a natural greedy algorithm. It first partitions the domain $S^0 = \{0,1\}^n$ into $k := 2^\ell$ subsets $S^0_1, \ldots, S^0_k$ and queries h on each subset. It sets $S^1 = S^0_i$ for some S^0_i of minimum h value; that is, $h(S^0_i) = \min_{1 \le j \le k} h(S^0_j)$. It then repeats this process, partitioning S^1 itself into k subsets $S^1_1, \ldots, S^1_k$ of size $2^{n-2\ell}$, querying h on each one, setting S^2 to be any subset with minimum h value, and so on. After $\lceil n/\ell \rceil - 1$ iterations, it finds a subset $S^{\lceil n/\ell \rceil - 1}$ with fewer than k elements. It then queries h on each singleton subset of $S^{\lceil n/\ell \rceil - 1}$ to find a singleton subset $S^{\lceil n/\ell \rceil} =: \{x^*\}$ of minimum h value, and outputs x^*.

It is clear that $\mathcal{A}^h$ makes at most $\lceil n/\ell \rceil \cdot 2^\ell$ queries to h. Moreover, we trivially have $f(x^*) = \min_{x \in S^{\lceil n/\ell \rceil}} f(x)$. And for each i, by the fact that h is a γ-approximate heuristic for f, we have that

$$\begin{aligned}\min_{x \in S^{i+1}} f(x) \le h(S^{i+1})& \\ &= \min_{1 \le j \le k} h(S^i_j) \\ &\le \min_{1 \le j \le k} \gamma \cdot \min_{x \in S^i_j} f(x) \\ &\le \gamma \min_{x \in S^i} f(x).\end{aligned}$$

It is therefore clear by induction that the output x^* satisfies $f(x^*) \le \gamma^{\lceil n/\ell \rceil} \min_{x \in S} f(x)$. ◀

In particular, when $n = \mathrm{poly}(\ell)$, the greedy algorithm achieves $\gamma' = \gamma^{\tilde{O}(n/\log q)}$, and by Corollary 9, this is the best approximation factor achievable (up to lower-order terms) by any oracle algorithm that succeeds with constant probability.

5 The Traveling Salesperson Problem

We consider undirected weighted complete graphs on n vertices with no self-loops where all edge weights are non-negative. For an edge $e = \{i, j\}$ of a graph G, $w(e)$ and $w(i, j)$ denote the weight of e.

We write C_n for the set of all Hamiltonian cycles in a complete n-vertex graph G. The weight of a cycle c is the sum of the weights of its edges: $w(c) = \sum_{i=1}^n w(c_i, c_{i+1 \bmod n})$. For $0 \le k \le n-1$, let P^k_n be the set of all length-k simple paths on an n-vertex graph:

$$P^k_n = \{(v_0, \ldots, v_k) \mid v_0, \ldots, v_k \in [n], v_0, \ldots, v_k \text{ all distinct}\}.$$

By $\mathrm{OPT}(G)$ we denote the weight of an optimal traveling salesperson (TSP) cycle in G, and for a path $p \in P_n^k$ by $\mathrm{OPT}(G, p)$ we denote the minimum weight of a TSP cycle in G containing p.

We will use the following result that says that a random graph contains a Hamiltonian cycle with all but negligible probability.

▶ **Lemma 11** (E.g., [1]). *A uniform random undirected graph with n vertices contains a Hamiltonian cycle with probability $\geq 1 - 2^{-n+o(n)}$.*

We define a "useless oracle" for TSP in Theorem 12. Namely, for every approximation factor $\gamma > 1$ and every probability of error ($\approx e^{-t}$), we give a distribution of graphs such that with high probability, the weight of an optimal Hamiltonian cycle extending a path of length k essentially does not depend on the path, but only on its length k. Intuitively, an approximate TSP oracle is useless for this distribution of graphs because (with high probability) it reveals no information about the input graph.

▶ **Theorem 12** (TSP useless oracle). *For all $\gamma > 1$ and $1 \leq t \leq \delta n/2$ where $\delta = (\gamma - 1)/(\gamma + 1)$, there exists a distribution $\mathcal{G}$ over n-vertex graphs, and values $v_0, \ldots, v_{n-1}$ such that*

- *$v_0 \leq n/\gamma$ and $v_{n-1} \geq (\gamma - 1)n^2/(5t)$,*
- *for all $0 \leq k \leq n-1$ and all paths $p \in P_n^k$,*

$$\Pr_{G\sim\mathcal{G}}\left[\mathrm{OPT}(G, p) \in [v_k,\ \gamma v_k]\right] \geq 1 - O(e^{-\delta^2 t/30}).$$

For reasons of space, we defer the proofs of this and subsequent theorems to the appendix. They can also be found in the full version [7].

We are now ready to prove the main result of this section showing an essentially tight bound on search to decision reductions with an approximate TSP oracle.

▶ **Definition 13** (TSP oracle). *A function h_G is a γ-approximate TSP estimation oracle for $G \in \mathcal{G}_n$ if for all $0 \leq k \leq n-1$, for all $p \in P_n^k$,*

$$\mathrm{OPT}(G, p) \leq h_G(p) \leq \gamma\,\mathrm{OPT}(G, p).$$

▶ **Theorem 14** (TSP oracle bounds). *For every $\gamma > 1$ and positive integer $\ell \leq \delta^3 n/20$ for $\delta = (\gamma-1)/(\gamma+1)$, there exists a distribution $\mathcal{G}$ over n-vertex graphs G and a γ-approximate TSP estimation oracle h_G for G such that for any algorithm $\mathcal{A}^{h_G}$ making at most q queries to h_G,*

$$\Pr[c \leftarrow \mathcal{A}^{h_G}()\ :\ w(c) \leq \gamma' \cdot \mathrm{OPT}(G)] \leq \varepsilon,$$

where $\gamma' := (\gamma - 1)\delta^2 n/(150\ell)$ and $\varepsilon := O\big(q \cdot e^{-\ell}\big)$.

Furthermore, for every $\gamma \geq 1$ and positive integer ℓ, there exists an algorithm $\mathcal{A}^{h_G}$ making at most $\binom{n}{\ell} \cdot \ell! \leq n^\ell$ queries to a γ-approximate TSP estimation oracle h_G that computes a $\gamma n/(\ell - 1)$-approximation to TSP.

In particular, using q queries, one can achieve $\gamma' = \gamma n/(\log q - 1)$, but no algorithm that succeeds with constant probability can do better than $\gamma' = O((\gamma - 1)n/\log q)$.

6 Constraint Satisfaction Problems

Informally, a constraint satisfaction problem (CSP) consists of constraints applied to variables; the goal is to find an assignment of values to variables that satisfies all or most of the constraints. For example, graph coloring is a CSP, where each edge corresponds to the constraint that the endpoints have different colors.

A CSP is specified by a non-empty family of constraint functions $\mathcal{F}$. Each constraint function $f \in \mathcal{F}$ is a function $f : [c]^k \to \{0,1\}$, where the alphabet size c and arity k are fixed. An assignment assigns a value from $[c]$ to each of the n variables $x_1, \ldots, x_n$. Formally, we represent this by a function $v : [n] \to [c]$. An assignment v satisfies a constraint $C = (f, (j_1, \ldots, j_k))$, written $v \models C$, if $f(v(x_{j_1}), \ldots, v(x_{j_k})) = 1$. We write $\mathsf{CSP}(\mathcal{F})$ for the CSP specified by $\mathcal{F}$; an instance $I \in \mathsf{CSP}(\mathcal{F})$ is simply a collection of constraints. For simplicity, we allow duplicate constraints. As is standard, we say I is satisfiable if there is an assignment v that satisfies all constraints of I. For example, the classical 3-SAT problem is $\mathsf{CSP}(\mathcal{F}_{3-\mathrm{SAT}})$, where $\mathcal{F}_{3-\mathrm{SAT}}$ is the family of constraint functions defined by disjunctive clauses (e.g., $f(x_1, x_2, x_3) = \neg x_1 \vee x_2 \vee x_3$).

We are interested in an approximation version of the constraint satisfaction problem, namely $\mathsf{MAX–CSP}(\mathcal{F})$ [10]. $\mathsf{MAX–CSP}(\mathcal{F})$ is the computational problem whose instances I are collections of $m(I)$ constraints on variables $x_1, \ldots, x_n$, and the objective is to find an assignment v to these variables that maximizes the number of satisfied constraints.

A *partial assignment* assigns values to a subset of the n variables. Formally, a partial assignment is represented by a partial function $w : [n] \to [c]$. An assignment v *extends* w if v agrees with w on all variables to which w assigns a value. Define $\mathsf{SAT}(I)$ to be the maximum number of satisfied constraints over all assignments. Define $\mathsf{SAT}_I(w)$ in the same way, except the maximum is taken over only the assignments extending w. In the special case where w is a total assignment, this is simply the number of constraints satisfied (unsatisfied) by that assignment. Similar to before, for $\beta < 1$, we define a function h_I to be a β-heuristic if $\beta\,\mathsf{SAT}_I(w) \le h_I(w) \le \mathsf{SAT}_I(w)$.

To state our lower bound for Max-CSPs in full generality, we must first define the hard distribution $\mathcal{D}_s$ over instances $I \in \mathsf{CSP}(\mathcal{F})$. $\mathcal{D}_s$ is defined by the following sampling process. All sampling steps are done uniformly at random. First we sample a “planted” assignment P. Then, for each of s steps $i = 1, 2, \ldots, s$, we sample an ordered tuple $\left(x_{J_1^{(i)}}, \ldots, x_{J_k^{(i)}}\right)$ of k distinct variables, and sample a constraint function F_i that is satisfied by the input $\left(P\big(x_{J_1^{(i)}}\big), \ldots, P\big(x_{J_k^{(i)}}\big)\right)$. If there is no such $F_i \in \mathcal{F}$, we write $C_i = \mathsf{NULL}$ to indicate that we do not add a constraint; otherwise, C_i is the constraint $(F_i, (J_1^{(i)}, \ldots, J_k^{(i)}))$. The sampled instance $I \sim \mathcal{D}_s$ consists of all non-NULL constraints C_i. Given an assignment v, it will be convenient to write $v \models i$ to mean that $C_i \ne \mathsf{NULL}$ and $v \models C_i$.

Let $a(\mathcal{F}) := \liminf_{n\to\infty} \min_v \Pr[v \text{ does not satisfy } C_i \mid C_i \ne \mathsf{NULL}]$. That is, $a(\mathcal{F})$ is (close to, for sufficiently large n) the minimal achievable expected fraction of unsatisfied constraints (in an instance drawn from $\mathcal{D}_s$) over all fixed assignments v. Our lower bound roughly states that even for satisfiable instances, it is hard to satisfy much more than a $(1 - a(\mathcal{F}))$ fraction of constraints. Although the definition of $a(\mathcal{F})$ looks complicated, it is actually a fairly natural measure of the hardness of choosing a fixed assignment to satisfy a random constraint from $\mathcal{F}$. For example, it is not hard to see that $a(\mathcal{F}_{3-\mathrm{SAT}}) = 1/8$, and $a(\mathcal{F}_{\mathrm{k-LIN}}) = 1/2$. These are exactly the expected fraction of random constraints not satisfied by any fixed assignment. We will write a for $a(\mathcal{F})$ when $\mathcal{F}$ is clear from context.

Say that a constraint family $\mathcal{F}$ is *constant satisfiable* (resp. *constant unsatisfiable*) if there is $b \in [c]$ for which every $f \in \mathcal{F}$ has $f(b, \ldots, b) = 1$ (resp. $f(b, \ldots, b) = 0$).

▶ **Theorem 15.** *For all constraint families $\mathcal{F}$ that are not constant unsatisfiable, there is $r > 0$ such that for all $0 < \varepsilon < 1$, there are $(1-\varepsilon)$-approximate estimation oracles h_I such that for all oracle algorithms $\mathcal{A}^{h_I}$ making at most q queries to h_I, letting $I \sim \mathcal{D}_s$ and $v \leftarrow \mathcal{A}^{h_I}()$, for sufficiently large n,*

$$\Pr[\mathsf{SAT}_I(v)/\mathsf{SAT}_I \ge (1+\delta)(1-a)] \le q \exp(-(\delta^2 + \varepsilon^2) \cdot r \cdot n),$$

where $s = 1000k^2 \log(c) n/\varepsilon^2$.

In particular, for any small $\delta > 0$, for sufficiently large n, an arbitrary assignment satisfies a $1 - a - \delta$ fraction of constraints with overwhelming probability, but no algorithm can satisfy a $1 - a + \delta$ fraction of constraints unless it makes exponentially many queries.

A few additional remarks are in order. The theorem is vacuous for trivially satisfiable constraint families $\mathcal{F}$, since the assignment v mapping every variable to b satisfies all constraints, which implies $a(\mathcal{F}) = 0$. But by the same logic, no non-trivial lower bound on the approximation ratio is possible for such families. Fortunately, if $\mathcal{F}$ is not trivially satisfiable, $a(\mathcal{F})$ is strictly positive. To see this, fix an assignment v, and let b be the majority value of v. By assumption, there is $f^* \in \mathcal{F}$ such that $f^*(b, \ldots, b) = 0$. Sample a random constraint $(f, (j_1, \ldots, j_k))$. Independent of $(j_1, \ldots, j_k)$, over the random choice of P, we have that $f^*(P(x_{j_1}), \ldots, P(x_{j_k})) = 1$ with probability at least $1/c^k$. Conditional on this, f is chosen to be f^* with probability at least $1/2^{c^k}$. Hence f is chosen to be f^* with probability at least $1/c^k \cdot 1/2^{c^k}$. Suppose $n \geq ck$. For any assignment v, we have $v(x_{j_1}) = \cdots = v(x_{j_k}) = b$ with probability (over the random choice of $j_1, \ldots, j_k$ independent of P and f^*) at least $\Pi_{i=0}^{k-1}(n/c - i)/(n - i) \geq \Pi_{i=0}^{k-1}(k - i)/(ck - i)$ (the last expression corresponds to the case of $n = ck$ and balanced v). Thus, for $n \geq ck$, the random constraint is unsatisfied with probability at least $1/c^k \cdot 1/2^{c^k} \cdot \Pi_{i=0}^{k-1}(k - i)/(ck - i) > 0$; it follows $a(\mathcal{F}) > 0$.

The trivial unsatisfiability condition is slightly less natural; however, some similar condition is necessary for lower bounds. Consider the problem of 3-coloring a graph to maximize the number of edges with one green endpoint and one blue endpoint. This is clearly a CSP. Starting with all nodes colored red, one can color two nodes blue and green and use a heuristic to determine if there is an edge between them. Proceeding in this way, using $O(n^2)$ queries, one can recover the whole graph and (using unbounded computation) recover the optimal coloring. So, in our model, we cannot hope to rule out a branch-and-bound algorithm for such a CSP. This CSP is ruled out by trivial unsatisfiability, because coloring all nodes red makes the only constraint in the family (the blue-green constraint) output 0.

References

1 Yahav Alon and Michael Krivelevich. Random graph's Hamiltonicity is strongly tied to its minimum degree. *The Electronic Journal of Combinatorics*, pages 1–30, 2020. 14

2 Mihir Bellare and Shafi Goldwasser. The complexity of decision versus search. *SIAM Journal on Computing*, 23(1):97–119, 1994. 3, 7

3 Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh. *Handbook of satisfiability*, volume 336. IOS press, 2021. 6

4 William J. Cook. *In pursuit of the traveling salesman.* Princeton University Press, 2011. 6

5 Uriel Feige. On estimation algorithms vs approximation algorithms. In *FSTTCS*, 2008. 3

6 Uriel Feige and Shlomo Jozeph. Separation between estimation and approximation. In *ITCS*, 2015. 3

7 Alexander Golovnev, Siyao Guo, Spencer Peters, and Noah Stephens-Davidowitz. The (im)possibility of simple search-to-decision reductions for approximation problems, 2022. arXiv:TR22-141. 1, 14

8 Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963. 10

9 Gengran Hu and Yanbin Pan. Improvements on reductions among different variants of SVP and CVP. In *WISA*, 2013. 5

10 Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P Williamson. The approximability of constraint satisfaction problems. *SIAM Journal on Computing*, 30(6):1863–1920, 2001. 15

11 Richard J. Lipton and Kenneth W. Regan. Branch and bound—Why does it work? https://rjlipton.wpcomstaging.com/2012/12/19/branch-and-bound-why-does-it-work/, December 2012. 7

12 Colin McDiarmid. Concentration. In *Probabilistic methods for algorithmic discrete mathematics*, pages 195–248. Springer, 1998. 10

13 Daniele Micciancio. Efficient reductions among lattice problems. In *SODA*, 2008. 4

14 David R. Morrison, Sheldon H. Jacobson, Jason J. Sauppe, and Edward C. Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, February 2016. doi:10.1016/j.disopt.2016.01.005. 2, 7

15 Noah Stephens-Davidowitz. Dimension-preserving reductions between lattice problems. Unpublished manuscript, 2015. URL: http://noahsd.com/latticeproblems.pdf. 4

16 Noah Stephens-Davidowitz. Search-to-decision reductions for lattice problems with approximation factors (slightly) greater than one. In *APPROX*, 2016. 4, 5

17 Zoya Svitkina and Lisa Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. *SIAM Journal on Computing*, 40(6), 2011. 8

18 Jan Vondrák. Symmetry and approximability of submodular maximization problems. In *FOCS*, 2009. 8

A Omitted Proofs

A.1 Proof of Theorem 12

Proof. $\mathcal{G}$ is defined by independently setting the weight of each edge e as follows:

$$w(e) = \begin{cases} 1 & \text{w.p. } 1/2\,, \\ \omega := 1 + (\gamma - 1)n/t & \text{w.p. } 1/2\,. \end{cases}$$

The following is the definition of v_k:

$$v_k = \begin{cases} n/\gamma & \text{if } k = 0\,, \\ n & \text{if } 1 \le k \le t\,, \\ k(\omega+1)/(\gamma+1) + n - k & \text{if } t < k < n - t\,, \\ \big(k/2 + n/(\gamma+1)\big)(\omega+1)/2 & \text{if } k \ge n - t\,. \end{cases}$$

The definitions of v_0 and v_{n-1} satisfy the first item of the theorem statement as $v_0 = n/\gamma$, and

$$v_{n-1} \ge (n-1)(\omega+1)/4 \ge (n-1)(\gamma-1)n/(4t) \ge (\gamma-1)n^2/(5t)\,.$$

It remains to bound from above $\Pr_{G\sim\mathcal{G}}\big[\mathrm{OPT}(G,p) \notin [v_k, \gamma v_k]\big]$ for every fixed $p \in P_n^k$. By the Chernoff-Hoeffding bound (Lemma 4), for every $\varepsilon \in (0,1)$, p contains from $(1-\varepsilon)k/2$ to $(1+\varepsilon)k/2$ edges of weight ω with probability at least $1 - 2e^{-\varepsilon^2 k/6}$. Therefore,

$$\Pr_{G\sim\mathcal{G}}\big[w(p) \in [(1-\varepsilon)k(\omega+1)/2,\ (1+\varepsilon)k(\omega+1)/2]\big] \ge 1 - O\big(e^{-\varepsilon^2 k/6}\big)\,. \qquad (2)$$

Let $G \sim \mathcal{G}$, and let G' be the graph consisting of the vertices of G not belonging to the path p, and the edges of G of weight 1. Note that G' is a uniformly random unweighted graph with $n-k-1$ vertices. By Lemma 11, G' contains a Hamiltonian cycle with probability $1 - 2^{-(n-k-1)(1-o(1))}$. With probability at least $1 - (3/4)^{n-k-1}$ the endpoints of p are connected by edges of weight 1 to a pair of consecutive points of the Hamiltonian cycle in G'. Thus, with probability at least $1 - O\big((3/4)^{n-k}\big)$, p can be extended to a TSP cycle in G by taking a Hamiltonian path in G', removing an edge from it, and connecting the two endpoints to the endpoints of p by edges of weight 1. Hence,

$$\Pr_{G\sim\mathcal{G}}[\mathrm{OPT}(G,p) = w(p) + n - k] \geq 1 - O\big((3/4)^{n-k}\big)\,. \tag{3}$$

Now we consider the following four cases.

- $k = 0$. For the distribution $\mathcal{G}$, Lemma 11 implies that for every $p \in P_n^0$, $\mathrm{OPT}(G,p) = \mathrm{OPT}(G) = n$ with probability $1 - 2^{-n+o(n)} \geq 1 - O(e^{-\delta^2 t/30})$.
- $1 \leq k \leq t$. For each path $p \in P_n^k$, $\mathrm{OPT}(G,p) \geq n = v_k$, and, by (3), with probability at least $1 - O\big((3/4)^{n-k}\big) \geq 1 - O\big((3/4)^{n/2}\big) \geq 1 - O(e^{-\delta^2 t/30})$, we have that

  $$\mathrm{OPT}(G,p) = w(p) + n - k \leq k\omega + n - k \leq \gamma n = \gamma v_k\,.$$

- $t < k < n - t$. From (3), with probability $1 - O\big((3/4)^t\big)$, $\mathrm{OPT}(G,p) = w(p) + n - k$. By setting $\varepsilon = \delta$, from (2), with probability at least $1 - O(e^{-\delta^2 t/6})$, $w(p) \in [k(\omega+1)/(\gamma+1),\ \gamma k(\omega+1)/(\gamma+1)]$. Together these bounds give us that

  $$\mathrm{OPT}(G,p) = w(p) + n - k \in [v_k, \gamma v_k]$$

  with probability at least $1 - O\big((3/4)^t\big) - O(e^{-\delta^2 t/6}) \geq 1 - O(e^{-\delta^2 t/30})$.
- $k \geq n - t$. From (2) with $\varepsilon = 1/2 - \frac{n}{k(\gamma+1)} \geq 1/2 - 2/(\gamma+3)$, with probability $1 - O(e^{-(1-4/(\gamma+3))^2 k/24}) \geq 1 - O(e^{-\delta^2 t/30})$ (where we used that $t \leq \delta n/2$ and $k \geq n(1-\delta/2)$), we have that

  $$\begin{aligned} w(p) &\in [(1-\varepsilon)k(\omega+1)/2,\ (1+\varepsilon)k(\omega+1)/2] \text{ and} \\ \mathrm{OPT}(G,p) &\in [w(p),\ w(p) + \omega(n-k)] \\ &\subseteq [(1-\varepsilon)k(\omega+1)/2,\ (\omega+1)(n-(1-\varepsilon)k/2)] \\ &= \big[\big(k/2 + n/(\gamma+1)\big)(\omega+1)/2,\ \big(2n - \big(k/2 + n/(\gamma+1)\big)\big)(\omega+1)/2\big] \\ &\subseteq [v_k, \gamma v_k] \end{aligned}$$

  for every $k \geq n - t \geq n(1-\delta) = 2n/(\gamma+1)$. ◀

A.2 Proof of Theorem 14

We prove the first part of the theorem using Corollary 7. For this, we first denote by $\mathcal{G}$ the distribution of graphs from Theorem 12, and by D the set of all cycles C_n. We define the distribution $\mathcal{D}$ of functions $f_G\colon D \to \mathbb{R}_{\geq 0}$ computing the length of a given cycle in $G \sim \mathcal{G}$. For a path $p \in P_n^k$, we denote by $S_p \subseteq C_n$ the set of cycles containing the path p, and we define

$$\mathcal{S} = \{S_p\colon p \in P_n^k,\ 0 \leq k \leq n-1\}\,.$$

It is easy to see that $C_n \in \mathcal{S}$ and for every cycle $c \in C_n$, $\{c\} \in \mathcal{S}$. Now for a set $S_p \in \mathcal{S}$ corresponding to a path $p \in P_n^k$, we define $g(S_p) = \gamma v_k$. By Theorem 12 with $t = 30\ell/\delta^2$, for each S_p, $\Pr[g(S_p)/\gamma \leq \min_{c\in S_p} f(c) \leq g(S_p)] \geq 1 - p$ for $p = O\big(e^{-\delta^2 t/30}\big) = O\big(e^{-\ell}\big)$. Now we apply Corollary 7 to our choices of f and g, and

$$\gamma' := \gamma^{-1} \cdot \min_{c\in D} \frac{g(\{c\})}{g(D)} \geq \gamma^{-1} \cdot v_{n-1}/v_0 \geq (\gamma-1)\delta^2 n/(150\ell)\,,$$

and have that

$$\Pr_{G\sim\mathcal{G}}[c \leftarrow \mathcal{A}^{h_G}(),\ w(c) \le \gamma' \operatorname{OPT}(G)] \le (q+2)\cdot O\big(e^{-\ell}\big) \le O\big(q\cdot e^{-\ell}\big)\,.$$

To prove the "furthermore" part, we consider the following algorithm. Let G be an input graph on n vertices, and let n be a multiple of $\ell-1$. The algorithm first queries $h_G(p)$ for each path $p\in P_n^{\ell-1}$ of length $\ell-1$. Then the algorithm returns a cycle $c=(c_0,\dots,c_{n-1})\in C_n$ that minimizes the sum

$$H(c) := h_G(c_0,\dots,c_{\ell-1}) + h_G(c_{\ell-1},\dots,c_{2(\ell-1)}) + \ldots + h_G(c_{n+1-\ell},\dots,c_{n-1},c_0)\,.$$

It is easy to see that the algorithm makes $|P_n^{\ell-1}| = \binom{n}{\ell}\cdot\ell!$ queries to h_G.

Since for every path $p\in P_n^{\ell-1}$, $w(p)\le h_G(p)$, the weight of the resulting cycle does not exceed $H(c)$. Now it remains to show that $\min_{x\in C_n} H(x) \le \gamma n \operatorname{OPT}(G)/(\ell-1)$. To this end, consider an optimal TSP cycle $c'\in C_n$ in G. Since for every subpath $p\in P_n^{\ell-1}$ of c', $h_G(p)\le\gamma\operatorname{OPT}(G)$, we have that

$$H(c) \le H(c') \le \gamma \operatorname{OPT}\cdot n/(\ell-1)\,.$$

A.3 Proof of Theorem 15

First, in the following lemma, we show that the "optimal satisfaction probability" $\rho_w(P)$ is concentrated as a function of P.

▶ **Lemma 16.**

$$\Pr\left[|\rho_w(P) - \mathbb{E}[\rho_w(P)]| \ge (\varepsilon/10)\,\mathbb{E}[\rho_w(P)]\right] \le 2\exp\Big(-n\varepsilon^2\,\mathbb{E}[\rho_w(P)]^2/(200k^2)\Big).$$

Proof. Notice that for all assignments v, P and indices $j\in[n]$, letting E be the event $(j=J_1^{(i)})\vee\cdots\vee(j=J_k^{(i)})$, we have

$$\rho_v(P) = \Pr[E]\cdot\Pr[v\models i \mid E, P] + \Pr[\overline{E}]\cdot\Pr[v\models i\mid \overline{E},P]$$

Observe that $\Pr[E]=k/n$, and $\Pr[v\models i\mid\overline{E},P]$ does not depend on $P(j)$. Thus, defining

$$v^{\oplus j} = \begin{cases}\neg v(x) & x=j\\ v(x) & x\ne j,\end{cases}$$

we have

$$|\rho_v(P) - \rho_v(P^{\oplus j})| \le k/n.$$

It follows that for all partial assignments w,

$$|\rho_w(P) - \rho_w(P^{\oplus j})| \le k/n.$$

Indeed, if no v improves by more than k/n in success probability, the maximum success probability over all v extending w cannot improve by more than k/n either. The desired result follows by McDiarmid's inequality (Lemma 5). ◀

It follows from the definition of $a=a(\mathcal{F})$ and the Chernoff-Hoeffding bound that there is a constant $r'>0$ such that for all assignments v:

$$\Pr_{I\sim\mathcal{D}}[\mathsf{SAT}_I(v)/\mathsf{SAT}_I \ge (1+\delta)(1-a)] = \Pr_{I\sim\mathcal{D}}[\mathsf{SAT}_I(v)\ge(1+\delta)(1-a)m(I)] \le \exp(-r'\cdot\delta^2\cdot n).$$

(Informally, any fixed assignment is unlikely to satisfy much more than a $1-a$ fraction of constraints.)

Notice that, for $x, y > 0$, if $x \in [(1-\varepsilon/3)y, (1+\varepsilon/3)y]$, then $x \in [(1-\varepsilon)z, z]$, where $z = (1+\varepsilon/3)y$. Thus, by the useless oracle corollary (Corollary 7), Theorem 15 will follow immediately from the claim below.

▷ Claim 17. Given a partial assignment w, let $\rho_w(P) = \max_{v:v \text{ extends } w} \Pr[v \models i \mid P]$. There is a constant $r > 0$ such that

$$\Pr_{I \sim \mathcal{D}_m}\left[|\mathsf{SAT}_I(w) - m\,\mathbb{E}[\rho_w(P)]| \geq (\varepsilon/3)m\,\mathbb{E}[\rho_w(P)]\right] \leq \exp(-r \cdot \varepsilon^2 \cdot n).$$

Proof. Fix a partial assignment w. We will argue that $\mathsf{SAT}_I(w)$ is concentrated conditional on P. Fixing P and an assignment v, $\mathsf{SAT}_I(v)$ is the sum of m independent coin tosses $\mathbb{1}(v \models i)$ with success probability $\rho_v(P)$. For the optimal v^* extending w, we have $\mathbb{E}[\mathsf{SAT}_I(v)] = m\rho_w(P)$, and the Chernoff-Hoeffding bound gives

$$\Pr\left[|\mathsf{SAT}_I(v^*) - m\rho_w(P)| \geq (\varepsilon/10)m\rho_w(P) \mid P\right] \leq 2\exp(-\varepsilon^2 m\rho_w(P)/600). \tag{4}$$

Moreover, for any v extending w, $\mathsf{SAT}_I(v)$ is the sum of m independent coin tosses with a smaller success probability $\rho_v(P) \leq \rho_w(P)$. Hence the upper bound of (4) holds, namely

$$\Pr[\mathsf{SAT}_I(v) \geq (1+\varepsilon/10)m\rho_w(P) \mid P] \leq \exp(-\varepsilon^2 m\rho_w(P)/600). \tag{5}$$

Recall that $\mathsf{SAT}_I(w)$ is the maximum over all v extending w of $\mathsf{SAT}_I(v)$. Inequality (4) is thus a high-probability lower bound on $\mathsf{SAT}_I(w)$, and (5) combined with a union bound over the (at most c^n) assignments v extending w gives a high-probability upper bound. Specifically, plugging in $m = \frac{1000k^2\log(c)n}{\varepsilon^2}$, we have

$$\begin{aligned}&\Pr\left[|\mathsf{SAT}_I(w) - m\rho_w(P)| \geq (\varepsilon/10)m\rho_w(P) \mid P\right]\\ &\leq (c^n+2)\exp(-\varepsilon^2 m\rho_w(P)/600)\\ &\leq \exp(-n\rho_w(P)/(100k^2)).\end{aligned} \tag{6}$$

Third, combining Inequality (6) with Lemma 16 yields

$$\Pr\left[|\mathsf{SAT}_I(w) - m\,\mathbb{E}[\rho_w(P)]| \leq (\varepsilon/3)m\,\mathbb{E}[\rho_w(P)]\right] \leq 2\exp(-n\varepsilon^2\,\mathbb{E}[\rho_w(P)]^2/(200k^2)), \tag{7}$$

where we have used $\mathbb{E}[\rho_w(P)] \leq 1 \Rightarrow \mathbb{E}[\rho_w(P)]^2 \leq \mathbb{E}[\rho_w(P)]$.

To finish the proof, we show that $\mathbb{E}[\rho_w(P)]$ is bounded below by a constant independent of n and w, if $n \geq ck$. Since $\mathcal{F}$ is not trivially unsatisfiable, for all $b \in [c]$, there is $f^* \in \mathcal{F}$ with $f(b, \ldots, b) = 1$. As we argued before with trivial satisfiability, when a random constraint is sampled, it is non-NULL and has constraint function f^* with probability at least $1/c^k \cdot 1/2^{c^k}$. Letting the variable indices in the constraint be $j_1, \ldots, j_k$, again for any assignment v (for $n \geq ck$) we have $v(x_{j_1}) = \cdots = v(x_{j_k}) = b$ with probability at least $\Pi_{i=0}^{k-1}(k-i)/(ck-i)$ (again, the last expression corresponds to the case of $n = ck$ and balanced v). Thus the constraint is satisfied with probability at least $1/c^k \cdot 1/2^{c^k} \cdot \Pi_{i=0}^{k-1}(k-i)/(ck-i)$, as needed. ◁

Approximating Red-Blue Set Cover and Minimum Monotone Satisfying Assignment

Eden Chlamtáč
Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Yury Makarychev
Toyota Technological Institute at Chicago (TTIC), IL, USA

Ali Vakilian
Toyota Technological Institute at Chicago (TTIC), IL, USA

Abstract

We provide new approximation algorithms for the Red-Blue Set Cover and Circuit Minimum Monotone Satisfying Assignment (MMSA) problems. Our algorithm for Red-Blue Set Cover achieves $\tilde{O}(m^{1/3})$-approximation improving on the $\tilde{O}(m^{1/2})$-approximation due to Elkin and Peleg (where m is the number of sets). Our approximation algorithm for MMSA$_t$ (for circuits of depth t) gives an $\tilde{O}(N^{1-\delta})$ approximation for $\delta = \frac{1}{3}2^{3-\lceil t/2 \rceil}$, where N is the number of gates and variables. No non-trivial approximation algorithms for MMSA$_t$ with $t \geq 4$ were previously known.

We complement these results with lower bounds for these problems: For Red-Blue Set Cover, we provide a nearly approximation preserving reduction from Min k-Union that gives an $\tilde{\Omega}(m^{1/4-\varepsilon})$ hardness under the Dense-vs-Random conjecture, while for MMSA we sketch a proof that an SDP relaxation strengthened by Sherali–Adams has an integrality gap of $N^{1-\varepsilon}$ where $\varepsilon \to 0$ as the circuit depth $t \to \infty$.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Theory of computation → Circuit complexity

Keywords and phrases Red-Blue Set Cover Problem, Circuit Minimum Monotone Satisfying Assignment (MMSA) Problem, LP Rounding

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.11

Category APPROX

Related Version *Full Version*: `https://arxiv.org/abs/2302.00213`

Funding *Eden Chlamtáč*: The work was done while the author was visiting and supported by TTIC.
Yury Makarychev: Supported by NSF awards CCF-1955173 and CCF-1934843.

1 Introduction

In this paper, we study two problems, *Red-Blue Set Cover* and its generalization *Circuit Minimum Monotone Satisfying Assignment.* Red-Blue Set Cover, a natural generalization of Set Cover, was introduced by Carr et al. [5]. Circuit Minimum Monotone Satisfying Assignment, a problem more closely related to Label Cover, was introduced by Alekhnovich et al. [2] and Goldwasser and Motwani [12].

▶ **Definition 1.** *In Red-Blue Set Cover, we are given a universe of $(k+n)$ elements U partitioned into disjoint sets of red elements (R) of size n and blue elements (B) of size k, that is $U = R \cup B$ and $R \cap B = \emptyset$, and a collection of sets $\mathcal{S} := \{S_1, \cdots, S_m\}$. The goal is to find a sub-collection of sets $\mathcal{F} \subseteq \mathcal{S}$ such that the union of the sets in $\mathcal{F}$ covers all blue elements while minimizing the number of covered red elements.*

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 11; pp. 11:1–11:19
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Besides Red-Blue Set Cover, we consider the Partial Red-Blue Set Cover *problem in which we are additionally given a parameter $\hat{k}$, and the goal is cover at least $\hat{k}$ blue elements while minimizing the number of covered red elements.*

▶ **Definition 2.** *The Circuit Minimum Monotone Satisfying Assignment problem of depth t, denoted as $MMSA_t$, is as follows. We are given a circuit C of depth t over Boolean variables $x_1, \ldots, x_n$. Circuit C has AND and OR gates: all gates at even distances from the root (including the output gate at the root) are AND gates; all gates at odd distances are OR gates. The goal is to find a satisfying assignment with the minimum number of variables x_i set to 1 (true).*

Note that C computes a monotone function and the assignment of all ones is always a feasible solution. Though the definitions of the problems are quite different, Red-Blue Set Cover and MMSA_t are closely related. Namely, Red-Blue Set Cover is equivalent to MMSA_3.[1] The correspondence is as follows: variables $x_1, \ldots, x_n$ represent red elements; AND gates in the third layer represent sets $S_1, \ldots, S_m$; OR gates in the second layer represent blue elements. The gate for a set S_j is connected to OR gates representing blue elements of S_j and variables x_i representing red elements of S_j. It is easy to see that an assignment to $x_1, \ldots, x_n$ satisfies the circuit if and only if there exists a sub-family $\mathcal{F} \subseteq \mathcal{S}$ that covers all the blue elements, and only covers red elements corresponding to variables x_i which are assigned 1 (but not necessarily all of them).

Background. Red-Blue Set Cover and its variants are related to several well-known problems in combinatorial optimization including *group Steiner* and *directed Steiner* problems, *minimum monotone satisfying assignment* and *ymmetric minimum label cover.* Arguably, the interest to the general MMSA_t problem is mostly motivated by its connection to complexity and hardness of approximation.

The Red-Blue Set Cover has applications in various settings such as *anomaly detection, information retrieval* and notably in *learning of disjunctions* [5]. Learning of disjunctions over $\{0,1\}^m$ is one of the basic problems in the PAC model. In this problem, given an arbitrary distribution $\mathcal{D}$ over $\{0,1\}^m$ and a target function $h^* : \{0,1\}^m \to \{-1,+1\}$ which denotes the true labels of examples, the goal is to find the best disjunction $f^* : \{0,1\}^m \to \{-1,+1\}$ with respect to $\mathcal{D}$ and h^* (i.e., $f^*(x)$ computes a disjunction of a subset of coordinates of x). The problem of learning disjunctions can be formulated as an instance of the (Partial) Red-Blue Set Cover problem [4]: we can think of the positive examples as blue elements (i.e., $h^*(x) = 1$) and the negative examples as red elements (i.e., $h^*(x) = -1$). Then, we construct a set S_i corresponding to each coordinate i and the set S_i contains an example x if the i-th coordinate of x is equal to 1. Let $C \subset \{S_1, \cdots, S_m\}$. Then, the disjunction f_C corresponding to C, i.e., $f_C(x) := \bigvee_{S_i \in C} x_i$, outputs one on an example x if in the constructed Red-Blue Set Cover instance, the element corresponding to x is covered by sets in C.

As we observe in Section 5, Red-Blue Set Cover is also related to the Min k-Union problem which is a generalization of Densest k-Subgraph [8]. In Min k-Union, given a collection of m sets $\mathcal{S}$ and a target number of sets k, the goal is to pick k sets from $\mathcal{S}$ whose union has the minimum cardinality. Notably, under a hardness assumption, which is an extension of the "Dense vs Random" conjecture for Densest k-Subgraph to hypergraphs, Min k-Union

[1] Also observe that MMSA_2 is equivalent to Set Cover.

cannot be approximated better than $\tilde{\Omega}(m^{1/4-\varepsilon})$ [9]. In this paper, we prove a hardness of approximation result for Red-Blue Set Cover by constructing a reduction from Min k-Union to Red-Blue Set Cover.

1.1 Related work

Carr et al. [5] formulated the Red-Blue Set Cover problem and presented a $2\sqrt{m}$-approximation algorithm for the problem when every set contains only one blue element. Later, Elkin and Peleg [11] showed that it is possible to obtain a $2\sqrt{m\log(n+k)}$ approximation in the general case of the problem. This remained the best known upper bound for Red-Blue Set Cover prior to our work. No non-trivial algorithms for MMSA_t were previously known for any $t \geq 4$.

On the hardness side, Dinur and Safra [10] showed that MMSA_3 is hard to approximate within a factor of $O(2^{\log^{1-\epsilon} m})$ where $\epsilon = 1/\log\log^c m$ for any constant $c < 1/2$, if $P \neq NP$. As was observed by Carr et al. [5], this implies a factor of $O(2^{\log^{1-\epsilon} m})$ hardness for Red-Blue Set Cover as well. The hardness result holds even for the special case of the problem where every set contains only one blue and two red elements.

Finally, Charikar et al. [7] gave a lower bound on a variant of MMSA in which the circuit depth t is not fixed. Assuming a variant of the Dense-vs-Random conjecture, they showed that for every $\varepsilon > 0$, the problem does not admit an $O(n^{1/2-\varepsilon})$ approximation, where n is the number of variables, and an $O(N^{1/3-\varepsilon})$ approximation, where N is the total number of gates and variables in the circuit.

Learning of Disjunctions. While algorithms for Red-Blue Set Cover return a disjunction with no error on positive examples, i.e., it covers all "blue" elements, it is straightforward to make those algorithms work for the case with two-sided error. A variant of the problem with a two-sided error is formally defined as *Positive–Negative Partial Set Cover* [15] where the author showed that a $f(m,n)$-approximation for Red-Blue Set Cover implies a $f(m+n,n)$-approximation for Positive-Negative Partial Set Cover. Our result also holds for Partial Red-Blue Set Cover and a c-approximation for Partial Red-Blue Set Cover can be used to output a c-approximate solution of Positive-Negative Partial Set Cover.

Awasthi et al. [4] designed an $O(n^{1/3+\alpha})$-approximation for any constant $\alpha > 0$. While the proposed algorithm of Awasthi et al. is an *agnostic learning* of disjunctions (i.e., the solution is not of form of disjunctions), employing an approximation algorithm of Red-Blue Set Cover, produces a disjunction as its output (i.e., the algorithms for Red-Blue Set Cover are *proper* learners).

Geometric Red-Blue Set Cover. The problem has been studied extensively in geometric models. Chan and Hu [6] studied the setting in which R and B are sets of points in $\mathbb{R}^2$ and $\mathcal{S}$ is a collection of unit squares. They proved that the problem still remains NP-hard in this restricted instance and presented a PTAS for this problem. Madireddy and Mudgal [13] designed an $O(1)$-approximation algorithm for another geometric variant of the problem, in which sets are unit disks. The problem has also been studied in higher dimensions with hyperplanes and axis-parallel objects as the sets [3, 14, 1].

1.2 Our Results

In this paper, we present new approximation algorithms for Red-Blue Set Cover, MMSA_4, and general MMSA_t. Additionally, we offer a new conditional hardness of approximation result for Red-Blue Set Cover. We also discuss the integrality gap of a basic SDP relaxation of MMSA_t strengthened by Sherali–Adams when $t \to \infty$.

We start by describing our result for Red-Blue Set Cover.

▶ **Theorem 3.** *There exists an $O(m^{1/3} \log^{4/3} n \log k)$-approximation algorithm for Red-Blue Set Cover where m is the number of sets, n is the number of red elements, and k is the number of blue elements.*

As we demonstrate later, our algorithm also works for the Partial Red-Blue Set Cover. Our approach partitions the instance into subinstances in which all sets have a bounded number of red elements, say between r and $2r$, and each red element appears in a bounded number of sets. Utilizing the properties of this partition, we show that we can always find a small collection of sets that preserves the right ratio of red to blue elements in order to make progress towards an $\tilde{O}(m^{1/3})$-approximation algorithm.[2] Then, by applying the algorithm iteratively until all blue elements are covered, we obtain the guarantee of Theorem 3. In each iteration, our analysis guarantees the feasibility of a local LP relaxation for which a simple randomized rounding obtains the required ratio of blue to red vertices.

Now we describe our results for the MMSA problem.

▶ **Theorem 4.** *There exists an $\tilde{O}(N^{1/3})$-approximation algorithm for MMSA$_4$, where N is the total number of gates and variables in the input instance.*

Our algorithm for MMSA$_4$ is inspired by our algorithm for MMSA$_3$, though due to the complexities of the problem, the algorithm is significantly more involved. In particular, there does not seem to be a natural preprocessing step analogous to the partition we apply for Red-Blue Set Cover, and so we need to rely on a higher-moment LP relaxation and a careful LP-based partition which is built into the algorithm.

▶ **Theorem 5.** *Let $t \geq 4$. Define $\delta = \frac{1}{3} \cdot 2^{3-\lceil t/2 \rceil}$. There exists an $\tilde{O}(N^{1-\delta})$-approximation algorithm for MMSA$_t$ where N is the total number of gates and variables in the input instance.*

Our algorithm for general MMSA$_t$ applies a recursion on the depth t, with our algorithms for Red-Blue Set Cover and MMSA$_4$ as the basis of the recursion. Each recursive step relies on an initially naive LP relaxation to which we add constraints as calls to the algorithm for smaller depth MMSA reveal new violated constraints.

We complement our upper bound for Red-Blue Set Cover with a hardness of approximation result.

▶ **Theorem 6.** *Assuming the Hypergraph Dense-vs-Random Conjecture, for every $\varepsilon > 0$, no polynomial-time algorithm achieves better than $O(m^{1/4-\varepsilon}/\log^2 k)$ approximation for Red-Blue Set Cover where m is the number of sets and k is the number of blue elements.*

To show the hardness, we present a reduction from Min k-Union to Red-Blue Set Cover that preserves the approximation up to a factor of polylog(k). Then, the hardness follows from the standard conjectured hardness of Min k-Union [9]. In our reduction, all elements of the given instance of Min k-Union are considered as the red elements in the constructed instance for Red-Blue Set Cover and we further complement each set with a sample size of $O(\log k)$ (with replacement) from a ground set of blue elements of size k. We prove that this reduction is approximation-preserving up to a factor of polylog(k).

[2] Here, we abuse the $\tilde{O}$ notation to hide polylog factors of n, k.

Organization. In Section 2, we restate Red-Blue Set Cover and introduce some notation. In Section 3, we present our algorithm for Red-Blue Set Cover. We adapt this algorithm for Partial Red-Blue Set Cover in Appendix A. Then, in Section 4 we give the algorithm for MMSA_t with $t \geq 5$. This algorithm relies on the algorithm for MMSA_4, which we describe later in Section 6. We present a reduction from Min k-Union to Red-Blue Set Cover, which yields a hardness of approximation result for Red-Blue Set Cover, in Section 5. The discussion on hardness of the general MMSA_t problem is deferred to the full version of the paper.

2 Preliminaries

To simplify the description and analysis of our approximation algorithm for Red-Blue Set Cover, we restate the problem in graph-theoretic terms. Essentially we restate the problem as MMSA_3. Specifically, we think of a Red-Blue Set Cover instance as a tripartite graph (B, J, R, E) in which all edges (E) are incident on J and either B or R. The vertices in J represent the set indices, and their neighbors in B (resp. R) represent the blue (resp. red) elements in these sets. Thus, our goal is to find a subset of the vertices in J that is a dominating set for B and has a minimum total number of neighbors in R. For short, we will denote the cardinality of these different vertex sets by $k = |B|$, $m = |J|$, and $n = |R|$.

Similarly, we think of a MMSA_4 instance as a tuple (B, J, R, S, E). Here, B, J, and R represent gates in the second, third, and fourth layers of the circuit (where layer i consists of the gates at distance $i - 1$ from the root), respectively; S represents the variables; E represent edges between gates/variables. Combinatorially, the goal is to obtain a subset of J as above, along with a minimum dominating set in S for the red neighbors (in R) of our chosen subset of J.

Notation. We use $\Gamma(\cdot)$ to represent neighborhoods of vertices, and for a vertex set U, we use $\Gamma(U)$ to denote the union of neighborhoods of vertices in U, that is $\bigcup_{u\in U} \Gamma(u)$. We also consider restricted neighborhoods, which we denote by $\Gamma_T(u) := \Gamma(u) \cap T$ or $\Gamma_T(U) = \Gamma(U) \cap T$. We will refer to the cardinality of such a set, i.e. $|\Gamma_T(u)|$ as the *T-degree* of u.

▶ Remark 7. Note that, for every set index $j \in J$, the set $\Gamma(j)$ is simply the set S_j in the set system formulation of the problem, and the set $\Gamma_R(j)$ (resp. $\Gamma_B(j)$) is simply the subset of red (resp. blue) elements in the set with index j. Similarly, $\Gamma_J(i)$ consists of indices j representing those sets S_j that contain element i, for any $i \in R \cup B$.

For Red-Blue Set Cover algorithms, we introduce a natural notion of progress:

▶ **Definition 8.** *We say that an algorithm for Red-Blue Set Cover* makes progress towards an $O(A \cdot \log k)$-approximation *if, given an instance with an optimum solution containing* OPT *red elements, the algorithm finds a subset $\hat{J} \subseteq J$ such that* $\frac{|\Gamma_R(\hat{J})|}{|\Gamma_B(\hat{J})|} \leq A \cdot \frac{\text{OPT}}{|B|}$.

It is easy to see that if we have an algorithm which makes progress towards an A-approximation, then we can run this algorithm repeatedly (with decreasing $|B|$ parameter, where initially $|B| = k$) until we cover all blue elements and obtain an $O(A \cdot \log k)$-approximation. For brevity, all logarithms are implicitly base 2 unless otherwise specified, that is, we write $\log(\cdot) \equiv \log_2(\cdot)$.

3 Approximation Algorithm for Red-Blue Set Cover

We begin by excluding a small number of red elements, and binning the sets J into a small number of bins with uniform red-degree. For an $O(A)$-approximation, the goal will be to exclude at most $A \cdot \mathrm{OPT}$ red elements (we may guess the value of OPT by a simple linear or binary search). This is handled by the following lemma:

▶ **Lemma 9.** *There is a polynomial time algorithm, which, given an input (B, J, R, E) and parameter n_0, returns a set of at most $\log n$ pairs (J_α, R_α) with the following properties:*

- *The sets $J_\alpha \subseteq J$ partition the set J.*
- *The sets $R_\alpha \subseteq R$ form a nested collection of sets, and the smallest among them (i.e., their intersection) has cardinality at least $n - n_0$. That is, at most n_0 red elements are excluded by any of these sets.*
- *For every α there is some r_α such that every set $j \in J_\alpha$ has R_α-degree (or restricted red set size) $|\Gamma_{R_\alpha}(j)| \in [r_\alpha, 2r_\alpha]$,*
- *and for every α, every red element $i \in R_\alpha$ has J_α-degree at most (that is, the number of red sets in J_α that contain i) $|\Gamma_{J_\alpha}(i)| \leq \frac{2mr_\alpha \log n}{n_0}$.*

Proof. Consider the following algorithm:

- Let r be the maximum red-degree (i.e., $\max_{j \in J} \deg_R(j)$).
- Repeat the following while $J \neq \emptyset$:
  - Delete the top $n_0 / \log n$ J-degree red elements from R, along with their incident edges.
  - After this deletion, let $J' = \{j \in J \mid \deg_R(j) \in [r/2, r]\}$.
  - If J' is non-empty, add the current pair (J', R) to the list of output pairs (excluding all elements deleted from R so far).
  - Remove the sets in J' from J (along with incident edges) and let $r \leftarrow r/2$.

By the decrease in r, it is easy to see that this partitions J into at most $\log n$ sets (or more specifically, log of the initial maximum red set size, $\max_{j \in J} \deg_R(j)$). Also note that at the beginning of each iteration, all red sets have size at most r, and so there are at most mr edges to R, and the top $n_0 / \log n$ J-degree red elements will have average degree (and in particular minimum degree) at most $mr/(n_0/\log n)$. Thus, after removing these red elements, all remaining red elements will have J-degree (and in particular J'-degree) at most the required bound of $\frac{2mr_\alpha \log n}{n_0}$ where $r_\alpha = r/2$. Finally, since there at most $\log n$ iterations, the total number of red elements removed across all iterations is at most n_0. ◀

Our algorithm works in iterations, where at every iteration, some subset of blue elements is covered and removed from B. However, nothing is removed from J or R. Thus the above lemma applies throughout the algorithm. Note that for an optimum solution $J_{\mathrm{OPT}} \subseteq J$, for at least one of the subsets J_α in the above partition, the sets in $J_{\mathrm{OPT}} \cap J_\alpha$ must cover at least a $(1/\log n)$-fraction of B. Thus, to achieve an $O(A)$ approximation, it suffices to apply the above lemma with parameter $n_0 = \mathrm{OPT} \cdot A$, and repeatedly make progress towards an A-approximation within one of the subgraphs induced on (B, J_α, R_α). We will only pay at most another $\mathrm{OPT} \cdot A$ in the final analysis by restricting our attention to these subinstances.

Let us fix some optimum solution $J_{\mathrm{OPT}} \subseteq J$ in advance. For any α in the above partition, let $J^\alpha_{\mathrm{OPT}} = J_\alpha \cap J_{\mathrm{OPT}}$ be the collection of sets in J_α that also belong to our optimum solution, and let $B_\alpha = \Gamma_B(J^\alpha_{\mathrm{OPT}})$ be the blue elements covered by the sets in J^α_{OPT}. Note that every blue element must belong by the feasibility of J_{OPT} to at least one B_α. In this context we can show the following:

▶ **Lemma 10.** *For any α in the partition described in Lemma 9, there exists a red element $i_0 \in R_\alpha$ such that its optimum J_α-restricted neighbors $\Gamma_{J^\alpha_{\mathrm{OPT}}}(i_0)$ cover at least $|B_\alpha| r_\alpha/\mathrm{OPT}$ blue elements.*

Proof. Consider the following subgraph of $(B_\alpha, J^\alpha_{\mathrm{OPT}}, E(B_\alpha, J^\alpha_{\mathrm{OPT}}))$. For every blue element $\ell \in B_\alpha$, retain exactly one edge to J^α_{OPT}. Let $\hat{F}$ be this set of edges.

Thus the blue elements B_α have at least $|B_\alpha| r_\alpha$ paths through $\hat{F} \times E(J^\alpha_{\mathrm{OPT}}, R_\alpha)$ to the red neighbors of J^α_{OPT} in R_α. Since there are at most OPT such red neighbors, at least one of them, say $i_0 \in \Gamma_{R_\alpha}(J^\alpha_{\mathrm{OPT}})$, must be involved in at least a $1/\mathrm{OPT}$ fraction of these paths. That is, at least $|B_\alpha| r_\alpha/\mathrm{OPT}$ paths. Since the $\hat{F}$-neighborhoods of the vertices in J^α_{OPT} are disjoint (by construction), these paths end in distinct blue elements, thus, at least $|B_\alpha| r_\alpha/\mathrm{OPT}$ elements in B_α. ◀

Of course, we cannot know which red element will have this property, but the algorithm can try all elements and run the remainder of the algorithm on each one. Now, given a red element $i_0 \in R_\alpha$, our algorithm proceeds as follows: Begin by solving the following LP.

$$\max \sum_{\ell \in B} z_\ell \tag{1}$$

$$\sum_{i \in R_\alpha} y_i \le \mathrm{OPT} \tag{2}$$

$$0 \le z_\ell \le \min\{1, \sum_{j \in \Gamma_{J_\alpha}(i_0) \cap \Gamma_{J_\alpha}(\ell)} x_j\} \qquad \forall \ell \in B \tag{3}$$

$$0 \le x_j \le y_i \le 1 \qquad \forall j \in \Gamma_{J_\alpha}(i_0), i \in \Gamma_{R_\alpha}(j) \tag{4}$$

In the intended solution, x_j is the indicator for the event that $j \in \Gamma_{J^\alpha_{\mathrm{OPT}}}(i_0)$, y_i is the indicator variable for the event that red vertex i is in the union of red sets indexed by $\Gamma_{J^\alpha_{\mathrm{OPT}}}(i_0)$ (and therefore in the optimum solution), and z_ℓ is the indicator variable for the event that the blue vertex $\ell \in B$ is is covered by some set in $\Gamma_{J^\alpha_{\mathrm{OPT}}}(i_0)$. This LP is always feasible (say, by setting all variables to 0), though since there are at most $\log n$ subinstances in the partition, for at least one α we must have $|B_\alpha| \ge |B|/\log n$, and then by Lemma 10, there is also some choice of $i_0 \in R_\alpha$ for which the objective function satisfies

$$\sum_{\ell \in B} z_\ell \ge \frac{|B_\alpha| r_\alpha}{\mathrm{OPT}} \ge \frac{|B| r_\alpha}{\mathrm{OPT} \cdot \log n}. \tag{5}$$

The algorithm will choose α and i_0 that maximize the rescaled objective function $\sum_{\ell \in B} z_\ell / r_\alpha$, guaranteeing this bound. Finally, at this point, we perform a simple randomized rounding, choosing every set $j \in \Gamma_{J_\alpha}(i_0)$ independently with probability x_j. The entire algorithm is described in Algorithm 1.

Now let $J^* \subseteq J_\alpha$ be the collection of sets added by this step in the algorithm. Let us analyze the number of blue elements covered by J^* and the number of red elements added to the solution. First, noting that this LP acts as a max coverage relaxation for blue elements, the expected number of blue elements covered will be at least $(1 - 1/e)\frac{|B| r_\alpha}{\mathrm{OPT} \cdot \log n}$, by the standard analysis and the bound (5).

Now let us bound the number of red elements added. Let $J_+ = \left\{ j \in \Gamma_{J_\alpha}(i_0) \;\middle|\; x_j \ge \frac{\mathrm{OPT}}{r_\alpha \hat{A}} \right\}$ for a value of $\hat{A}$ to be determined later. By Constraint (4), every red neighbor $i \in \Gamma_{R_\alpha}(J_+)$ will also have $y_i \ge \mathrm{OPT}/(r_\alpha \hat{A})$, and so by Constraint (2), there can be at most $r_\alpha \hat{A}$ such neighbors. On the other hand, the expected number of red elements added by the remaining sets $j \in J^* \setminus J_+$ is bounded by

Algorithm 1 Approximation Algorithm for Blue-Red Set Cover.

```
Input: $B, J, R, E$
guess OPT                                                          ▷ e.g. using binary search
$J_{\text{ALG}} = \emptyset$                                        ▷ $J_{\text{ALG}}$ stores the current solution
find decomposition $\{(J_\alpha, R_\alpha)\}_\alpha$ as in Lemma 9, with $n_0 = \text{OPT} \cdot m^{1/3} \log^{4/3} n \log^2 k$.
while $B \neq \emptyset$ do
    for all $\alpha$ and $i_0 \in R_\alpha$ do
        Solve LP (1)-(4). Let $\text{LP}(\alpha, i_0)$ be its objective value
    end for
    choose the value of $\alpha$ and $i_0$ which maximizes $\text{LP}(\alpha, i_0)/r_\alpha$
    let $x, y, z$ be an optimal solution for this LP
    use solution $x$ and the method of conditional expectations to find $J^* \subseteq J_\alpha$
     s.t. $|\Gamma_{R_\alpha}(J^*)|/|\Gamma_B(J^*)| \leq O(1) \cdot m^{1/3} \log^{4/3} n \cdot \text{OPT}/|B|$   ▷ see the proof for details
    let $J_{\text{ALG}} = J_{\text{ALG}} \cup J^*$
    let $B = B \setminus \Gamma_B(J^*)$
    remove edges incident to deleted vertices from $E$
end while
return $J_{\text{ALG}}$
```

$$\begin{aligned}
\mathbb{E}\left[\left|\bigcup_{j \in J^* \setminus J_+} \Gamma_{R_\alpha}(j)\right|\right] &\leq 2r_\alpha \mathbb{E}\left[|J^* \setminus J_+|\right] && \text{by } R_\alpha\text{-degree bounds for } j \in J_\alpha \\
&= 2r_\alpha \cdot \sum_{j \in \Gamma_{J_\alpha}(i_0) \setminus J_+} x_j \\
&\leq 2r_\alpha \cdot \frac{\text{OPT}}{r_\alpha \hat{A}} |\Gamma_{J_\alpha}(i_0) \setminus J_+| && \text{by definition of } J_+ \\
&\leq \frac{2\text{OPT}}{\hat{A}} \cdot \frac{2mr_\alpha \log n}{\text{OPT} \cdot \hat{A}} && \text{by } J_\alpha\text{-degree bounds for } i \in R_\alpha \\
&= \frac{4mr_\alpha \log n}{\hat{A}^2}
\end{aligned}$$

These two bounds are equal when $r_\alpha \hat{A} = 4mr_\alpha \log n/\hat{A}^2$, that is, when $\hat{A} = (4m \log n)^{1/3}$, giving us a total bound on the expected number of red elements added in this step of

$$\mathbb{E}[|\Gamma_{R_\alpha}(J^*)|] \leq 2r_\alpha(4m \log n)^{1/3} \leq \frac{2r_\alpha(4m \log n)^{1/3}}{(1 - 1/e)|B|r_\alpha/(\text{OPT} \cdot \log n)} \cdot \mathbb{E}[|\Gamma_B(J^*)|].$$

Thus, $\mathbb{E}[|\Gamma_{R_\alpha}(J^*)|] - \frac{2(4m \log^4 n)^{1/3}\text{OPT}}{(1-1/e)|B|} \cdot \mathbb{E}|\Gamma_B(J^*)| \leq 0$. Using the method of conditional expectations, we can derandomize the algorithm and find J^* with a non-empty blue neighbor set such that $\frac{|\Gamma_{R_\alpha}(J^*)|}{|\Gamma_B(J^*)|} \leq O(1) \cdot m^{1/3} \log^{4/3} n \cdot \frac{\text{OPT}}{|B|}$. Thus, we make progress (according to Definition 8) towards an approximation guarantee of $\tilde{A} \cdot k$ for $\tilde{A} = O\left(m^{1/3} \log^{4/3} n\right)$, which, as noted, ultimately gives us the same approximation guarantee for Red-Blue Set Cover, proving Theorem 3.

4 Approximating MMSA$_t$ for $t \geq 5$

We now turn to the general problem of approximating MMSA$_t$ for arbitrarily large (but fixed) t. We will build on our approximation algorithm for MMSA$_4$ (described in Section 6) by recursively calling approximation algorithms for the problem with smaller values of t, and using the result of this approximation as a separation oracle in certain cases.

We will denote the total size of our input by N, and we will denote our approximation factor for MMSA$_t$ by A_t. We will only describe an algorithm for even depth. There is a very slightly simpler but quite similar algorithm for odd depth, however the guarantee we are able to achieve for MMSA$_{2t-1}$ is nearly the same as for MMSA$_{2t}$ (up to an $O(\log N)$ factor). Since MMSA$_{2t-1}$ is essentially a special case of MMSA$_{2t}$, we focus only on even levels.

▶ **Lemma 11.** *For $t \geq 2$, if MMSA$_{2t}$ can be approximated to within a factor of A_{2t}, then we can approximate MMSA$_{2t+2}$ (and thus MMSA$_{2t+1}$) to within $O(\sqrt{N \cdot A_{2t}} \log N)$.*

Proof. Denote our input as a layered graph with vertex layers $V_1, \ldots, V_{2t+2}$. Ideally, we would like to discard any vertex $j \in V_{2t}$ such that covering its neighbors $\Gamma_{2t+1}(j)$ requires more than OPT vertices in V_{2t+2}, however, checking this precisely requires solving Set Cover. Instead, we discard any vertex $j \in V_{2t}$ for which the smallest *fractional* set cover[3] in V_{2t+2} of its neighbors $\Gamma_{V_{2t+1}}(j)$ has value greater than OPT. Such vertices cannot be included without incurring cost greater than OPT and so we know they do not participate in any optimum solution. We begin with the following basic LP:

$$\sum_{h \in V_{2t+2}} w_h \leq \text{OPT} \tag{6}$$

$$y_i \leq \sum_{h \in \Gamma_{2t+2}(i)} w_h \qquad \forall i \in V_{2t+1} \tag{7}$$

$$x_j \leq y_i \qquad \forall j \in V_{2t}, i \in \Gamma_{V_{2t+1}}(j) \tag{8}$$

$$x_j, y_i, w_h \in [0,1] \qquad \forall j \in V_{2t} \forall i \in V_{2t+1} \forall h \in V_{2t+2} \tag{9}$$

Note that, as stated, this LP is trivial. Indeed, in the absence of any additional constraints, the all-zero solution is feasible. However, we will add new violated constraints as the algorithm proceeds.

Given a solution to the above linear program, our algorithm for MMSA$_{2t+2}$ is as follows:

- Let $V_{2t}^+ = \{j \in V_{2t} \mid x_j \geq 2(1 + \ln N)/A_{2t+2}\}$. Add these vertices to the solution.
- Let $V_{2t+2}^+ = \Gamma_{V_{2t+2}}(\Gamma_{V_{2t+1}}(V_{2t}^+))$ be the neighbors-of-neighbors of V_{2t}^+.
- Apply a greedy $(1 + \ln N)$-approximation for Set Cover to obtain a set cover (in V_{2t+2}) for $\Gamma_{V_{2t+1}}(V_{2t}^+)$, and add this set cover to the solution as well.
- Create an instance of MMSA$_{2t}$ by removing layers V_{2t+1}, V_{2t+2}, all vertices in V_{2t}^+, as well as their neighbors in V_{2t-1}, that is, $\Gamma_{V_{2t-1}}(V_{2t}^+)$, as these are already covered by V_{2t}^+.
- Apply an A_{2t}-approximation algorithm for MMSA$_{2t}$ to this instance, and let $U_{\text{ALG}} \subseteq V_{2t} \setminus V_{2t}^+$ be the result (or at least the portion belonging to layer $2t$).
- If $|U_{\text{ALG}}| \leq A_{2t+2}/(2 + 2\ln N)$, add the vertices in U_{ALG} to the solution, as well as a greedy set cover (in V_{2t+2}) for the neighborhood $\Gamma_{V_{2t+1}}(U_{\text{ALG}})$.
- Otherwise (if $|U_{\text{ALG}}| > A_{2t+2}/(2 + 2\ln N)$), continue the Ellipsoid algorithm using the new violated constraint

$$\sum_{j \in V_{2t} \setminus V_{2t}^+} x_j \geq \left\lfloor \frac{A_{2t+2}}{2(1 + \ln N) A_{2t}} \right\rfloor + 1, \tag{10}$$

and restart the algorithm (discarding the previous solution) using the new LP solution.

[3] That is, $\min\left\{\sum_{h \in S} z_h \;\middle|\; \forall i \in \Gamma_{V_{2t+1}}(j) : \sum_{h \in \Gamma_{V_{2t+2}}(i)} z_h \geq 1;\ \forall h \in S : z_h \geq 0\right\}$.

Let us now analyze this algorithm. By (8), we know that all neighbors $i \in V_{2t+1}$ of V_{2t}^+ have LP value $y_i \geq 2(1+\ln N)/A_{2t+2}$. Thus, by (7), if for every vertex $h \in V_{2t+2}$ we define $w_h^+ = w_h \cdot A_{2t+2}/(2+2\ln N)$, then this is a fractional Set Cover for the V_{2t+1}-neighborhood $\Gamma_{V_{2t+1}}(V_{2t}^+)$, and by (6) it has total fractional value at most $\mathrm{OPT} \cdot A_{2t+2}/(2+2\ln N)$. Thus, the greedy Set Cover $(1+\ln N)$-approximation algorithm will cover this neighborhood using at most $\mathrm{OPT} \cdot A_{2t+2}/2$ vertices in V_{2t+2}. After this step, we may add at most $\mathrm{OPT} \cdot A_{2t+2}/2$ additional vertices in V_{2t+2} to our solution to obtain an A_{2t+2}-approximation.

Now, suppose our MMSA_{2t} approximation returns a set U_{ALG} of cardinality $|U_{\mathrm{ALG}}| \leq A_{2t+2}/(2+2\ln N)$. Clearly, adding to our solution the vertices of U_{ALG} and a V_{2t+2}-Set Cover for its neighborhood $\Gamma_{V_{2t+1}}(U_{\mathrm{ALG}})$ gives a feasible solution to our MMSA_{2t+2} instance. Moreover, since by our preprocessing, the neighborhood $\Gamma_{V_{2t+1}}(j)$ of every $j \in U_{\mathrm{ALG}}$ has a fractional Set Cover in V_{2t+2} of value at most OPT, it follows that the union of all these neighborhoods, that is $\Gamma_{V_{2t+1}}(U_{\mathrm{ALG}})$, has a fractional set cover in V_{2t+2} of value at most $\mathrm{OPT} \cdot |U_{\mathrm{ALG}}| \leq \mathrm{OPT} \cdot A_{2t+2}/(2+2\ln N)$. And so applying a greedy Set Cover algorithm for the neighborhood $\Gamma_{V_{2t+1}}(U_{\mathrm{ALG}})$ contributes at most an additional $\mathrm{OPT} \cdot A_{2t+2}/2$ vertices in V_{2t+2} to our solution, as required.

Finally, let us examine the validity of the final step (the separation oracle). If the A_{2t}-approximation for MMSA_{2t} was not able to find a solution of size at most $A_{2t+2}/(2+2\ln N)$, then by definition, the value of any solution to our MMSA_{2t} instance must be greater than $A_{2t+2}/((2+2\ln N)A_{2t})$. This is a subinstance of our original instance, so any solution to our original MMSA_{2t+2} instance must also contain more than $A_{2t+2}/((2+2\ln N)A_{2t})$ vertices in V_{2t}. Thus, Constraint (10) is valid for any optimum solution. But when is it violated?

By definition of V_{2t}^+, the current total LP value of $V_{2t} \setminus V_{2t}^+$ is at most $2(1+\ln N)N/A_{2t+2}$. And so the current LP solution violates (10) if

$$\frac{2(1+\ln N)N}{A_{2t+2}} \leq \frac{A_{2t+2}}{2(1+\ln N)A_{2t}} \qquad \Longleftrightarrow \qquad A_{2t+2} \geq 2(1+\ln N)\sqrt{N \cdot A_{2t}}.$$

Thus, we can obtain an approximation factor of $A_{2t+2} = O(\sqrt{N \cdot A_{2t}} \log N)$ as claimed. ◀

Thus, by induction on t, with the guarantee of Theorem 4 for MMSA_4 as the basis of the induction, and Lemma 11 for the inductive steps, we get a general approximation algorithm for MMSA_t with approximation ratio $O\left(N^{1-\frac{1}{3}2^{3-\lceil t/2 \rceil}} \cdot (\log N)^{2+O(2^{-t/2})}\right)$.

5 Reduction from Min k-Union to Red-Blue Set Cover

In this section, we first present a reduction from Min k-Union to Red-Blue Set Cover and then prove a hardness result for Red-Blue Set Cover. We start with formally defining the Min k-Union problem.

▶ **Definition 12** (Min k-Union). *In the Min k-Union problem, we are given a set X of size n, a family $\mathcal{S}$ of m sets $S_1, \ldots, S_m$, and an integer parameter $k \geq 1$. The goal is to choose k sets $S_{i_1}, \ldots, S_{i_k}$ so as to minimize the cost $\left|\bigcup_{t=1}^k S_{i_t}\right|$. We will denote the cost of the optimal solution by $\mathrm{OPT}_{MU}(X, \mathcal{S}, k)$.*

Note that Min k-Union resembles the Red-Blue Set Cover Cover problem: in both problems, the goal is to choose some subsets $S_{i_1}, \ldots, S_{i_r}$ from a given family $\mathcal{S}$ so as to minimize the number of elements or red elements in their union. Importantly, however, the feasibility requirements on the chosen subsets $S_{i_1}, \ldots, S_{i_r}$ are different in Red-Blue Set Cover Cover and Min k-Union; in the former, we require that the chosen sets cover all k blue points but in the latter, we simply require that the number of chosen sets be k. Despite this difference, we show that there is a simple reduction from Min k-Union to Red-Blue Set Cover.

▷ Claim 13. There is a randomized polynomial-time reduction from Min k-Union to Red-Blue Set Cover that given an instance $\mathcal{I} = (X, \mathcal{S}, k)$ of Min k-Union returns an instance $\mathcal{I}' = (R, B, \{S'_i\}_{i\in[m]})$ of Red-Blue Set Cover satisfying the following two properties:

1. If $S'_{j_1}, \dots, S'_{j_r}$ is a feasible solution for $\mathcal{I}'$ then $k' \le r$ and the cost of solution $S_{j_1}, \dots, S_{j_{k'}}$ for Min k'-Union where $k' = \lfloor k/\ell \rfloor$ and $\ell = \lceil \log_e k \rceil + 1$ does not exceed that of $S'_{j_1}, \dots, S'_{j_r}$ for Red-Blue Set Cover:
$$\mathrm{cost}_{MU}(S_{j_1}, \dots, S_{j_{k'}}) \equiv \left|\bigcup_{t=1}^{k'} S_{i_t}\right| \le \left|\bigcup_{t=1}^{r} (S'_{i_t} \cap R)\right| = \mathrm{cost}_{RB}(S'_{j_1}, \dots, S'_{j_r}).$$
This is true always no matter what random choices the reduction makes.
2. $\mathrm{OPT}_{RB}(R, B, \{S'_i\}_i) \le \mathrm{OPT}_{MU}(X, \mathcal{S}, k)$ with probability at least $1 - 1/e$.

Proof. We define instance $\mathcal{I}'$ as follows. Let $R = X$ and $B = [k]$. For every $i \in [m]$, let $R_i = S_i$; B_i be a set of ℓ elements randomly sampled from $[k]$ with replacement, and $S'_i = R_i \cup B_i$. All random choices are independent. Now we verify that this reduction satisfies the required properties.

Consider a feasible solution $S'_{j_1}, \dots, S'_{j_r}$ for $\mathcal{I}'$. Since this solution is feasible, $\cup_{t=1}^{r} B_t = B$. Now $|B_t| \le \ell$ and thus $r \ge |B|/\ell = k/\ell \ge k'$, as required. Further,
$$\mathrm{cost}_{MU}(S_{j_1}, \dots, S_{j_{k'}}) \equiv \left|\bigcup_{t=1}^{k'} S_{j_t}\right| = \left|\bigcup_{t=1}^{k'} R_{j_t}\right| \le \left|\bigcup_{t=1}^{r} R_{j_t}\right| \equiv \mathrm{cost}_{RB}(S'_{j_1}, \dots, S'_{j_r}).$$
We have verified that item 1 holds. Now, let $S_{i_1}, \dots, S_{i_k}$ be an optimal solution for $\mathcal{I}$. We claim that $S'_{i_1}, \dots, S'_{i_k}$ is a feasible solution for $\mathcal{I}'$ with probability at least $1 - 1/e$. To verify this claim, we need to lower bound the probability that $B_{i_1} \cup \dots \cup B_{i_k} = B$. Indeed, set $B_{i_1} \cup \dots \cup B_{i_k}$ consists of $k\ell$ elements sampled from B with replacement. The probability that a given element $b \in B$ is not in $B_{i_1} \cup \dots \cup B_{i_k}$ is at most $(1 - 1/k)^{k\ell} \le e^{-\ell} \le \frac{1}{ek}$. By the union bound, the probability that there is some $b \in B \setminus (B_{i_1} \cup \dots \cup B_{i_k})$ is at most $k \times \frac{1}{ek} = \frac{1}{e}$. Thus, there is no such b with probability at least $1 - 1/e$ and consequently $B_{i_1} \cup \dots \cup B_{i_k} = B$. In that case, the cost of solution $S'_{i_1}, \dots, S'_{i_k}$ for Red-Blue Set Cover equals $\left|\bigcup_{i=1}^{k} R'_{i_t}\right| = \left|\bigcup_{i=1}^{k} S_{i_t}\right|$, the cost of the optimal solution for Min k-Union. ◁

▶ **Corollary 14.** *Assume that there is an $\alpha(m, n)$ approximation algorithm $\mathcal{A}$ for Red-Blue Set Cover (where α is a non-decreasing function of m and n).Then there exists a randomized polynomial-time algorithm $\mathcal{B}$ for Min k-Union that finds k' sets $S_{i_1}, \dots, S_{i_{k'}}$ such that*
$$\left|\bigcup_{t=1}^{k'} S_{i_t}\right| \le \alpha(m, n)\mathrm{OPT}_{MU}(X, \mathcal{S}, k).$$
The failure probability is at most $1/n$.

Proof. We simply apply the reduction to the input instance of Min k-Union and then solve the obtained instance of Red-Blue Set Cover using algorithm $\mathcal{A}$. To make sure that the failure probability is at most $1/n$, we repeat this procedure $\lceil \log_e n \rceil$ times and output the best of the solutions we found. ◀

▶ **Theorem 15.** *Assume that there is an $\alpha(m,n)$ approximation algorithm $\mathcal{A}$ for Red-Blue Set Cover (where α is a non-decreasing function of m and n). Then there exists an $O(\log^2 k)\alpha(m,n)$ approximation algorithm for Min k-Union.*

Proof. Our algorithm iteratively uses algorithm $\mathcal{B}$ from the corollary to find an approximate solution. First, it runs $\mathcal{B}$ on the input instance and gets $k_1 = k'$ sets. Then it removes the sets it found from the instance and reduces the parameter k to $k - k_1$. Then the algorithm runs $\mathcal{B}$ on the obtained instance and gets k_2 sets. It again removes the obtained sets and reduces k to $k - k_1 - k_2$ (here k is the original value of k). It repeats these steps over and over until it finds k sets in total. That is, $k_1 + \cdots + k_T = k$ where T is the number of iterations the algorithm performs.

Observe that each of the instances of Min k-Union constructed in this process has cost at most $\mathrm{OPT}_{MU}(X, \mathcal{S}, k)$. Indeed, consider the subinstance $\mathcal{I}_{t+1}$ we solve at iteration $t+1$. Consider k sets that form an optimal solution for $(X, \mathcal{S}, k)$. At most $k_1 + \cdots + k_t$ of them have been removed from $\mathcal{I}_{t+1}$ and thus at least $k - k_1 - \cdots - k_t$ are still present in $\mathcal{I}_{t+1}$. Let us arbitrarily choose $k - k_1 - \cdots - k_t$ sets among them. The chosen sets form a feasible solution for $\mathcal{I}_{t+1}$ of cost at most $\mathrm{OPT}_{MU}(X, \mathcal{S}, k)$.

Thus, the cost of a partial solution we find at each iteration t is at most $\alpha(m,n) \cdot \mathrm{OPT}_{MU}(X, \mathcal{S}, k)$. The total cost is at most $\alpha(m,n) \cdot T \cdot \mathrm{OPT}_{MU}(X, \mathcal{S}, k)$. It remains to show that $T \leq O(\log^2 k)$. We observe that the value of k reduces by a factor at least $1 - 1/\ell$ in each iteration, thus after t iterations it is at most $(1 - 1/\ell)^t k$. We conclude that the total number of iterations T is at most $O(\ell \log k) = O(\log^2 k)$, as desired. ◀

Now we obtain a conditional hardness result for Reb-Blue Set Cover from a corollary from the Hypergraph Dense-vs-Random Conjecture.

▶ **Corollary 16** (Chlamtáč et al. [9]). *Assuming the Hypergraph Dense-vs-Random Conjecture, for every $\varepsilon > 0$, no polynomial-time algorithm for Min k-Union achieves better than $\Omega(m^{1/4-\varepsilon})$ approximation.*

Theorem 6 immediately follows.

6 Approximation Algorithm for MMSA$_4$

Consider an instance (B, J, R, S, E) of MMSA$_4$. As we did for Red-Blue Set Cover, we will focus on making progress towards a good approximation. Due to space constraints, we have omitted most proofs for statements in this section. Complete proofs can be found in the full version of the paper.

▶ **Definition 17.** *We say that an algorithm for MMSA$_4$* makes progress towards an $O(A)$-approximation *if, given an instance with an optimum solution containing at most* OPT *vertices in S, the algorithm finds a subset $\hat{J} \subseteq J$ and a subset $\hat{S} \subseteq S$ such that $\Gamma_R(\hat{J}) \subseteq \Gamma_R(\hat{S})$ (a valid partial solution) and $\frac{|\hat{S}|}{|\Gamma_B(\hat{J})|} \leq A \cdot \frac{\mathrm{OPT}}{|B|}$.*

As before, it is easy to see that given such an algorithm, we can run such an algorithm repeatedly to obtain an actual $\tilde{O}(A)$ approximation for MMSA$_4$. In fact, in the rest of this section we will *only* discuss an algorithm which makes progress towards an $O(A)$-approximation.

For the sake of formulating an LP relaxation with a high degree of uniformity, we will actually focus on a partial solution which covers a large fraction of blue elements in a uniform manner:

▶ **Lemma 18.** *For any cover $J_0 \subseteq J$ of the blue elements B, there exist subsets $J' \subseteq J_0$ and $B' \subseteq B$ and a parameter $\Delta > 0$ with the following properties:*

- *Every vertex $j \in J'$ has B'-degree in the range $\deg_{B'}(j) \in [\Delta, 2\Delta]$.*
- *Every blue element $\ell \in B'$ has at least one neighbor in J'_Δ and at most $2e\ln(2k)$ neighbors.*
- *We have the cardinality bound $|B'| \geq |B|/(\log k \log m)$.*

Simplifying Assumptions. We can make the following assumptions which will be useful in the analysis of our algorithm. First, we may assume that for every $j \in J$, the red neighborhood $\Gamma_R(j)$ has a fractional set cover in S of weight at most OPT. That is, the standard LP relaxation for covering $\Gamma_R(j)$ using S has optimum value at most OPT. If we have guessed the correct value of OPT, then we know that no $j \in J$ whose red neighborhood cannot be covered with cost OPT can participate in an optimum solution, and can therefore be discarded. We may also assume that for some $\varepsilon > 0$, the value Δ above is at most k/m^ε. The reason is that otherwise, the blue elements B' can be covered with at most $\tilde{O}(m^\varepsilon)$ vertices in J, and we know that for each of these, its red neighborhood can be covered by a set of size $\tilde{O}(\mathrm{OPT})$ in S, and thus we can make progress towards an $\tilde{O}(m^\varepsilon)$ approximation.

Guessing the value of $\Delta \in [k]$ above and the value of the optimum OPT, we can write the following LP relaxation:

$$\sum_{h\in S} w_h \leq \mathrm{OPT} \tag{11}$$

$$\sum_{\ell\in B} z_\ell \geq |B|/(\log k \log m) \tag{12}$$

$$z_\ell \leq \sum_{j\in\Gamma_J(\ell)} x_j^\ell \leq 2e\ln(2k) z_\ell \qquad \forall \ell \in B \tag{13}$$

$$\Delta x_j \leq \sum_{\ell\in\Gamma_B(j)} x_j^\ell \leq 2\Delta x_j \qquad \forall j \in J \tag{14}$$

$$0 \leq x_j^\ell \leq x_j, z_\ell \leq 1 \qquad \forall \ell \in B \forall j \in J \tag{15}$$

$$\sum_{h\in\Gamma_S(i)} w_h \geq y_i \qquad \forall i \in R \tag{16}$$

$$0 \leq x_j \leq y_i \qquad \forall (j,i) \in E(J,R) \tag{17}$$

We further strengthen this LP by partially lifting the above constraints. Specifically, for every $a \in J \cup S$, $j \in J$, $h \in S$, $i \in R$, and $\ell \in B$ we introduce variables $X_h^{(a)}, X_\ell^{(a)}, X_{\ell,j}^{(a)}, X_j^{(a)}, X_i^{(a)}$, and lift all the above constraints accordingly. For a precise definition, see Appendix B. For any $j \in J$ such that $x_j > 0$ or $h \in S$ such that $w_h > 0$, we will denote the "conditioned" variables by $\hat{w}_h^{(j)} = X_h^{(j)}/x_j$, $\hat{z}_\ell^{(h)} = X_\ell^{(h)}/w_h$, etc.

▶ Remark 19. The above linear program is a relaxation for the partial solution given by Lemma 18. Specifically, given an optimal solution $(J_{\mathrm{OPT}}, S_{\mathrm{OPT}})$, applying the lemma to $J_0 = J_{\mathrm{OPT}}$, we have the following feasible solution: Set the variables z_ℓ and x_j to be indicators for B' and J' as in the lemma, respectively, and the variables x_j^ℓ to be indicators for $J' \times B'$. Set the the variables w_h to be indicators for S_{OPT}, and the variables y_i to be indicators for the red neighbors $\Gamma_R(J')$ of J'.

Let us examine some useful properties of this relaxation. First of all, we note that it approximately determines the total LP value of J (since the LP assigns total LP value $\tilde{\Theta}(|B|)$ to B):

▷ Claim 20. Any solution satisfying constraints (13)-(15), has total LP weight in J bounded by $\frac{1}{2\Delta}\sum_{\ell\in B} z_\ell \le \sum_{j\in J} x_j \le \frac{2e\ln(2k)}{\Delta}\sum_{\ell\in B} z_\ell$.

These constraints also determine a useful combinatorial property: in any feasible solution, the number of blue neighbors a subset of J has is (at least) proportional to the LP value of that set.

▷ Claim 21. For any solution satisfying constraints (13)-(15), and any subset of vertices $\hat{J} \subseteq J$, the number of blue neighbors of $\hat{J}$ is bounded from below by:

$$|\Gamma_B(\hat{J})| \ge \frac{1}{4e\ln(2k)\log k \log m} \cdot \frac{x(\hat{J})}{x(J)} \cdot |B|.$$

A fractional variant of the above covering property for the blue vertices is the following:

▷ Claim 22. For any solution satisfying constraints (13)-(15), and any subset of vertices $\hat{J} \subseteq J$, at least $\varepsilon|B|$ vertices $\ell \in B$ satisfy $\sum_{j\in\Gamma_J(\ell)} x_j^\ell \ge \frac{1}{4\log k\log m} \cdot \frac{x(\hat{J})}{x(J)}$, where $\varepsilon = \frac{1}{8e\ln(2k)\log k\log m} \cdot \frac{x(\hat{J})}{x(J)}$.

Let us now analyze the approximation guarantee of Algorithm 2. We begin by stating simple lower bounds on the total LP value of the set J_0 as well as the vertices in the set.

▶ **Lemma 23.** *The set J_0 defined in Algorithm 2 has LP value at least $x(J)/(2\log m)$ and the lower bound on the individual LP values in the set is bounded by $x_0 \ge 1/m$.*

Next, we examine the bucketing of neighbors in S, and give a lower bound on the number of vertices in these bucketed sets.

▶ **Lemma 24.** *In Algorithm 2, for every vertex $j \in J_0$, and every red neighbor $i \in \Gamma_R(j)$, the bucketed set of neighbors $\hat{\Gamma}_j(i)$ of i has cardinality bounded from below by $|\hat{\Gamma}_j(i)| \ge 1/(6\beta_{ji}\log|S|\log(|S|^2 m))$.*

Proof. Fix vertices $j \in J_0$ and $i \in \Gamma_R(j)$. Let us begin by examining our choice of β_{ji}. Note that lifting Constraint 17, we get $x_j = X_j^{(j)} \le X_i^{(j)} (\le x_j)$, and so $\hat{y}^{(j)} = X_i^{(j)}/x_j = 1$. Lifting Constraint (16), we thus get $\sum_{h\in\Gamma_S(i)} \hat{w}_h^{(j)} \ge 1$. Note that the total LP weight of the set $S'_{ji} = \{h \in \Gamma_S(i) \mid \hat{w}_h^{(j)} \le 1/|S|^2\}$ is at most $1/|S| \le \hat{w}^{(j)}(\Gamma_S(i))/3$. Therefore, the total $\hat{w}^{(j)}$ LP weight of the bucketed sets S_s^{ji} for s such that $2^{-s} \ge 1/|S|^2$ is at least $\frac{2}{3}\hat{w}^{(j)}(\Gamma_S(i))$, and at least one of these bucketed sets has LP weight at least a $1/(2\log|S|)$-fraction of this, or at least $\hat{w}^{(j)}(\Gamma_S(i))/(3\log|S|) \ge 1/(3\log|S|)$. This gives a lower bound on the LP weight of the bucket which defines β_{ji}. Also, the heaviest bucket cannot be S_s^{ji} for s such that $2^{-s} \le 1/|S|^2$, since even the total weight of these buckets is at most $1/|S| = o(1/(3\log|S|))$. In particular, this means that $\beta_{ji} \ge 1/|S|^2$. Moreover, for s such that $2^{-s} = \beta_{ji}$, since the total conditional LP weight of S_s^{ji} is at least $1/(3\log|S|)$, and every vertex in the set has conditional LP value at most $2\beta_{ji}$, the cardinality of the set must be at least $1/(6\beta_{ji}\log|S|)$.

Now let us examine the second stage of bucketing. Note that for every $h \in \Gamma_S(i)$, we have $w_h \ge X_h^{(j)} = \hat{w}_h^{(j)} \cdot x_j \ge \beta_{ji}x_0 \ge 1/(|S|^2 m)$ (and, of course, $w_h \le 1$). Therefore, the number of non-empty buckets $\hat{S}_t^{ji}$ is at most $\log(|S|^2 m)$, and at least one of them must have cardinality at least $|S_s^{ji}|/\log(|S|^2 m)$, which, along with our lower bound on $|S_s^{ji}|$ above, gives us the required lower bound on $|\hat{\Gamma}_j(i)|$. ◀

Note that from the above proof, we also get upper-bounds on the number of values of β_{ji} and γ_{ji} that can produce non-empty buckets. In particular, we get the following bound:

▶ **Observation 25.** *The total number of possible values for β_{ji} is at most $2\log|S|$, and the total number of possible values for γ_{ji} is at most $\log(|S|^2m)$. Along with the range of values for D, the total number of triples $\langle\beta,\gamma,D\rangle$ for which $J^D_{\beta,\gamma}$ is non-empty is at most $2\log^2|S|\log(|S|^2m)$.*

The algorithm proceeds by separating the buckets corresponding to parameters for which the simple rounding (which samples a random subset of J of size $\tilde{\Omega}(J)$) makes progress towards an approximation guarantee of $\tilde{O}(A)$. If a large fraction of vertices in J_0 participate exclusively in such buckets, then the algorithm applies this rounding. The following lemma gives the analysis of the algorithm in this case.

▶ **Lemma 26.** *In Algorithm 2, if $|J_1| < |J_0|/2$, then with high probability the algorithm samples a subset $J_{\mathrm{ALG}} \subseteq J$ which covers an $\tilde{\Omega}(1)$-fraction of blue vertices, and a subset of S of size $\tilde{O}(A\cdot\mathrm{OPT})$ which covers all the red neighbors of J_{ALG}.*

Finally, we turn to the remaining case in Algorithm 2, when $|J_1| \le |J_0|/2$. The analysis of this case rests on a back-degree argument similar (though significantly more involved) to the argument in Lemma 10 for Red-Blue Set Cover. Indeed, we show the following:

▶ **Lemma 27.** *If $|J_1| \ge |J_0|/2$, then for β,γ,D,h_0 and the set J_{ALG} as defined by the algorithm in this case, we have $\sum_{j\in J_{\mathrm{ALG}}}\hat{x}_j^{(h_0)} \ge \frac{|J_0|Dx_0\beta}{\mathrm{OPT}}\cdot\frac{1}{4\log^2|S|\log(|S|^2m)\log m}$.*

Furthermore, for every vertex $j \in \tilde{J}$ (as defined by the algorithm), we have $\hat{x}_j^{h_0} \in [x_0\beta/(2\gamma), 4x_0\beta/\gamma]$.

We can now show that in this case, the algorithm makes progress towards an $\tilde{O}(m/A^2)$-approximation. Trading this off with the progress towards an $\tilde{O}(A)$-approximation as guaranteed by Lemma 26, we get an $\tilde{O}(m^{1/3})$-approximation by setting $A = m^{1/3}$.

▶ **Lemma 28.** *In Algorithm 2, if $|J_1| \ge |J_0|/2$, then with high probability, the algorithm makes progress towards an approximation guarantee of $\tilde{O}(m/A^2)$.*

Proof. Let us first bound the number of blue vertices covered by J_{ALG}. By Lemma 27, we have

$$\begin{aligned}
\hat{x}^{h_0}(J_{\mathrm{ALG}}) &\ge \frac{|J_0|x_0D\beta}{\mathrm{OPT}}\cdot\frac{1}{4\log^2|S|\log(|S|^2m)\log m} \\
&\ge \frac{x(J_0)D\beta}{\mathrm{OPT}}\cdot\frac{1}{4\log^2|S|\log(|S|^2m)\log m} && \text{since } \forall j\in J_0: 2x_0 \ge x_j \\
&\ge \frac{x(J)D\beta}{\mathrm{OPT}}\cdot\frac{1}{4\log^2|S|\log(|S|^2m)\log m} && \text{by Lemma 23} \\
&> x(J)\cdot\frac{A}{x(J_0)}\cdot\frac{1}{4\log^2|S|\log(|S|^2m)\log m}. && \text{since } \forall\langle\beta,\gamma,D\rangle\in P_1: \frac{\beta D}{\mathrm{OPT}} > \frac{A}{x(J_0)}
\end{aligned}$$

Thus, since the conditioned LP solution satisfies the basic LP, we can apply Claim 21 to this solution and get that the size of the blue neighborhood of J_{ALG} can be bounded by

$$\begin{aligned}
|\Gamma_B(J_{\mathrm{ALG}})| &\ge \frac{1}{4e\ln(2k)\log k\log m}\cdot\frac{A}{x(J_0)}\cdot\frac{1}{4\log^2|S|\log(|S|^2m)\log m}\cdot|B| \\
&= \frac{1}{16\ln(2k)\log k\log^2|S|\log(|S|^2m)\log^2 m}\cdot\frac{A}{x(J_0)}\cdot|B|.
\end{aligned}$$

Note that by the LP constraints and Lemma 27, for every red neighbor $i \in \Gamma_R(J_{\mathrm{ALG}})$, we have $\hat{y}_i^{h_0} \geq x_i^{h_0} \geq x_0\beta/(2\gamma)$, and so by (16), the rescaled solution $(\hat{w}_h^{h_0} \cdot 2\gamma/(x_0\beta))_{h\in S}$ is a fractional set cover for $\Gamma_R(J_{\mathrm{ALG}})$. Thus, sampling every $h \in S$ with probability $\min\{1, 2\ln n \cdot \hat{w}_h^{h_0} \cdot 2\gamma/(x_0\beta)\}$ produces a valid set cover with high probability. It remains to analyze the size of this set cover. Indeed, since $\hat{w}^{h_0}(S) \leq \mathrm{OPT}$, our sampling procedure produces a set of expected size

$$\begin{aligned}
\mathbb{E}[|S_{\mathrm{ALG}}|] &\leq \frac{4\gamma\ln n}{x_0\beta} \cdot \mathrm{OPT} \\
&\leq 8\ln n \cdot \frac{\gamma}{\beta} \cdot \frac{|J_0|}{x(J_0)} \cdot \mathrm{OPT} && \text{since } \forall j \in J_0 : x_j \leq 2x_0 \\
&\leq 8\ln n \cdot \frac{\gamma}{\beta} \cdot \frac{m}{x(J_0)} \cdot \mathrm{OPT} \\
&< 8\ln n \cdot \frac{1}{A} \cdot \frac{m}{x(J_0)} \cdot \mathrm{OPT}, && \text{since } \forall \langle \beta, \gamma, D\rangle \in P_1 : \frac{\beta}{\gamma} > A
\end{aligned}$$

and so with high probability we have $|S_{\mathrm{ALG}}| = O(\mathrm{OPT} \cdot m\log n/(A \cdot x(J_0)))$.

Putting our two bounds together, we get that in this case, the algorithm makes progress towards an approximation guarantee of

$$\frac{|B|}{|\Gamma_B(J_{\mathrm{ALG}})|} \cdot \frac{|S_{\mathrm{ALG}}|}{\mathrm{OPT}} = \tilde{O}(1) \cdot \frac{x(J_0)}{A} \cdot \frac{m}{A \cdot x(J_0)} = \tilde{O}(1) \cdot \frac{m}{A^2}. \qquad ◀$$

References

1 V. P. Abidha and Pradeesha Ashok. Red blue set cover problem on axis-parallel hyperplanes and other objects. *CoRR*, abs/2209.06661, 2022. doi:10.48550/arXiv.2209.06661.

2 Michael Alekhnovich, Samuel R. Buss, Shlomo Moran, and Toniann Pitassi. Minimum propositional proof length is np-hard to linearly approximate. *J. Symb. Log.*, 66(1):171–191, 2001. doi:10.2307/2694916.

3 Pradeesha Ashok, Sudeshna Kolay, and Saket Saurabh. Multivariate complexity analysis of geometric red blue set cover. *Algorithmica*, 79(3):667–697, 2017. doi:10.1007/s00453-016-0216-x.

4 Pranjal Awasthi, Avrim Blum, and Or Sheffet. Improved guarantees for agnostic learning of disjunctions. In Adam Tauman Kalai and Mehryar Mohri, editors, *COLT 2010 - The 23rd Conference on Learning Theory, Haifa, Israel, June 27-29, 2010*, pages 359–367. Omnipress, 2010. URL: http://colt2010.haifa.il.ibm.com/papers/COLT2010proceedings.pdf#page=367.

5 Robert D. Carr, Srinivas Doddi, Goran Konjevod, and Madhav V. Marathe. On the red-blue set cover problem. 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 345–353. ACM/SIAM, 2000. URL: http://dl.acm.org/citation.cfm?id=338219.338271.

6 Timothy M. Chan and Nan Hu. Geometric red-blue set cover for unit squares and related problems. *Comput. Geom.*, 48(5):380–385, 2015. doi:10.1016/j.comgeo.2014.12.005.

7 Moses Charikar, Yonatan Naamad, and Anthony Wirth. On approximating target set selection. In Klaus Jansen, Claire Mathieu, José D. P. Rolim, and Chris Umans, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France*, volume 60 of *LIPIcs*, pages 4:1–4:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.APPROX-RANDOM.2016.4.

8 Eden Chlamtác, Michael Dinitz, Christian Konrad, Guy Kortsarz, and George Rabanca. The densest k-subhypergraph problem. *SIAM J. Discret. Math.*, 32(2):1458–1477, 2018. doi:10.1137/16M1096402.

9 Eden Chlamtác, Michael Dinitz, and Yury Makarychev. Minimizing the union: Tight approximations for small set bipartite vertex expansion. 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 881–899. SIAM, 2017. doi:10.1137/1.9781611974782.56.

10 Irit Dinur and Shmuel Safra. On the hardness of approximating label-cover. *Inf. Process. Lett.*, 89(5):247–254, 2004. doi:10.1016/j.ipl.2003.11.007.

11 Michael Elkin and David Peleg. The hardness of approximating spanner problems. *Theory Comput. Syst.*, 41(4):691–729, 2007. doi:10.1007/s00224-006-1266-2.

12 Michael H. Goldwasser and Rajeev Motwani. Intractability of assembly sequencing: Unit disks in the plane. In Frank K. H. A. Dehne, Andrew Rau-Chaplin, Jörg-Rüdiger Sack, and Roberto Tamassia, editors, *Algorithms and Data Structures, 5th International Workshop, WADS '97, Halifax, Nova Scotia, Canada, August 6-8, 1997, Proceedings*, volume 1272 of *Lecture Notes in Computer Science*, pages 307–320. Springer, 1997. doi:10.1007/3-540-63307-3_70.

13 Raghunath Reddy Madireddy and Apurva Mudgal. A constant-factor approximation algorithm for red-blue set cover with unit disks. *Algorithmica*, 85(1):100–132, 2023. doi:10.1007/s00453-022-01012-z.

14 Raghunath Reddy Madireddy, Subhas C. Nandy, and Supantha Pandit. On the geometric red-blue set cover problem. In Ryuhei Uehara, Seok-Hee Hong, and Subhas C. Nandy, editors, *WALCOM: Algorithms and Computation - 15th International Conference and Workshops, WALCOM 2021, Yangon, Myanmar, February 28 - March 2, 2021, Proceedings*, volume 12635 of *Lecture Notes in Computer Science*, pages 129–141. Springer, 2021. doi:10.1007/978-3-030-68211-8_11.

15 Pauli Miettinen. On the positive-negative partial set cover problem. *Inf. Process. Lett.*, 108(4):219–221, 2008. doi:10.1016/j.ipl.2008.05.007.

A Adapting and Applying our Algorithm to Partial Red-Blue Set Cover

Let us now consider the variation in which we are given a parameter $\hat{k}$, and are only required to cover at least $\hat{k}$ elements in a feasible solution. The algorithm and analysis work with almost no change other than the following.

In the algorithm, the stopping condition of the loop is of course no longer once we have covered all blue elements, but once we have covered at least $\hat{k}$ of them.

A slightly more subtle change involves the analysis of the LP rounding in the final iteration. The notion of progress towards a certain approximation guarantee may not be valid if the ratio of red elements to blue elements covered is still as small as required, but the number of blue elements added is far more than we need. Rather than derandomize the rounding, one can show that it succeeds (despite this issue) with high probability. Let us briefly sketch the argument here.

First, note that we can always preemptively discard any sets with more than OPT red elements, and so we may assume that $r_\alpha \leq \text{OPT}$. Suppose we need to cover an additional k^* elements in order to reach the target of $\hat{k}$ blue elements total. Since our bound on $\mathbb{E}[|\Gamma_{R_\alpha}(J^*)|]$ is a linear function of our bound on $\mathbb{E}[|J^* \setminus J_+|]$, by a Chernoff bound we have $|\Gamma_{R_\alpha}(J^*)| = \tilde{O}(r_\alpha A)$ with all but exponentially small probability. On the other hand, $|\Gamma_B(J^*)|$ is always at most $|B|$, so by Markov, we have

$$\text{Prob}\left[|\Gamma_B(J^*)| \leq \frac{\mathbb{E}[|\Gamma_B(J^*)|]}{2}\right] \leq \frac{|B| - \mathbb{E}[|\Gamma_B(J^*)|]}{|B| - \mathbb{E}[|\Gamma_B(J^*)|]/2} \leq 1 - \frac{1}{2|B|}.$$

Thus, repeating the rounding a polynomial number of times (in a given iteration), with all but exponentially small probability we can find a set $\hat{J} \subseteq J$ that satisfies both

$$|\Gamma_B(\hat{J})| \geq \frac{\mathbb{E}[|\Gamma_B(J^*)|]}{2} \qquad \text{and} \qquad |\Gamma_{R_\alpha}(\hat{J})|] = \tilde{O}(r_\alpha A).$$

Now if $\mathbb{E}[|\Gamma_B(J^*)|] \leq 2k^*$, then we have the required ratio and bound on the number of new red elements by the previous analysis. If $\mathbb{E}[|\Gamma_B(J^*)|] > 2k^*$, then this will be the last iteration, as we will cover at least the required k^* additional blue elements, and the number of red elements added at this final stage is at most $\tilde{O}(r_\alpha A) \leq \tilde{O}(A \cdot \mathrm{OPT})$, so we maintain the desired approximation ratio.

B Additional LP Constraints for MMSA$_4$

The following is a complete list of lifted constraints that we use in addition to the basic LP relaxation for MMSA$_4$:

$$\begin{aligned}
&X_j^{(h)} = X_h^{(j)} && \forall j \in J, \forall h \in S \\
&X_j^{(j)} = x_j && \forall j \in J \\
&\sum_{h \in \Gamma_S(i)} X_h^{(j)} \geq X_i^{(j)} && \forall j \in J \forall i \in R \\
&X_j^{(j)} \leq X_i^{(j)} && \forall j \in J \forall i \in \Gamma_R(j) \\
&0 \leq X_a^{(j)} \leq x_j && \forall j \in J \forall a \in \{j\} \cup R \cup S \\
&\sum_{h' \in S} w_{h'}^{(h)} \leq \mathrm{OPT} && \forall h \in S \\
&\sum_{\ell \in B} X_\ell^{(h)} \geq |B|/(\log k \log m) w_h && \forall h \in S \\
&X_\ell^{(h)} \leq \sum_{j \in \Gamma_J(\ell)} X_{\ell,j}^{(h)} \leq 2e \ln(2k) X_\ell^{(h)} && \forall h \in S \forall \ell \in B \\
&\Delta X_j^{(h)} \leq \sum_{\ell \in \Gamma_B(j)} X_{\ell,j}^{(h)} \leq 2\Delta X_j^{(h)} && \forall h \in S \forall j \in J \\
&0 \leq X_{\ell,j}^{(h)} \leq X_j^{(h)}, X_\ell^{(h)} \leq w_h && \forall h \in S \forall \ell \in B \forall j \in J \\
&\sum_{h' \in \Gamma_S(i)} X_{h'}^{(h)} \geq X_i^{(h)} && \forall h \in S \forall i \in R \\
&X_j^{(h)} \leq X_i^{(h)} && \forall h \in S \forall (j,i) \in E(J,R) \\
&0 \leq X_a^{(h)} \leq w_h && \forall h \in S \forall a \in B \cup J \cup R \cup S
\end{aligned}$$

Algorithm 2 Approximation Algorithm for MMSA$_4$.

```
Input: $B, J, R, S, E$
Guess $\mathrm{OPT}, \Delta$ and solve the LP                ▷ e.g. using binary search
Choose parameter $s$ such that the LP weight of the bucket $J_s = \{j \in J \mid 2^{-s} \le x_j \le 2^{-(s-1)}\}$, that is, $\sum_{j \in J_s} x_j$ is maximized, and let $x_0 = 2^{-s}$ and $J_0 = J_s$.
for every $j \in J_0$ and $i \in \Gamma_R(j)$ do
    Choose a new parameter $s$ such that the conditioned LP weight of the bucket $S^{ji}_s = \{h \in S \mid 2^{-s} \le \hat{w}^{(j)}_h \le 2^{-(s-1)}\}$, that is, $\sum_{h \in S^{ji}_s} \hat{w}^{(j)}_h$, is maximized, and let $\beta_{ji} = 2^{-s}$.
    Choose parameter $t$ such that the sub-bucket $\hat{S}^{ji}_t = \{h \in S^{ji}_s \mid 2^{-t} \le w_h \le 2^{-(t-1)}\}$ has maximum cardinality $|\hat{S}^{ji}_t|$, and let $\gamma_{ji} = 2^{-t}$ and $\hat{\Gamma}_j(i) = \hat{S}^{ji}_t$.
end for
for every $j \in J_0$ and $\beta, \gamma$ do
    Let $\Gamma^R_{\beta,\gamma}(j) = \{i \in \Gamma_R(j) \mid \beta_{ji} = \beta, \gamma_{ji} = \gamma\}$.
    Let $\Gamma^S_{\beta,\gamma}(j) = \bigcup_{i \in \Gamma^R_{\beta,\gamma}(j)} \hat{\Gamma}_j(i)$.
end for
for every $\beta, \gamma$ and $D \in \{2^{s-1} \mid s \in \lceil \log |S| \rceil\}$ do
    Let $J^D_{\beta,\gamma} = \{j \in J_0 \mid \Gamma^R_{\beta,\gamma}(j) \ne \emptyset, |\Gamma^S_{\beta,\gamma}(j)| \in [D, 2D]\}$.
    Let $T^D_{\beta,\gamma} = \{\langle j, \Gamma^R_{\beta,\gamma}(j), \Gamma^S_{\beta,\gamma}(j)\rangle \mid j \in J^D_{\beta,\gamma}(j)\}$.
end for
Let $P_1 = \{\langle \beta, \gamma, D \mid \beta/\gamma > A, \beta D > A \cdot \mathrm{OPT}/x(J_0)\rangle\}$, and $J_1 = \bigcup_{\langle \beta,\gamma,D\rangle \in P_1} J^D_{\beta,\gamma}$.
if $|J_1| < |J_0|/2$ then
    Let $J_{\mathrm{ALG}} = \emptyset$.
    for all $j \in J_0 \setminus J_1$ do
        Independently add $j$ to $J_{\mathrm{ALG}}$ with probability $x_0$.
    end for
    Let $S_{\mathrm{ALG}} = \emptyset$.
    for every $\beta$ do
        Let $S_\beta = \bigcup_{j \in J_{\mathrm{ALG}}} \bigcup_\gamma \Gamma^S_{\beta,\gamma}(j)$.
        for all $h \in S_\beta$ do
            Independently add $h$ to $S_{\mathrm{ALG}}$ with probability $\min\{1, \beta \cdot 12 \log |S| \log(|S|^2 m) \ln n\}$.
        end for
    end for
else if $|J_1| \ge |J_0|/2$ then
    Choose $\langle \beta, \gamma, D\rangle \in P_1$ that maximize the cardinality $|J^D_{\beta,\gamma}|$, and let $J_2 = J^D_{\beta,\gamma}$.
    Let $S_{\tilde{D}} = \{h \in S \mid \{j' \in J_2 \mid h \in \Gamma^S_{\beta,\gamma}(j')\}| \in [\tilde{D}, 2\tilde{D}]\}$ for every $\tilde{D} \in \{2^{s-1} \mid s \in \lceil \log |J_2| \rceil\}$.
    Choose $\tilde{D}$ that maximizes the cardinality $|\{\langle j, h\rangle \in J_2 \times S_{\tilde{D}} \mid h \in \Gamma^S_{\beta,\gamma}(j)\}|$, and let $\tilde{S} = S_{\tilde{D}}$.
    Choose $h_0 \in \tilde{S}$ that maximizes the total LP value $\sum_{j \in J_2 : \Gamma^S_{\beta,\gamma}(j) \ni h_0} \hat{x}^{(h_0)}_j$.
    Let $J_{\mathrm{ALG}} = \{j \in J_2 \mid h_0 \in \Gamma^S_{\beta,\gamma}(j)\}$.
    Let $S_{\mathrm{ALG}} = \emptyset$.
    for every $h \in \bigcup_{j \in J_{\mathrm{ALG}}} \Gamma_S(\Gamma_R(j))$ do
        Independently add $h$ to $S_{\mathrm{ALG}}$ with probability $\min\{1, \hat{w}^{h_0}_h \cdot 4\gamma \ln n/(x_0 \beta)\}$.
    end for
end if
```

Efficient Algorithms and Hardness Results for the Weighted k-Server Problem

Anupam Gupta ✉
Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

Amit Kumar ✉
Computer Science and Engineering Department, Indian Institute of Technology, Delhi, India

Debmalya Panigrahi ✉
Computer Science, Duke University, Durham, NC, USA

Abstract

In this paper, we study the weighted k-server problem on the uniform metric in both the offline and online settings. We start with the offline setting. In contrast to the (unweighted) k-server problem which has a polynomial-time solution using min-cost flows, there are strong computational lower bounds for the weighted k-server problem, even on the uniform metric. Specifically, we show that assuming the unique games conjecture, there are no polynomial-time algorithms with a sub-polynomial approximation factor, even if we use c-resource augmentation for $c < 2$. Furthermore, if we consider the natural LP relaxation of the problem, then obtaining a bounded integrality gap requires us to use at least ℓ resource augmentation, where ℓ is the number of distinct server weights. We complement these results by obtaining a constant-approximation algorithm via LP rounding, with a resource augmentation of $(2 + \varepsilon)\ell$ for any constant $\varepsilon > 0$.

In the online setting, an $\exp(k)$ lower bound is known for the competitive ratio of any randomized algorithm for the weighted k-server problem on the uniform metric. In contrast, we show that 2ℓ-resource augmentation can bring the competitive ratio down by an exponential factor to only $O(\ell^2 \log \ell)$. Our online algorithm uses the two-stage approach of first obtaining a fractional solution using the online primal-dual framework, and then rounding it online.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Online Algorithms, Weighted k-server, Integrality Gap, Hardness

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.12

Category APPROX

Funding *Anupam Gupta*: NSF awards CCF-1955785, CCF-2006953, and CCF-2224718.
Debmalya Panigrahi: NSF awards CCF-1750140 (CAREER) and CCF-1955703.

1 Introduction

The k-Server problem is a foundational problem in online algorithms and has been extensively studied over the last 30 years [10]. In this problem, there are a set of k servers that must serve requests arriving online at the vertices of an n-point metric space. The goal is to minimize the total movement cost of the servers. The k-Server problem was defined by Manasse et al. [22], who also showed a lower bound of k on the competitive ratio of any deterministic algorithm for this problem. Koutsoupias and Papadimitriou [20] gave a $(2k-1)$-compeititive algorithm for k-Server. There has been much progress in the recent past on obtaining randomized algorithms with polylogarithmic (in k and n) competitive ratio [2, 13, 21, 14]. The Weighted k-Server version of this problem, introduced by Fiat and Ricklin [17], allows the servers to have non-uniform positive weights; the cost of moving a server is now scaled by its weight. In this paper, we consider the Weighted k-Server problem on a uniform metric, namely when all n points of the metric space are at unit

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 12; pp. 12:1–12:19
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

distance from each other, which means that the cost of moving a server between any two distinct points is simply the weight of the server. Note that the corresponding unweighted problem for the uniform metric is the extensively studied PAGING problem [10]. Indeed, one of the original motivations for studying the WEIGHTED k-SERVER problem came from a version of paging with non-uniform replacement costs for different cache slots [17]. In the rest of this paper, we will implicitly assume that the underlying metric space is a uniform metric.

The original paper of Fiat and Ricklin [17] introducing the WEIGHTED k-SERVER problem (on uniform metrics) gave a deterministic algorithm with a competitive ratio of about $2^{2^{2k}}$. They also showed a lower bound of $(k+1)!/2$ for deterministic algorithms. Chiplunkar and Viswanathan [15] improved this lower bound to $(k+1)!-1$, and gave a randomized algorithm that is 1.6^{2^k}-competitive against *adaptive* online adversaries; this also implies a deterministic competitive ratio of $2^{2^{k+1}}$ using the simulation technique of Ben-David et al. [8]. Bansal, Elias, and Koumutsos [6] showed that this competitive ratio is essentially tight for deterministic algorithms by showing a lower bound of $2^{2^{k-4}}$. They also gave a deterministic *work function algorithm* with a competitive ratio of $2^{2^{k+O(\log k)}}$. If the number of distinct server weights is ℓ and there are k_j servers of weight W_j, then the competitive ratio of their algorithm is $\exp(O(\ell k^3 \prod_{j=1}^{\ell}(k_j+1)))$, which is an exponential improvement over the general bound when ℓ is a constant. Unlike the k-SERVER and PAGING problems, it is unknown if randomization qualitatively improves the competitive ratio for WEIGHTED k-SERVER, although the best known lower bound for randomized algorithms against oblivious adversaries is only singly exponential in k [1] as against the doubly exponential lower bound for deterministic algorithms.

The above competitive ratios depend only on k, and are independent of the size n of metric space. Moreover, the hard instances are for metric spaces with the number of points n that are exponentially larger than the number of servers k. This is not a coincidence, since better algorithms exist for smaller values of n. Indeed, the WEIGHTED k-SERVER problem can be modeled as a metrical task system, where each state ω is a configuration (specifying the location of each of the k servers), and the distance between any two states ω, ω' is the cost to move between the configurations. Since there are $N = n^k$ states, one can obtain an n^k-competitive deterministic algorithm [11], and an $O(\text{poly}(k \log n))$-competitive randomized algorithm against *oblivious* adversaries [7, 3, 12, 16]; all these algorithms use $\text{poly}(n^k)$ time to explicitly maintain and manipulate the entire metric space, and hence are not efficient.

In this paper we ask: *is it possible to get efficient (randomized)* online *algorithms that have competitive ratios of the form* $\text{poly}(k \log n)$*, or even better? Is it possible to get such approximation ratios even in the* offline *setting?* We show that obtaining improved competitive or approximation ratios in polynomial time is possible, provided we allow for *resource augmentation* in the number of servers.

Resource augmentation in online algorithms has been widely studied in paging and scheduling settings (see e.g. [19, 23]). It is often a much needed assumption that allows for obtaining bounded or improved competitive ratios for such problems. Bansal et al. [5] studied the k-SERVER problem on trees under resource augmentation.

1.1 Our Results

Our first result establishes computational hardness of approximating the WEIGHTED k-SERVER problem in the offline setting. Unlike PAGING or k-SERVER, which are exactly solvable offline in polynomial time, we show that under the Unique Games conjecture, the WEIGHTED k-SERVER problem cannot be approximated to any subpolynomial factor even when we allow c-resource augmentation for any constant $c < 2$.

▶ **Theorem 1** (Hardness). *For any constant $\varepsilon > 0$, it is UG-hard to obtain an $N^{1/2-\varepsilon}$-approximation algorithm for* Weighted *k*-Server *with two weight classes, even when we are allowed c-resource augmentation for any constant $c < 2$. Here N represents the size of the input (including the request sequence length).*

Next, we show that the natural time indexed LP relaxation for Weighted k-Server (see LP) has a large integrality gap, unless we allow for a resource augmentation of almost ℓ, the number of distinct server weights.

▶ **Theorem 2** (Integrality Gap). *For any constant $\varepsilon > 0$, the integrality gap of the relaxation LP for* Weighted *k*-Server *is unbounded, even with $(\ell - \varepsilon)$-resource augmentation.*

It is worth noting that an optimal fractional solution to LP can be easily rounded to give an ℓ-approximation ratio with ℓ-resource augmentation. Indeed, we know that for each request, there exists a weight class which services this request to an extent of at least $1/\ell$. We can then scale this fractional solution by a factor ℓ and reduce this problem to ℓ instances of standard Paging problem. The integrality gap result shows that any rounding algorithm with bounded competitive ratio must incur almost ℓ-resource augmentation. We complement this integrality gap result with our main technical result, which gives an offline $O(1/\varepsilon)$-approximation with $(2+\varepsilon)\ell$-resource augmentation, for any $\varepsilon \in (0, 1)$.

▶ **Theorem 3** (Offline Algorithm). *Let $\mathcal{I}$ be an instance of* Weighted *k*-Server *with k_j servers of weight W_j for all $j \in [\ell]$. For any $\varepsilon \in (0, 1)$, there is a polynomial time algorithm for $\mathcal{I}$ that uses at most $2(1+\varepsilon)\ell \cdot k_j$ servers of weights W_j for each $j \in [\ell]$ and has server movement cost at most $O(1/\varepsilon)$ times the optimal cost of $\mathcal{I}$.*

Finally, we obtain an online algorithm for Weighted k-Server with 2ℓ-resource augmentation. The competitive ratio of the online algorithm is $O(\ell^2 \log \ell)$. (In constrast to the offline setting, it is no longer clear how to achieve an ℓ-competitive algorithm even if we augment the number of servers by a factor of ℓ.)

▶ **Theorem 4** (Online Algorithm). *Let $\mathcal{I}$ be an instance of* Weighted *k*-Server *with k_j servers of weight W_j for all $j \in [\ell]$. There is a randomized (polynomial time) online algorithm for $\mathcal{I}$ that uses at most $2\ell k_j$ servers of weights W_j for each $j \in [\ell]$ and has expected server movement cost at most $O(\ell^2 \log \ell)$ times the optimal cost of $\mathcal{I}$.*

Since $\ell \leq k$, the competitive ratio of the online algorithm is $O(k^2 \log k)$. This implies that an $O(\ell^2)$-resource augmentation achieves at least an exponential improvement in the competitive ratio of the Weighted k-Server problem. Moreover, by rounding the weights to powers of 2, we can assume that $\ell \leq O(\log W)$, where W is the aspect ratio of the server weights. Hence, the competitive ratio of the online algorithm is $O(\log^2 W \log\log W)$. Finally, note that for $\ell = O(1)$, the above theorem gives a $O(1)$-competitive online algorithm with $O(1)$-resource augmentation. This can be seen as a generalization of the classic result for the Paging problem that achieves a randomized competitive ratio of $O(\log \frac{k}{k-h+1})$ where the algorithm's cache has k slots while the adversary's has only $h < k$ slots [24].

1.2 Our Techniques

In this section, we give an overview of the main techniques in the paper. The UG hardness of Weighted k-Server is based on a reduction from the Vertex Cover problem. Given an instance of the vertex cover problem, the corresponding Weighted k-Server consists of one point in the uniform metric space for each vertex of the graph. The main observation is that if we know the minimum vertex cover size, we can keep one heavy weight server at each point corresponding to this vertex cover, which never change their positions. One can then generate an input sequence where the optimal solution pays a small cost, whereas an algorithm which does not cover an edge using heavy servers pays a much higher cost. The UG-hardness result for Vertex Cover translates to a corresponding resource augmentation lower bound for Weighted k-Server. Extending this approach to more than two weight classes (with stronger lower bounds on resource augmentation) turns out to be more challenging because the length of the input sequence becomes exponential in n. Instead, we show that the natural LP relaxation has a large integrality gap. The large gap instance consists of cycling through a sequence of subsets of the metric spaces with carefully varying frequency. The fractional solution is able to maintain a low cost by uniformly spreading servers over such cycles, but the integral solution is forced to service some of the cycles by small number of servers only.

Our main technical result shows how to round a solution to the LP relaxation. The relaxation has both covering and packing type constraints, and the rounding carefully addresses one set of constraints without violating the other. We first scale the LP by a factor of about 2ℓ, thus increasing both the resource augmentation and the cost. As a result, each request σ_t is covered to an extent of 2ℓ, and we can split this coverage across those weight classes which cover σ_t to an extent of at least 1. Now for a fixed weight class, we consider the requests which are covered by it to an extent of at least 1. We show how to integrally round this solution so that this coverage property is satisfied and yet, we do not violate any packing constraint. After this, we show that the packing constraints can be ignored. This allows to scale down the LP solution by a factor ℓ (which saves the cost by this factor) and uses total unimodularity of the constraint matrix to round it.

We extend our approximation algorithm to the online setting. The first step is to maintain an online fractional solution to the LP relaxation. Standard (weighted) paging algorithms for this problem rely on the fact that even the optimal offline algorithm needs to place a server at a requested location. But this turns out to be trickier here as we do not know the weight of the server which serves this location in the optimal solution. So we serve a request by ensuring that fractional mass from each weight classes is transferred at the same rate. The overall analysis proceeds by a careful accounting in the potential function. The online fractional solution satisfies the stronger guarantee that each request is served by servers of a particular weight class only. This allows us to reduce the rounding problem to independent instances of the Paging problem.

We now give an overview of the rest of the submission. In §2, we give details of the integrality gap construction; we defer the UG hardness proof to §A. The offline rounding of the LP relaxation is given in §3, and then we extend this algorithm to the online case in §4.

1.3 Preliminaries

In the Weighted k-Server problem on the uniform metric, we are given a set of n points $V = \{1, \ldots, n\}$, such that $d(v, v') = 1$ for each $v \neq v'$. There are k servers, labeled $1, \ldots, k$, with server i having weight $w_i \geq 0$. The input specifies a request sequence $(\sigma_1, \ldots, \sigma_T)$ of length T, with each request σ_t arriving at *time* t being a point in V. A solution $f : [k] \times \{0, \ldots, T\} \to V$ specifies the position of each server at each time $t \in [T]$ (where the initial positions $f(i, 0)$ are specified as part of the problem statement) such that for each time t there exists some server i_t such that $f(i_t, t) = \sigma_t$. The cost of the solution f is the total weighted distance travelled by the servers, i.e.,

$$1/2 \sum_{i=1}^{k} w_i \sum_{t=1}^{T} \mathbb{1}[f(i,t) \neq f(i,t-1)].$$

The goal is to find a solution with the minimum cost. We say that an instance has ℓ *weight classes* if the set $\{w_1, \ldots, w_k\}$ has cardinality ℓ. For an instance with ℓ different weight classes, we denote the distinct weights by $W_1, \ldots, W_\ell$, and let k_j denote the number of servers of weight W_j, with $\sum_j k_j = k$. For such an instance and a parameter $c \geq 1$, we say that the algorithm uses *c-resource augmentation* if it uses $\lfloor ck_j \rfloor$ servers of weight W_j for each $j = 1, \ldots, \ell$.

We now describe the natural LP relaxation for WEIGHTED k-SERVER. It has a variable $x(v, j, t)$ for each request time t, weight class $j \in [\ell]$ and vertex $v \in V$; it denotes the fractional mass of servers of weight W_j that are present at point v at time t. Let σ_t denote the vertex requested at time t. It is easy to verify that this is a valid relaxation.

$$\min {}^1\!/\!_2 \sum_{j\in[\ell]} W_j \sum_t \sum_{v\in V} |x_{v,j,t} - x_{v,j,t-1}| \tag{LP}$$

$$\sum_{v\in V} x_{v,j,t} \leq k_j \qquad \forall t, j \in [\ell] \tag{1}$$

$$\sum_{j\in[\ell]} x_{\sigma_t,j,t} \geq 1 \qquad \forall t \tag{2}$$

$$x_{v,j,t} \geq 0 \qquad \forall t, v \in V, j \in [\ell]$$

2 An Integrality Gap for the Natural Linear Program

In this section, we show that the relaxation LP for WEIGHTED k-SERVER has a large integrality gap, unless we allow for a resource augmentation of almost ℓ, the number of distinct server weights.

Recall that the ℓ weights are denoted $W_1 \gg \cdots \gg W_\ell$, and there are k_j servers of weight W_j. Our theorem is the following:

▶ **Theorem 2** (Integrality Gap). *For any constant $\varepsilon > 0$, the integrality gap of the relaxation LP for WEIGHTED k-SERVER is unbounded, even with $(\ell - \varepsilon)$-resource augmentation.*

An Instance for Two Classes. To gain some intuition, we first consider the special case of $\ell = 2$, and prove the result without giving any resource augmentation. There are $n/2$ servers of weight W and $n/4$ servers of weight 1, thereby giving a total of $k = 3n/4$ servers. The input is given in "phases". Each phase is specified by a distinct subset S of V, where $|S| = n/2$. During the phase corresponding to a subset S, we cycle through all subsets S' of S with $|S'| = |S|/2 = n/4$. Given such a subset S' of S, we send requests which cycle through the points in S' for L times, where L is large enough.

One fractional solution for such a sequence is defined as follows: we assign $1/2$ unit of weight-W server at each of the n locations. During the phase for a subset S, we assign $1/2$ unit of server of unit weight at each of the locations in S. The cost of the fractional solution is at most $Z := \binom{n}{n/2} \cdot n/4$ (not accounting for the initial movement of the servers). However, an integral solution either moves at least one heavy server, or else pays at least L during one of the phases, thereby must pay at least $\min(W, L)$. Assuming $W, L \gg Z$ gives an arbitrarily large integrality gap. (We can account for the initial movement of the fractional servers by repeating the process some M times: the integral solution would pay at least $\min(W, L)$ in each such iteration and the fractional solution would pay at most Z, so that the initial movement cost would get amortized away.)

The Instance for ℓ Classes. We extend this construction to larger values of ℓ by defining phases in a recursive manner on a nested sequence of subsets of V, with each phase containing several repetitions of the same sequence. Instead of decreasing by a factor 2, the number of servers of each weight class now goes down by a factor of $C \geq \ell$. This allows the integrality gap result to hold even when the integral solution is allowed a resource augmentation of nearly ℓ.

For some $r \leq \ell - 1$, we call a tuple $(S_0, \ldots, S_r)$ *valid* if (i) $S_0 = V$ and each $S_j \subseteq S_{j-1}$, and (ii) $|S_j| = |S_{j-1}|/C = n/C^j$. The request sequence is generated by calling Algorithm 1 with the trivial valid sequence $(S_0 = V)$. Given a valid tuple $(S_0, \ldots, S_r)$, the procedure cycles through all subsets $S \subseteq S_r$ of size $|S_r|/C$ and recursively calls $\mathsf{Generate}(S_0, \ldots, S_r, S)$; this process is repeated L_r times. Finally, in the base case when $r = \ell - 1$, it cycles through all the locations in S_ℓ for $L_{\ell-1}$ times. For a suitably large choice of M, we define for each $r \in [\ell]$:

$$L_r := M^r \qquad \text{and} \qquad W_r := M^{\ell - r}. \tag{3}$$

Finally, the number of servers of weight W_r is given by $k_r := \frac{n}{\ell C^{r-1}}$.

Algorithm 1 Procedure $\mathsf{Generate}(S_0, S_1, \ldots, S_r)$.

```
1.1 Input: A valid tuple $(S_0, S_1, \ldots, S_r)$
1.2 repeat
1.3     if $r$ is equal to $\ell - 1$ then
1.4         Send a request at each location in $S_{\ell-1}$.
1.5     else
1.6         for each subset $S$ of $S_r$ with $|S| = \frac{|S_r|}{C}$ do
1.7             // Move $1/\ell$ mass of servers of weight $W_{r+2}$ to $S$
1.8             Call Generate$(S_0, \ldots, S_r, S)$.
1.9 until $L_r$ iterations have occurred
```

Analyzing the Integrality Gap

We bound the cost of the optimal fractional solution for the above input sequence.

▶ **Lemma 5.** *There is a fractional solution of total cost $O(f(n)M^{\ell-2})$ for the input sequence constructed by Algorithm 1, where $f(n)$ is a function solely of n.*

Proof. Our fractional solution maintains the invariant: when the procedure $\mathsf{Generate}(S_0, \ldots, S_r)$ is called, we have $1/\ell$ fractional mass of servers of weight $W_1, \ldots, W_{r+1}$ respectively at each location in S_r. For the base case $r = 0$, we place $1/\ell$ server mass at each location in $S_0 = V$; recall that $k_1 = n/\ell$. For the inductive step, suppose this invariant is satisfied for a certain value of r where $0 \leq r < \ell - 1$; we need to show that it is satisfied for $r + 1$ as well. Indeed, the induction hypothesis implies that we have $1/\ell$ amount of server mass of weight $W_1, \ldots, W_{r+1}$ at each location in S_r, and hence at each location in S_{r+1}. Moreover, as line 1.7 indicates, we move $1/\ell$ fractional mass of servers of weight W_{r+2} to each location in S_{r+1} to satisfy the invariant condition. This costs $W_{r+2}\, k_{r+2}/\ell$; moreover, this is feasible because the total number of servers of weight W_{r+2} needed is $\frac{|S_{r+1}|}{\ell} = \frac{n}{\ell C^{r+1}} = k_{r+2}$. Finally, when $r = \ell - 1$, the invariant shows that 1 unit of server mass is present at each of the locations in S_ℓ, and hence the requests generated in line 1.4 can be served without any additional movement of servers.

We now account for the movement cost. The total server movement cost during $\mathsf{Generate}(S_0, \ldots, S_r)$ (not counting the movement costs in the recursive calls) is at most $O(L_r\, k_{r+1}\, W_{r+2}) = O(k_{r+1}\, M^{\ell-2})$. Since $k_{r+1} \le n$ and the number of calls to Generate is a function only of n, the overall movement cost can be expressed as $O(f(n) \cdot M^{\ell-2})$. (Again, by repeating the entire process multiple times we can amortize away the initial movement cost; we avoid this step for the sake of clarity.) ◀

The next lemma shows that any integral solution must have much higher cost.

▶ **Lemma 6.** *Let $\varepsilon > 0$ be a small enough constant. Assume that the integral solution is allowed $(\ell - \varepsilon)k_r$ servers of weight W_r for each $r \in [\ell]$. Any integral solution for the input sequence generated by Algorithm 1 (with $C = \ell/\varepsilon$) has movement cost at least $M^{\ell-1}$.*

We defer the proof to Appendix B; combining Lemma 5 and Lemma 6 proves Theorem 2.

3 An Offline Algorithm via LP Rounding

We now show an algorithm for the offline setting, that rounds any fractional solution to the LP relaxation (LP), and achieves the following guarantee:

▶ **Theorem 3** (Offline Algorithm). *Let $\mathcal{I}$ be an instance of Weighted k-Server with k_j servers of weight W_j for all $j \in [\ell]$. For any $\varepsilon \in (0,1)$, there is a polynomial time algorithm for $\mathcal{I}$ that uses at most $2(1+\varepsilon)\ell \cdot k_j$ servers of weights W_j for each $j \in [\ell]$ and has server movement cost at most $O(1/\varepsilon)$ times the optimal cost of $\mathcal{I}$.*

Instead of working with the relaxation (LP), we work with an equivalent relaxation which turns out to be easier to interpret. For each vertex $v \in V$, index $j \in [\ell]$ and time interval I, we have a variable $y_{v,j,I}$, which denotes the fractional mass of server of weight W_j residing at v during the entire time interval I. The variable $x_{v,j,t}$ used in (LP) can be expressed as follows:

$$x_{v,j,t} = \sum_{I: t \in I} y_{v,j,I}. \tag{4}$$

Let $\mathbf{I}$ denote the set of all intervals during the request timeline. The new linear program relaxation for Weighted k-Server is the following:

$$\begin{aligned}
&\min\ {}^1\!/_2 \sum_{j\in[\ell]} W_j \sum_{I\in\mathbf{I}} \sum_{v\in V} y_{v,j,I} && && \text{(LP2)}\\
&\text{s.t.}\ \sum_{j\in[\ell]} \sum_{I:t\in I} y_{\sigma_t,j,I} \ge 1 && \forall t && (5)\\
&\qquad \sum_{v\in V} \sum_{I:t\in I} y_{v,j,I} \le k_j && \forall t, j \in [\ell] && (6)\\
&\qquad\qquad y_{v,j,I} \ge 0 && \forall t, j \in [\ell], v \in V. &&
\end{aligned}$$

Note that the covering constraint (5) enforces having at least one unit of (fractional) server mass at the location σ_t requested for each time t. The packing constraint (6) enforces that the total (fractional) server mass of weight W_j used at any time t is at most the number of servers of this weight, namely k_j. Given a solution $y_{v,j,I}$ to LP2, the variables $x_{v,j,t}$ defined using (4) define a feasible solution to LP of the same cost.

Fix any constant $\varepsilon \in (0,1)$. We now prove Theorem 3 by rounding an optimal fractional solution $y_{v,j,I}$ to LP2. The rounding algorithm has two stages. The first stage scales and discretizes the LP variables to integers such that

Algorithm 2 Procedure $\mathsf{ScaleRound}(x, y, v, W_j)$.

```
Input: A fractional solution $(y_{v,j,I}, x_{v,j,t})$ to LP2, a location $v$ and a weight $W_j$
Initialize variables $\bar{y}_{v,j,I}$ to 0 for all intervals $I$.
(Scale): Define $\widetilde{y}_{v,j,I} = (2 + \varepsilon/2)\,\ell \cdot y_{v,j,I}$ and therefore,
    $\widetilde{x}_{v,j,t} = \sum_{I:t\in I} \widetilde{y}_{v,j,I} = (2 + \varepsilon/2)\,\ell \cdot x_{v,j,t}$ for each $I \in \mathbf{I}$.
(Round): for $h = 1, 2, \ldots, \ell$ do
    Initialize LastEvent = DOWN, LastTime = 0.
    repeat
        if LastEvent = UP then
            Let $t$ be the first DOWN after LastEvent
            Update LastEvent = DOWN, LastTime = $t$.
        else
            (LastEvent = DOWN) Let $t$ be the first UP after LastEvent
            Add $I = [\mathsf{LastTime}, t)$ to $\mathbf{I}_{v,j}(h)$ and increase $\bar{y}_{v,j,I}$ by 1.
            Update LastEvent = DOWN, LastTime = $t$.
    until we have reached the end of the timeline $[0, T]$
```

1. the packing constraints are satisfied up to a factor of $(2 + \varepsilon)\ell$,
2. the covering constraints are satisfied with a scaled covering requirement of ℓ instead of 1, i.e., $\sum_j \sum_{I:t\in I} y_{\sigma_t,j,I} \geq \ell$, for all times t, and
3. the cost of the fractional solution increases by a factor of $O(\ell/\varepsilon)$.

In the second stage, we remove the packing constraints from the LP; this results in the resulting interval covering LP being integral. Next, we scale the solution from the first stage down by ℓ, getting a feasible fractional solution to the standard LP relaxation for the interval covering problem. Finally, we use the integrality of the interval covering LP relaxation to obtain an integral solution for LP2. We present these two stages in the next two sections.

3.1 Stage I: Scaling and Discretization

The first stage of the rounding algorithm operates independently on each location $v \in V$ and for each server weight W_j; the formal algorithm $\mathsf{ScaleRound}(x, y, v, W_j)$ is given in Algorithm 2. We work with both the $y_{v,j,I}$ variables and the equivalent $x_{v,j,t}$ variables defined in (4); this representational flexibility makes it convenient to explain the algorithm. To begin, we scale the LP variables $y_{v,j,I}$ by a factor $(2 + \varepsilon/2)\ell$ to obtain $\widetilde{y}_{v,j,I}$ (we also define the auxiliary variables $\widetilde{x}_{v,j,t}$ by scaling $x_{v,j,t}$ similarly).

Discretization. Next we discretize the scaled variables $\widetilde{y}_{v,j,I}$ and $\widetilde{x}_{v,j,t}$ to nonnegative integers $\bar{y}_{v,j,I}$ and $\bar{x}_{v,j,t}$ respectively. To start, let us describe the discretization of $\widetilde{x}_{v,j,t}$ to obtain $\bar{x}_{v,j,t}$. Intuitively, we would like to define $\bar{x}_{v,j,t}$ as $\lfloor \widetilde{x}_{v,j,t} \rfloor$, i.e., the largest step function with unit step sizes entirely contained in $\widetilde{x}_{v,j,t}$, but this can amplify small fluctuations around integer values, and hence may increases the cost. To avoid this, we introduce *hysteresis* in our discretization, by setting different thresholds for increasing and decreasing the value of $\widetilde{x}_{v,j,t}$. We view $\widetilde{x}_{v,j,t}$ as a time-varying profile and define horizontal *slabs* in it corresponding to the restriction of the range of $\widetilde{x}_{v,j,t}$ to $[h, h + 1)$ for some integer h. For each such slab, we identify intervals I of width at most 1 and at least $1/2$ and set the increase the corresponding

$\bar{y}_{v,j,I}$ value by 1. In more detail, for each such level h, we identify a subset $\mathbf{I}_{v,j}(h)$ of intervals for which the corresponding $\bar{y}_{v,j,I}$ variable is to be increased by 1. We identify an alternating sequence of *up* and *down* events in the timeline $[0, T]$ as follows:

- UP event: At time t, there is an UP event at level h if $\widetilde{x}_{v,j,t^-} < h$ and $\widetilde{x}_{v,j,t} \geq h$, and the previous event at level h was a DOWN event.
- DOWN event: At time t, there is a DOWN event at level h if the previous event at level h was an UP, and $\widetilde{x}_{v,j,t^-} > h - \varepsilon/2$ and $\widetilde{x}_{v,j,t} \leq h - \varepsilon/2$, or $t = T$, the end of the timeline. (The reader should think of $\varepsilon/2$ as the "hysteresis gap" between the up and down events at any level.)

To make the definition complete, we set $\widetilde{x}_{v,j,t}$ to 0 at $t = 0^-$ and at $t = T^+$, and start with a DOWN at time 0. Finally, we add intervals stretching from each UP to the next DOWN to the set $\mathbf{I}_{v,j}(h)$ of intervals. By construction, these intervals are mutually disjoint. Finally, whenever an interval I is added to such a set $\mathbf{I}_{v,j}(h)$, we increment the corresponding variable $\bar{y}_{v,j,I}$. Thus we have:

$$\bar{y}_{v,j,I} = |\{h : I \in \mathbf{I}_{v,j}(h)\}|, \text{ and correspondingly, } \bar{x}_{v,j,t} = \sum_{I:t\in I} \bar{y}_{v,j,I}.$$

The next lemma shows that $\bar{x}_{v,j,t}$ can be thought of as a discretized form of $\widetilde{x}_{v,j,t}$:

▶ **Lemma 7.** *The following holds for variables $\bar{x}_{v,j,t}$:*

$$\widetilde{x}_{v,j,t} - 1 < \bar{x}_{v,j,t} < \widetilde{x}_{v,j,t} + \varepsilon/2. \tag{7}$$

Proof. Suppose $\widetilde{x}_{v,j,t} \in [r, r+1)$. Consider the **for** loop in line 2.4 in Algorithm 2 for a value $h \leq r$. We claim that at time t, the value of the variable LastEvent must be UP. Suppose not. Let t' be the value of LastTime at time t (i.e., t' is the last time before and including t when an UP or a DOWN occurred). Since a DOWN event happened at time t', $\widetilde{x}_{v,j,t'} < h$. Since $\widetilde{x}_{v,j,t} \geq h$, an UP event must occur during $(t', t]$, a contradiction. Therefore must have added an interval containing time t to $\mathbf{I}_{v,j}(h)$. Thus, $\bar{x}_{v,j,t}$ gets increased during each such iteration, i.e., $\bar{x}_{v,j,t} \geq r > \widetilde{x}_{v,j,t} - 1$. This proves the first inequality in (7).

We now prove the second inequality. Let h be an integer satisfying $h \geq \widetilde{x}_{v,j,t} + \varepsilon/2$. Consider the iteration of the **for loop** in Algorithm 2 for this particular value of h. We claim that the value of the variable LastEvent at time t must be DOWN. Suppose not, and let t' denote the value of the variable LastTime. Then an UP happened at time t' and so $\widetilde{x}_{v,j,t'} \geq h$. Since $\widetilde{x}_{v,j,t} \leq h - \varepsilon/2$, a DOWN event must have happened during $(t', t]$, a contradiction. Hence, we do not add any interval containing time t to the set $\mathbf{I}_{v,j}(h)$. Therefore, $\bar{x}_{v,j,t} < \widetilde{x}_{v,j,t} + \varepsilon/2$, which proves the second inequality in (7). ◀

The next lemma establishes the key properties of the variables $\bar{y}_{v,j,I}$ and $\bar{x}_{v,j,t}$.

▶ **Lemma 8.** *The following properties hold the for the variables $\bar{y}_{v,j,I}$:*

(i) *(Cost) The LP cost increases by at most $O(\ell/\varepsilon)$ when the original variables $y_{v,j,I}$ are replaced by the new variables $\bar{y}_{v,j,I}$:*

$$\sum_{v,j,I} W_j \cdot \bar{y}_{v,j,I} \leq O(\ell/\varepsilon) \cdot \sum_{v,j,I} W_j \cdot y_{v,j,I}.$$

(ii) *(Covering) The variables $\bar{y}_{v,j,I}$ satisfy the scaled covering constraints of* (LP2)

$$\sum_{j,I:t\in I} \bar{y}_{v,j,I} \geq \ell \quad \forall t.$$

(iii) *(Packing) The variables $\bar{y}_{v,j,I}$ approximately satisfy the packing constraints of* (LP2)*:*

$$\sum_{v,I:t\in I} \bar{y}_{v,j,I} \le (2+\varepsilon)\ell k_j \quad \forall j \in [\ell], t.$$

Proof. We first prove the cost bound: the cost of the solution $\bar{y}_{v,j,I}$ is the weight of all intervals added to the sets $\mathbf{I}_{v,j}(h)$ for all v, j, h. I.e.,

$$\sum_{v,j,I} W_j \cdot \bar{y}_{v,j,I} = \sum_{v,j} W_j \cdot \sum_{h\in[\ell]} |\mathbf{I}_{v,j}(h)|. \tag{8}$$

Fix a vertex v and indices j, h. For a non-negative number x, and non-negative integer h, define the *h-level truncation* of x to be $\mathsf{trunc}_h(x) := \min(1, (x-h)^+)$, where $(a)^+ := \max(a, 0)$ for any real a. Observe that $x = \sum_{h\ge 0} \mathsf{trunc}_h(x)$. In fact, for any two non-negative integers x and y:

$$|x-y| = \sum_{h'\ge 0} |\mathsf{trunc}_{h'}(x) - \mathsf{trunc}_{h'}(y)|. \tag{9}$$

Now let $I_1 = [s_1, t_1), \ldots, I_u = [s_u, t_u)$ be the intervals added to $\mathbf{I}_{v,j}(h)$ (in left to right order). Define $t_0 = 0$. We know that for any $i \in [u]$, an UP happens at s_u and a DOWN happens at t_u. Therefore, $\mathsf{trunc}_h(\widetilde{x}_{v,j,s_u}) - \mathsf{trunc}_h(\widetilde{x}_{v,j,t_{u-1}}) \ge \varepsilon/2$. Hence,

$$\begin{aligned}\varepsilon W_j/2 \cdot |\mathbf{I}_{v,j}(h)| &\le W_J \cdot \sum_{i=1}^{u} |\mathsf{trunc}_h(\widetilde{x}_{v,j,s_u}) - \mathsf{trunc}_h(\widetilde{x}_{v,j,t_{u-1}})| \\ &\le W_j \cdot \sum_{t'=1}^{T} |\mathsf{trunc}_h(\widetilde{x}_{v,j,t-1}) - \mathsf{trunc}_h(\widetilde{x}_{v,j,t})|,\end{aligned}$$

where the last inequality follows from triangle inequality. Summing over all h and using (9), we get

$$\varepsilon W_j/2 \cdot \bar{y}_{v,j,I} \le W_j \cdot \sum_{t'=1}^{T} |\widetilde{x}_{v,j,t-1}) - \widetilde{x}_{v,j,t}|.$$

Summing over all vertices v and indices $j \in [\ell]$, we see that the cost of the solution $\bar{y}_{v,j,I}$ is at most $2/\varepsilon$ times that of $\widetilde{y}_{v,j,I}$. Finally, the fact that $\widetilde{y}_{v,j,I}$ are obtained by scaling $y_{v,j,I}$ by a factor $(2+\varepsilon/2)\ell$, we get the desired bound on the cost of $\bar{y}_{v,j,I}$ solution.

Next, we prove the covering property. Since $x_{v,j,t}$ is a feasible solution to LP2, we have for any time t:

$$\sum_j x_{\sigma_t,j,t} \ge 1, \text{ and therefore, } \sum_j \widetilde{x}_{\sigma_t,j,t} \ge (2+\varepsilon/2)\ell.$$

Using Lemma 7, we have $\widetilde{x}_{\sigma_t,j,t} < \bar{x}_{\sigma_t,j,t} + 1$, so

$$\sum_{j\in\ell} (\bar{x}_{\sigma_t,j,t} + 1) > (2+\varepsilon/2)\ell, \text{ and therefore, } \sum_j \bar{x}_{\sigma_t,j,t} > \ell.$$

Finally, we prove the packing property. Since $x_{v,j,t}$ is a feasible solution to the LP, we have for any $j \in [\ell]$ and time t,

$$\sum_v x_{v,j,t} \le k_j, \text{ and therefore, } \sum_v \widetilde{x}_{v,j,t} \le (2+\varepsilon/2)\ell k_j.$$

Again Lemma 7 gives $\widetilde{x}_{v,j,t} > \bar{x}_{v,j,t} - \varepsilon/2$, which implies

$$\sum_j (\bar{x}_{v,j,t} - \varepsilon/2)^+ < (2 + \varepsilon/2)\ell k_j. \tag{10}$$

Since $\bar{x}_{v,j,t}$ is a nonnegative integer,

$$\bar{x}_{v,j,t} > 0 \implies \bar{x}_{v,j,t} \geq 1 \overset{\text{Lemma 7}}{\implies} \widetilde{x}_{v,j,t} > \bar{x}_{v,j,t} - \varepsilon/2 \geq 1 - \varepsilon/2.$$

Since $\sum_v \widetilde{x}_{v,j,t} \leq k_j$, it follows that the number of locations v for which $\bar{x}_{v,j,t} > 0$ is at most $\frac{k_j}{1-\varepsilon/2} < 2k_j$, if $\varepsilon < 1$. Using this fact in Equation (10), we get

$$\begin{aligned}\sum_v \bar{x}_{v,j,t} &= \sum_{v:\bar{x}_{v,j,t}>0} \bar{x}_{v,j,t} = \sum_{v:\bar{x}_{v,j,t}>0} (\bar{x}_{v,j,t} - \varepsilon/2) + \sum_{v:\bar{x}_{v,j,t}>0} \varepsilon/2 \\ &\leq \sum_v (\bar{x}_{v,j,t} - \varepsilon/2)^+ + 2k_j \cdot \varepsilon/2 \leq (2 + \varepsilon/2)\ell k_j + \varepsilon k_j.\end{aligned}$$

Since $\ell \geq 2$ (otherwise, we have the unweighted problem), we get

$$\sum_v \bar{x}_{v,j,t} \leq (2 + \varepsilon)\ell k_j.$$ ◀

3.2 Stage II: Weighted Interval Cover

In the second stage of the rounding algorithm, we first scale the (integer-valued) variables $\bar{y}_{v,j,I}$ down by a factor of ℓ to obtain new variables $y^*_{v,j,I}$:

$$y^*_{v,j,I} := \bar{y}_{v,j,I}/\ell \text{ and therefore, } x^*_{v,j,t} = \sum_{I:t\in I} y^*_{v,j,I} = \bar{x}_{v,j,t}/\ell. \tag{11}$$

Our goal is to round the fractional variables $y^*_{v,j,I}$ to $\{0,1\}$ values. In fact, our rounding will ensure that if the rounded value equals 1 then $y^*_{v,j,I} > 0$. Since $\bar{y}_{v,j,I}$ is integral, the packing property in Lemma 8 implies that for any time t, vertex v, and index $j \in [\ell]$, there are at most $(2+\varepsilon)\ell k_j$ intervals $I \ni t$ for which $\bar{y}_{v,j,I} > 0$. The rounding property of our algorithm will ensure that the final integral solution, which lies in the support of $y^*_{v,j,I}$, will also satisfy that there are at most $(2+\varepsilon)\ell k_j$ intervals containing any time t. Since we are allowed a resource augmentation of $(2+\varepsilon)\ell$ factor in the number of servers of weight W_j, we can serve the requests with the set of available servers. Henceforth, we can ignore the packing constraint (6) for our rounded solution. As a result, the relaxation LP2 decouples into n independent relaxations, one for each location $v \in V$.

In this decoupled instance, we get the following LP relaxation for each location v, where for each class $j \in [\ell]$, we define $\mathbf{I}_{v,j} := \{I \mid y^*_{v,j,I} > 0\}$ as the set of intervals I with a nonzero value of $y^*_{v,j,I}$ and $\mathcal{R}(v)$ as the set of times t when v is requested:

$$\begin{aligned} &\min 1/2 \sum_{j\in[\ell]} W_j \cdot \sum_{I\in\mathbf{I}_{v,j}} y_{v,j,I} && \text{(LP}_v\text{)} \\ &\text{s.t.} \sum_j \sum_{I\in\mathbf{I}_{v,j}:t\in I} y_{v,j,I} \geq 1 \qquad \forall t \in \mathcal{R}_v \\ &\qquad\qquad y_{v,j,I} \geq 0. \end{aligned}$$

By the covering property of Lemma 8, the variables $y^*_{v,j,I}$ defined in (11) are feasible solutions for (LP$_v$) for all locations v. Furthermore, by the lemma's cost property (and the scaling down by ℓ), the total cost $\sum_v \sum_j W_j \cdot \sum_I y^*_{v,j,I}$ is at most $O(1/\varepsilon)$ times the optimal cost of (LP2).

Finally, the constraint matrix for (LP_v) satisfies the consecutive-ones property: if the constraints are ordered chronologically, then a variable $y_{v,j,I}$ appears in the constraints corresponding to times $t \in I$ where $\sigma_t = v$, which is a contiguous subsequence of all times t where $\sigma_v = j$. Constraint matrices with this property are totally unimodular (see, e.g., [18]). Therefore, each of the solutions $\{y^*_{v,j,I} : j \in [\ell], I \in \mathbf{I}_{v,j}\}$ for LP_v can be rounded to a feasible integral solution without any increase in cost, which proves Theorem 3.

4 Online Algorithm

In this section, we describe an efficient online algorithm for Weighted k-Server and prove the following result:

▶ **Theorem 4** (Online Algorithm). *Let $\mathcal{I}$ be an instance of* Weighted k-Server *with k_j servers of weight W_j for all $j \in [\ell]$. There is a randomized (polynomial time) online algorithm for $\mathcal{I}$ that uses at most $2\ell k_j$ servers of weights W_j for each $j \in [\ell]$ and has expected server movement cost at most $O(\ell^2 \log \ell)$ times the optimal cost of $\mathcal{I}$.*

We begin by re-writing the LP relaxation (LP2) in terms of the "anti-page" variables, as in [4]. Recall that (LP2) has variables $y_{v,j,I}$ representing the (fractional) weight W_j server mass present at location v during the interval I; instead we first rewrite it in terms of the "page" variables $x_{v,j,t}$, which denote the total amount of weight W_j server mass at location v at time t, as given in (4). The objective of this LP in terms of $x_{v,j,t}$ is:

$$\sum_{v,j,I} W_j \cdot y_{v,j,I} = \sum_{v,j,I} W_j \cdot (x_{v,j,t} - x_{v,j,t^-})^+.$$

We can constrain any algorithm to values $x_{v,j,t} \in [0,1]$ for all v, j, t (since having multiple servers at a location is not beneficial). This allows us to work with non-negative *anti-page* variables $z_{v,j,t} := 1 - x_{v,j,t}$. The objective, now rewritten in terms of these new variables $z_{v,j,t}$, becomes:

$$\sum_{v,j,I} W_j \cdot (x_{v,j,t} - x_{v,j,t^-})^+ = \sum_{v,j,I} W_j \cdot (z_{v,j,t^-} - z_{v,j,t})^+. \tag{12}$$

We shall also maintain the following invariant for each server weight W_j and time t:

$$\sum_v x_{v,j,t} = k_j \qquad \Longleftrightarrow \qquad \sum_v z_{v,j,t} = n - k_j \quad \forall j, t. \tag{13}$$

We write the covering constraint (5) (or equivalently (2)) in terms of $z_{v,j,t}$ as:

$$\sum_j z_{\sigma_t,j,t} \le \ell - 1 \tag{14}$$

The algorithm follows the standard relax-and-round paradigm in the online setting. The first step is to compute a feasible fractional solution to an LP consisting of objective (12) and constraints (13) and (14), in an online setting. We show in §4.1 that we can find a fractional solution that uses $O(\ell k_j)$ servers of weight W_j for each class j, and has a competitive ratio of $O(\ell^2)$. The second step is to give an online rounding algorithm to convert this fractional solution to an integral solution: our rounding algorithm given in §4.2 uses the standard online rounding algorithm for the paging problem and increases the cost of the solution by a constant factor.

4.1 Online Fractional Algorithm

In this section, we give an online algorithm for maintaining a fractional solution to the LP involving $z_{v,j,t}$ variables. We obtain the following result:

▶ **Theorem 9.** *There is a deterministic (polynomial time) online fractional algorithm that maintains the condition that for every request time t, there exists an index $j \in [\ell]$ such that there is unit server mass of weight W_j at location σ_t at time t. The algorithm uses $2\ell k_j$ servers of weight W_j for each $j \in [\ell]$, and whose cost is at most $O(\ell^2 \log \ell)$ times that of an optimal fractional solution.*

Note that the condition in the theorem is stronger than (14), the feasibility condition for (LP2), because we are using server from a single weight class to service this request.

Consider a time t, and the request arriving at location σ_t. We first set $z_{v,j,t} = z_{v,j,t^-}$ for all $v \in V, j \in \ell$. Now the algorithm moves fractional server mass to σ_t until a relaxed version of the covering constraint (14) for time t gets satisfied. The relaxed constraint is given by

$$\exists j \in [\ell] \text{ such that } z_{\sigma_t,j,t} \leq 1 - \frac{1}{2\ell}. \tag{15}$$

Indeed, if the constraint is violated, then for each vertex $v \neq \sigma_t$ and each $j \in [\ell]$, if v has non-zero server mass of weight W_j (i.e., $z_{v,j,t} < 1$), then the algorithm moves server mass of weight W_j from v to σ_t using the following differential equation. (The derivative is with respect to a variable s which starts from 0 and increases at uniform rate.)

$$\dot{z}_{v,j,t} = \frac{1}{W_j |S_j|} \cdot (z_{v,j,t} + \delta) \qquad \forall j \in [\ell], \forall v \in S_j. \tag{16}$$

Here, $S_j \subseteq V$ denotes the instantaneous set of locations (i.e., at the current value of the variable s) that have $z_{v,j,t} < 1$, not including the location σ_t, and $\delta > 0$ is a parameter that we shall fix later. Correspondingly, we reduce $z_{\sigma_t,j,t}$ by the total amount of server mass of weight W_j entering σ_t:

$$\dot{z}_{\sigma_t,j,t} = -\frac{1}{W_j |S_j|} \cdot \sum_{v \in S_j} (z_{v,j,t} + \delta) \qquad \forall j \in [\ell]. \tag{17}$$

Note that server mass is moved away other locations and into location σ_t only if $z_{\sigma_t,j,t} > 1 - \frac{1}{2\ell}$ for all j. Since $z_{\sigma_t,j,t} \leq 1$ for all j, it follows that $z_{v,j,t} \in [1 - \frac{1}{2\ell}, 1]$ for all j, t. Hence,

$$z_{v,j,t} \geq 1 - \frac{1}{2\ell} \text{ for all } j,t \quad \Longrightarrow \quad |S_j| \geq 2\ell k_j - 1 \geq \frac{3\ell k_j}{2} \geq 3 \text{ for all } j,t, \tag{18}$$

since $\ell \geq 2, k_j \geq 1$.

To analyze the algorithm, we use a potential function Φ. The potential function depends on the offline (integral) optimal solution – let us call it $\mathcal{O}$, and let $\text{opt}_{v,j,t}$ be the indicator variable for the location v containing a server of weight W_j at time t. The potential at time t is defined as follows:

$$\Phi(t) := \sum_{v,j:\text{opt}_{v,j,t}=0} W_j \cdot \ln\left(\frac{1+\delta}{z_{v,j,t}+\delta}\right).$$

Let $\mathsf{cost}(t)$ denote the algorithm's server movement cost at time t and $\mathsf{cost}^{\mathcal{O}}(t)$ denote the corresponding quantity for the optimum solution $\mathcal{O}$. Our goal is to show that:

$$\frac{\mathsf{cost}(t)}{4\ell} + \Phi(t+1) - \Phi(t) \leq \ln(1 + 1/\delta) \cdot \mathsf{cost}^{\mathcal{O}}(t). \tag{19}$$

The following properties of $\Phi(t)$ can verified easily:

- **Nonnegativity:** Φ is always nonnegative, since $z_{v,j,t} \le 1$.
- **Lipschitzness property:** When the optimal solution moves a server of weight W_j from one location to another, the increase in Φ is at most $W_j \cdot \ln(1 + 1/\delta)$.

The Lipschitzness property implies that (19) holds when $\mathcal{O}$ serves the request at σ_t. It remains the analyze the cost and change in potential when the algorithm changes its solution. Consider the process when we transfer server mass to σ_t.

We first bound the online algorithm's cost. Since all the weight classes incur the same server movement cost while transferring to σ_t, the movement cost is ℓ times the movement cost incurred while transferring servers of a fixed class, say $j^\star$. The latter is at most

$$W_{j^\star} \sum_{v \in S_{j^\star}} \dot{z}_{v,j^\star,t} \overset{(16)}{=} \frac{1}{|S_{j^\star}|} \sum_{v \in S_{j^\star}} (z_{v,j^\star,t} + \delta) \quad = \frac{|S_{j^\star}| + 1 - k_{j^\star} + \delta |S_{j^\star}|}{|S_{j^\star}|} \le 1 + \delta. \tag{20}$$

Thus, the upper bound on the $\frac{\mathsf{cost}(t)}{4\ell}$ term in the LHS of (19) is at most $\frac{1+\delta}{4} \le 1/3$ provided $\delta \le 1/3$.

Next, we lower bound the rate of decrease of potential Φ. We begin by bounding the rate of decrease in potential due to because of server mass leaving all locations except σ_t:

$$\begin{aligned}\Delta^- &= - \sum_{j \in [\ell], v \neq \sigma_t : \mathrm{opt}_{v,j,t} = 0} \frac{W_j}{z_{v,j,t} + \delta} \cdot \dot{z}_{v,j,t} \quad \overset{(16)}{=} - \sum_{j, v \in S_j : \mathrm{opt}_{v,j,t} = 0} \frac{1}{z_{v,j,t} + \delta} \cdot \frac{z_{v,j,t} + \delta}{|S_j|} \\ &= - \sum_j \frac{|\{v \in S_j : \mathrm{opt}_{v,j,t} = 0\}|}{|S_j|} \quad \overset{(18)}{\le} - \sum_j \frac{|S_j| - k_j}{|S_j|} \le -\ell \left(1 - \frac{2}{3\ell}\right) = -\ell + 2/3.\end{aligned} \tag{21}$$

Next, we bound the rate of increase in potential due to server classes $j \neq j^*$ because of server mass entering σ_t:

$$\begin{aligned}\Delta^+ &= \sum_{j \neq j^*} \frac{W_j}{z_{\sigma_t,j,t} + \delta} \cdot \dot{z}_{\sigma_t,j,t} \quad \overset{(16)}{=} \sum_{j \neq j^*, v \in S_j} \frac{W_j}{z_{\sigma_t,j,t} + \delta} \cdot \frac{z_{v,j,t} + \delta}{|S_j| W_j} \\ &= \sum_{j \neq j^*} \frac{\sum_{v \in S_j} (z_{v,j,t} + \delta)}{|S_j| (z_{\sigma_t,j,t} + \delta)} \quad = \sum_{j \neq j^*} \frac{(|S_j| - k_j + (1 - z_{\sigma_t,j,t})) + \delta \cdot |S_j|}{|S_j| (z_{\sigma_t,j,t} + \delta)} \\ &\overset{(18)}{\le} \sum_{j \neq j^*} \frac{(|S_j| - k_j + 1/2\ell) + \delta \cdot |S_j|}{|S_j| (1 - 1/2\ell + \delta)} \quad \overset{(18)}{\le} \sum_{j \neq j^\star} \frac{1 - 2/3\ell + 1/6\ell + \delta}{1 - 1/2\ell + \delta} \quad \le \ell - 1,\end{aligned}$$

provided $\delta = 1/2\ell$. Combining with (21), we see that the overall change in potential is $\Delta^- + \Delta^+ \le -1/3$. Consequently, we get that the change in potential pays for the increase in the algorithm's cost (divided by 4ℓ) – which shows (19) – when the fractional solution changes.

This implies that we have an algorithm for maintaining $z_{v,j,t}$ that satisfies (15). In terms of the competitive ratio, the algorithm loses 4ℓ in (19) and $\ln(1 + 1/\delta) = O(\log \ell)$ in the Lipschitzness of the potential function. Note that (15) implies that for all t, there exists j such that $x_{\sigma_t,j,t} \ge \frac{1}{2\ell}$. We scale the fractional variables to obtain $\widetilde{x}_{v,j,t} := \min(2\ell x_{v,j,t}, 1)$; then, for all t, there exists j such that $\widetilde{x}_{\sigma_t,j,t} = 1$. Note that this satisfies the condition in Theorem 9. Equivalently, the corresponding "anti-page" variables $\widetilde{z}_{v,j,t} := 1 - \widetilde{x}_{v,j,t}$ satisfy the following condition for all t:

$$\exists j \text{ such that } \widetilde{z}_{\sigma_t,j,t} = 0. \tag{22}$$

The last scaling step creates a resource augmentation of 2ℓ, and increases the competitive ratio to $O(\ell^2 \log \ell)$. This completes the proof of Theorem 9.

4.2 Rounding the Fractional Solution Online

We round the fractional solution for each weight class j separately. Let T_j represent the request times t when (22) is satisfied by weight class j. Note that the solution $\widetilde{z}_{v,j,t}$ for weight class j represents a feasible fractional solution for an instance of the paging problem with $2\ell k_j$ cache slots, where there is a page request for each time $t \in T_j$ at location σ_t.

We now invoke the following known online rounding algorithm for the paging problem separately in each weight class j to complete the proof of Theorem 4.

▶ **Lemma 10** ([9]). *There is a randomized (polynomial time) online algorithm that converts any feasible fractional solution for an instance of the* Paging *problem to an integral solution using the same number of cache slots, and incurs constant times the cost of the fractional solution.*

5 Discussion

In this work, we have given the first efficient offline and online algorithms with non-trivial guarantees for Weighted k-Server. Several interesting problems remains open:

1. For the case of two distinct weight classes, we show in Appendix A that it is UG-Hard to obtain an $\Omega(N^c)$-approximation algorithm for some constant $c > 0$, even with $(2-\varepsilon)$-resource augmentation. Can we extend such a hardness result to more weight classes? For example, can we show that for three distinct weight classes, it is UG-Hard to obtain a C-approximation algorithm for any *constant* C, even with $(3-\varepsilon)$-resource augmentation? The principal reason why our hardness proof for $\ell = 2$ does not extend here is because one needs to recursively cycle through all subsets (of a certain size) of V to create an integrality gap instance for the natural LP relaxation. If the size of these subsets is large, then the length of the input becomes very large. If the size of these subsets is small, then it is not clear how to extend this to a hardness proof.
2. In Section 3, we give an offline constant approximation algorithm which requires slightly more than 2ℓ-resource augmentation. Can we get a constant approximation algorithm (or even an optimal algorithm) with exactly ℓ-resource augmentation? We conjecture that the integrality gap of LP is constant (or even 1) if the integral solution is allowed ℓ-resource augmentation.
3. In the online case, we give a $O(\ell^2 \log \ell)$-competitive algorithm with 2ℓ-resource augmentation in Section 4. Can we get a constant-competitive algorithm with $O(\ell)$-resource augmentation, i.e., a result in the same vein as our offline algorithm?

References

1 Nikhil Ayyadevara and Ashish Chiplunkar. The randomized competitive ratio of weighted k-server is at least exponential. In *29th Annual European Symposium on Algorithms*, volume 204 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 9, 11. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2021.

2 Nikhil Bansal, Niv Buchbinder, Aleksander Madry, and Joseph Naor. A polylogarithmic-competitive algorithm for the k-server problem. *J. ACM*, 62(5):40:1–40:49, 2015. doi:10.1145/2783434.

3 Nikhil Bansal, Niv Buchbinder, and Joseph Naor. Metrical task systems and the k-server problem on HSTs. In *Automata, languages and programming. Part I*, volume 6198 of *Lecture Notes in Comput. Sci.*, pages 287–298. Springer, Berlin, 2010. doi:10.1007/978-3-642-14165-2_25.

4 Nikhil Bansal, Niv Buchbinder, and Joseph Naor. A primal-dual randomized algorithm for weighted paging. *J. ACM*, 59(4):19, 2012.

5 Nikhil Bansal, Marek Eliás, Lukasz Jez, and Grigorios Koumoutsos. The (h, k)-server problem on bounded depth trees. *ACM Trans. Algorithms*, 15(2):28:1–28:26, 2019. doi:10.1145/3301314.

6 Nikhil Bansal, Marek Eliáš, and Grigorios Koumoutsos. Weighted k-server bounds via combinatorial dichotomies. In *58th Annual IEEE Symposium on Foundations of Computer Science – FOCS 2017*, pages 493–504. IEEE Computer Soc., Los Alamitos, CA, 2017. doi:10.1109/FOCS.2017.52.

7 Yair Bartal, Avrim Blum, Carl Burch, and Andrew Tomkins. A polylog(n)-competitive algorithm for metrical task systems. In *STOC '97 (El Paso, TX)*, pages 711–719. ACM, New York, 1999.

8 S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11(1):2–14, 1994. doi:10.1007/BF01294260.

9 Avrim Blum, Carl Burch, and Adam Kalai. Finely-competitive paging. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 450–458. IEEE Computer Society, 1999. doi:10.1109/SFFCS.1999.814617.

10 Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, USA, 1998.

11 Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal on-line algorithm for metrical task system. *J. Assoc. Comput. Mach.*, 39(4):745–763, 1992. doi:10.1145/146585.146588.

12 Sébastien Bubeck, Michael B. Cohen, James R. Lee, and Yin Tat Lee. Metrical task systems on trees via mirror descent and unfair gluing. *SIAM J. Comput.*, 50(3):909–923, 2021. doi:10.1137/19M1237879.

13 Sébastien Bubeck, Michael B. Cohen, Yin Tat Lee, James R. Lee, and Aleksander Madry. k-server via multiscale entropic regularization. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 3–16. ACM, 2018. doi:10.1145/3188745.3188798.

14 Niv Buchbinder, Anupam Gupta, Marco Molinaro, and Joseph (Seffi) Naor. k-servers with a smile: Online algorithms via projections. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 98–116. SIAM, 2019. doi:10.1137/1.9781611975482.7.

15 Ashish Chiplunkar and Sundar Vishwanathan. Randomized memoryless algorithms for the weighted and the generalized k-server problems. *ACM Trans. Algorithms*, 16(1):Art. 14, 28, 2020. doi:10.1145/3365002.

16 Christian Coester and James R. Lee. Pure entropic regularization for metrical task systems. *Theory Comput.*, 18:Paper No. 23, 24, 2022. doi:10.4086/toc.2022.v018a023.

17 Amos Fiat and Moty Ricklin. Competitive algorithms for the weighted server problem. *Theoret. Comput. Sci.*, 130(1):85–99, 1994. doi:10.1016/0304-3975(94)90154-6.

18 D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15(3):835–855, 1965.

19 Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 214–221. IEEE Computer Society, 1995. doi:10.1109/SFCS.1995.492478.

20 Elias Koutsoupias and Christos H. Papadimitriou. On the k-server conjecture. *J. ACM*, 42(5):971–983, 1995. doi:10.1145/210118.210128.

21 James R. Lee. Fusible hsts and the randomized k-server conjecture. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 438–449. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00049.

22 Mark S. Manasse, Lyle A. McGeoch, and Daniel Dominic Sleator. Competitive algorithms for server problems. *J. Algorithms*, 11(2):208–230, 1990. doi:10.1016/0196-6774(90)90003-W.

23 Daniel Dominic Sleator and Robert Endre Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, 1985. doi:10.1145/2786.2793.

24 Neal E. Young. On-line caching as cache size varies. In Alok Aggarwal, editor, *Proceedings of the Second Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 28-30 January 1991, San Francisco, California, USA*, pages 241–250. ACM/SIAM, 1991. URL: http://dl.acm.org/citation.cfm?id=127787.127832.

A The Unique Games Hardness

In this section, we consider the special case of Weighted k-Server when there are only two weight classes. Assume wlog that the two distinct weights are 1 and W, where $W \gg 1$. Our first main result shows that getting a good approximation algorithm with $(2-\varepsilon)$-resource augmentation for any constant $\varepsilon > 0$ is as hard as getting a better-than-two approximation for the vertex cover problem.

▶ **Theorem 1** (Hardness). *For any constant $\varepsilon > 0$, it is UG-hard to obtain an $N^{1/2-\varepsilon}$-approximation algorithm for Weighted k-Server with two weight classes, even when we are allowed c-resource augmentation for any constant $c < 2$. Here N represents the size of the input (including the request sequence length).*

Proof. We give a reduction from the Vertex Cover problem. Let $d = d(\varepsilon)$ be a constant to be fixed later, and let $c < 2$ be a constant as in the statement of the theorem. Let $\mathcal{I} = (G = (V, E), t)$ be an instance of the Vertex Cover problem on n vertices. We know that it is UG-hard to distinguish between the following two cases: (i) G has a vertex cover of size at most t, or (ii) every vertex cover of G must have size strictly larger than ct.

We map $\mathcal{I}$ to an instance $\mathcal{I}'$ of Weighted k-Server as follows: the set of points in $\mathcal{I}'$ is given by $V \cup \{v_0\}$, where v_0 is a special vertex. There are t servers of weight $W = n^d$ and one server of unit weight. Let the edges in E be $e_1, \ldots, e_m$. A subsequence of the request sequence consists of m *phases*, where we have a phase for each edge e_i. During phase i corresponding to edge $e_i = (u_i, v_i)$, the request sequence toggles between u_i and v_i for W times. Finally, the subsequence is repeated W times. In other words, it is the following sequence

$$\Big(\underbrace{u_1, v_1, u_1, v_1, \ldots, u_1, v_1}_{W \text{ times}}, \ldots, \underbrace{u_m, v_m, u_m, v_m, \ldots, u_m, v_m}_{W \text{ times}}\Big)^W.$$

We also have to specify the initial location of the servers. Assume that all servers are at location v_0 in the beginning. This completes the description of the instance $\mathcal{I}'$. Observe that N, the number of requests in instance $\mathcal{I}'$ is $O(m \cdot n^{2d})$.

▷ Claim 11. Suppose G has a vertex cover of size at most t. Then the cost of the optimal solution for $\mathcal{I}'$ is at most $2mW$.

Proof. Let $V' \subseteq V$ be a vertex cover of size t. Consider the following solution: we move the t heavy servers from v_0 to V' at the beginning. From now on, the heavy servers will not move at all. During a phase corresponding to an edge $e_i = (u_i, v_i)$, we know that at least one of these vertices will be occupied by a heavy server. If the other end-point, say v_i, is not occupied by a heavy server, we move the server of weight 1 to v_i. Now we have two servers occupying u_i and v_i respectively until the end of this phase. The total movement cost is incurred either at the beginning (which is tW overall), or at the beginning of each phase (when the cost is 1). Since there are mW phases, the overall cost is at most $tW + mW \leq 2mW$. ◁

▷ Claim 12. Suppose every vertex cover in G has size strictly larger than ct. Then cost of the optimal solution for $\mathcal{I}'$, even with c-resource augmentation, is at least W^2.

Proof. Consider any solution for $\mathcal{I}'$. Recall that the input consists of W subsequences, call these $S_1, \ldots, S_W$, where each subsequence S_j consists of m phases, one for each edge of G. We claim that during each such subsequence S_j, the solution must pay movement cost of at least W. Indeed, consider a subsequence S_j. If the solution moves a heavy server during S_j, then the claim follows directly. Else observe that the size of any vertex cover is strictly larger than the number of heavy servers ct, so there is some edge $e_i = (u_i, v_i)$ not covered by the heavy servers during S_j. Now the phase for e_i in S_j would require the unit weight server to toggle between u_i and v_i for W times. In either case, the cost of each subsequence is at least W, and the overall cost of the solution is at least W^2. ◁

The above two results along with the UG-hardness result for Vertex Cover impliy that it is UG-hard to obtain a $\frac{W^2}{2mW}$-approximation for Weighted k-Server with two weight classes. This ratio is equal to $\frac{W}{2m} \geq n^{d-2} \geq N^{1/2-\varepsilon}$, assuming d is $\Omega(1/\varepsilon)$, which proves Theorem 1. ◀

B Missing proofs from §2

▶ **Lemma 6.** *Let $\varepsilon > 0$ be a small enough constant. Assume that the integral solution is allowed $(\ell - \varepsilon)k_r$ servers of weight W_r for each $r \in [\ell]$. Any integral solution for the input sequence generated by Algorithm 1 (with $C = \ell/\varepsilon$) has movement cost at least $M^{\ell-1}$.*

Proof of Lemma 6. We prove the following more general statement by reverse induction on r: any integral solution for the sequence generated by Generate$(S_0, \ldots, S_r)$ for a valid tuple $(S_0, \ldots, S_r)$ which does not use any server of weight class $W_1, \ldots, W_r$ (at any location in S_r) has cost at least $M^{\ell-1}$. It suffices to prove this statement, because the case when $r = 0$ implies the lemma.

Consider the base case when $r = \ell - 1$. Consider the sequence generated by such a procedure Generate$(S_0, \ldots, S_r)$ such that no server of weight $W_1, \ldots, W_{\ell-1}$ is used for serving the requests at $S_{\ell-1}$. Thus all requests generated by this procedure must be served by servers of weight W_ℓ. Now, $|S_{\ell-1}| = \frac{n}{C^{\ell-1}}$, whereas the number of weight W_ℓ servers available to the algorithm is $(\ell - \varepsilon)k_\ell < \frac{n}{C^{\ell-1}}$. Therefore, during each iteration of the **repeat-until** loop in lines 1.2–1.8 in Algorithm 1, at least one server of weight W_ℓ must move. So the overall movement cost during this input sub-sequence is at least $W_\ell \cdot L_{\ell-1} = M^{\ell-1}$. This proves the base case.

The inductive case is proved in an analogous manner. Suppose the statement is true for $r+1$, and now consider the sub-sequence generated by $Gen(S_0, \ldots, S_r)$ for some valid tuple $(S_0, \ldots, S_r)$. Assume that no server of weight $W_1, \ldots, W_r$ is present at any node in S_r during this time. We claim that the algorithm must incur movement cost of at least W_{r+1} during each iteration of the **repeat-until** loop. Indeed, fix such an iteration. Two cases arise: (a) The algorithm moves a server of weight W_{r+1} then the claim follows trivially, or (b) No server of weight W_{r+1} is moved during this period. Now observe that $|S_r| = \frac{n}{C^r}$, and the number of weight W_{r+1} servers available to the algorithm is $(\ell - \varepsilon)k_{r+1} = |S_r| - \varepsilon k_{r+1} = |S_r| \left(1 - \frac{1}{C}\right)$. Thus, there is a subset S_{r+1} of S of size $\frac{|S_r|}{C} = \frac{n}{C^{r+1}}$ where no server of weight W_{r+1} appears during this input sub-sequence. Consider the recursive call Generate$(S_0, \ldots, S_r, S_{r+1})$ in line 1.8. The induction hypothesis implies that the movement cost during this recursive call is at least $M^{\ell-1} \geq W_{r+1}$.

Thus, we have shown that the movement cost during each iteration of the **repeat-until** loop during Generate$(S_0, \ldots, S_r)$ is at least W_{r+1}. Since there are L_r such iterations, the overall movement cost is at least $W_{r+1} \cdot L_r = M^{\ell-1}$. This completes the proof of the induction hypothesis, and implies the lemma. ◀

A Constant-Factor Approximation for Quasi-Bipartite Directed Steiner Tree on Minor-Free Graphs

Zachary Friggstad ✉
Department of Computing Science, University of Alberta, Canada

Ramin Mousavi ✉
Department of Computing Science, University of Alberta, Canada

Abstract

We give the first constant-factor approximation algorithm for quasi-bipartite instances of Directed Steiner Tree on graphs that exclude fixed minors. In particular, for K_r-minor-free graphs our approximation guarantee is $O(r \cdot \sqrt{\log r})$ and, further, for planar graphs our approximation guarantee is 20.

Our algorithm uses the primal-dual scheme. We employ a more involved method of determining when to buy an edge while raising dual variables since, as we show, the natural primal-dual scheme fails to raise enough dual value to pay for the purchased solution. As a consequence, we also demonstrate integrality gap upper bounds on the standard cut-based linear programming relaxation for the Directed Steiner Tree instances we consider.

2012 ACM Subject Classification Theory of computation → Routing and network design problems

Keywords and phrases Directed Steiner tree, Combinatorial optimization, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.13

Category APPROX

Related Version *Full Version*: https://arxiv.org/abs/2111.02572

1 Introduction

In the Directed Steiner Tree (DST) problem, we are given a directed graph $G = (V, E)$ with edge costs $c(e) \geq 0$ for all $e \in E$, a root node $r \in V$, and a collection of terminals $X \subseteq V \setminus \{r\}$. The nodes in $V \setminus (X \cup \{r\})$ are called *Steiner* nodes. The goal is to find a minimum cost subset $F \subseteq E$ such that there is an $r - t$ path using only edges in F for every terminal $t \in X$. Note any feasible solution that is inclusion-wise minimal must be an arborescence rooted at r. Throughout, we let n denote $|V|$.

One key aspect of DST lies in the fact that it generalizes many other important problems, e.g. Set Cover, (non-metric, multilevel) Facility Location, and Group Steiner Tree. Halperin and Krauthgamer [23] showed Group Steiner Tree cannot be approximated within $O(\log^{2-\epsilon} n)$ for any $\epsilon > 0$ unless NP $\subseteq$ ZTIME $(n^{\text{polylog}\,(n)})$ and therefore the same result holds for DST.

Building on a height-reduction technique of Calinescu and Zelikovsky [5, 36], Charikar et al. give the best approximation for DST which is an $O(|X|^\epsilon)$-approximation for any constant $\epsilon > 0$ [8] and also an $O(\log^3 |X|)$-approximation in $O(n^{\text{polylog}(|X|)})$ time (quasi-polynomial time). More recently, Grandoni, Laekhanukit, and Li [21] obtained a quasi-polynomial time $O(\frac{\log^2 |X|}{\log \log |X|})$-approximation factor for Directed Steiner Tree which is the best possible for quasi-polynomial time algorithms, assuming both the Projection Game Conjecture

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 13; pp. 13:1–13:18
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

and NP $\not\subseteq \bigcap_{0<\delta<1}$ ZTIME(2^{n^δ}). Ghuge and Nagarajan [18] studied a variant of DST called the Directed Tree Orienteering problem and presented an $O(\frac{\log |X|}{\log\log |X|})$-approximation in quasi-polynomial time which yields the same approximation guarantee as in [21] for DST.

Methods based on linear programming have been less successful. Zosin and Khuller [38] showed the integrality gap of a natural flow-based LP relaxation is $\Omega(\sqrt{|X|})$ but n, the number of vertices, in this example is exponential in terms of $|X|$. More recently, Li and Laekhanukit [27] provided an example showing the integrality gap of this LP is at least polynomial in n. On the positive side, [33] shows for ℓ-layered instances of DST that applying $O(\ell)$ rounds of the Lasserre hierarchy to a slight variant of the natural flow-based LP relaxation yields a relaxation with integrality gap $O(\ell \cdot \log |X|)$. This was extended to the LP-based Sherali-Adams and Lovász-Schrijver hierarchies by [14].

We consider the cut-based relaxation (**Primal-LP**) for DST, which is equivalent to the flow-based relaxation considered in [38, 27]; the flow-based relaxation is an extended formulation of (**Primal-LP**). Let $\delta^{in}(S)$ be the set of directed edges entering a set $S \subseteq V$,

$$\begin{aligned} \text{minimize:} \quad & \sum_{e\in E} c(e)\cdot x_e && \textbf{(Primal-LP)} \\ \text{subject to:} \quad & x(\delta^{in}(S)) \geq 1 \quad \forall S \subseteq V \setminus \{r\},\ S \cap X \neq \emptyset && (1) \\ & x \geq 0 \end{aligned}$$

It is useful to note that if $|X| = 1$ (the shortest $s-t$ path problem) or $X \cup \{r\} = V$ (the minimum cost arborescence problem), the extreme points of (**Primal-LP**) are integral, see [29] and [11] respectively.

The undirected variant of Steiner Tree has seen more activity[1]. A series of papers steadily improved over the simple 2-approximation [37, 25, 30, 32] culminating in a $\ln 4 + \epsilon$ for any constant $\epsilon > 0$ [4]. Bern and Plassmann [2] showed that unless P = NP there is no approximation factor better than $\frac{96}{95}$ for Steiner Tree. However, there is a PTAS for Steiner Tree on planar graphs [3] and more generally [1] obtains a PTAS for Steiner Forest on graphs of bounded-genus.

Another well-studied restriction of Steiner Tree is to quasi-bipartite graphs. These are the instances where no two Steiner nodes are connected by an edge (i.e., $V \setminus (X \cup \{r\})$ is an independent set). Quasi-bipartite instances were first studied by Rajagopalan and Vazirani [31] in order to study the bidirected-cut relaxation of the Steiner Tree problem: this is exactly (**Primal-LP**) where we regard both directions of an undirected edge as separate entities. Feldmann et al. [13] studied Steiner Tree on graphs that do not have an edge-induced claw on Steiner vertices, i.e., no Steiner vertex with three Steiner neighbours, and presented a faster ln(4)-approximation than the algorithm of [4]. Currently, the best approximation in quasi-bipartite instances of Steiner Tree is $\frac{73}{60}$-approximation [19].

A natural question is to study the complexity of DST on these restricted instances. Hibi and Fujito [24] presented an $O(\log |X|)$-approximation algorithm for this case. Assuming P $\neq$ NP, this result asymptotically matches the lower bound $(1 - o(1)) \cdot \ln |X|$ for any $\epsilon > 0$; this lower bound comes from the hardness of Set Cover [12, 10] and the fact that the quasi-bipartite DST problem generalizes the Set Cover problem. Friggstad, Könemann, and Shadravan [15] showed that the integrality gap of (**Primal-LP**) is also $O(\log |X|)$ by a primal-dual algorithm and again this matches the lower bound on the integrality gap of this

[1] One usually does not specify the root node in Steiner Tree, the goal is simply to ensure all terminals are connected.

LP up to a constant. Chan et al. [7] studied the k-connected DST problem on quasi-bipartite instances in which the goal is to find a minimum cost subgraph H such that there are k edge-disjoint paths (in H) from r to each terminal in X. They gave an upper bound of $O(\log |X| \cdot \log k)$ on the integrality gap of the standard cut-based LP (put k instead of 1 in the RHS of the constraints in (**Primal-LP**)) by presenting a polynomial time randomized rounding algorithm.

Very recently, [17] show a combinatorial $O(\log |X|)$-approximation algorithm for DST on planar graphs based on shortest path separators in planar graph by Thorup [35]. It is also worth noting that Demaine, Hajiaghayi, and Klein [9] show that if one takes a standard flow-based relaxation for DST in planar graphs and further constraints the flows to be "non-crossing", then the solution can be rounded to a feasible DST solution while losing only a constant factor in the cost. To date, we do not know how to compute a low-cost, non-crossing flow in polynomial time for DST instances on planar graphs.

It remains an open question whether DST on planar graphs admits a constant factor approximation or even a PTAS, or not. We make some progress on this question. We show quasi-bipartite DST on planar graphs and more generally graphs excluding a fixed minor admit a constant factor approximation. In contrast to the approach in [17], our algorithm is LP-based and bounds the integrality gap of the natural cut-based LP. Our algorithm also works in the more general setting of minor-free graphs, whereas the DST approximation in [17] is specific only to planar graphs.

1.1 Primal-Dual Approximations for Steiner Tree Problems

Consider the Node-Weighted Steiner Tree (NWST) problem which is similar to undirected Steiner Tree except the weight function is on the Steiner vertices instead of edges and can also be viewed as a special case of DST. Guha et al. [22] presented a primal-dual algorithm with guarantee of $O(\ln n)$ which is asymptotically tight since NWST also generalizes set cover. Könemann, Sadeghian, and Sanità [26] give an $O(\log n)$-approximation using the primal-dual framework for a generalization of NWST called Node-Weighted Prize Collecting Steiner Tree[2].

Demaine, Hajiaghayi, and Klein [9] considered a generalization of NWST called Node-Weighted Steiner Forest (NWSF) on planar graphs and using the generic primal-dual framework of Goemans and Williamson [20] they showed a 6-approximation and further they extended their result to minor-free graphs. Later Moldenhauer [28] simplified their analysis and showed an approximation guarantee of 3 for NWSF on planar graphs.

An interesting, non-standard use of the primal-dual scheme is in the work of Chakrabarty, Devanur, and Vazirani [6] for undirected, quasi-bipartite instances of Steiner Tree. They introduce a new "simplex-embedding" LP relaxation and their primal-dual scheme raises dual variables with different rates. It is worth noting that although they also obtain upper bound for the integrality gap of the so-called *bidirected cut relaxation* (BCR) of quasi-bipartite instances of Steiner Tree, the algorithm and the simplex-embedding LP relaxation itself are valid only in the undirected setting.

[2] A key aspect of their algorithm is that it is also *Lagrangian multiplier preserving*.

1.2 Our contributions

We present a primal-dual algorithm for quasi-bipartite DST on minor-free graphs.

Generally, it is difficult to effectively utilize primal-dual algorithms in directed network design problems. This is true in our setting as well: we begin by showing a standard primal-dual algorithm (similar to the primal-dual algorithm for the minimum-cost arborescence problem) does not grow sufficiently-large dual to pay for the set of edges it purchases within any constant factor.

We overcome this difficulty by highlighting different roles for edges in connecting the terminals to the root. For some edges, we maintain two slacks: while raising dual variables these two slacks for an edge may be filled at different rates (depending on the edge's role for the various dual variables being raised) and we purchase the edge when one of its slacks is exhausted. Furthermore, unlike the analysis of standard primal-dual algorithms where the charging scheme is usually more local (i.e., charging the cost of purchased edges to the dual variables that are "close by"), we need to employ a more global charging scheme. Our approach also provides an $O(1)$ upper bound on the integrality gap of the natural cut-based relaxation (**Primal-LP**) for graphs that exclude a fixed minor.

We summarize our results here.

▶ **Theorem 1.** *There is an $O(r \cdot \sqrt{\log r})$-approximation algorithm for* Directed Steiner Tree *on quasi-bipartite, K_r-minor free graphs. Moreover, the algorithm gives an upper bound of $O(r \cdot \sqrt{\log r})$ on the integrality gap of* (**Primal-LP**) *for DST instances on such graphs.*

▶ Remark 2. The running time of our algorithm is $O(|V|^c)$ where c is a fixed constant that is independent of r. Also, we only require that every (simple) minor of the graph has bounded average degree to establish our approximation guarantee. In particular, if every minor of the input (quasi-bipartite) graph has degree at most d, then the approximation factor will be $O(d)$.

▶ **Theorem 3.** *There is a 20-approximation algorithm for* Directed Steiner Tree *on quasi-bipartite, planar graphs. Moreover, the algorithm gives an upper bound of 20 on the integrality gap of* (**Primal-LP**) *for* Directed Steiner Tree *instances on such graphs.*

We also verify that Steiner Tree (and, thus, Directed Steiner Tree) remains NP-hard even when restricted to quasi-bipartite, planar instances. Similar results are known, but we prove this one explicitly since we were not able to find this precise hardness statement in any previous work.

▶ **Theorem 4.** *Steiner Tree instances on bipartite planar graphs where the terminals are on one side and the Steiner nodes are on the other side is* NP*-hard.*

The above hardness result shows DST instances on quasi-bipartite, planar graphs is NP-hard as well.

1.3 Organization of the paper

In Section 2, we state some definition and notation where we use throughout the paper. In Section 3 we present an example that shows the most natural primal-dual algorithm fails to prove our approximation results, this helps the reader understand the key difficulty we need to overcome to make a primal-dual algorithm work and motivates our more refined approach. In Section 4 we present our primal-dual algorithm and in Section 5 we present the analysis.

The analysis contains three main subsections where in each section we present a charging scheme. The first two charging schemes are straightforward but the last one requires some novelty. Finally, we put all these charging schemes together in Subsection 5.4 and prove Theorems 1 & 3. The proof of the hardness result (Theorem 4) is deferred to the full version of the paper [16].

2 Preliminaries

In this paper, graphs are simple directed graphs unless stated otherwise. By simple we mean there are no parallel edges[3]. Note that we can simply keep the cheapest edge in a group of parallel edges if the input graph is not simple; the optimal value for DST problem does not change.

Throughout this paper, we fix a directed graph $G = (V, E)$, a cost function $c : E \rightarrow \mathbb{R}_{\geq 0}$, a root r, a set of terminals $X \subseteq V \setminus \{r\}$, and no edge between any two Steiner nodes, as the input to the DST problem. We denote the optimal value for DST instance by OPT.

Given a subgraph G' of G we define $\delta^{in}_{G'}(S) = \{e = (u, v) \in E(G') : \ u \in V \setminus S, \ v \in S\}$ (i.e., the set of edges in G' entering S) we might drop the subscript if the underlying subgraph is G itself. For an edge $e = (u, v)$, we call u the **tail** and v the **head** of e. By a **dipath** we mean a directed path in the graph. By **SCCs** of $F \subseteq E$ we mean the strongly connected components of (V, F) that contains either the root node or at least one terminal node. So for example, if a Steiner node is a singleton strongly connected component of (V, F) then we do not refer to it as an SCC of F. Due to the quasi-bipartite property, these are the only possible strongly connected components in the traditional sense of (V, F) that we will not call SCCs. Observe F is a feasible DST solution if and only if each SCC is reachable from r.

An *arborescence* $T = (V, E)$ rooted at $r \in V$ is a directed tree oriented away from the root such that every vertex in V is reachable from r. By *height* of a vertex u in T we mean the number of edges between r (the root) and u in the dipath from r to u in T. We let T_u denotes the subtree of T rooted at u.

Our discussions, algorithm, and the analysis rely on the concept of *active sets*, so we define them here.

▶ **Definition 5** (Violated set). *Given a DST instance and a subset $F \subseteq E$, we say $S \subseteq V \setminus \{r\}$ where $S \cap X \neq \emptyset$ is a* violated *set with respect to F if $\delta^{in}_F(S) = \emptyset$.*

▶ **Definition 6** (Active set). *Given a DST instance and a subset $F \subseteq E$, we call a minimal violated set (no proper subset of it, is violated) an* active *set (or active moat) with respect to F.*

We use the following definition throughout our analysis and (implicitly) in the algorithm.

▶ **Definition 7** (F-path). *We say a dipath P is a F-path if all the edges of P belong to $F \subseteq E$. We say there is a F-path from a subset of vertices to another if there is a F-path from a vertex of the first set to a vertex of the second set.*

In quasi-biparitite graphs, active moat have a rather "simple" structure, our algorithm will leverage the following properties.

▶ **Lemma 8.** *Consider a subset of edges F and let A be an active set with respect to F. Then, A consists of exactly one SCC C_A of F, and any remaining in $A \setminus C_A$ are Steiner nodes. Furthermore, for every Steiner node in $A \setminus C_A$ there are edges in F that are oriented from the Steiner node to C_A.*

[3] Two edges are parallel if their endpoints are the same and have the same orientation.

Proof. By definition of violated sets, A does not contain r. If A contains only one terminal, then the first statement holds trivially. So consider two terminals t and t' in A. We show there is a F-path from t to t' and vice versa. Suppose not and wlog assume there is no F-path from t' to t. Let $B := \{v \in A : \exists F - path\ from\ v\ to\ t\}$. Note that B is a violated set and $B \subsetneq A \setminus \{t'\}$ which violates the fact that A is a minimal violated set. Therefore, exactly one SCC of F is in A.

Next we prove the second statement. Let s be a Steiner node (if exists) in $A \setminus C_A$. If there is no edge in F oriented from s to C_A, then $A \setminus \{s\}$ is a violated set, because the graph is quasi-bipartite and the fact that A is a violated set itself, contradicting the fact that A is a minimal violated set. ◀

Note that the above lemma limits the interaction between two active moats. More precisely, two active moats can only share Steiner nodes that lie outside of the SCCs in the moats.

▶ **Definition 9** (The SCC part of active moats). *Given a set of edges F and an active set A (with respect to F), we denote by C_A the SCC (with respect to F) inside A.*

We use C_A rather than C_A^F because the set F will always be clear from the context.

Finally we recall bounds on the size of K_r-minor free graphs that we use at the end of our analysis.

▶ **Theorem 10** (Thomason 2001 [34]). *Let $G = (V, E)$ be a K_r-minor free graph with no parallel edges. Then, $|E| \leq O(r \cdot \sqrt{\log r})|V|$ and this bound is asymptotically tight. The constant in the O-notation in the above theorem is at most 3 for large enough r.*

Bipartite planar graphs are K_5-minor free, but we know of explicit bounds sizes. The following is the consequence of Euler's formula that will be useful in our tighter analysis for quasi-bipartite, planar graphs.

▶ **Lemma 11.** *Let $G = (V, E)$ be a bipartite planar graph with no parallel edges. Then, $|E| \leq 2 \cdot |V|$.*

3 Standard primal-dual algorithm and a bad example

Given a DST instance with $G = (V, E)$, $r \in V$ as the root, and $X \subseteq V - \{r\}$ as the terminal set, we define $\mathcal{S} := \{S \subsetneq V : r \notin S,\ and\ S \cap X \neq \emptyset\}$. We consider the dual of (**Primal-LP**).

$$\text{maximize:} \quad \sum_{S \in \mathcal{S}} y_S \qquad \textbf{(Dual-LP)}$$

$$\text{subject to:} \quad \sum_{\substack{S \in \mathcal{S}: \\ e \in \delta^{in}(S)}} y_S \leq c(e) \quad \forall e \in E \qquad (2)$$

$$y \geq 0$$

As we discussed in the introduction, a standard primal-dual algorithm solves arborescence problem on any directed graph [11]. Naturally, our starting point was to investigate this primal-dual algorithm for DST instances. We briefly explain this algorithm here. At the beginning we let $F := \emptyset$. Uniformly increase the dual constraints corresponding to active moats and if a dual constraint goes tight, we add the corresponding edge to F. Update the active sets based on F (see Definition 6) and repeat this procedure. At the end, we do a reverse delete, i.e., we go over the edges in F in the reverse order they have been added to F and remove it if the feasibility is preserved. Unfortunately, for DST instances in quasi-bipartite planar graphs, there is a bad example (see Figure 1), that shows the total growth of the dual variables is $2 + k \cdot \epsilon$ while the optimal solution costs $2 + k + k \cdot \epsilon$ for arbitrarily large k. So the dual objective is not enough to pay for the cost of the edges in F (i.e., we have to multiply the dual objective by $O(k)$ to be able to pay for the edges in F).

What is the issue and how can we fix it? One way to get an $O(1)$-approximation is to ensure at each iteration the number of edges (in the final solution) whose dual constraints are losing slack at this iteration is proportioned to the number of active moats. In the bad example (Figure 1), when the bottom moat is paying toward the downward blue edges, there are only two active moats but there are k downward blue edges that are currently being paid for by the growing dual variables.

To avoid this issue, we consider the following idea: once the bottom active moat grew enough so that the dual constraints corresponding to all the downward blue edges are tight we purchase an arbitrary one of them, say (r, z_k) for our discussion here. Once the top active moat reaches z_1 instead of skipping the payment for this edge (since the dual constraint for (w_2, z_1) is tight), we let the active moat pay towards this edge again by ignoring previous payments to the edge, and then we purchase it once it goes tight. Note that now we violated the dual constraint for (w_2, z_1) by a multiplicative factor of 2. Do the same for all the other downward blue edges (except (r, z_k) that was purchased by the bottom moat). Now it is easy to see that we grew enough dual objective to approximately pay for the edges that we purchased. We make this notion precise by defining different roles for downward blue edges in the next section. In general, each edge can serve up to two roles and has two "buckets" in which it receives payment: each moat pays towards the appropriate bucket depending on the role that edge serves for that moat. An edge is only purchased if one of its buckets is filled and some tiebreaking criteria we mention below is satisfied.

4 Our primal-dual algorithm

As we discussed in the last section, we let the algorithm violate the dual constraint corresponding to an edge by a factor of 2 and hence we work with the following modified Dual-LP:

$$\begin{aligned}
\text{maximize:} \quad & \sum_{S \in \mathcal{S}} y_S && \textbf{(Dual-LP-Modified)} \\
\text{subject to:} \quad & \sum_{\substack{S \in \mathcal{S}: \\ e \in \delta^{in}(S)}} y_S \leq 2 \cdot c(e) \quad \forall e \in E && (3) \\
& y \geq 0
\end{aligned}$$

Note that the optimal value of (**Dual-LP-Modified**) is at most twice the optimal value of (**Dual-LP**) because consider a feasible solution y for the former LP then $\frac{y}{2}$ is feasible for the latter LP.

Let us define the different buckets for each edge that are required for our algorithm.

Antenna, expansion and killer buckets. We say edge $e = (u, v)$ is an *antenna* edge if $u \notin X \cup \{r\}$ and $v \in X$, in other words, if the tail of e is a Steiner node and the head of e is a terminal. For every antenna edge we associate an antenna bucket with size $c(e)$. For every non-antenna edge e, we associate two buckets, namely *expansion* and *killer* buckets, each of size $c(e)$. The semantics of these labels will be introduced below.

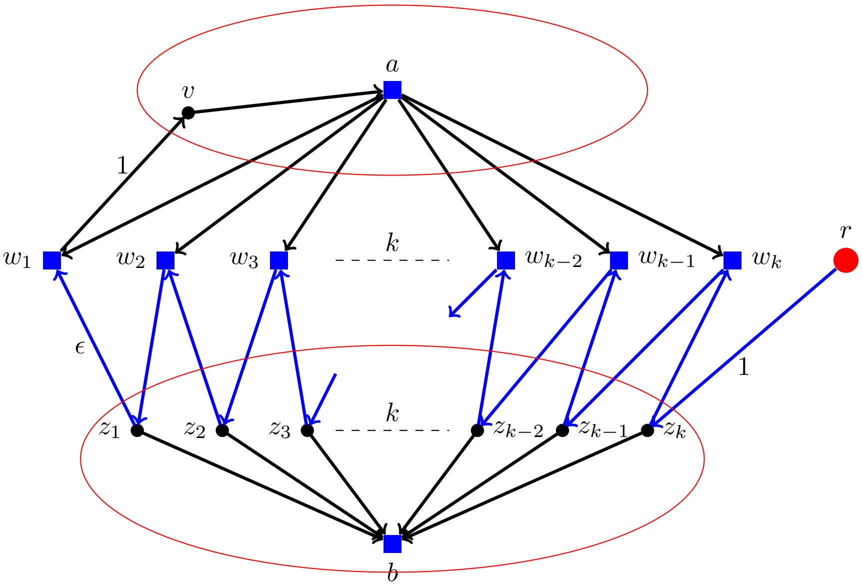


Figure 1 This is an example to show why a standard primal-dual algorithm fails. The square vertices are terminals. The downward blue edges (i.e., (w_i, z_{i-1})'s for $2 \leq i \leq k$) have cost 1, the upward blue edges (i.e., (z_i, w_i)'s for $1 \leq i \leq k$) have cost ϵ. The cost of the black edges are 0 except (w_1, v) who has cost 1. Note any feasible solution contains all the blue edges and the cost of an optimal solution is $k + k \cdot \epsilon + 1$. However, it is easy to see the total dual variables that are grown using a standard primal-dual algorithm is $2 + k \cdot \epsilon$.

Now we, informally, describe our algorithm, see Algorithm 1 for the detailed description. Recall the definition of active moats (Definition 6).

Growth phase. At the beginning of the algorithm we set $F := \emptyset$ and every singleton terminal is an active moat. As long as there is an active moat with respect to F do the following: uniformly increase the dual variables corresponding to the active moats. Let $e \notin F$ be an antenna edge with its head in an active moat, then the active moat pays towards the antenna bucket of e. Now suppose $e = (u, v) \notin F$ is a non-antenna edge, so $u \in X \cup \{r\}$. For every active moat A that contains v, if C_A (see Definition 9) is a subset of an active set A' with respect to $F \cup \{e\}$, then A pays toward the expansion bucket of e and otherwise A pays towards the killer bucket of e.

Uniformly increase the dual variables corresponding to active moats until a bucket for an edge e becomes full (antenna bucket in case e is an antenna edge, and expansion or killer bucket if e is a non-antenna edge), add e to F. Update the set of active moats $\mathcal{A}$ according to set F.

Pruning. Finally, we do the standard reverse delete meaning we go over the edges in F in the reverse order they have been added and if the resulting subgraph after removing an edge is still feasible for the DST instance, remove the edge and continue.

The following formalizes the different roles of a non-antenna edge that we discussed above.

▶ **Definition 12** (Relation between non-antenna edges and active moats). *Given a subset of edges $F \subseteq E$, let $\mathcal{A}$ be the set of all active moats with respect to F. Consider a non-antenna edge $e = (u, v)$ (so $u \in X \cup \{r\}$). Suppose $v \in A$ where $A \in \mathcal{A}$. Then,*

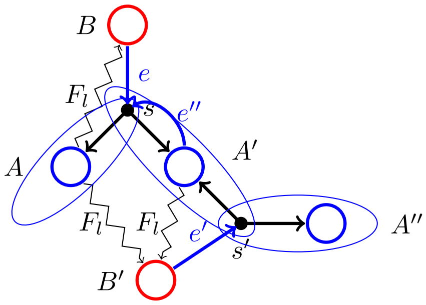


Figure 2 Above is a part of a graph at the beginning of iteration l in the algorithm. F_l denotes the set F at this iteration. The circles are SCCs in (V, F_l). Blue circles are inside some active moats shown with ellipses. The black dots s and s' are Steiner nodes. The black edges and the zigzag paths are in F_l. The edges e, e', and e'' have not been purchased yet (i.e., $e, e', e'' \notin F_l$). Since C_A is a subset of an active moat namely $A \cup B \cup \{s\}$ with respect to $F_l \cup \{e\}$, e is an expansion edge with respect to A. However, e is a killer edge with respect to A' and e'' is a killer edge with respect to A. Finally, e' is a killer edge with respect to A' (and A'') because there is a $F_l \cup \{e'\}$-path from C_A to $C_{A'}$ (and $C_{A''}$), therefore $C_{A'}$ (and $C_{A''}$) cannot be inside an active moat with respect to $F_l \cup \{e'\}$.

- *we say e is an expansion edge with respect to A under F if there is a subset of vertices A' that is active with respect to $F \cup \{e\}$ such that $C_A \subsetneq A'$,*
- *otherwise we say e is a killer edge with respect to A.*

For example, all exiting edges from r that are not in F is a killer edge with respect to any active moat (under F) it enters. See Figure 2 for an illustration of the above definition.

Intuition behind this definition. A non-antenna edge $e = (u, v)$ is a killer edge with respect to an active moat A, if and only if, there is a dipath in $F \cup \{e\}$ from r or $C_{A'}$ to C_A where $A' \neq A$ is an active moat with respect to F. Note that adding e to F will make the dual variable corresponding to A stop growing and that is why we call e a killer edge with respect to A. For example, in Figure 2, both e and e' are killer edges with respect to A'. On the other hand, if $e = (u, v)$ is an expansion edge with respect to A, then C_A will be a part of a "bigger" active moat with respect to $F \cup \{e\}$ and hence the name expansion edge for e. For example, in Figure 2, e is an expansion edge with respect to A because in $F \cup \{e\}$, $A \cup B \cup \{s\}$ is an active moat whose SCC contains C_A.

The complete description of the algorithm is given in Appendix A. Note that the purchased edge e_l at iteration l enters some active moat at iteration l.

After the algorithm finishes, then we label non-antenna edges by expansion/killer as determined by the following rule:

▶ **Definition 13** (Killer and expansion edges)**.** *Consider iteration l of the algorithm where we added a non-antenna edge e_l to F. We label e_l as expansion (killer) if the expansion (killer) bucket of e becomes full at iteration l, break ties arbitrarily.*

Following remark helps to understand the above definition better.

▶ Remark 14. It is possible that one bucket becomes full for an edge yet we do not purchase the edge with that bucket label (killer or expansion) due to tiebreaking when multiple buckets become full. For example, this would happen in our bad example for the downward blue edges: their killer buckets are full yet all but one are purchased as expansion edges.

Let us explain the growth phase of Algorithm 1 on the bad example in Figure 1. Since the early iterations of the algorithm on this example are straightforward, we start our explanation from the iteration where the active moats are $A = \{b, z_1, z_2, ..., z_k\}$ and $A' = \{a, v\}$.

Every (w_i, z_{i-1}) for $2 \leq i \leq k$ is a killer edge with respect to A so A pays toward the killer buckets of these edges. At the same iteration, (w_1, v) is an expansion edge with respect to A' so A' pays toward the expansion bucket of this edge. Now the respected buckets for all mentioned edges are full. Arbitrarily, we pick one of these edges, let us say (w_k, z_{k-1}), and add it to F. Then, A stops growing. In the next iteration, we only have one active moat A'. Since (w_1, v) is still expansion edge with respect to A' and its (expansion) bucket is full, in this iteration we add (w_1, v) to F and after updating the active moats, again we only have one active moat $\{a, v, w_1\}$ which by abuse of notation we denote it by A'. Next iteration we buy the antenna edge (z_1, w_1) and the active moat now is $A' = \{a, v, w_1, z_1\}$. In the next iteration, the crucial observation is that the killer bucket of (w_2, z_1) is full (recall the A payed toward the killer bucket of (w_2, z_1)); however, (w_2, z_1) is an expansion edge with respect to A' so A' will pay towards its expansion bucket and then purchases it. Similarly, the algorithm buys (w_i, z_{i-1})'s except (w_k, z_{k-1}) because this edge is in F already (recall we bought this edge with A). Finally, (r, z_k) is a killer edge with respect to the active moat in the last iteration and we purchase it.

5 The analysis

Because of the space constraints, we defer most of the proofs to the full version of the paper [16] and defer to the full version.

The general framework for analyzing primal-dual algorithms is to use the dual constraints to relate the cost of purchased edges and the dual variables. However, here we do not use the dual constraints and rather we use the buckets we created for each edge. Recall $\overline{F}$ is the solution output by Algorithm 1. We define $\overline{F}_{\text{Killer}}$ to be the set of edges in $\overline{F}$ that was purchased as killer edge (recall definition 13). Similarly define $\overline{F}_{\text{Exp}}$ and $\overline{F}_{\text{Ant}}$. For each iteration l, we denote by F_l the set F at this iteration, $\mathcal{A}_l$ denotes the set of active moats with respect to F_l, and ϵ_l is the amount we increased the dual variables (corresponding to active moats) with at iteration l. Finally, Let y^* be the dual solution for (**Dual-LP-Modified**) constructed in the course of the algorithm. We use the following notation throughout the analysis.

▶ **Definition 15.** *Fix an iteration l. For any $A \in \mathcal{A}_l$, let*

$$\Delta^l_{\text{Killer}}(A) := \{e \in \overline{F}_{\text{Killer}} : \ e \textit{ is killer with respect to } A \textit{ under } \ F_l\},$$

in other words, $\Delta^l_{\text{Killer}}(A)$ is the set of all killer edges in $\overline{F}$ such that they are killer edge with respect to A at iteration l. Similarly define $\Delta^l_{\text{Exp}}(A)$.

Let $\Delta^l_{\text{Ant}}(A) := \{e \in \overline{F}_{\text{Ant}} : \ e \in \delta^{in}(A)\}$. Finally, we define

$$\Delta^l(A) := \Delta^l_{\text{Killer}}(A) \cup \Delta^l_{\text{Exp}}(A) \cup \Delta^l_{\text{Ant}}(A).$$

Note $\Delta^l_{\text{Killer}}(A)$, $\Delta^l_{\text{Exp}}(A)$, and $\Delta^l_{\text{Ant}}(A)$ are pairwise disjoint for any $A \in \mathcal{A}_l$.

Suppose we want to show that the performance guarantee of Algorithm 1 is $2 \cdot \alpha$ for some $\alpha \geq 1$, it suffices to show the following: for any iteration l we have

$$\sum_{S \in \mathcal{A}_l} |\Delta^l(S)| \leq \alpha \cdot |\mathcal{A}_l|. \tag{4}$$

Once we have (4), then the $(2 \cdot \alpha)$-approximation follows easily:

$$
\begin{aligned}
\sum_{e \in \overline{F}} c(e) &= \sum_{e \in \overline{F}_{\text{Killer}}} \sum_{l} \sum_{\substack{S \in \mathcal{A}_l: \\ e \in \Delta^l_{\text{Killer}}(S)}} \epsilon_l + \sum_{e \in \overline{F}_{\text{Exp}}} \sum_{l} \sum_{\substack{S \in \mathcal{A}_l: \\ e \in \Delta^l_{\text{Exp}}(S)}} \epsilon_l + \sum_{e \in \overline{F}_{\text{Ant}}} \sum_{l} \sum_{\substack{S \in \mathcal{A}_l: \\ e \in \Delta^l_{\text{Ant}}(S)}} \epsilon_l && (5)\\
&= \sum_{l} \epsilon_l \cdot \sum_{S \in \mathcal{A}_l} |\Delta^l(S)| && (6)\\
&\leq \alpha \cdot \sum_{l} |\mathcal{A}_l| \epsilon_l && (7)\\
&= \alpha \cdot \sum_{S \subseteq V \setminus \{r\}} y^*_S && (8)\\
&\leq \alpha \cdot \big(2 \cdot \mathrm{OPT}(\mathbf{Dual-LP})\big) && (9)\\
&= 2 \cdot \alpha \cdot \mathrm{OPT}(\mathbf{Primal-LP}) && (10)\\
&\leq 2 \cdot \alpha \cdot \mathrm{OPT}, && (11)
\end{aligned}
$$

where the first equality follows from the algorithm, the second equality is just an algebraic manipulation, (7) follows from (4). Equality (8) follows from the fact we uniformly increased the dual variables corresponding to active moats by ϵ_l at iteration l, (9) follows from feasibility of $\frac{y^*}{2}$ for (**Dual-LP**), and (10) follows from strong duality theorem for linear programming.

It remains to show (4) holds. Consider iteration l. Using the bound on the total degree of nodes in G (using minor-free properties) to show (4), it suffices to bound the number of edges in $\bar{F}_{\text{Ant}} \cup \bar{F}_{\text{Killer}} \cup \bar{F}_{\text{Exp}}$ that are being paid by some active moat at iteration l, by $O(|\mathcal{A}_l|)$. We provide charging schemes for each type of edges, separately. Since G is quasi-bipartite, it is easy to show that for each active moat $A \in \mathcal{A}_l$, there is at most one antenna edge in $\bar{F}$ that enters A, this is proved in Section 5.1. The charging scheme for killer edges is also simple as one can charge a killer edge to an active moat that it kills; this will be formalized in Section 5.2. However, the charging scheme for expansion edges requires more care and novelty. The difficulty comes from the case that an expansion edge is not pruned because it would disconnect some terminals that are not part of any active moat that e is entering this iteration.

Our charging scheme for expansion edges is more global. In a two-stage process, we construct an auxiliary tree that encodes some information about which nodes can be reached from SCCs using edges in F_l (which is the information we used in the definition of expansion edge). Then using a token argument, we leverage properties of our construction to show the number of expansion edges is at most twice the number of active moats in any iteration. These details are presented in 5.3. Finally, in Section 5.4 we put all the bounds we obtained together and derive our approximation factors.

5.1 Counting the number of antenna edges in an iteration

Fix an iteration l. Recall F_l denotes the set F at iteration l, and $\mathcal{A}_l$ denotes the set of active moats with respect to F_l. It is easy to bound the number of antenna edges in $\overline{F}$ against $|\mathcal{A}_l|$. We do this in the next lemma.

▶ **Lemma 16.** *At the beginning of each iteration l, we have* $\sum\limits_{A \in \mathcal{A}_l} |\Delta^l_{\text{Ant}}(A)| \leq |\mathcal{A}_l|$.

5.2 Counting the number of killer edges in an iteration

We introduce a notion called *alive* terminal which helps us to bound the number of killer edges at a fixed iteration against the number of active moats in that iteration. Also this notion explains the name killer edge. Throughout the algorithm, we show every active moat contains exactly one alive terminal and every alive terminal is in an active moat.

We consider how terminals can be "killed" in the algorithm by associating active moats with terminals that have not yet been part of a moat that was killed. At the beginning of the algorithm, we mark every terminal alive, note that every singleton terminal set is initially an active moat as well. Let $e_l = (u, v)$ be the edge that was added to F_l at iteration l. If $e_l = (u, v)$ is a non-antenna edge, then for every active set A such that e_l is a killer edge with respect to A under F_l, mark the alive terminal in A as *dead*[4]. If $e_l = (u, v)$ is an antenna edge, then for every active moat A such that $e_l \in \delta^{in}(A)$ and C_A is **not** in any active moat with respect to $F_l \cup \{e_l\}$, then mark the alive terminal in A as dead[5].

The important observation here is that by definition, if e_l is a killer edge, then there must be an active set that satisfies the above condition, hence there is at least one alive terminal that will be marked dead because of e_l. In the case that e_l is bought as killer edge, arbitrarily pick an alive terminal t_{e_l} that dies because of e_l and assign e_l to t_{e_l}. Note that t_{e_l} was alive until e_l was added to F_l.

▶ **Definition 17.** *Fix an iteration l. We define*

$$\overline{F}^{l}_{\text{Killer}} := \bigcup_{A \in \mathcal{A}_l} \Delta^{l}_{\text{Killer}}(A),$$

in other words, $\overline{F}^{l}_{\text{Killer}}$ is the set of all killer edges in $\overline{F}$ such that some active moat(s) is paying toward their killer bucket at iteration l.

Now we can state the main lemma of this section.

▶ **Lemma 18.** *At the beginning of each iteration l, we have $|\overline{F}^{l}_{\text{Killer}}| \leq |\mathcal{A}_l|$.*

Note that the above lemma does not readily bound $\sum_{A \in \mathcal{A}_l} |\Delta^{l}_{\text{Killer}}(A)|$ against $|\mathcal{A}_l|$ which is required to prove inequality (4). We need the properties of minor-free graphs to do so. In the next section we prove a similar result for expansion edges and then using the properties of the underlying graph, we demonstrate our approximation guarantee.

5.3 Counting the number of expansion edges in an iteration

The high level idea to bound the number of expansion edges is to look at the graph $\overline{F} \cup F_l$ and contract all SCCs[6] of (V, F_l). Then, we construct an auxiliary tree that highlights the role of expansion edges to the connectivity of active moats. Finally, using this tree we provide our charging scheme and show the number of edges in $\overline{F}_{\text{Exp}}$ that are being paid by some active moats at iteration l is at most twice the number of active moats.

[4] It is possible, e_l is bought as an expansion edge but kills some alive terminals. For example, in Figure 2 suppose e is being added to F_l at iteration l as an expansion edge (note that A pays toward the expansion bucket of e). Then, we mark the alive terminal in A' as dead because e is a killer edge with respect to A' under F_l.

[5] For example, suppose the antenna edge $e_l = (u, v) \in \delta^{in}(A)$ is being added to F_l and u is in $C_{A'}$ for some active moat A'. Then, after adding e_l to F_l, we mark the alive terminal in A as dead.

[6] Recall that we do NOT call a Steiner node that is a singleton strongly connected component of (V, F_l) an SCC. So every SCC in (V, F_l) is either $\{r\}$ or contains at least one terminal node.

We fix an iteration l for this section. First let us recall some notation and definition that we use extensively in this section. (i) $\overline{F}$ is the output solution of the algorithm. (ii) $F_l \subseteq E$ is the set of purchased edges in the growing phase up to the beginning of iteration l (i.e., set F in the algorithm at iteration l). (iii) $\mathcal{A}_l$ is the set of active moats with respect to F_l (see Definition 6). Recall each $A \in \mathcal{A}_l$ is consists of an SCC (with respect to edges in F_l) and bunch of Steiner nodes. Denote by C_A the SCC part of A.

We define an analogous of Definition 17 for expansion edges.

▶ **Definition 19.** *Fix an iteration l. Then, we define*

$$\overline{F}^l_{\mathrm{Exp}} := \bigcup_{A \in \mathcal{A}_l} \Delta^l_{\mathrm{Exp}}(A),$$

in other words, $\overline{F}^l_{\mathrm{Exp}}$ is the set of all expansion edges in $\overline{F} \setminus F_l$ such that some active moat(s) is paying toward their expansion bucket at iteration l.

This section is devoted to prove the following inequality.

▶ **Lemma 20.** *At the beginning of each iteration l of the algorithm, we have $|\overline{F}^l_{\mathrm{Exp}}| \leq 2 \cdot |\mathcal{A}_l|$.*

Sketch of the proof. We start by giving a sketch of the proof of Lemma 20. Consider the subgraph $F_l \cup \overline{F}$ of G. Contract every SCC of (V, F_l) and denote the resulting subgraph by H (keeping all copies of parallel edges that may result). For every non-root, non-Steiner node $v \in V(H)$, we call v active if it is a contraction of an SCC that is a subset of an active moat in $\mathcal{A}_l$, otherwise we call v inactive. Note that r is a singleton SCC in (V, F_l) and therefore $r \in V(H)$. We call an edge in $E(H)$ an expansion edge, if its corresponding edge is in $\overline{F}^l_{\mathrm{Exp}}$. Note that every non root vertex in $V(H)$ is either labeled active/inactive, or it is a Steiner node. Lemma 20 follows if we show the number of expansion edges in H is at most twice the number of active vertices in H. As we stated at the beginning of this section, we use an arborescence that highlights the role of expansion edges to the connectivity of active vertices in H. A bit more formally, we show if every expansion edge is "good" with respect to the arborescence, which is formalized below, then every expansion edge is "close" to an active vertex in H and we use this in our charging scheme.

Given an arborescence T, define $\mathrm{Elevel}_T(v)$ to be the expansion level of v with respect to T, i.e., the number of expansion edges on the dipath from r to v in T.

▶ **Definition 21.** *Given an arborescence T and an expansion edge $e = (u, v)$, we say e is a* good *expansion edge with respect to T if one of the following cases happens:*

- *Type 1: If u has an active ancestor w such that* $\mathrm{Elevel}_T(w) = \mathrm{Elevel}_T(u)$.
- *Type 2: If e is not of type 1 and the subtree rooted at u has an active vertex w such that* $\mathrm{Elevel}_T(w) \leq \mathrm{Elevel}_T(u) + 1$.

Every expansion edge that is not of type 1 or type 2, is called a bad *expansion edge with respect to T.*

A starting point for an arborescence that every expansions edge is good, is a shortest path arborescence rooted at r in H where each expansion edge has cost 1 and the rest of the edges have cost 0. However, as Figure 3 shows, there could be some *bad* expansion edges in this arborescence. For example, e is a *bad* expansion edge with respect to the arborescence in Figure 3 (b). Since B_2, the tail of e, is an inactive vertex, there must be an active vertex, namely A_3, that has a dipath from A_3 to B_2 in F_ℓ. Then, we can "cut" the subtree rooted at B_2 and "paste" it under A_3 as shown in Figure 3(c). It is easy to verify that now every expansion edge is good with respect to the arborescence in Figure 3(c). We formalize this

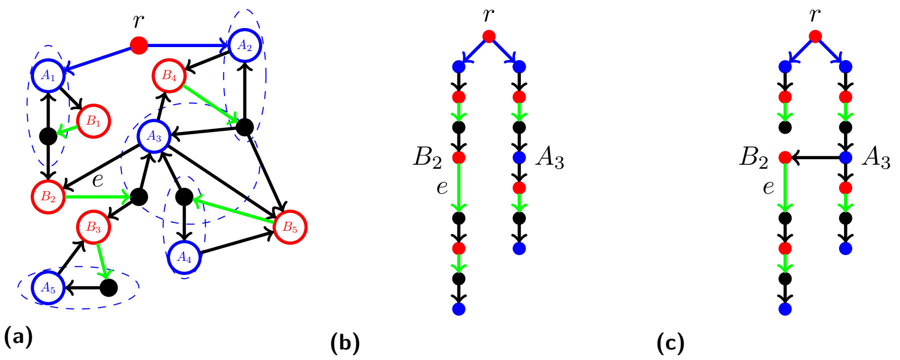


Figure 3 (a) shows subgraph $F_l \cup \bar{F}$ of G, in particular, the SCCs of (V, F_l) are shown with circles but the nodes inside SCCs are not shown for simplicity. The blue SCCs are inside some active moats shown with dashed ellipses. Contracting all the SCCs result in the graph H discussed before. Black edges are in F_l, blue edges are in $\bar{F} \setminus F_l$, and green edges are in $\bar{F}^l_{\text{Exp}}$. In (b), we have a shortest path arborescence rooted at r in H where the cost of edges is one if it is green and zero otherwise. Recall the definition of H at the beginning of this sketched proof. Note that e is a bad expansion edge with respect to this arborescence. In (c), we show how to construct an arborescence using cut-and-paste procedure so that every expansion edge is a good expansion edge in the resulting arborescence.

“cut and paste” procedures in Algorithm 2 in the full version of the paper and prove the output of the algorithm is an arborescence with the property that every expansion edge is good. Given an arborescence that every expansion edge is good, we show there is a rather natural charging scheme that proves Lemma 20.

Charging scheme. At the beginning we label every token **unused**. We process all the vertices with height l. For each expansion edge whose tail has height l we assign an unused token to it and change the label of the assigned token to **used**. Then we move to height $l-1$ and repeat the process. Fix height l. We do the following for every vertex u with this height: if there is no expansion edge whose tail is u then mark u as processed. Otherwise let $(u, v_1), \ldots, (u, v_k)$ be all the expansion edges whose tail is u. Note that by definition of type 1 and 2, either (i) all (u, v_i)'s are type 1 or (ii) all are type 2. Base on these two cases we do the following:

(i) Let $(u, v_1), \ldots, (u, v_k)$ be the expansion edges of type 1. For each $1 \leq i \leq k$ there is at least one unused token in $T^*_{v_i}$. Pick one such unused token and assign it to (u, v_i) and change its label to used. Mark u as processed.

(ii) Let $(u, v_1), \ldots, (u, v_k)$ be the expansion edges of type 2. For each $1 \leq i \leq k$ there is at least one unused token in $T^*_{v_i}$. Pick one such unused token and assign it to (u, v_i) and change its label to used. Furthermore, after this there is at least one more unused token in T^*_u. Mark u as processed.

The proof of correctness of this charging scheme is based on induction on the height l. ◀

5.4 Putting everything together

Fix an iteration l. We use Lemmas 18 & 20 and the properties of graph G to bound $\sum_{A \in \mathcal{A}_l} |\Delta^l_{\text{Killer}}(A) \cup \Delta^l_{\text{Exp}}(A)|$. Consider an active moat A and its SCC C_A. We show there is at most one killer/expansion edge that enters C_A. So the remaining killer/expansion edges must enter some Steiner node in $A \setminus C_A$. We use this fact later.

▷ Claim 22. Fix an iteration l and an active moat $A \in \mathcal{A}_l$. There is at most one edge in $\Delta^l_{\text{Killer}}(A) \cup \Delta^l_{\text{Exp}}(A)$ whose head is in C_A.

Consider the graph $F_l \cup \overline{F}$. Remove all vertices that are not in an active moat at this iteration. For each active moat A, remove all Steiner nodes in $A \setminus C_A$ that are not the head of any edge in $\overline{F}^l_{\text{Killer}} \cup \overline{F}^l_{\text{Exp}}$. Then, for each $A \in \mathcal{A}_l$ contract C_A to a single vertex and call the contracted vertex by C_A. Finally, if there are parallel edges, arbitrarily keep one of them and remove the rest[7]. Call the resulting graph G'.

Now we relate the sum we are interested in to bound with the sum of the indegree of vertices in G'.

▷ Claim 23. For each active moat $A \in \mathcal{A}_l$, we have

$$|\Delta^l_{\text{Killer}}(A) \cup \Delta^l_{\text{Exp}}(A)| \le |\delta^{in}_{G'}(C_A)| + 1. \tag{12}$$

Next, using Lemmas 18 & 20 we bound the number of vertices in G'.

▷ Claim 24. Fix an iteration l. Then, $|V(G')| \le 4 \cdot |\mathcal{A}_l|$.

Finally, we prove Theorems 1 & 3.

Proof of Theorem 1. Since G is K_r-minor free so does G'. So we can write

$$\begin{aligned}\sum_{A \in \mathcal{A}_l} \left|\Delta^l_{\text{Killer}}(A) \cup \Delta^l_{\text{Exp}}(A)\right| &\le \sum_{A \in \mathcal{A}_l} \left(|\delta^{in}_{G'}(C_A)| + 1\right) \\ &= |E(G')| + |\mathcal{A}_l| \\ &\le O(r \cdot \sqrt{\log r}) \cdot 4 \cdot |\mathcal{A}_l| + |\mathcal{A}_l| \\ &= O(r \cdot \sqrt{\log r})|\mathcal{A}_l|,\end{aligned} \tag{13}$$

where the inequality follows from Claim 23 and the second inequality follows from Claim 24 together with Theorem 10.

Next we show (4) holds for $\alpha = O(r \cdot \sqrt{\log r})$.

$$\begin{aligned}\sum_{A \in \mathcal{A}_l} |\Delta^l(A)| &= \sum_{A \in \mathcal{A}_l} |\Delta^l_{\text{Killer}}(A) \cup \Delta^l_{\text{Exp}}(A)| + \sum_{A \in \mathcal{A}_l} |\Delta^l_{\text{Ant}}(A)| \\ &\le O(r \cdot \sqrt{\log r})|\mathcal{A}_l| + |\mathcal{A}_l| \\ &= O(r \cdot \sqrt{\log r})|\mathcal{A}_l|,\end{aligned}$$

where inequality follows from inequality (13) and Lemma 16.

As we discussed at the beginning of Section 5 that if (4) holds for α then we have a $(2 \cdot \alpha)$-approximation algorithm. Hence, Algorithm 1 is an $O(r \cdot \sqrt{\log r})$-approximation for DST on quasi-bipartite, K_r-minor free graphs. ◀

Proof of Theorem 3. The proof is exactly the same as proof of Theorem 1 except instead of $O(r \cdot \sqrt{\log r})$ in (13) we have 2 because G' is a bipartite planar graph, see Lemma 11. Now we can write $\sum_{A \in \mathcal{A}_l} \left|\Delta^l_{\text{Killer}}(A) \cup \Delta^l_{\text{Exp}}(A)\right| \le 9 \cdot |\mathcal{A}_l|$ and $\sum_{A \in \mathcal{A}_l} |\Delta^l(A)| \le 10 \cdot |\mathcal{A}_l|$. Therefore, (4) holds for $\alpha = 10$ and hence we have a 20-approximation algorithm, as desired. ◀

[7] Note that all the parallel edges are antenna edges and so removing them does not affect the quantity $\sum_{A \in \mathcal{A}_l} |\Delta^l_{\text{Killer}}(A) \cup \Delta^l_{\text{Exp}}(A)|$ we are trying to bound.

References

1 MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Dániel Marx. Approximation schemes for steiner forest on planar graphs and graphs of bounded treewidth. *Journal of the ACM (JACM)*, 58(5):1–37, 2011.

2 Marshall Bern and Paul Plassmann. The steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32(4):171–176, 1989.

3 Glencora Borradaile, Philip Klein, and Claire Mathieu. An o (n log n) approximation scheme for steiner tree in planar graphs. *ACM Transactions on Algorithms (TALG)*, 5(3):1–31, 2009.

4 Jarosław Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. Steiner tree approximation via iterative randomized rounding. *Journal of the ACM (JACM)*, 60(1):1–33, 2013.

5 Gruia Calinescu and Alexander Zelikovsky. The polymatroid steiner problems. *J. Combonatorial Optimization*, 33(3):281–294, 2005.

6 Deeparnab Chakrabarty, Nikhil R Devanur, and Vijay V Vazirani. New geometry-inspired relaxations and algorithms for the metric steiner tree problem. *Mathematical programming*, 130(1):1–32, 2011.

7 Chun-Hsiang Chan, Bundit Laekhanukit, Hao-Ting Wei, and Yuhao Zhang. Polylogarithmic approximation algorithm for k-connected directed steiner tree on quasi-bipartite graphs. *arXiv preprint*, 2019. arXiv:1911.09150.

8 Moses Charikar, Chandra Chekuri, To-Yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation algorithms for directed steiner problems. *Journal of Algorithms*, 33(1):73–91, 1999.

9 Erik D Demaine, MohammadTaghi Hajiaghayi, and Philip N Klein. Node-weighted steiner tree and group steiner tree in planar graphs. *ACM Transactions on Algorithms (TALG)*, 10(3):1–20, 2014.

10 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 624–633, 2014.

11 Jack Edmonds. Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240, 1967.

12 Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

13 Andreas Emil Feldmann, Jochen Könemann, Neil Olver, and Laura Sanità. On the equivalence of the bidirected and hypergraphic relaxations for steiner tree. *Mathematical programming*, 160(1):379–406, 2016.

14 Zachary Friggstad, Jochen Könemann, Young Kun-Ko, Anand Louis, Mohammad Shadravan, and Madhur Tulsiani. Linear programming hierarchies suffice for directed steiner tree. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 285–296. Springer, 2014.

15 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)*, volume 53 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:11, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

16 Zachary Friggstad and Ramin Mousavi. A constant-factor approximation for quasi-bipartite directed steiner tree on minor-free graphs. *arXiv preprint*, 2021. arXiv:2111.02572.

17 Zachary Friggstad and Ramin Mousavi. An O(log k)-Approximation for Directed Steiner Tree in Planar Graphs. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, volume 261 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 63:1–63:14, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

18 Rohan Ghuge and Viswanath Nagarajan. Quasi-polynomial algorithms for submodular tree orienteering and other directed network design problems. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1039–1048. SIAM, 2020.

19 Michel X Goemans, Neil Olver, Thomas Rothvoß, and Rico Zenklusen. Matroids and integrality gaps for hypergraphic steiner tree relaxations. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1161–1176, 2012.

20 Michel X Goemans and David P Williamson. The primal-dual method for approximation algorithms and its application to network design problems. *Approximation algorithms for NP-hard problems*, pages 144–191, 1997.

21 Fabrizio Grandoni, Bundit Laekhanukit, and Shi Li. O (log2 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*, pages 253–264, 2019.

22 Sudipto Guha, Anna Moss, Joseph Naor, and Baruch Schieber. Efficient recovery from power outage. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 574–582, 1999.

23 Eran Halperin and Robert Krauthgamer. Polylogarithmic inapproximability. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 585–594, 2003.

24 Tomoya Hibi and Toshihiro Fujito. Multi-rooted greedy approximation of directed steiner trees with applications. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 215–224. Springer, 2012.

25 Marek Karpinski and Alexander Zelikovsky. New approximation algorithms for the steiner tree problems. *Journal of Combinatorial Optimization*, 1(1):47–65, 1997.

26 Jochen Könemann, Sina Sadeghian, and Laura Sanita. An lmp o (log n)-approximation algorithm for node weighted prize collecting steiner tree. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 568–577. IEEE, 2013.

27 Shi Li and Bundit Laekhanukit. Polynomial integrality gap of flow lp for directed steiner tree. *arXiv preprint*, 2021. arXiv:2110.13350.

28 Carsten Moldenhauer. Primal-dual approximation algorithms for node-weighted steiner forest on planar graphs. *Information and Computation*, 222:293–306, 2013.

29 Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity.* Courier Corporation, 1998.

30 Hans Jürgen Prömel and Angelika Steger. A new approximation algorithm for the steiner tree problem with performance ratio 5/3. *Journal of Algorithms*, 36(1):89–101, 2000.

31 Sridhar Rajagopalan and Vijay V Vazirani. On the bidirected cut relaxation for the metric steiner tree problem. In *SODA*, volume 99, pages 742–751, 1999.

32 Gabriel Robins and Alexander Zelikovsky. Tighter bounds for graph steiner tree approximation. *SIAM Journal on Discrete Mathematics*, 19(1):122–134, 2005.

33 Thomas Rothvoß. Directed steiner tree and the lasserre hierarchy. *arXiv preprint*, 2011. arXiv:1111.5473.

34 Andrew Thomason. The extremal function for complete minors. *Journal of Combinatorial Theory, Series B*, 81(2):318–338, 2001.

35 Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *Journal of the ACM (JACM)*, 51(6):993–1024, 2004.

36 Alexander Zelikovsky. A series of approximation algorithms for the acyclic directed steiner tree problem. *Algorithmica*, 18(1):99–110, 1997.

37 Alexander Z Zelikovsky. An 11/6-approximation algorithm for the network steiner problem. *Algorithmica*, 9(5):463–470, 1993.

38 Leonid Zosin and Samir Khuller. On directed steiner trees. In *SODA*, volume 2, pages 59–63. Citeseer, 2002.

A Full description of the primal-dual algorithm

Algorithm 1 Primal-Dual Algorithm for DST on Quasi-Bipartite Graphs.

Input: Directed quasi-bipartite graph $G = (V, E)$ with edge costs $c(e) \geq 0$ for $e \in E$, a set of terminal $X \subseteq V \setminus \emptyset$, and a root vertex r.
Output: An arborescence $\overline{F}$ rooted at r such that each terminal is reachable from r in $\overline{F}$.

```
$\mathcal{A} \leftarrow \{\{v\} : v \in X\}$. {The active moats each iteration, initially all singleton terminal set.}
$y^* \leftarrow 0$. {The dual solution}
$F \leftarrow \emptyset$. {The edges purchased}
$l \leftarrow 0$. {The iteration counter}
$b_e^{\text{Ant}} \leftarrow 0$, $b_e^{\text{Exp}} \leftarrow 0$ and $b_e^{\text{Killer}} \leftarrow 0$. {The buckets}
Growing phase:
while until $\mathcal{A} \neq \emptyset$ do
  Find the maximum value $\epsilon \geq 0$ such that the following holds:
    (a) for every antenna edge $e$ we have $b_e^{\text{Ant}} + \sum_{\substack{A\in\mathcal{A}:\\ e\in\delta^{in}(A)}} \epsilon \leq c(e)$.
    (b) for every non-antenna edge $e$ we have $b_e^{\text{Exp}} + \sum_{\substack{A\in\mathcal{A}:\\ e \textit{ is expansion}\\ \textit{with resp. to } A}} \epsilon \leq c(e)$.
    (c) for every non-antenna edge $e$ we have $b_e^{\text{Killer}} + \sum_{\substack{A\in\mathcal{A}:\\ e \textit{ is killer with}\\ \textit{resp. to } A}} \epsilon \leq c(e)$.
  Increase the dual variables $y^*$ corresponding to each active moat by $\epsilon$.
  for every antenna edge $e$ do
    $b_e^{\text{Ant}} \leftarrow b_e^{\text{Ant}} + \sum_{\substack{A\in\mathcal{A}:\\ e\in\delta^{in}(A)}} \epsilon$.
  end for
  for every non-antenna edge $e$ do
    $b_e^{\text{Exp}} \leftarrow b_e^{\text{Exp}} + \sum_{\substack{A\in\mathcal{A}:\\ e \textit{ is expansion}\\ \textit{with resp. to } A}} \epsilon$.
    $b_e^{\text{Killer}} \leftarrow b_e^{\text{Killer}} + \sum_{\substack{A\in\mathcal{A}:\\ e \textit{ is killer with}\\ \textit{resp. to } A}} \epsilon$.
  end for
  pick any single edge $e_l \in \cup_{A\in\mathcal{A}} \delta^{in}(A)$ with one of (a)-(c) being tight (break ties arbitrarily).
  $F \leftarrow F \cup \{e_l\}$.
  update $\mathcal{A}$ based on the minimal violated sets with respect to $F$.
  $l \leftarrow l + 1$.
end while
Deletion phase:
$\overline{F} \leftarrow F$.
for $i$ from $l$ to 0 do
  if $\overline{F} \setminus \{e_i\}$ is a feasible solution for the DST instance then
    $\overline{F} \leftarrow \overline{F} \setminus \{e_i\}$.
  end if
end for
return $\overline{F}$
```

Algorithms for 2-Connected Network Design and Flexible Steiner Trees with a Constant Number of Terminals

Ishan Bansal ✉
Operations Research and Information Engineering, Cornell University, Ithaca, NY, USA

Joe Cheriyan ✉ 🏠
Department of Combinatorics and Optimization, University of Waterloo, Canada

Logan Grout ✉
Operations Research and Information Engineering, Cornell University, Ithaca, NY, USA

Sharat Ibrahimpur ✉ 🏠 iD
Department of Mathematics, London School of Economics and Political Science, UK

Abstract

The k-Steiner-2NCS problem is as follows: Given a constant (positive integer) k, and an undirected connected graph $G = (V, E)$, non-negative costs c on the edges, and a partition $(T, V \setminus T)$ of V into a set of terminals, T, and a set of non-terminals (or, Steiner nodes), where $|T| = k$, find a min-cost two-node connected subgraph that contains the terminals. The k-Steiner-2ECS problem has the same inputs; the algorithmic goal is to find a min-cost two-edge connected subgraph that contains the terminals.

We present a randomized polynomial-time algorithm for the unweighted k-Steiner-2NCS problem, and a randomized FPTAS for the weighted k-Steiner-2NCS problem. We obtain similar results for a capacitated generalization of the k-Steiner-2ECS problem.

Our methods build on results by Björklund, Husfeldt, and Taslaman (SODA 2012) that give a randomized polynomial-time algorithm for the unweighted k-Steiner-cycle problem; this problem has the same inputs as the unweighted k-Steiner-2NCS problem, and the algorithmic goal is to find a min-cost simple cycle C that contains the terminals (C may contain any number of Steiner nodes).

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Approximation algorithms, Capacitated network design, Network design, Parametrized algorithms, Steiner cycle problem, Steiner 2-edge connected subgraphs, Steiner 2-node connected subgraphs

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.14

Category APPROX

Related Version *Previous Version*: `https://doi.org/10.48550/arXiv.2206.11807`

Funding *Joe Cheriyan*: Supported in part by NSERC, RGPIN-2019-04197.
Sharat Ibrahimpur: Received funding from the following sources: NSERC grant 327620-09 and an NSERC DAS Award, European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. ScaleOpt–757481), and Dutch Research Council NWO Vidi Grant 016.Vidi.189.087.

1 Introduction

The k-Steiner-cycle problem is as follows: Given a constant k, and an undirected connected graph $G = (V, E)$, non-negative costs c on the edges, and a partition $(T, V \setminus T)$ of V into a set of terminals, T, and a set of non-terminals (or, Steiner nodes), where $|T| = k$, find a minimum-cost simple cycle C that contains all the terminals (and any subset of Steiner nodes). Note that this is an optimization problem and not a search problem. To the best of our knowledge,

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 14; pp. 14:1–14:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

no polynomial-time (deterministic or randomized) algorithm is known for finding an optimal solution of the (weighted) k-Steiner-cycle problem, even for $k = 3$; this problem has been open for several decades, see [7]. Björklund, Husfeldt, and Taslaman (SODA 2012) [3] give a randomized polynomial-time algorithm for the unweighted k-Steiner-cycle problem; also, see Taslaman's thesis [26]. Further research on the same problem is presented by Wahlström [27], and by Fafianie and Kratsch [10]. The algorithm of [3] extends easily to a randomized FPTAS for the weighted k-Steiner-cycle problem, by using techniques from Ibarra & Kim [15] and Hochbaum & Shmoys [14], see Proposition 6. All the results in our paper are based on the key result of Björklund et al. [3]. Below, in the section on related work, we discuss several papers that pertain to the k-Steiner-cycle problem, but we stress that the methods and techniques of these papers have no direct implications for the k-Steiner-cycle problem.

Network design encompasses a wide class of problems that find applications in sectors like transportation, facility location, information security, resource connectivity, etc. Due to its wide scope and usefulness, the area of network design has been studied for decades and it has spawned major algorithmic innovations. Most of the problems in network design are NP-Hard, and researchers in the area have focused on designing good approximation algorithms. The nodes of a network are designated as terminals (i.e., "essential" nodes) or non-terminals (i.e., Steiner nodes or "optional" nodes). A well-known goal of network design is to construct cheap networks that can survive the failure of one element (i.e., one edge or one node); "surviving" means that all the (remaining) terminals stay connected even after the deletion of one element, that is, there exists a path between every pair of (remaining) terminals after deleting one element.

In the *Steiner-2ECS* problem, the input is an undirected graph $G = (V, E)$, a set of terminals $T \subseteq V$, and non-negative costs c on the edges. The goal is to find a minimum-cost 2-edge-connected subgraph containing all the terminals. The *Steiner-2NCS* problem is defined similarly, where the goal is to find a minimum-cost 2-node-connected subgraph containing all the terminals. Both these problems are NP-Hard. The best known approximation algorithms achieve an approximation ratio of two, see [29].

Another paradigm to address NP-Hard problems is to develop parameterized algorithms. In this setting, a parameter of the input is chosen (e.g., the number of terminals, or the size of an optimal solution) and the goal is to develop algorithms whose running time depends on the input size and the parameter.

Feldmann, Mukherjee and van Leeuwen [11] presented parametrized algorithms for the Steiner-2ECS and Steiner-2NCS problems (among others) where the parameter is the optimal solution size, which is denoted by ℓ. They showed that if ℓ is bounded by a constant, then these problems can be solved in polynomial time. In particular, they present a *fixed parameter tractable* (FPT) algorithm that runs in time $n^{O(1)} f(\ell)$ and computes an optimal solution, where $f(\cdot)$ denotes some computable function.

Feldmann et al. [11] recently high-lighted the following open question in the area of network design: Is there a polynomial-time algorithm for the Steiner-2NCS problem, where the number of terminals is a constant? We use the term k-Steiner-2NCS problem (respectively, k-Steiner-2ECS problem) to refer to the special case of the Steiner-2NCS problem (respectively, Steiner-2ECS problem) with k terminals. Usually, we assume that the number of terminals is a constant, i.e., $k = O(1)$. We present a randomized polynomial-time algorithm for the unweighted k-Steiner-2NCS problem. We also consider the weighted k-Steiner-2NCS problem, and we provide a randomized fully polynomial time approximation scheme (FPTAS) for this problem.

We obtain similar results for the following generalization of the k-Steiner-2ECS problem: In the k-Flexible Steiner Tree (k-FST) problem, the input consists of an undirected graph $G = (V, E)$, a partition of the edge-set E into a set $\mathcal{S}$ of *safe edges* and a set $\mathcal{U}$ of *unsafe edges*, a set of terminals $T \subseteq V$ with $|T| = k$, and non-negative costs c on the edges. The goal is to find a minimum-cost connected subgraph $H = (U, F)$ such that $T \subseteq U$ and for any unsafe edge $e \in F$, the graph $H - e$ is connected. This problem is equivalent to the capacitated k-Steiner-2ECS problem, where the input is the same as that of the k-FST problem, except that there is a a positive integral capacity u on the edges (instead of the partition $E = \mathcal{S} \bigcup \mathcal{U}$); the goal is to find a minimum-cost connected subgraph $H = (U, F)$ such that $T \subseteq U$ and for any unit-capacity edge $e \in F$, the graph $H - e$ is connected. (The two problems are equivalent because the unsafe edges of the k-FST problem correspond to the unit-capacity edges of the latter problem, and the safe edges of the k-FST problem correspond to edges of capacity at least two of the latter problem.) We present a randomized polynomial-time algorithm for the unweighted k-FST problem; this easily extends to a randomized FPTAS for the weighted k-FST problem.

1.1 Our Results and Techniques

We prove that the k-Steiner-2NCS problem can be solved in randomized slicewise polynomial time, and hence it is in the complexity class randomized **XP**.

▶ **Theorem 1.** *For any $\eta > 0$, $\epsilon > 0$ and constant k,*

(i) *there exists a randomized algorithm for the unweighted* k-Steiner-2NCS *problem that outputs an optimal solution with probability $1 - \eta$ in time $O(\binom{n}{2k} \cdot B(3k) \cdot \binom{3k}{2}^k \cdot 2^{3k} \cdot n^{O(1)} \cdot \log \frac{k}{\eta}) = O\left(n^{O(k)} \cdot \log \frac{1}{\eta}\right)$, where $B(i)$ denotes the ith ordered Bell number.*

(ii) *there exists a randomized algorithm for the weighted* k-Steiner-2NCS *problem that runs in time $O(n^{O(k)} \cdot (1/\epsilon)^{O(k)} \cdot \log(1/\eta))$ such that, with probability at least $1 - \eta$, the solution returned by the algorithm costs at most $(1 + \epsilon)$ times the cost of an optimal solution.*

Our methods build on results by Björklund, Husfeldt, and Taslaman (SODA 2012) [3] that give a randomized polynomial-time algorithm for the unweighted k-Steiner-cycle problem. Given an instance of the k-Steiner-2NCS problem, we guess an ear decomposition of an optimal solution by enumeration, and repeatedly use the algorithm in [3] to construct an optimal subgraph. It can be seen that naively attaching new ears does not lead to an optimal solution. Hence, we also keep track of the high degree nodes in the optimal subgraph. Subsequently, we obtain our randomized FPTAS for the weighted k-Steiner-2NCS problem, by using the scaling techniques from Ibarra & Kim [15] and Hochbaum & Shmoys [14]. We present similar results for the k-FST problem.

▶ **Theorem 2.** *For any $\eta > 0$, $\epsilon > 0$ and constant k,*

(i) *there exists a randomized algorithm for the unweighted* k-FST *problem that outputs an optimal solution with probability $1 - \eta$ in time $O\left(n^{O(k)} \cdot \log \frac{1}{\eta}\right)$.*

(ii) *there exists a randomized algorithm for the weighted* k-FST *problem that runs in time $O(n^{O(k)} \cdot (1/\epsilon)^{O(k)} \cdot \log(1/\eta))$ such that, with probability at least $1 - \eta$, the solution returned by the algorithm costs at most $(1 + \epsilon)$ times the cost of an optimal solution.*

Our methods here rely on the block decomposition of an optimal solution, in conjunction with our results on the k-Steiner-2NCS problem. We use Theorem 1 to find the individual blocks optimally, and then paste these blocks together using the results by Adjiashvili, Hommelsheim, Mühlenthaler, and Schaudt [1] who give a polynomial-time algorithm for finding an optimal solution to the 2-FST problem; see Proposition 1 and Theorem 5 of [1]. As a corollary, we obtain the same results for the capacitated k-Steiner-2ECS problem.

▶ **Corollary 3.**

(i) *The capacitated* k-Steiner-2ECS *problem can be solved in slicewise polynomial time with high probability, hence, it is in randomized* $\mathbf{XP}$.

(ii) *There exists a randomized FPTAS for the weighted capacitated* k-Steiner-2ECS *problem.*

1.2 Related Work

1.2.1 k-Steiner-cycle problem

One of the key applications of the Graph Minors theory of Roberston and Seymour [23] is a polynomial-time algorithm for the decision/search version of the k disjoints paths problem for constant k. In this problem, we are given an undirected graph $G = (V, E)$ and k source-sink pairs s_i, t_i, and the goal is to decide (or, find) if there exist k node-disjoint paths P_i where the end-nodes of P_i are s_i and t_i. The Graph Minors theory (as of now) cannot address the optimization version of this problem.

There are many papers on the k disjoint paths problem, and a few on problems related to the k-Steiner-cycle problem. Kawarabayashi [19] presented improved algorithms for the search version of the latter problem, by improving on the methods from Graph Minors theory. There are a few other relevant results from the last few decades; for example, Fleischner and Woeginger present results for the unweighted 3-Steiner-cycle problem, see [13].

Recently, Lochet [21] and Bentert et al. [2] presented interesting algorithms for the so-called k disjoint shortest paths problem, i.e., each of the paths P_i (of the disjoint paths problem) is required to be a *shortest* path between s_i and t_i; this problem is *not* directly related to the optimization problems of interest to us.

1.2.2 k-Steiner-2NCS problem and k-Steiner-2ECS problem

Network design problems involving finding a cheapest subgraph subject to connectivity requirements have been studied for decades. One of the simplest such problems is the *minimum spanning tree* problem which is known to have polynomial-time algorithms. However, increasing the connectivity requirements makes these problems intractable. The 2-edge-connected spanning subgraph (2-ECSS) problem is Max-SNP-Hard, see [6]. In the weighted setting, the best known approximation algorithm achieves an approximation ratio of two, see Khuller and Vishkin [20]. The best known approximation ratio for the minimum-cost 2-node-connected spanning subgraph (2-NCSS) problem is two, see the survey by Nutov [22].

The analogous problems with Steiner nodes are usually harder. For instance, the minimum-cost Steiner tree problem is already NP-Hard [18]. The best known approximation ratio for this problem is ≈ 1.39, due to Byrka et al. [5]. The best known approximation ratios for the (weighted) Steiner-2ECS/Steiner-2NCS problem is two, see [29, 16, 12].

In the context of parameterized algorithms, Dreyfus and Wagner [9] showed that the Steiner tree problem can be solved in FPT time where the parameter is the number of terminals. Feldmann et al. [11] showed that the Steiner-2ECS and Steiner-2NCS problems can be solved in FPT time where the parameter is the size of an optimal solution.

Sami [24], in his master's thesis, has some results related to our paper. He notes that there is a reduction from the k-Steiner-2ECS problem to the k-Steiner-2NCS problem. $\big($We can "inflate" each node v of the graph G of the k-Steiner-2ECS problem to a complete graph, i.e., clique, C_v on $\deg_G(v)$ nodes with edges of cost zero, and replace the edges incident to v in G by edges incident to distinct nodes of C_v while preserving the edge costs.) Moreover, he shows that the FPT (in the solution size parameter ℓ) algorithm of [11] for the k-Steiner-2ECS problem can be combined with a result of Jordan [17] to give an FPT (in parameter $k = |T|$) algorithm for the k-Steiner-2ECM problem where the solution subgraph may pick multiple copies of any edge (and incurs cost c_e for each copy of e).

Borradaile and Klein [4] presented a PTAS for the planar case of the Steiner-2ECM problem (i.e., the multi-edge variant of the Steiner-2ECS problem).

2 Preliminaries

This section has definitions and preliminary results. Our notation and terms are consistent with [8], and readers are referred to that text for further information.

Let $G = (V, E)$ be a loopless multi-graph with non-negative costs $c \in \mathbb{R}^E_{\geq 0}$ on the edges. We take G to be the input graph, and we use n to denote $|V(G)|$. For a set of edges $F \subseteq E(G)$, $c(F) := \sum_{e \in F} c(e)$, and for a subgraph G' of G, $c(G') := \sum_{e \in E(G')} c(e)$.

For a positive integer k, we use $[k]$ to denote the set $\{1, \ldots, k\}$.

For a graph H and a set of nodes $S \subseteq V(H)$, $\Gamma_H(S) := \{w \in V(H) \setminus S \ : \ v \in S, vw \in E(H)\}$, thus, $\Gamma_H(S)$ denotes the set of neighbours of S.

For a graph H and a set of nodes $S \subseteq V(H)$, $\delta_H(S)$ denotes the set of edges that have one end node in S and one end node in $V(H) \setminus S$; moreover, $H[S]$ denotes the subgraph of H induced by S, and $H - S$ denotes the subgraph of H induced by $V(H) \setminus S$. For a graph H and a set of edges $F \subseteq E(H)$, $H - F$ denotes the graph $(V(H),\ E(H) \setminus F)$. We may use relaxed notation for singleton sets, e.g., we may use $\delta_H(v)$ instead of $\delta_H(\{v\})$, and we may use $H - v$ instead of $H - \{v\}$, etc.

We may not distinguish between a subgraph and its node set; for example, given a graph H and a set S of its nodes, we use $E(S)$ to denote the edge set of the subgraph of H induced by S.

2.1 2EC, 2NC and related notions

A multi-graph H is called k-edge connected if $|V(H)| \geq 2$ and for every $F \subseteq E(H)$ of size $< k$, $H - F$ is connected. Thus, H is 2-edge connected if it has ≥ 2 nodes and the deletion of any one edge results in a connected graph. A multi-graph H is called k-node connected if $|V(H)| > k$ and for every $S \subseteq V(H)$ of size $< k$, $H - S$ is connected. We use the abbreviations *2EC* for “2-edge connected,” and *2NC* for “2-node connected.”

For any instance H, we use $\textsc{opt}(H)$ to denote the minimum cost of a feasible subgraph (i.e., a subgraph that satisfies the requirements of the problem). When there is no danger of ambiguity, we use $\textsc{opt}$ rather than $\textsc{opt}(H)$.

By a *bridge* we mean an edge of a connected (sub)graph whose removal results in two connected components, and by a *cut-node* we mean a node of a connected (sub)graph whose deletion results in two or more connected components. A maximal connected subgraph without a cut-node is called a *block*. Thus, every block of a given graph G is either a maximal 2NC subgraph, or a bridge (and its incident nodes), or an isolated node. For any node v of G, let $\Gamma^{blocks}_G(v)$ denote the set of 2NC blocks of G that contain v.

2.2 Ear decompositions

An *ear decomposition* of a graph is a partition of the edge set into paths or cycles, $P_0, P_1, \ldots, P_\ell$, such that P_0 is the trivial path with one node, and each P_i $(1 \leq i \leq \ell)$ is either (1) a path that has both end nodes in $V_{i-1} = V(P_0) \cup V(P_1) \cup \ldots \cup V(P_{i-1})$ but has no internal nodes in V_{i-1}, or (2) a cycle that has exactly one node in V_{i-1}. For an ear P_i, let $\mathrm{int}(P_i)$ denote the set of nodes $V(P_i) \setminus V_{i-1}$. Each of $P_1, \ldots, P_\ell$ is called an *ear*; note that P_0 is not regarded as an ear. We call $P_i, i \in \{1, \ldots, \ell\}$, an *open ear* if it is a path, and we call it a *closed ear* if it is a cycle. An *open* ear decomposition $P_0, P_1, \ldots, P_\ell$ is one such that all the ears $P_2, \ldots, P_\ell$ are open. (The ear P_1 is always closed.)

▶ **Proposition 4** (Whitney [28]).

(i) *A graph is 2EC $\iff$ it has an ear decomposition.*

(ii) *A graph is 2NC $\iff$ it has an open ear decomposition.*

2.3 Algorithms for basic computations

There are well-known polynomial-time algorithms for implementing all of the basic computations in this paper, see [25]. We state this explicitly in all relevant results, but we do not elaborate on this elsewhere.

3 FPTAS for k-Steiner-cycle

Björklund, Husfeldt, and Taslaman [3] presented a randomized algorithm for finding a min-cost simple cycle that contains a given set of terminals T of an unweighted, undirected graph $G = (V, E)$ with a running time of $2^k n^{O(1)}$, where $k = |T|$ and $n = |V|$. In other words, they present a randomized **FPT**-algorithm for the unweighted k-Steiner-cycle problem.

▶ **Theorem 5.** *Consider a graph $G = (V, E)$ and a set of terminals $T \subseteq V$ of size k. Let $\eta > 0$ be a parameter. A minimum-size* k-Steiner-cycle *can be found, if one exists, by a randomized algorithm in time $2^k n^{O(1)} \log \frac{1}{\eta}$ with probability at least $1 - \eta$.*

We present a simple (randomized) FPTAS for the weighted k-Steiner-cycle problem, based on the algorithm of [3].

▶ **Proposition 6.** *Consider a graph $G = (V, E)$ with nonnegative costs $c \in \mathbb{R}^E_{\geq 0}$ on the edges, and a set of k terminals $T \subseteq V$. Let $\varepsilon, \eta > 0$ be some parameters. There is a randomized algorithm that finds a $(1 + \varepsilon)$-approximate* k-Steiner-cycle*, if one exists, with probability at least $1 - \eta$. The running time of the algorithm is $O\Big(2^k \cdot n^{O(1)} \cdot \big(\frac{1}{\varepsilon}\big)^{O(1)} \cdot \log \frac{1}{\eta}\Big)$.*

Proof. Let $E = \{e_1, e_2, \ldots, e_m\}$ where $c_{e_1} \leq c_{e_2} \leq \cdots \leq c_{e_m}$. Let $\eta' := \eta/2$. Let $j \in [m]$ denote the smallest index such that the graph $(V, \{e_1, \ldots, e_j\})$ contains a k-Steiner-cycle. Note that if G does not have a k-Steiner-cycle, then the weighted-version of the problem is trivially infeasible. Using at most m applications of Theorem 5 with the η-parameter set to η', we can find the index j with probability at least $1 - \eta/2$. Suppose that we have the correct index j. Let $\beta := c(e_j)$. Let Q^* denote an optimal k-Steiner-cycle in G, and $\textsc{opt} := c(Q^*)$ denote the optimal cost. By the definition of j, $\beta \leq \textsc{opt} \leq n\beta$. In particular, every edge in Q^* has cost at most $n\beta$. We now describe our randomized algorithm for obtaining a k-Steiner-cycle with cost at most $(1 + \varepsilon)\textsc{opt}$. First, we discard all edges e of G with cost $c_e > n\beta$. Let $\mu := \varepsilon\beta/n$; this is our "scaling parameter". For each edge e, define $\tilde{c}_e := \mu \cdot \max(1, \lceil c_e/\mu \rceil)$. Note that $\tilde{c}_e = \mu$ if $c_e = 0$. $\big($Observe that this rounding introduces errors, but the total error incurred on any cycle is $\le n\,\mu \le \epsilon\beta \le \epsilon\textsc{opt}$.) Consider the graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ obtained from G by replacing each edge e by a path of $\tilde{c}_e/\mu$ edges (of unit cost). Note that $|\widetilde{V}| \le |V| + |E| \cdot (n\beta)/\mu = O(mn^2/\varepsilon)$. Using a single application of Theorem 5, we can obtain a minimum-size k-Steiner-cycle $\widetilde{Q} \subseteq \widetilde{E}$ with probability at least $1 - \eta/2$ in $O\Big(2^k \cdot (\frac{n^2 m}{\varepsilon})^{O(1)} \cdot \log\frac{1}{\eta}\Big)$ time. Let Q denote the k-Steiner-cycle in G corresponding to $\widetilde{Q}$. By our choice of $\tilde{c}$, we have $c(Q) \le \tilde{c}(Q) \le \mu \cdot |\widetilde{Q}|$. Since the optimal k-Steiner-cycle Q^* consists of at most n edges each with cost at most $n\beta$, the (unweighted) k-Steiner-cycle $\widetilde{Q}^*$ in $\widetilde{G}$ corresponding to Q^* satisfies $\mu|\widetilde{Q}^*| \le \tilde{c}(Q^*) \le c(Q^*) + n\mu \le \textsc{opt}(1+\varepsilon)$. By the above discussion, we can obtain a k-Steiner-cycle Q satisfying $c(Q) \le \mu|\widetilde{Q}| \le \mu|\widetilde{Q}^*| \le (1+\varepsilon)\textsc{opt}$ with probability at least $1-\eta$. Clearly, the overall running time is $O\Big(2^k \cdot n^{O(1)} \cdot (\frac{1}{\varepsilon})^{O(1)} \cdot \log\frac{1}{\eta}\Big)$. ◀

4 Algorithms for k-Steiner-2NCS

In this section, we consider the k-Steiner-2NCS problem. First, we present a randomized polynomial-time algorithm for finding an optimal subgraph for the special case of unweighted k-Steiner-2NCS; then, using the method from Section 3 we extend our algorithm to a (randomized) FPTAS for weighted k-Steiner-2NCS.

We denote an instance of the k-Steiner-2NCS problem by $(G = (V, E),\ c \in \mathbb{R}^E_{\ge 0},\ T \subseteq V)$; G is the input graph with non-negative edge costs c, and T is the set of terminals, $|T| \ge 3$ (we skip the easy case of $|T| = 2$). We assume (w.l.o.g.) that G is a feasible subgraph, that is, all terminals are contained in one block of G.

For any graph H, let $D_3(H)$ denote the set of nodes that have degree ≥ 3 in H.

▶ **Lemma 7.** *Let $H = (V', E')$ be an (edge) minimal 2NC subgraph that contains T. Then H has an open ear decomposition $P_0, P_1, \ldots, P_\ell$ such that*

(i) *each of the ears P_i ($i \in [\ell]$) contains a terminal as an internal node (i.e., $\mathrm{int}(P_i) \cap T \neq \emptyset$), and P_0 contains a terminal,*

(ii) *$|D_3(H)| \le 2(|T| - 2)$.*

Proof.

(i) Pick any terminal to be P_0. Suppose we have constructed open ears $P_1, \ldots, P_{i-1}$ and that each $\mathrm{int}(P_j)(j \in [i-1])$ contains a terminal. Let $F = \cup_{j=1}^{i-1} E(P_j)$. Let t be a terminal in $T \setminus V(F)$ (we have the required ear decomposition, if $T \subseteq V(F)$). Suppose $i = 1$; then, G has two openly disjoint paths between t and P_0, and we take P_1 to be the edge-set of these two paths. Suppose $i \ge 2$; then, G has a two-fan P between t and $V(F)$ (i.e., P is the union of two paths between t and $V(F)$ that have only the node t in common); we take P_i to be P.

(ii) Clearly, $\ell \le |T| - 1$ for the ear decomposition of part (i), and each of the ears $P_2, \ldots, P_\ell$ contributes at most 2 (new) nodes to $D_3(H)$. ◀

The next lemma states an extension of Proposition 4.

▶ **Lemma 8.** *Let $G = (V, E)$ be a graph, and let $H = (V', E')$ be a 2NC subgraph of G. Let P be a path of G that has both end nodes in V'. Then, $H \cup P = (V' \cup V(P), E' \cup E(P))$ is a 2NC subgraph of G.*

Each set $S \subseteq V$ of size $\le 2|T| - 4$ is a candidate for $D_3(H)$ for a 2NC subgraph H that contains T, and we call $T \cup S$ the set of *marker* nodes.

Our algorithm has several nested loops. The outer-most loop picks a set $S \subseteq V$ of size $\leq 2|T| - 4$, and then applies the following main loop. Each iteration of the main loop attempts to construct a 2NC subgraph that contains the set of marker nodes $T \cup S$, by iterating over all ordered partitions $(\widetilde{T}_1, \widetilde{T}_2, \ldots, \widetilde{T}_r)$ of $T \cup S$ such that $|\widetilde{T}_1| \geq 2$ and the number of sets in the partition, r, is a positive integer, $r \leq k = |T|$.

Consider one of these ordered partitions $(\widetilde{T}_1, \widetilde{T}_2, \ldots, \widetilde{T}_r)$. We attempt to find a min-cost Steiner-cycle C_1 that contains $\widetilde{T}_1$ using the algorithm of [3]; if G has no Steiner-cycle that contains $\widetilde{T}_1$, then this iteration has failed, otherwise, we take C_1 to be the first (closed) ear of an open ear decomposition of our candidate 2NC subgraph that contains $T \cup S$. Then, for $i = 1, \ldots, r-1$, we pick a pair of nodes $s_i, t_i \in \widetilde{T}_1 \cup \cdots \cup \widetilde{T}_i$, and attempt to find a min-cost Steiner-path P_{i+1} between s_i and t_i that contains $\widetilde{T}_{i+1}$; if G has no such Steiner-path, then this iteration has failed, otherwise, we augment the current subgraph $H := C_1 \cup P_2 \cup \cdots \cup P_i$ by P_{i+1}.

The algorithm maintains an edge-set $\hat{F}$; initially, $\hat{F} = E$, and, at termination, $\hat{F}$ is the edge-set of a min-cost 2NC subgraph that contains T.

Pseudo-code for the algorithm is presented below.

We use $\mathcal{A}_{\text{BHT-cycle}}(G,\ \widetilde{T}_1,\ \eta)$ to denote a call to the Steiner-cycle algorithm of [3] where the inputs are the graph G, the terminal set $\widetilde{T}_1 \subseteq V(G)$, and the desired probability of failure η. With probability at least $1-\eta$, this call either returns the edge-set of a minimum-size cycle of G that contains all nodes of $\widetilde{T}_1$ or reports an error if G has no such cycle.

We use $\mathcal{A}_{\text{BHT-path}}(G,\ \widetilde{T}_1,\ s,\ t,\ \eta)$ to denote a call to the following subroutine that attempts to find an s,t-path of G that contains all nodes of $\widetilde{T}_1$. First, construct an auxiliary graph G' from G by adding a node u' and two edges $u's, u't$. Then call $\mathcal{A}_{\text{BHT-cycle}}(G',\ \widetilde{T}_1 \cup \{u', s, t\},\ \eta)$; report an error if the call returns an error, and, otherwise, return the path obtained by deleting the node u' (and its two incident edges) from the cycle returned by the call.

Algorithm 1 $\mathcal{A}_{2NC}(G, T, \eta)$ for the unweighted k-Steiner-2NCS problem.

```
$\eta' \leftarrow \eta/k$
$\hat{F} \leftarrow E$
for $S \subseteq V$ such that $|S| \leq 2k$ do
    for $r = 1, \ldots, k$ do
        for Ordered partitions $(\widetilde{T}_1, \ldots, \widetilde{T}_r)$ of $T \cup S$ such that $|\widetilde{T}_1| \geq 2$ do
            for $i = 1, \ldots, r-1$ and node pairs $(s_i, t_i) \in \cup_{j=1}^{i} \widetilde{T}_i$, where $s_i \neq t_i$ do
                $H \leftarrow \mathcal{A}_{\text{BHT-cycle}}(G, \widetilde{T}_1,\ \eta') \quad \bigcup_{i=1}^{r-1} \mathcal{A}_{\text{BHT-path}}(G, \widetilde{T}_{i+1}, s_i, t_i,\ \eta')$
                continue the loop if any call to any subroutine reports an error
                if $|E(H)| < |\hat{F}|$ then
                    $\hat{F} \leftarrow E(H)$
                end if
            end for
        end for
    end for
end for
return $\hat{F}$
```

▶ **Lemma 9.** *Let $H^* = (V^*, E^*)$ be an optimal subgraph for k-*Steiner-2NCS*. Assume that each of the calls to the subroutines (namely, $\mathcal{A}_{\textit{BHT-cycle}}, \mathcal{A}_{\textit{BHT-path}}$) returns a valid subgraph whenever one exists. Let $H = (U, \hat{F})$ denote the output of the above algorithm. Then H is a 2NC subgraph, $U \supseteq T$, and $|\hat{F}| \leq |E^*| = \text{OPT}$.*

Proof. By Lemma 7, H^* has an open ear decomposition $P_1, P_2, \dots, P_{r^*}$ such that each of the ears P_i contains at least one terminal as an internal node; hence, $r^* \leq k = |T|$. Let $S^* = D_3(H^*)$ be the set of nodes of degree ≥ 3 of H^*; clearly, $|S^*| \leq 2r^* \leq 2k$.

For $i = 1, \dots, r^*$, let $T_i^* = P_i \cap (T \cup S^*)$. For $i = 1, \dots, r^* - 1$, let (s_i^*, t_i^*) denote the end nodes of the ear P_{i+1}; clearly, $(s_i^*, t_i^*) \in \cup_{j=1}^{i}(T_i^*)$.

Now consider the loop in the algorithm where $S = S^*$, $r = r^*$, $\widetilde{T}_i = T_i^*$ for $i = 1, \dots, r^*$ and $(s_i, t_i) = (s_i^*, t_i^*)$ for $i = 1, \dots, r^* - 1$. Observe that the calls to the subroutines $\mathcal{A}_{\text{BHT-cycle}}$ and $\mathcal{A}_{\text{BHT-path}}$ return minimum-size subgraphs, hence, $|\mathcal{A}_{\text{BHT-cycle}}(G, \widetilde{T}_1)| \leq |P_1|$ and $|\mathcal{A}_{\text{BHT-path}}(G, \widetilde{T}_{i+1}, s_i, t_i)| \leq |P_{i+1}|$ for $i = 1, \dots, r^* - 1$. Since $|E^*| = \sum_{i=1}^{r^*} |P_i|$, we conclude that the 2NC subgraph H found by this iteration satisfies $|E(H)| \leq |E^*|$. Thus, the algorithm outputs an optimal 2NC subgraph that contains T. ◀

4.1 Proof of Theorem 1

Proof. As seen in the proof of Lemma 9, if the subroutines $\mathcal{A}_{\text{BHT-cycle}}$ and $\mathcal{A}_{\text{BHT-path}}$ run correctly when $S = S^*$, $r = r^*$, $\widetilde{T}_i = T_i^*$ for $i = 1, \dots, r^*$ and $(s_i, t_i) = (s_i^*, t_i^*)$ for $i = 1, \dots, r^* - 1$ corresponding to an ear decomposition of an optimal solution H^*, then the above algorithm outputs an optimal solution. During this loop, there are at most $r^* \leq k = |T|$ calls to the subroutines $\mathcal{A}_{\text{BHT-cycle}}$ and $\mathcal{A}_{\text{BHT-path}}$. Hence, with probability at least $(1 - \eta')^k \geq 1 - \eta$, Algorithm $\mathcal{A}_{2NC}$ outputs an optimal solution.

The running time is analyzed as follows: the term $2k \cdot \binom{n}{2k} = O(\binom{n}{2k})$ comes from choosing $S \subseteq V$, $|S| \leq 2k$ (in the outer-most loop), the term $B(3k)$ comes from choosing ordered partitions of $S \cup T$, the term $\binom{3k}{2}^k$ comes from choosing the node pairs (s_i, t_i) for the $r - 1 (\leq k)$ calls to $\mathcal{A}_{\text{BHT-path}}$, and the term $2^{3k} n^{O(1)} \log \frac{k}{\eta}$ comes from the running time of the algorithm of [3] for the Steiner-cycle problem, with error probability $\frac{\eta}{k}$.

Having solved the unweighted k-Steiner-2NCS problem, we can directly use the methods from Section 3 to obtain an FPTAS for the weighted k-Steiner-2NCS problem. ◀

5 FPTAS for k-FST and k-Steiner-2ECS

In this section we present a randomized polynomial-time algorithm for finding an optimal subgraph for the special case of unweighted k-FST; then, using the method from Section 3, we extend our algorithm to a (randomized) FPTAS for weighted k-FST. We assume that $k = |T| \geq 3$ is a positive integer. Note that the capacitated k-Steiner-2ECS problem can be reduced to the k-FST problem by defining the set of safe edges to be the set of edges with capacity at least 2 and defining the set of unsafe edges to be the set of edges with capacity exactly 1. Hence the results in this section can also be applied to the capacitated k-Steiner-2ECS problem.

Adjiashvili, Hommelsheim, Mühlenthaler, and Schaudt [1] give a polynomial-time algorithm for finding an optimal solution to the 2-FST problem; see Proposition 1 and Theorem 5 of [1]. We refer to their 2-FST algorithm as $\mathcal{A}_{\text{2-FST}}$. We refer to (inclusion-wise) minimal feasible solutions to a 2-FST problem on G as *1-protected paths*.

Informally speaking, our randomized polynomial-time algorithm for k-FST represents minimal feasible solutions as 2NC blocks connected together using 1-protected paths. To simplify our presentation, we first modify the k-FST instance $G = (V, \mathcal{S} \sqcup \mathcal{U}, T)$ as follows. For each terminal $v \in T$, we create a new node v' and a new safe edge vv'. Let T' denote the set of these new nodes and let E' denote the set of the new safe edges. Consider the modified instance $G' = (V \cup T', (\mathcal{S} \cup E') \sqcup \mathcal{U}, T')$. Observe that (U, F) is a feasible solution to the original instance if and only if $(U \cup T', F \cup E')$ is a feasible solution to the modified instance.

▶ **Definition 10** (Block-Tree). *A block-tree of a graph G is a tree $B(G)$ with the following properties:*

1. *The nodes of $B(G)$ are in one-to-one correspondence with the 2NC blocks of G.*
2. *If two 2NC blocks are connected by a bridge in G, then the two corresponding nodes in $B(G)$ are adjacent.*
3. *For each cut-node v of G, the subgraph of $B(G)$ induced by $\Gamma_G^{blocks}(v)$ is connected ($\Gamma_G^{blocks}(v)$ is the set of 2NC blocks of G that contain v). In other words, the unique path of $B(G)$ between any two nodes of $\Gamma_G^{blocks}(v)$ has all its internal nodes in $\Gamma_G^{blocks}(v)$.*

Informally speaking, a block-tree of a graph G represents how the 2NC blocks of G are connected together. Each edge of the block-tree either represents a bridge of G or connects a pair of 2NC blocks of G that share a common cut-node. Let H be a minimal feasible k-FST solution. Due to the modification above, we may assume that every leaf of $B(H)$ corresponds to a block of H that contains exactly one terminal. Then any path in $B(H)$ corresponds to a 1-protected path of H that connects either (i) two cut-nodes, or (ii) a cut-node and a terminal, or (iii) two terminals.

For our algorithmic application, nodes of $B(H)$ of degree two are redundant, and this motivates the notion of a "non-redundant" block-tree.

▶ **Definition 11** (Condensed Block-Tree). *A condensed block-tree of a graph G is a tree $\hat{B}(G)$ obtained from a block-tree $B(G)$ with the following properties:*

1. *The nodes of $\hat{B}(G)$ are nodes b of $B(G)$ such that $\deg_{B(G)}(b) \neq 2$.*
2. *Two nodes b_1 and b_2 are adjacent in $\hat{B}(G)$ if and only if every internal node in the path connecting b_1 and b_2 in $B(G)$ has degree two.*

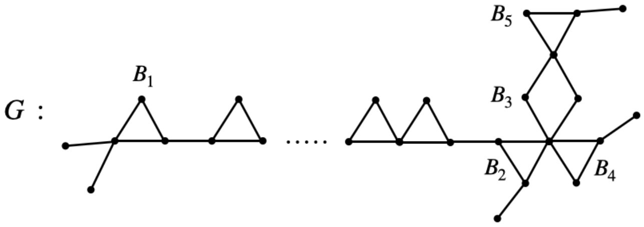


Figure 1 The original graph G.

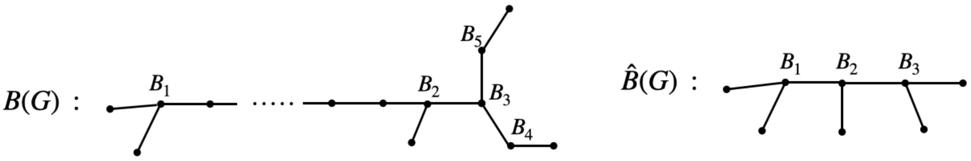


Figure 2 A block tree $B(G)$ and the corresponding condensed block tree $\hat{B}(G)$.

For any minimal feasible solution H of k-FST and a condensed block-tree $\hat{B}(H)$, we refer to the 2NC blocks of H that correspond to internal nodes of $\hat{B}(H)$ as *high-degree blocks.* The leaves of $\hat{B}(H)$ correspond to the terminals. Edges of $\hat{B}(H)$ correspond to 1-protected paths in H that connect either (i) two high-degree blocks, or (ii) a high-degree block and a terminal, or (iii) two terminals. The end-points of these 1-protected paths are either cut-nodes of H or terminals. Note that some of these 1-protected paths could be trivial paths corresponding to cut-nodes that are common to two high-degree blocks. We now state some useful lemmas that follow from the handshaking lemma applied to $\hat{B}(H)$.

▶ **Lemma 12.** *The number of internal nodes (i.e., non-leaf nodes) of $\hat{B}(H)$ is at most $k-2$ where k is the number of terminals.*

▶ **Lemma 13.** *The total number of cut-nodes (with repetitions) in high-degree blocks of H is at most $3k-6$ where k is the number of terminals.*

Now, we describe our algorithm for unweighted k-FST. We guess (via enumeration) the high-degree blocks of an optimal solution OPTSOLN corresponding to some condensed block-tree $\hat{B}(\text{OPTSOLN})$. The guess would include the number of high-degree blocks and the cut-nodes in each of these high-degree blocks. This is done by picking $3k-6$ nodes of V with replacement and then picking a partition $\widehat{\mathcal{P}}$ of these $3k-6$ nodes into at most $k-2$ sets. Let $r \leq k-2$ be the number of sets in the partition $\widehat{\mathcal{P}}$. Thus, $\widehat{\mathcal{P}} = (X_1, X_2, \ldots, X_r)$. For each set X_i, we use algorithm $\mathcal{A}_{2NC}$ to find B_i, a minimum size 2NC subgraph of G containing the specified cut-nodes in X_i, possibly, with some additional Steiner nodes. Finally, we construct a tree that connects these 2NC subgraphs and terminals via 1-protected paths using the following subroutine.

First, for every pair of nodes $(u, v) \in V' \times V'$, we use algorithm $\mathcal{A}_{\text{2-FST}}$ to find $G^{\min}_{uv}$, the minimum size 1-protected path connecting u and v in G'. We then construct a complete graph $K(X_1, \ldots, X_r)$ with $r+k$ nodes that has one node for each set X_i and one node corresponding to each terminal $\{t\}$. The cost of an edge between two nodes of K corresponding to node sets V_1 and V_2 is given by $\min\{|E(G^{\min}_{uv})| : u \in V_1, v \in V_2\}$. Note that if there is no 1-protected path connecting a node in V_1 to a node in V_2, then we fix the cost of the edge to be infinity. Thus an edge $\bar{e}$ of K corresponds to a subgraph $G^{\min}_{\bar{e}}$ in G' which is the minimum size 1-protected path whose end points are in V_1 and V_2 respectively. We then find a minimum spanning tree $\bar{T}$ in K. Note that if $\bar{T}$ has infinite cost, then we output an error. Else, we output the subgraph of G' defined by $G^{\min}(X_1, \ldots, X_r) := \cup_{\bar{e} \in \bar{T}} G^{\min}_{\bar{e}}$.

Algorithm 2 $\mathcal{A}_{\text{k-FST}}(G', T', \eta)$ for the unweighted k-FST problem.

```
$\eta' \leftarrow \eta/k$
$H \leftarrow G'$
for $S = (v_1, \ldots, v_{3k-6}) \in V^{3k-6}$ do
    for $r = 1, \ldots, k-2$ do
        for partitions $(X_1, \ldots, X_r)$ of $S$ do
            $\hat{H} \leftarrow G^{\min}(X_1, \ldots, X_r) \cup_{i=1}^{r} \mathcal{A}_{2NC}(G, X_i,\ \eta')$
            continue the loop if any call to any subroutine reports an error
            if $|E(\hat{H})| < |EH|$ then
                $H \leftarrow \hat{H}$
            end if
        end for
    end for
end for
return H
```

▶ **Lemma 14.** *Let $H^* = (V^*, E^*)$ be an optimal subgraph for k-FST. Assume that each of the calls to the subroutine $\mathcal{A}_{2NC}(G, X_i, \eta')$ returns a valid subgraph $2NC(G, X_i)$ whenever one exists. Let $H = (U, F)$ denote the output of the above algorithm. Then, H is a feasible k-FST solution and $|F| \le |E^*| = \text{OPT}$.*

Proof. We argue that the subgraph $\hat{H}$ in any iteration of the algorithm is a feasible k-FST solution. This holds because the algorithm finds 2NC subgraphs $2NC(G, X_i)$ and then connects them to one another and to the terminals using the 1-protected paths in $G^{\min}(X_1, \dots, X_r)$. Thus, $T \subseteq V(\hat{H})$ and any unsafe edge $e \in E(\hat{H})$ either lies in a 2NC subgraph of $\hat{H}$ or a 1-protected path in $\hat{H}$, hence, $\hat{H} - e$ is connected.

Now consider a condensed block-tree $\hat{B}(H^*)$. Let $B_1^*, \dots, B_{r^*}^*$ be the high-degree blocks of H^* and let X_i^* be the set of cut-nodes in B_i^*. By Lemma 13, the total number of cut-nodes in all the high-degree blocks B_i^* is at most $3k - 6$. We may assume that it is exactly $3k - 6$ by duplicating a cut-node $v \in X_1^*$ multiple times. Consider the iteration of the algorithm where $r = r^*$ and $X_i = X_i^*$ for $i = 1, \dots, r^*$. Then,

$$|E(2NC(G, X_i))| \le |E(B_i^*)| \qquad \forall i = 1, \dots, r.$$

Recall that the nodes of $\hat{B}(H^*)$ correspond to the high-degree blocks B_i^* (and hence to the node sets X_i^*) or to the terminals $\{t\}$. Also an edge $\bar{e}$ of $\hat{B}(H^*)$ between nodes corresponding to node sets V_1 and V_2 represents a 1-protected path $H_{\bar{e}}^*$ in H^* whose end-points lie in V_1 and V_2 respectively. Hence $\hat{B}(H^*)$ may be viewed as a subgraph of $K(X_1, \dots, X_r)$. Furthermore, since any two nodes in H^* have a 1-protected path between them, $\hat{B}(H^*)$ must be connected. Finally, by construction of K, $|E(H_{\bar{e}}^*)| \ge c(\bar{e})$ where $c(\bar{e})$ is the cost of the edge $\bar{e}$ in K. This implies that the cost of the minimum spanning tree in K is at most $\sum_{\bar{e} \in \hat{B}(H^*)} |E(H_{\bar{e}}^*)|$. Hence,

$$|E(G^{\min}(X_1, \dots, X_r))| \le \sum_{\bar{e} \in \hat{B}(H^*)} |E(H_{\bar{e}}^*)|.$$

Combining the two inequalities above we obtain

$$\begin{aligned}
|E(\hat{H})| &= |E(G^{\min}(X_1, \dots, X_r) \cup_{i=1}^{r} 2NC(G, X_i))| \\
&\le |E(G^{\min}(X_1, \dots, X_r))| + \sum_{i=1}^{r} |E(2NC(G, X_i))| \\
&\le \sum_{\bar{e} \in \hat{B}(H^*)} |E(H_{\bar{e}}^*)| + \sum_{i=1}^{r} |E(B_i^*)| \\
&= |E^*|
\end{aligned}$$

The last equation holds because E^* partitions into the edge-sets of the high-degree blocks B_i^* and the edge-sets of the 1-protected paths $H_{\bar{e}}^*$. This completes the proof of the lemma. ◀

5.1 Proof of Theorem 2

Lemma 14 proves that algorithm $\mathcal{A}_{\text{k-FST}}$ outputs an optimal solution to the k-FST problem with high probability. Let α denote the running time of the algorithm $\mathcal{A}_{\text{2-FST}}$ and let β denote the running time of the algorithm $\mathcal{A}_{\text{2NC}}$. Then, the running time of the algorithm $\mathcal{A}_{\text{k-FST}}$ is bounded by

$$O(n^2\alpha \cdot n^{3k-6} 2^k n^2 \beta \cdot k^2 \log k) = O\left(\alpha \cdot \beta \cdot n^{3k}\right).$$

Since $\mathcal{A}_{\text{2-FST}}$ has runtime $n^{O(1)}$ and $\mathcal{A}_{2\text{NC}}$ has runtime $n^{O(k)}$, we can conclude that the running time of the algorithm $\mathcal{A}_{\text{k-FST}}$ is $O\left(n^{O(k)} \cdot \log \frac{1}{\eta}\right)$.

Having solved the unweighted k-FST problem, we can directly use the methods from Section 3 to obtain an FPTAS for the weighted k-FST problem.

References

1 David Adjiashvili, Felix Hommelsheim, Moritz Mühlenthaler, and Oliver Schaudt. Fault-Tolerant Edge-Disjoint s-t Paths — Beyond Uniform Faults. In *Proceedings of the 18th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, volume 227, pages 5:1–5:19, 2022. doi:10.4230/LIPIcs.SWAT.2022.5.

2 Matthias Bentert, André Nichterlein, Malte Renken, and Philipp Zschoche. Using a Geometric Lens to Find k Disjoint Shortest Paths. In *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 198, pages 26:1–26:14, 2021. doi:10.4230/LIPIcs.ICALP.2021.26.

3 Andreas Björklund, Thore Husfeldt, and Nina Taslaman. Shortest Cycle Through Specified Elements. In *Proceedings of the 23rd Symposium on Discrete Algorithms (SODA)*, pages 1747–1753, 2012. doi:10.1137/1.9781611973099.139.

4 Glencora Borradaile and Philip Klein. The Two-Edge Connectivity Survivable-Network Design Problem in Planar Graphs. *ACM Transactions on Algorithms*, 12(3):30:1–30:29, 2016. doi:10.1145/2831235.

5 Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanita. An Improved LP-based Approximation for Steiner Tree. In *Proceedings of the 42nd Symposium on Theory of Computing*, pages 583–592, 2010. doi:10.1145/1806689.1806769.

6 Artur Czumaj and Andrzej Lingas. On Approximability of the Minimum-Cost k-Connected Spanning Subgraph Problem. In *Proceedings of the 10th Symposium on Discrete Algorithms (SODA)*, pages 281–290, 1999. URL: https://dl.acm.org/doi/pdf/10.5555/314500.314573.

7 Nathaniel Dean. Open Problems. In Neil Robertson and Paul D. Seymour, editors, *Graph Structure Theory, Proceedings of a AMS-IMS-SIAM Joint Summer Research Conference on Graph Minors*, volume 147 of *Contemporary Mathematics*, pages 677–688. American Mathematical Society, 1991.

8 Reinhard Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer, 2017. doi:10.1007/978-3-662-53622-3.

9 Stuart E Dreyfus and Robert A Wagner. The Steiner Problem in Graphs. *Networks*, 1(3):195–207, 1971. doi:10.1002/net.3230010302.

10 Stefan Fafianie and Stefan Kratsch. An Experimental Analysis of a Polynomial Compression for the Steiner Cycle Problem. In *Proceedings of the 14th International Symposium on Experimental Algorithms (SEA)*, volume 9125 of *Lecture Notes in Computer Science*, pages 367–378, 2015. doi:10.1007/978-3-319-20086-6_28.

11 Andreas Emil Feldmann, Anish Mukherjee, and Erik Jan van Leeuwen. The Parameterized Complexity of the Survivable Network Design Problem. In *Proceedings of the 5th Symposium on Simplicity in Algorithms (SOSA)*, pages 37–56, 2022. doi:10.1137/1.9781611977066.4.

12 Lisa Fleischer, Kamal Jain, and David P. Williamson. Iterative rounding 2-approximation algorithms for minimum-cost vertex connectivity problems. *Journal of Computer and System Sciences*, 72(5):838–867, 2006. doi:10.1016/j.jcss.2005.05.006.

13 Herbert Fleischner and Gerhard J. Woeginger. Detecting cycles through three fixed vertices in a graph. *Information Processing Letters*, 42(1):29–33, 1992. doi:10.1016/0020-0190(92)90128-I.

14 Dorit S. Hochbaum and David B. Shmoys. A Unified Approach to Approximation Algorithms for Bottleneck Problems. *Journal of the ACM*, 33(3):533–550, 1986. doi:10.1145/5925.5933.

15 Oscar H. Ibarra and Chul E. Kim. Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems. *Journal of the ACM*, 22(4):463–468, 1975. doi:10.1145/321906.321909.

16 Kamal Jain. A Factor 2 Approximation Algorithm for the Generalized Steiner Network Problem. *Combinatorica*, 21(1):39–60, 2001. doi:10.1007/s004930170004.

17 Tibor Jordán. On minimally k-edge-connected graphs and shortest k-edge-connected Steiner networks. *Discrete Applied Mathematics*, 131(2):421–432, 2003. doi:10.1016/S0166-218X(02)00465-1.

18 Richard M Karp. On the Computational Complexity of Combinatorial Problems. *Networks*, 5(1):45–68, 1975. doi:10.1002/net.1975.5.1.45.

19 Ken-ichi Kawarabayashi. An Improved Algorithm for Finding Cycles Through Elements. In *Proceedings of the 13th Integer Programming and Combinatorial Optimization (IPCO)*, volume 5035, pages 374–384. Springer, 2008. doi:10.1007/978-3-540-68891-4_26.

20 Samir Khuller and Uzi Vishkin. Biconnectivity Approximations and Graph Carvings. *Journal of the ACM*, 41(2):214–235, 1994. doi:10.1145/174652.174654.

21 William Lochet. A Polynomial Time Algorithm for the k-Disjoint Shortest Paths Problem. In *Proceedings of the 32nd Symposium on Discrete Algorithms (SODA)*, pages 169–178, 2021. doi:10.1137/1.9781611976465.12.

22 Zeev Nutov. Node-Connectivity Survivable Network Problems. In Teofilo F. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics, Second Edition, Volume 2: Contemporary and Emerging Applications*. Chapman and Hall/CRC, 2018.

23 Neil Robertson and Paul D Seymour. Graph Minors. XIII. The Disjoint Paths Problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.

24 Sasha Sami. *Parameterized Algorithms for 2-Edge Connected Steiner Subgraphs*. Univerzita Karlova, Matematicko-fyzikální fakulta, 2023. URL: https://dspace.cuni.cz/handle/20.500.11956/179482.

25 Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, 2003.

26 Nina Taslaman. *Exponential-Time Algorithms and Complexity of NP-Hard Graph Problems*. Number 83 in ITU-DS. IT-Universitetet i København, 2013. URL: https://pure.itu.dk/ws/portalfiles/portal/39516792/nina_thesis.pdf.

27 Magnus Wahlström. Abusing the Tutte Matrix: An Algebraic Instance Compression for the K-set-cycle Problem. In *Proceedings of the 30th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 20, pages 341–352, 2013. doi:10.4230/LIPIcs.STACS.2013.341.

28 Hassler Whitney. Non-Separable and Planar Graphs. *Transactions of the American Mathematical Society*, 34:339–362, 1932. doi:10.1090/S0002-9947-1932-1501641-2.

29 David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. URL: http://www.cambridge.org/knowledge/isbn/item5759340/.

Oblivious Algorithms for the Max-kAND Problem

Noah G. Singer

Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

Motivated by recent works on streaming algorithms for constraint satisfaction problems (CSPs), we define and analyze *oblivious algorithms* for the Max-kAND problem. This is a class of simple, combinatorial algorithms which round each variable with probability depending only on a quantity called the variable's *bias*. Our definition generalizes a class of algorithms defined by Feige and Jozeph (Algorithmica '15) for Max-DICUT, a special case of Max-2AND.

For each oblivious algorithm, we design a so-called *factor-revealing* linear program (LP) which captures its worst-case instance, generalizing one of Feige and Jozeph for Max-DICUT. Then, departing from their work, we perform a fully explicit analysis of these (infinitely many!) LPs. In particular, we show that for all k, oblivious algorithms for Max-kAND provably outperform a special subclass of algorithms we call "superoblivious" algorithms.

Our result has implications for streaming algorithms: Generalizing the result for Max-DICUT of Saxena, Singer, Sudan, and Velusamy (SODA'23), we prove that certain separation results hold between streaming models for *infinitely many* CSPs: for *every* k, $O(\log n)$-space sketching algorithms for Max-kAND known to be optimal in $o(\sqrt{n})$-space can be beaten in (a) $O(\log n)$-space under a random-ordering assumption, and (b) $O(n^{1-1/k}D^{1/k})$ space under a maximum-degree-D assumption. Even in the previously-known case of Max-DICUT, our analytic proof gives a fuller, computer-free picture of these separation results.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Discrete optimization

Keywords and phrases streaming algorithm, approximation algorithm, constraint satisfaction problem (CSP), factor-revealing linear program

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.15

Category APPROX

Related Version *Full Version with Proofs*: `https://arxiv.org/abs/2305.04438`

Supplementary Material *Software (Source Code)*: `https://github.com/singerng/oblivious-csps` archived at `swh:1:dir:0662828b8cd21f298e6a1163db40e7b7d2253825`

Funding This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE2140739. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

Acknowledgements I would like to thank Madhu Sudan and Santhoshini Velusamy for generous feedback and comments on the manuscript; Pravesh Kothari and Peter Manohar for helpful discussions; and anonymous reviewers at APPROX whose feedback helped improve the presentation in the paper.

1 Introduction

In this work, we study a restricted but natural class of randomized algorithms called *oblivious algorithms* for a family of *constraint satisfaction problems (CSPs)* called Max-kAND for $k \geq 2$. In the Max-kAND problem, an algorithm is presented with a list of m constraints on

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 15; pp. 15:1–15:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

n Boolean variables; each constraint is a disjunction on k literals (e.g., $x_1 \wedge \neg x_4 \wedge \neg x_5$ for the case of Max-3AND); and the goal is to satisfy the highest possible fraction of constraints. We begin by introducing these problems and algorithms informally and discussing the context and motivation for our work.

1.1 Background

Context on Max-kAND

Constraints of Max-kAND are maximally "precise" among all Boolean CSPs: each constraint specifies *exactly* what its k variables must be assigned to in order for that constraint to be satisfied.[1] Numerous works have developed algorithms for Max-kAND [43, 22, 23, 8] as well as NP-hardness-of-approximation results [44, 42, 34, 15, 35]; we now know that $\Theta(k/2^k)$-approximations are the best achievable in polynomial time assuming $\mathsf{P} \neq \mathsf{NP}$ [8, 35].

Further attention has been devoted to important special cases of Max-kAND. One particularly important example is the Max-DICUT problem, a special case of Max-2AND where each constraint is of the form "$x \wedge \neg y$". Max-DICUT can be viewed alternatively as a directed graph optimization problem, where the goal is to find a directed cut (S, T) maximizing the number of edges (s, t) such that $s \in S$ and $t \in T$. Approximation algorithms for Max-DICUT (and sometimes Max-2AND) were developed in [18, 16, 33, 30], and its hardness-of-approximation was studied in [24]. Max-3AND was studied in [47, 45].

The importance of Max-kAND, the extensive work on its polynomial-time approximability, and the "preciseness" of its constraints have inspired significant study on its approximability in restricted algorithmic settings. For instance, Trevisan [43] showed that the natural linear programming (LP) relaxation for Max-kAND beats the trivial (uniformly random rounding) algorithm's approximation ratio by a factor of 2, and that this LP's "nice" structure allows it to be solved (approximately) by distributed algorithms. Max-DICUT in particular has been studied extensively in various restrictive algorithmic frameworks and models, including "combinatorial" algorithms [21], spectral partitioning algorithms [46], local search algorithms [1, 2], parallel algorithms [6], near-linear time algorithms [40], and online algorithms [5].

Oblivious algorithms for Max-kAND

In this paper, we define a restricted class of algorithms for Max-kAND called *oblivious algorithms* (which were previously studied in the case of Max-DICUT by Feige and Jozeph [17]). Oblivious algorithms fall into the paradigm of *probabilistic rounding* algorithms. These algorithms somehow choose, for each variable v, a probability $p_v \in [0, 1]$, and then assign x_v to 1 w.p. p_v and 0 w.p. $1 - p_v$ (independently of all other variables); the goal is then to choose these probabilities efficiently while ensuring that the resulting assignment satisfies many of the constraints in expectation.

Informally, oblivious algorithms are probabilistic rounding algorithms which choose rounding probabilities for each variable v in a "very local" way. It is simplest to define these algorithms, as in [17], in the case of (the graph-theoretic view of) Max-DICUT. In this context, oblivious algorithms choose a rounding probability for each vertex depending only its *bias*. In a directed graph G, the bias of a vertex v is a real number in the interval $[-1, +1]$,

[1] In particular, as observed by Trevisan [43], an *arbitrary* Boolean predicate ϕ of arity k with r satisfying assignments can be converted to r "disjoint" applications of Max-kAND constraints; this transformation makes an instance of Max-CSP(ϕ) into an instance of Max-kAND and drops the value by a factor exactly r. In turn, this means that algorithms for Max-kAND can approximate the acceptance probability of k-bit probabalistically checkable proofs (PCP) verifiers.

and is defined as the difference between v's in- and out-degrees divided by its total degree. (Note that the in- and out-degrees of a vertex completely describe what the vertex can "see" about the graph around it by exploring paths of length 1.) The natural generalization of bias to a variable in Max-kAND is the difference between its number of positive and negative appearances in the instance, divided by the total number of appearances,[2] and oblivious algorithms again are those which round each variable depending only on its bias.

In this paper, we also define a class of algorithms called *superoblivious algorithms*. These are a subclass of oblivious algorithms which round each variable depending only on the *sign* of the bias (i.e., positive, negative, or zero) and not the magnitude; for instance, they ignore the distinction between variables of bias $+0.9$ (95% of appearances are positive) and bias $+0.1$ (55% of appearances are positive). A principal focus of this paper is showing that for all k, oblivious algorithms can achieve strictly better approximation ratios vs. superoblivious algorithms for Max-kAND, and we motivate this focus in the following section.

Max-kAND and streaming algorithms

Max-kAND and its special case Max-DICUT have recently received heavy attention in one particular family of models, namely, those of *streaming algorithms* [20, 14, 7, 37, 36].[3] In these models, the algorithm has highly bounded space (relative to n, the number of variables in the instance) and takes one or more passes over the list of constraints in the instance before outputting an estimate for the value of the instance. It turns out that certain separation results established between streaming models in [36, 37] are closely related to the distinction between oblivious and superoblivious algorithms described in the previous subsection, and to illuminate this connection, we pause to give a quick recap of these works on streaming algorithms for CSPs.

Chou, Golovnev, and Velusamy [14] designed a $O(\log n)$-space streaming algorithm which (arbitrarily-close-to-)4/9-approximates the value of instances of Max-DICUT, and showed that this algorithm is optimal in $o(\sqrt{n})$ space. Chou, Golovnev, Sudan, and Velusamy [12, 13] vastly generalized this "dichotomy" result to hold for all CSPs (even over non-Boolean alphabets): They showed that for every CSP predicate ϕ, there is a ratio α^*, below which there are $O(\log n)$-space approximation algorithms, and above which there are no $o(\sqrt{n})$-space algorithms.[4]

The recent works of Saxena, Singer, Sudan, and Velusamy [37, 36] demonstrated that Max-DICUT exhibits a streaming-complexity-theoretic phenomenon previously unbeknownst to any CSP: It admits approximation algorithms in certain streaming regimes which beat the optimal $o(\sqrt{n})$-space approximation ratio of 4/9 [14]. Specifically, better-than-4/9-approximations are achievable either by (i) using $\widetilde{O}(\sqrt{n})$ space or (ii) assuming that the input stream of constraints is randomly ordered. These works relied heavily on the investigation by Feige and Jozeph [17] of oblivious algorithms for Max-DICUT. Indeed, Feige and Jozeph [17] constructed a $0.483 > 4/9$-approximation oblivious algorithm for Max-DICUT, and the main thrust of [37, 36] is to use stronger streaming models to "implement" the [17] algorithm by sampling random edges and calculating the biases of their endpoints, thereby beating the 4/9-approximation of [14].

[2] Or, more generally in weighted instances, the total weight of the clauses in which it appears positively vs. those in which it appears negatively.

[3] Other works in the broader area of approximation algorithms for CSPs in streaming models include [3, 4, 29, 26, 27, 9, 10, 11, 19, 27, 28, 39]. See also Sudan's survey [41] and the author's undergraduate thesis [38].

[4] Technical note: The lower bounds in [12, 13] only hold in general for a special class of streaming algorithms called sketching algorithms; we ignore this point in the exposition.

The principal motivation for the current work is extending the results of [37, 36] to Max-kAND for all k: That is, showing that tweaking the streaming model can lead to improved approximations vs. the model studied in the dichotomy theorem of [12]. It is not difficult to show that oblivious algorithms can be "implemented" in these modified streaming settings, analogously to the Max-DICUT case. Thus, letting α_k^* denote the $O(\log n)$-vs.-$\Omega(\sqrt{n})$-space dichotomy threshold for Max-kAND, the goal becomes to show that oblivious algorithms beat the ratio α_k^*. A recent work of Boyland, Hwang, Prasad, Singer, and Velusamy [7] studied exactly this ratio, and showed that for each k, the optimal algorithm for Max-kAND essentially "implements" a superoblivious algorithm. (Note that *a priori* one cannot simulate a superoblivious algorithm in $O(\log n)$ space; the algorithm in [7] instead measures some quantitative properties of the input instance which turn out to relate to the behavior of a certain superoblivious algorithm.) In particular, the threshold ratio α_k^* equals the ratio achieved by *some* superoblivious algorithm. Thus, to prove the separation results, it suffices to show that oblivious algorithms strictly outperform superoblivious algorithms (for all k); as mentioned above, this is the main focus of the current work.

Linear programming as a tool for analyzing approximation algorithms

The key technical ingredient in the current work is the definition and analysis of a so-called *factor-revealing linear program (LP).* In such an LP, feasible solutions encode instances of the problem at hand (i.e., Max-kAND); when we fix an algorithm in the designated class (i.e., an oblivious algorithm), the objective function "reveals" the approximation ratio the algorithm achieves on any given instance. Similar programs were first studied in depth for facility location problems by Jain, Mahdian, Markakis, Saberi, and Vazirani [25], and have been examined in other contexts such as online bipartite matching [31]. Our LP is a generalization of the one developed by Feige and Jozeph [17] for Max-DICUT.

To show that there is a 0.483-approximation oblivious algorithm for Max-DICUT, Feige and Jozeph [17] used computer analysis of their LP; as mentioned above, the fact that $0.483 > 4/9$ – that is, oblivious algorithms outperform superoblivious algorithms for Max-DICUT – in turn powered the works of [37, 36] in establishing that there are improved approximations in stronger streaming models. However, this result of [17] was used as a black box. We believe that our approach, which contrasts between "superoblivious" and more general "oblivious" algorithms and attacks the corresponding LPs from an analytical perspective, gives a more natural and systematic explanation for why Max-DICUT admits these improved approximations – and implies that Max-kAND does as well, for all k.

1.2 Results

Next, we turn to statements of our results. In our notation, an oblivious algorithm for Max-kAND is denoted $\texttt{Obl}_k^{\mathbf{t},\mathbf{p}}$, where $\mathbf{t}$ and $\mathbf{p}$ are, respectively, a *bias partition* which splits the space of possible biases (i.e., the interval $[-1,+1]$) into discrete intervals, and a *rounding vector* $\mathbf{p}$ specifying a probability with which to round variables for each of these intervals. We denote by $\alpha(\texttt{Obl}_k^{\mathbf{t},\mathbf{p}})$ the approximation ratio achieved by this algorithm. (See Section 2 below for formal definitions of these objects.)

To state the theorems properly, we first define some relevant quantities which first arose in the context of small-space sketching algorithms for Max-kAND in the work of Boyland *et al.* [7]. We define

$$\gamma_k \overset{\text{def}}{=} \begin{cases} \frac{1}{k} & k \text{ odd} \\ \frac{1}{k+1} & k \text{ even,} \end{cases} \tag{1}$$

and

$$p_k^* \stackrel{\text{def}}{=} \frac{1}{2}(1+\gamma_k) \tag{2}$$

and

$$\alpha_k^* \stackrel{\text{def}}{=} 2 \cdot (p_k^*(1-p_k^*))^{\lfloor k/2 \rfloor}. \tag{3}$$

That is, for even k, $\alpha_k^* = 2^{-(k-1)}(1-1/(k+1))^{k/2}(1+1/(k+1))^{k/2}$, and for odd k, $\alpha_k^* = 2^{-(k-1)}(1-1/k)^{(k-1)/2}(1+1/k)^{(k-1)/2}$. In particular, at $k=2$, we have $p_k^* = 2/3$ and $\alpha_k^* = 4/9$.

Our first theorem states that the optimal superoblivious algorithm for Max-kAND achieves ratio α_k^*, and that this algorithm rounds with probability p_k^*:

▶ **Theorem 1** (Characterization for superoblivious algorithms). *For every $k \geq 2$, there is a unique superoblivious algorithm $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}$ achieving ratio $\alpha(\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}) = \alpha_k^*$, and all other superoblivious perform strictly worse. (In particular, for $\mathbf{t} = (0,1)$, every rounding vector $\mathbf{p} = (p)$ satisfies $\alpha(\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}) \leq \alpha_k^*$, with equality if and only if $p = p_k^*$.)*

Our main theorem then states that one can improve over these superoblivious algorithms using other oblivious algorithms, and indeed, it suffices to consider only slight "perturbations" of the optimal superoblivious algorithms:

▶ **Theorem 2** (Main theorem: Better oblivious algorithms). *For every $k \geq 2$, there exists a bias partition $\mathbf{t}$ and a rounding vector $\mathbf{p}$ such that the oblivious algorithm $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}$ achieves $\alpha(\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}) \gneq \alpha_k^*$. (In particular, there exists $\epsilon^* > 0$ such that for all $0 < \epsilon \leq \epsilon^*$, there exists $0 < \delta < 1$ such that $\mathbf{t} = (\delta, 1)$ and $\mathbf{p} = (p_k^* + \epsilon)$ satisfy $\alpha(\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}) \gneq \alpha_k^*$.)*

These theorems are both proven by analyzing the dual of a certain natural "factor-revealing" linear program. Arguably, this "dual" perspective systematizes the *ad hoc* analyses of small-space sketching algorithms for Max-DICUT in [14] and for Max-kAND in [7]. Indeed, the analysis in those paper examined certain systems of linear inequalities using "elementary" reasoning (i.e., taking nonnegative linear combinations), and it is exactly this type of reasoning which is captured by the technology of dual linear programs. Our perspective also sheds direct light on why oblivious algorithms outperform the 4/9 ratio for Max-2AND was key to the streaming separations established for Max-DICUT by Saxena *et al.* [37, 36].

Indeed, our results also have the following implications for streaming algorithms, generalizing connections established by Saxena, Singer, Sudan, and Velusamy [37] for the special case of Max-DICUT. We say an instance Ψ of Max-kAND is in *input form* if it is unweighted (i.e., the weight of every clause is 1), though multiple copies of the same clause are allowed. These instances will be the input to our algorithms though this is essentially without loss of generality as general instances can be "rounded" to such instances via standard arguments.

▶ **Theorem 3** (Random-order streaming algorithm). *For all $k \geq 2$, there exists $\alpha > \alpha_k^*$ such that for all $\epsilon > 0$, the following holds. There is an $O(\log n)$-space streaming algorithm which, for every instance Ψ of* Max-kAND *in input form with n variables and* $\mathrm{poly}(n)$ *clauses, given as input Ψ's clauses in a* randomly-ordered *stream, outputs an $(\alpha - \epsilon)$-approximation to the* Max-kAND *value of Ψ with probability* 99/100.

▶ **Theorem 4** (Bounded-degree streaming algorithm). *For all $k \geq 2$, there exists $\alpha > \alpha_k^*$ such that for all $\epsilon > 0$, the following holds: For all $D \geq 2$, there is an $O(D^{1/k} n^{1-1/k} \log n / \epsilon^{2/k})$-space streaming algorithm which, for every instance Ψ of input form with n variables and* maximum degree *$\leq D$ (i.e., every variable is contained in $\leq D$ clauses), given as input Ψ's clauses in an* adversarially-ordered *stream, outputs an $(\alpha_k^* + \epsilon)$-approximation to the* Max-kAND *value of Ψ with probability* 99/100.

Table 1 The table below displays concrete approximation ratio for Max-kAND, for $k \in \{2, \dots, 5\}$. In the second column, we write the trivial upper bound of $2^{-(k-1)}$ on the approximation ratio of all oblivious algorithms (and more generally all "local" algorithms) for Max-kAND (see Observation 7 below). In the third column, we write α_k^*, the approximation ratio achieved by the best superoblivious algorithm (and also by the best $o(\sqrt{n})$-space sketching algorithm [7]). In the fourth column, we highlight that the only previously known CSP for which oblivious algorithms outperformed superoblivious algorithms was Max-2AND (from [17]). In the fifth, we include approximation ratios from our "perturbed superoblivious" algorithms, as in Theorem 2 with $\delta = 0.01$ and $\epsilon = 0.001$. Finally, in the sixth column, we report ratios achieved by much more complex algorithms which we constructed. These algorithms are parametrized by triples (ℓ, x, y), where ℓ specifies the number of bias classes, and x, y specify a rounding vector. ℓ is chosen such that the number of variables, which is roughly $(2\ell)^k$ (see Section 3.1 below), is in at most the hundreds of thousands, in order for the LP solver to run in a reasonable amount of time. The bias partition is a uniform partition of $[0, 1]$ into ℓ intervals and, imitating the algorithm in [17, Proof of Theorem 1.3], our rounding vector are "two-piece piecewise-linear functions": the first part of the vector, up to bias x, interpolates linearly between probability $\frac{1}{2}$ and y, and the second part interpolates linearly between probability y and 1. The values in the last two columns were calculated with a Python script and the LP solver `glpk`; the code is available online at `https://github.com/singerng/oblivious-csps`. For the final column, the parameters (x, y) were chosen using a grid search; solving the final LPs took 1 hour, 56 minutes on a 2021 Macbook Pro.

k	**Upper bound**	**Superobl.**	**Prev.**	**New, pert.**	**New, piecewise lin.**
2	$1/2 = 0.5$	$4/9 \approx 0.4444$	0.4835 [17]	0.4457	0.4844 @ $(200, 0.5, 1.0)$
3	$1/4 = 0.25$	$2/9 \approx 0.2222$		0.2226	0.2417 @ $(30, 0.7, 1.0)$
4	$1/8 = 0.125$	$72/625 = 0.1152$		0.1157	0.1188 @ $(11, 0.8, 0.8)$
5	$1/16 = 0.0625$	$36/625 \approx 0.0576$		0.0578	0.0589 @ $(7, 0.95, 0.8)$

Both of these results are interesting because, as shown by Boyland *et al.* [7] (analyzing families of algorithms and lower bounds due to Chou *et al.* [12]), there are $O(\log n)$-space streaming algorithms which output (arbitrarily close to) α_k^*-approximations for adversarially-ordered streams, and this is the best achievable ratio in $o(\sqrt{n})$ space.[5] Therefore, our results show that by relaxing either the adversarial-ordering assumption or the space bound, one can achieve better algorithms (the latter under a bounded-degree assumption). Analogous results to Theorems 3 and 4 were obtained for Max-DICUT by Saxena, Singer, Sudan, and Velusamy [36] for the special case of Max-DICUT, there contrasting with algorithms and lower bounds due to Chou, Golovnev, and Velusamy [14].

We also include some explicit improved approximation ratios calculated using computer search and LP solvers in Table 1.

1.3 Technical overview

The first main technical step in the paper is to develop, for each oblivious algorithm $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}$ (defined by a bias partition $\mathbf{t}$ and a rounding vector $\mathbf{p}$), a linear program (LP) which characterizes the approximation ratio of $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}$; this LP is contained in Lemma 11 below, and is a generalization of the LP developed for oblivious algorithms in Max-DICUT in [17]. The LP has a simple structure: Each feasible solution corresponds to a certain family of instances of Max-kAND on which $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}$ produces the same approximation to the Max-kAND

[5] Again, this lower bound is currently only known to hold for a subclass of streaming algorithms called *sketching* algorithms, but the algorithm in Theorem 4 appears to be such an algorithm.

value, and the objective equals this approximation value. In particular, we will assign to each clause in an instance Ψ of Max-kAND a "pattern" based on the biases of and negations on the variables, and use the observation that the probability that any particular clause is satisfied depends only on its pattern.

Next, we formulate the dual LP for this original "primal" LP (see Lemma 12 below). Here is where we benefit massively from the fact that the performance of oblivious algorithms is captured by a linear program, because by the magic of LP duality, it is possible to constructively show that oblivious algorithms perform *well*: While feasible solutions to the *primal* LP *upper-bound* the ratio achieved by an oblivious algorithm $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}$, feasible solutions to the *dual* LP *lower-bound* the ratio! In other words, to prove that an oblivious algorithm performs well on *all* instances, it suffices to construct a *single* feasible dual solution.

To prove Theorem 2, we now want to compare the dual LP for superoblivious algorithms and their "perturbations", and show that we can get "improved" feasible solutions in the latter case. It turns out that in this setting, the primal LP has $O(k^5)$ variables and only 7 inequality constraints; therefore the dual LP has 7 variables and $O(k^5)$ inequality constraints. We make the crucial observation that in the superoblivious case there is an optimal dual solution which is *sparse*: It is supported on only 3 variables. Since this dual solution is so simple, we can analytically prove its feasibility for all k. The statement of the kind of dual solution we are seeking is in Lemma 13 below.

And moreover, this inequality will show that in the superoblivious case, when we plug in our special solution, all but 6 of the $O(k^5)$ dual constraints have slack! Thus, for very small values of δ, it will be sufficient to slightly perturb this special solution in a way that makes these 6 "core" constraints strictly satisfied. This involves careful analysis based on certain elementary inequalities, using simple inequalities such as that $\frac{1-\epsilon}{1-k\epsilon} > \frac{1+\epsilon}{1+k\epsilon}$ for all $\epsilon > 0$ and $k > 1$. The analysis outlined in this paragraph is omitted from this version of the paper; see the full version.

1.4 Future questions

Streaming algorithms

We hypothesize that the bounded-degree assumption in Theorem 4 can be relaxed to give an $\widetilde{O}(n^{1-1/k})$-space algorithm for *all* instances (in input form with poly(n) clauses). Specifically, Saxena, Singer, Sudan, and Velusamy [36] developed sketching techniques enabling such a guarantee for Max-DICUT, the bounded-degree counterpart being provided by their earlier work [37]; perhaps these ideas can be extended to Max-kAND.

More CSPs

It would also be interesting to extend the framework in this paper to more CSPs, both other Boolean CSPs and to CSPs over larger alphabets. To the best of our knowledge, it is even plausible that every CSP which admits nontrivial $O(\log n)$-space sketching algorithms (as analyzed in [13]) also admits "oblivious-style" approximation algorithms, which in turn yield better sublinear-space streaming algorithms. A good starting point here would be to analyze symmetric Boolean CSPs, since in that setting we know that all CSPs which do not support one-wise distributions of satisfying assignments admit such nontrivial sketching algorithms (see [12, Proposition 2.10]); this class includes Max-kAND, for which we developed such results in this paper, but we could hope for improved "oblivious-style" algorithms for other such CSPs, e.g. symmetric threshold functions.

"Uniform" hard instances

In the special case of Max-DICUT, Feige and Jozeph [17] constructed what might be called a *uniformly hard* instance of Max-DICUT: For this single instance, *every* oblivious algorithm achieves a ratio less than than 0.49. (This strengthens the $\frac{1}{2}$ bound from a "trivial" instance, a single bidirected edge; see also Observation 7 below.) It would be interesting to construct similar instances for Max-kAND, $k \geq 3$, especially if such this construction could be made analytic. We note that such an object corresponds to a feasible solution to the linear program in Lemma 11 for which *every* choice of rounding vector has objective strictly less than $\frac{1}{2}$, and therefore for Max-kAND, any proof would require certifying that a certain degree-k polynomial is bounded below $\frac{1}{2}$ over $p \in [0, 1]$.

An optimal rounding curve?

To construct an 0.483-approximate oblivious algorithm for Max-DICUT, Feige and Jozeph [17] rounded vertices using (a discretization of) a sigmoid-shaped piecewise-linear function: This function rounds vertices with bias $b \in (0, 1]$ to 1 with probability

$$p(b) = \begin{cases} \frac{1}{2} + b & 0 \leq b \leq \frac{1}{2} \\ 1 & \frac{1}{2} \leq b \leq 1 \end{cases}.$$

But is it possible to analytically calculate the optimal rounding function (and is it unique)? Given any discretization of biases into intervals, one could in principle enumerate all basic feasible solutions to the LP, and then calculate the best rounding vector; that is, each rounding vector will induce an objective function for the LP, and the best rounding vector maximizes the minimum objective over all basic feasible solution. Towards this, it might be helpful to get a handle on the vertices of this LP's polytope, and whether there is some simple way to enumerate them.

Outline

We define Max-kAND and oblivious algorithms formally in Section 2 and develop the linear-programming characterization for the approximation ratio of oblivious algorithms, and some other basic tools, in Section 3. We begin analyzing the dual LP to prove Theorem 2 in Section 4. The remainder of the proof of Theorem 2, and the proofs of the remaining theorems, are omitted; see the full version.

2 Definitions: Max-kAND and oblivious algorithms

We now give formal definitions for the Max-kAND problem and for oblivious algorithms.

The Max-kAND problem

For the remainder of the paper, we adopt a (nonstandard) convention which views variables in Max-kAND as taking $\{-1, +1\}$ values (as opposed to $\{0, 1\}$); this is for notational convenience in defining bias and similar concepts.

▶ **Definition 5** (Max-kAND). *An* instance *of the* Max-kAND *problem on n variables is given by a sequence of constraints $C_1, \ldots, C_m$, with $C_j = (V_j^+, V_j^-, w_j)$, consisting of "positive variables" $V_j^+ \subseteq [n]$ and "negative variables" $V_j^- \subseteq [n]$ with $V_j^+ \cap V_j^- = \emptyset$ and $|V_j^+ \cup V_j^-| = k$, and a weight $w_j \geq 0$. An* assignment *for this problem is given by $(\mathbf{x}) = (x_1, \ldots, x_n) \in \{\pm 1\}^n$, and the* value *of this assignment is*

$$\mathsf{val}_\Psi(\mathbf{x}) \overset{\text{def}}{=} \frac{\sum_{j=1}^m \mathbb{1}[x_v = +1\ \forall v \in V_j^+ \wedge x_v = -1\ \forall v \in V_j^-] \cdot w_j}{\sum_{j=1}^m w_j}.$$

The value *of the instance* Ψ *is*

$$\mathsf{val}_\Psi \overset{\text{def}}{=} \max_{\mathbf{x}\in\{\pm1\}^n} \mathsf{val}_\Psi(\mathbf{x}).$$

Next, towards defining the bias of a variable in an instance, for any variable $v \in [n]$, we define its *positive* and *negative weight*:

$$w_\Psi^+(v) \overset{\text{def}}{=} \sum_{j=1}^m \mathbb{1}[v \in V_j^+] \cdot w_j \text{ and } w_\Psi^-(v) \overset{\text{def}}{=} \sum_{j=1}^m \mathbb{1}[v \in V_j^-] \cdot w_j. \tag{4}$$

Bias

Then, we define the *bias* of a variable as:[6]

$$\mathsf{bias}_\Psi(v) \overset{\text{def}}{=} \frac{w_\Psi^+(v) - w_\Psi^-(v)}{w_\Psi^+(v) + w_\Psi^-(v)}. \tag{5}$$

Bias partitions

Next, we define notations for a partition of the space of possible biases $[-1,+1]$ into $L = 2\ell+1$ intervals labeled by $\{-\ell,\ldots,+\ell\}$. The data of such a partition is a "bias partition" vector $\mathbf{t} = (t_0,\ldots,t_\ell)$ with $0 \le t_0 < \cdots < t_{+\ell} = 1$. Each such vector $\mathbf{t}$ defines a set of intervals $\mathsf{Int}^{\mathbf{t}}_{-\ell},\ldots,\mathsf{Int}^{\mathbf{t}}_{+\ell}$ such that each real number in $[-1,+1]$ belongs to exactly one interval. Specifically, we define:

- $\mathsf{Int}^{\mathbf{t}}_0 := [-t_0,+t_0]$, and
- $\mathsf{Int}^{\mathbf{t}}_{-i} := [-t_i,-t_{i-1})$ and $\mathsf{Int}^{\mathbf{t}}_{+i} := (+t_{i-1},t_i]$ for each $i \in \{1,\ldots,\ell\}$.

Our choice of which ends of these intervals are open and which are closed is an arbitrary convention; the only important property of the decomposition of $[-1,+1]$ into intervals is that it is *symmetric*.

We also let t_i^+ and t_i^- denote the upper and lower bounds on $\mathsf{Int}^{\mathbf{t}}_i$, respectively, i.e., $t_i^+ = \sup \mathsf{Int}^{\mathbf{t}}_i$ and $t_i^- = \inf \mathsf{Int}^{\mathbf{t}}_i$, so that e.g., $t_i^+ = t_i$ for $i \ge 0$, whereas $t_i^+ = t_{-(i+1)}$ for $i < 0$.

Oblivious algorithms

Given a partition of $[-1,+1]$ into bias class intervals (defined by some vector $\mathbf{t}$ as in the previous paragraph), we now define an algorithm which randomly rounds each variable of an Max-kAND instance according to which interval its bias falls into. The data of such a rounding scheme is a *rounding vector* $\mathbf{p} = (p_1,\ldots,p_\ell)$ of probabilities; given this vector, a vertex with bias in $\mathsf{Int}^{\mathbf{t}}_{+i}$ is rounded to 1 with probability p_i.

More precisely, we define an oblivious algorithm as follows:

[6] Throughout the paper, we assume every variable appears in at least one constraint, and therefore that $w_\Psi^+(v) + w_\Psi^-(v) > 0$. (This is WLOG, since variables appearing in no constraints can be ignored for the purposes of Max-kAND.)

▶ **Definition 6** (Oblivious algorithm for Max-kAND). *Let $L = 2\ell + 1 \geq 3$ be an odd integer. Let $\mathbf{t} = (t_0, \dots, t_\ell)$ be a bias partition and $\mathbf{p} = (p_1, \dots, p_\ell)$ a rounding vector. For any $k \geq 2$, the oblivious algorithm $\texttt{Obl}_k^{\mathbf{t},\mathbf{p}}$ for* Max-kAND *behaves as follows: Given an instance Ψ, for each variable $v \in \{1, \dots, n\}$ independently:*

- *If $\mathsf{bias}_\Psi(v) \in \mathsf{Int}_0^{\mathbf{t}}$, assign $x_v \mapsto 1$ w.p. $\frac{1}{2}$, $x_v \mapsto -1$ w.p. $\frac{1}{2}$.*
- *If $\mathsf{bias}_\Psi(v) \in \mathsf{Int}_{+i}^{\mathbf{t}}$ for $i \in \{1, \dots, \ell\}$, assign $x_v \mapsto 1$ w.p. p_i, $x_v \mapsto -1$ w.p. $1 - p_i$.*
- *If $\mathsf{bias}_\Psi(v) \in \mathsf{Int}_{-i}^{\mathbf{t}}$ for $i \in \{1, \dots, \ell\}$, assign $x_v \mapsto 1$ w.p. $1 - p_i$, $x_v \mapsto -1$ w.p. p_i.*

We denote by $\texttt{Obl}_k^{\mathbf{t},\mathbf{p}}(\Psi)$ the expected value of the assignment produced by this algorithm,[7] *and by*

$$\alpha(\texttt{Obl}_k^{\mathbf{t},\mathbf{p}}) \stackrel{\text{def}}{=} \inf_{\Psi} \frac{\texttt{Obl}_k^{\mathbf{t},\mathbf{p}}(\Psi)}{\mathsf{val}_\Psi}$$

the approximation ratio achieved by this algorithm.

In the simplest interesting case, we have $\ell = 1$, $\mathbf{t} = (0, 1)$, and $\mathbf{p} = (p)$. These algorithms, which we call *superoblivious* algorithms, ignore the *magnitude* of the bias of each variable, rounding only based on sign: E.g., negatively-biased variables are rounded to 1 w.p. $1 - p$.

▶ **Observation 7** (A "symmetric" hard instance). *There is a simple lower-bound construction which shows that* no *oblivious algorithm for* Max-kAND *can achieve a ratio better than $2^{-(k-1)}$. (Note that the constant α_k^* defined above, which according to Theorem 1 is the optimal ratio of superoblivious algorithms, equals this upper bound times a "discounting" factor.) Consider any k and the instance with two equally weighted constraints $C^+ = (+1, \dots, +1), (1, \dots, k)$ and $C^- = (-1, \dots, -1), (1, \dots, k)$; that is, the two constraints want $x_1, \dots, x_k$ to be all-$(+1)$'s and all-(-1)'s, respectively. Every variable has bias zero so it will be rounded uniformly by every oblivious algorithm, yielding value 2^{-k}, while the "greedy" all-$(+1)$'s (or all-(-1)'s) assignment achieves value $\frac{1}{2}$. (Indeed, this "lower bound" holds for any class of algorithms which cannot "break the symmetry" between these two greedy assignments.) Thus, for* every *oblivious algorithm $\texttt{Obl}_k^{\mathbf{t},\mathbf{p}}$, $\alpha(\texttt{Obl}_k^{\mathbf{t},\mathbf{p}}) \leq 2^{-(k-1)}$.*

▶ **Observation 8** (Generalizing oblivious algorithms). *There are a few natural ways to generalize our definition of oblivious algorithms:*

- *We could consider rounding functions which are not "antisymmetric". That is, we could round variables with bias $+b$ and variables with bias $-b$ with probabilities which are not complementary. In particular, for $b = 0$, we could round bias-0 variables to 1 with probability $p \neq \frac{1}{2}$. However, on the instance described in the previous observation, such an algorithm outputs an assignment of value $\frac{1}{2}p^k + \frac{1}{2}(1-p)^k$, and this value is uniquely maximized at $p = \frac{1}{2}$.*
- *We could use "general rounding functions", that is, arbitrary functions mapping each bias value to a rounding probability, to round each vertex. Under this "rounding functions" view, the above definition via intervals corresponds to "step fucntions" where the domain is broken up into finitely many intervals on which the function is constant. However, the focus of the current work is analyzing the approximation ratio of oblivious algorithms via linear programming, and such "continuous" algorithms would not be (directly) amenable to analysis via linear programming.*

[7] We abuse this notation and often think of $\texttt{Obl}_k^{\mathbf{t},\mathbf{p}}(\Psi)$ as the *output* of the oblivious algorithm, i.e., we think of the oblivious algorithm's goal as outputting a (scalar) estimate of the value of the instance; this holds especially in the context of streaming algorithms.

- *We could use a "max" algorithm, taking an ensemble of oblivious algorithms and outputting the best assignment produced by any of them.*

For Max-2AND*, Feige and Jozeph [17] showed that even allowing all these generalizations simultaneously stills fall short of achieving the ultimate goal of a $\frac{1}{2}$-approximation, and they described a specific instance witnessing this.*

3 The linear-programming framework for oblivious algorithms

In this section, we develop a linear program which captures the "worst-case instance" for any oblivious algorithm, and therefore can be used to calculate the approximation ratio (Lemma 11), as well as the corresponding dual linear program (Lemma 12). These will be applied to bound the approximation ratios of certain oblivious algorithms in the following sections.

3.1 Clause patterns

Let Ptn_k^L denote the set of vectors $\mathbf{c} = (\mathbf{c}^+, \mathbf{c}^-) = (c^+_{-\ell}, \dots, c^+_{+\ell}, c^-_{-\ell}, \dots, c^-_{-\ell})$ whose entries are natural numbers and sum to k. These are useful because they describe each particular clause from the perspective of an L-class oblivious algorithms. In particular, given a clause C, we denote its *pattern* $\mathsf{ptn}^{\mathbf{t}}(C) = (c^+_{-\ell}, \dots, c^+_{+\ell}, c^-_{-\ell}, \dots, c^-_{-\ell}) \in \mathsf{Ptn}_k^L$ where c_i^+ and c_i^- denote the number of positive and negative literals in C whose variables have bias class i, respectively, for each $i \in \{-\ell, \dots, +\ell\}$. That is, e.g.,

$$c_i^+ = |\{v \in V_j^+ : \mathsf{bias}_\Psi(v) \in \mathsf{Int}_i^{\mathbf{t}}\}|.$$

Now, for any rounding vector $\mathbf{p} = (p_1, \dots, p_\ell)$, we define

$$\mathsf{prob}^{\mathbf{p}}(\mathbf{c}) = 2^{-(c_0^+ + c_0^-)} \prod_{i=1}^{\ell} p_i^{c^+_{+i} + c^-_{-i}} (1 - p_i)^{c^-_{+i} + c^+_{-i}} \tag{6}$$

for each $\mathbf{c} \in \mathsf{Ptn}_k^L$.[8] Then we have:

▷ Claim 9. Let Ψ be an instance of Max-kAND with clauses $C_1, \dots, C_m$ with weights $w_1, \dots, w_m$, respectively. Then

$$\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}(\Psi) = \frac{\sum_{j=1}^m \mathsf{prob}^{\mathbf{p}}(\mathsf{ptn}^{\mathbf{t}}(C_j)) \cdot w_j}{\sum_{j=1}^m w_j}.$$

Proof. By linearity of expectation, it suffices to show that each clause C_j is satisfied w.p. $\mathsf{prob}^{\mathbf{p}}(\mathsf{ptn}^{\mathbf{t}}(C_j))$. We can rewrite

$$\mathsf{prob}^{\mathbf{p}}(\mathbf{c}) = 2^{-c_0^+} 2^{-c_0^-} \prod_{i=1}^{\ell} p_i^{c^+_{+i}} p_i^{c^-_{-i}} (1 - p_i)^{c^-_{+i}} (1 - p_i)^{c^+_{-i}}.$$

Recalling that each variable is assigned independently, and the clause is satisfied iff each literal is, the above expression precisely represents the probability that the clause is satisfied. (E.g., if there is a negative literal whose variable has bias class $+i$, this literal is satisfied with probability $1 - p_i$; the number of such factors in the probability is c^-_{+i}.) ◁

We observe that $|\mathsf{Ptn}_k^L| = \binom{k+2L-1}{2L-1}$ by the "stars-and-bars" formula. For instance, if $L = 3$ (as will be the case in the explicit analysis in the following sections), we have $|\mathsf{Ptn}_k^L| = O(k^5)$.

[8] In this expression we adopt the convention $0^0 = 1$, i.e., if $p_i = 0$ but $c^-_{+i} + c^+_{-i} = 0$ then we ignore the factor 0.

3.2 The factor-revealing linear program

We denote by $\mathsf{PosPtn}_k^L \subseteq \mathsf{Ptn}_k^L$ the space of clause patterns without negations, i.e., $\mathbf{c}$ such that $c_{-\ell}^- = \cdots = c_{+\ell}^- = 0$. For two vectors $\mathbf{x} = (x_1, \ldots, x_n), \mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{R}^n$, let $\mathbf{x} \odot \mathbf{y} = (x_1 y_1, \ldots, x_n y_n)$ denote their entrywise product. To design the linear program, we will need the following useful proposition:

▶ **Proposition 10** (Flipping). *Let Ψ be an instance of* Max-kAND*, and for any assignment $\mathbf{y} = (y_1, \ldots, y_n) \in \{\pm 1\}^n$, let $\mathsf{flip}^{\mathbf{y}}(\Psi)$ denote the instance of* Max-kAND *where we "flip" the variables v with $y_v = -1$; that is, each clause $C_j = (V_j^+, V_j^-, w_j)$ in Ψ becomes a clause $D_j = (U_j^+, U_j^-, w_j)$ where $U_j^+ = \{v \in V_j^+ : y_v = +1\} \cup \{v \in V_j^- : y_v = -1\}$ and $U_j^- = \{v \in V_j^+ : y_v = -1\} \cup \{v \in V_j^- : y_v = +1\}$. Then:*

- *For every assignment $\mathbf{x} \in \{\pm 1\}^n$, $\mathsf{val}_\Psi(\mathbf{x}) = \mathsf{val}_{\mathsf{flip}^{\mathbf{y}}(\Psi)}(\mathbf{x} \odot \mathbf{y})$.*
- *In particular, if $\mathbf{x}$ is an optimal assignment to Ψ, then $\mathbf{x} \odot \mathbf{y}$ is an optimal assignment to $\mathsf{flip}^{\mathbf{y}}(\Psi)$.*
- $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}(\Psi) = \mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}(\mathsf{flip}^{\mathbf{y}}(\Psi))$.

Proof. Follows immediately from definitions. ◀

▶ **Lemma 11** (Primal characterization). *For every bias partition $\mathbf{t} = (t_0, \ldots, t_\ell)$ and rounding vector $\mathbf{p} = (p_1, \ldots, p_\ell)$, the approximation ratio $\alpha(\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}})$ achieved by $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}$ equals the value of the following linear program:*

$$\begin{cases} \text{minimize} & \displaystyle\sum_{\mathbf{c} \in \mathsf{Ptn}_k^L} \mathsf{prob}^{\mathbf{p}}(\mathbf{c}) \cdot W(\mathbf{c}) & \\ \text{s.t.} & W(\mathbf{c}) \geq 0 & \forall \mathbf{c} \in \mathsf{Ptn}_k^L \\ & \displaystyle\sum_{\mathbf{c} \in \mathsf{PosPtn}_k^L} W(\mathbf{c}) = 1 & \\ & t_i^-(W^+(i) + W^-(i)) \leq W^+(i) - W^-(i) & \forall i \in \{-\ell, \ldots, +\ell\} \\ & W^+(i) - W^-(i) \leq t_i^+(W^+(i) + W^-(i)) & \forall i \in \{-\ell, \ldots, +\ell\} \end{cases}$$

where we define the linear functions

$$W^+(i) = \sum_{\mathbf{c} \in \mathsf{Ptn}_k^L} c_i^+ W(\mathbf{c}) \text{ and } W^-(i) = \sum_{\mathbf{c} \in \mathsf{Ptn}_k^L} c_i^- W(\mathbf{c}).$$

Proof. Let α_{alg} denote the approximation ratio of $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}$, and α_{LP} the minimum value of the linear program. This proof generalizes [17, Proof of Theorem 1.2].

$\boldsymbol{\alpha_{\text{LP}} \leq \alpha_{\text{alg}}}$. We show that for every instance Ψ of Max-kAND, there is a feasible LP solution $\{W(\mathbf{c})\}_{\mathbf{c} \in \mathsf{Ptn}_k^L}$ of objective value $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}(\Psi)/\mathsf{val}_\Psi$.

Towards this claim, by Proposition 10, we can assume WLOG that the all-(+1)'s assignment is optimal for Ψ. Also, we assume WLOG by rescaling that Ψ has total weight $\frac{1}{\mathsf{val}_\Psi}$, i.e., $\sum_{j=1}^m w_j = \frac{1}{\mathsf{val}_\Psi}$. Now, let $C_1, \ldots, C_m$ denote the constraints of Ψ, and let $W(\mathbf{c}) := \sum_{j=1}^m \mathbb{1}[\mathsf{ptn}^{\mathbf{t}}(C_j) = \mathbf{c}] w_j$. We claim that $\{W(\mathbf{c})\}_{\mathbf{c} \in \mathsf{Ptn}_k^L}$ is feasible and has objective value $\sum_{\mathbf{c} \in \mathsf{Ptn}_k^L} \mathsf{prob}^{\mathbf{p}}(\mathbf{c}) W(\mathbf{c}) = \mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}(\Psi)/\mathsf{val}_\Psi$.

First, we check feasibility. Clearly all $W(\mathbf{c})$'s are nonnegative. Next, we have

$$\begin{aligned}
\mathsf{val}_\Psi &= \mathsf{val}_\Psi(+\mathbf{1}) && \text{(all-(+1)'s is optimal)}\\
&= \frac{\sum_{j=1}^m w_j \mathbb{1}[|V_j^-| = 0]}{\sum_{j=1}^m w_j} && \text{(def. of } \mathsf{val}_\Psi(+\mathbf{1})\text{)}\\
&= \frac{\sum_{\mathbf{c}\in\mathsf{PosPtn}_k^L} W(\mathbf{c})}{1/\mathsf{val}_\Psi} && \text{(def. of } \mathsf{PosPtn}_k^L \text{ and total weight assumption)}
\end{aligned}$$

which rearranges to $\sum_{\mathbf{c}\in\mathsf{PosPtn}_k^L} W(\mathbf{c}) = 1$.

Now, recall the definitions of $\mathsf{bias}_\Psi, w_\Psi^+, w_\Psi^-$ from Section 2. Fix a bias class $i \in \{-\ell, \dots, +\ell\}$. For any variable v with bias class i, we have $\mathsf{bias}_\Psi(v) \in \mathsf{Int}_i^{\mathbf{t}}$, so $t_i^- \leq \mathsf{bias}_\Psi(v) \leq t_i^+$, so multiplying through by $w_\Psi^+(v) + w_\Psi^-(v)$, we get

$$t_i^-(w_\Psi^+(v) + w_\Psi^-(v)) \leq w_\Psi^+(v) - w_\Psi^-(v) \leq t_i^+(w_\Psi^+(v) + w_\Psi^-(v)).$$

Letting $\mathcal{V}_i$ denote the set of all variables in Ψ with bias class i, we can sum over these equations to get

$$t_i^- \sum_{v\in\mathcal{V}_i}(w_\Psi^+(v) + w_\Psi^-(v)) \leq \sum_{v\in\mathcal{V}_i}(w_\Psi^+(v) - w_\Psi^-(v)) \leq t_i^+ \sum_{v\in\mathcal{V}_i}(w_\Psi^+(v) + w_\Psi^-(v)).$$

We claim that

$$W^+(i) = \sum_{v\in\mathcal{V}} w_\Psi^+(v),$$

and similarly $W^-(i) = \sum_{v\in\mathcal{V}} w_\Psi^-(v)$. These equalities imply that $W(\cdot)$ satisfies the feasibility constraints, and it remains to prove them. Now recall $w_\Psi^+(v) = \sum_{j=1}^m \mathbb{1}[v \in V_j^+]w_j$; therefore,

$$\sum_{v\in\mathcal{V}} w_\Psi^+(v) = \sum_{j=1}^m \sum_{v\in V_j^+} \mathbb{1}[\mathsf{bias}_\Psi(v) \in \mathsf{Int}_i^{\mathbf{t}}]w_j,$$

and j-th term in this sum is precisely c_i^+ where $\mathbf{c} = (\mathbf{c}^+, \mathbf{c}^-) = \mathsf{ptn}^{\mathbf{t}}(C_j)$. The proof for $W^-(i)$ is similar.

Finally, by Claim 9 and our assumption $\sum_{j=1}^m w_j = 1/\mathsf{val}_\Psi$, we have

$$\mathtt{Obl}_k^{\mathbf{t},\mathbf{P}}(\Psi) = \frac{\sum_{j=1}^m w_j \mathsf{prob}^{\mathbf{P}}(\mathsf{ptn}^{\mathbf{t}}(C_j))}{\sum_{j=1}^m w_j} = \frac{\sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} W(\mathbf{c}) \cdot \mathsf{prob}^{\mathbf{P}}(\mathbf{c})}{1/\mathsf{val}_\Psi},$$

which rearranges to $\mathtt{Obl}_k^{\mathbf{t},\mathbf{P}}(\Psi)/\mathsf{val}_\Psi = \sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} W(\mathbf{c}) \cdot \mathsf{prob}^{\mathbf{P}}(\mathbf{c})$, as desired.

$\alpha_{\mathrm{LP}} \geq \alpha_{\mathrm{alg}}$. This argument is essentially converse to the former argument, but there are two technical issues: (i) the linear program does not encode strict inequality constraints, while an oblivious algorithm needs to (in the sense that e.g., if $t_0 = 0$, then the algorithm rounds vertices with bias 0 and bias $+\epsilon$ differently), and (ii) since an Max-kAND constraint cannot use a variable twice, we might need many variables with the same bias in the instance we create. We omit the proof here; see the full version. ◀

▶ **Lemma 12** (Dual characterization). *For every bias partition* $\mathbf{t} = (t_0, \dots, t_\ell)$ *and rounding vector* $\mathbf{p} = (p_1, \dots, p_\ell)$*, the approximation ratio* $\alpha(\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}})$ *achieved by* $\mathtt{Obl}_k^{\mathbf{t},\mathbf{p}}$ *equals the value of the following linear program:*

$$\begin{cases} \max & z \\ \text{s.t.} & \mathbb{1}[\mathbf{c} \in \mathsf{PosPtn}_k^L] \cdot z \\ & \quad + \sum_{i=-\ell}^{+\ell} \left(((1-t_i^-)c_i^+ - (t_i^- + 1)c_i^-)y_i^- + ((t_i^+ - 1)c_i^+ + (1 + t_i^+)c_i^-)y_i^+\right) \le \mathsf{prob}^{\mathbf{P}}(\mathbf{c}) & \forall \mathbf{c} \in \mathsf{Ptn}_k^L \\ & y_i^- \ge 0 & \forall i \in \{-\ell, \dots, +\ell\} \\ & y_i^+ \ge 0 & \forall i \in \{-\ell, \dots, +\ell\} \end{cases}$$

Proof. To place the primal LP (from Lemma 11) in a more standard form, we rewrite the primal inequality $t_i^-(W^+(i)+W^-(i)) \le W^+(i)-W^-(i)$ as $(t_i^- -1)W^+(i)+(t_i^- +1)W^-(i) \le 0$; expanding the definitions of $W^+(i)$ and $W^-(i)$, this is equivalent to $(t_i^- -1)\sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} c_i^+ W(\mathbf{c}) + (t_i^- +1)\sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} c_i^- W(\mathbf{c}) \le 0$. Similarly, the inequality $W^+(i)-W^-(i) \le t_i^+(W^+(i)+W^-(i))$ becomes $(1-t_i^+)\sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} c_i^+ W(\mathbf{c}) - (1+t_i^+)\sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} c_i^- W(\mathbf{c}) \le 0$. Therefore, the primal LP is equivalent to the following standard-form LP:

$$\begin{cases} \text{minimize} & \sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} \mathsf{prob}^{\mathbf{P}}(\mathbf{c}) \cdot W(\mathbf{c}) \\ \text{s.t.} & W(\mathbf{c}) \ge 0 & \forall \mathbf{c} \in \mathsf{Ptn}_k^L \\ & \sum_{\mathbf{c}\in\mathsf{PosPtn}_k^L} W(\mathbf{c}) = 1 \\ & (1-t_i^+) \sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} c_i^+ W(\mathbf{c}) - (1+t_i^+) \sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} c_i^- W(\mathbf{c}) \le 0 & \forall i \in \{-\ell, \dots, +\ell\} \\ & (t_i^- - 1) \sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} c_i^+ W(\mathbf{c}) + (t_i^- + 1) \sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} c_i^- W(\mathbf{c}) \le 0 & \forall i \in \{-\ell, \dots, +\ell\} \end{cases}$$

By LP duality, the above LP has the same value as its dual LP, which is the LP in the hypothesis.[9] ◀

4 Proving Theorem 2 by analyzing "dual slack"

In this section, we outline the first step towards proving Theorem 2 by constructing dual solutions which witness lower bounds on the approximation ratio of oblivious algorithms. This first step is the following lemma, which gives a clean sufficient condition for a lower bound on the approximation ratio by constructing a certain sparse dual solution and applying the dual program (Lemma 12):

[9] See e.g. [32, p. 85]). One has to be careful with the signs, since our primal LP is a *minimization* LP. Instead, we can consider the LP which maximizes $-\sum_{\mathbf{c}\in\mathsf{Ptn}_k^L} \mathsf{prob}^{\mathbf{P}}(\mathbf{c}) \cdot W(\mathbf{c})$ (whose output is the negation of our desired output).
Applying duality to this LP gives one which minimizes z' such that $\mathbb{1}[\mathbf{c} \in \mathsf{PosPtn}_k^L] \cdot z' + \sum_{i=-\ell}^{\ell} \left(((1-t_i^-)c_i^+ - (t_i^- + 1)c_i^-)y_i^- + ((t_i^+ - 1)c_i^+ + (1 + t_i^+)c_i^-)y_i^+\right) \ge -\mathsf{prob}^{\mathbf{P}}(\mathbf{c})$. Transforming to a maximization problem equivalent to our original LP (since we had a negation!), we maximize $-z'$ such that $\mathbb{1}[\mathbf{c} \in \mathsf{PosPtn}_k^L] \cdot z' + \sum_{i=-\ell}^{\ell} \left(((1-t_i^-)c_i^+ - (t_i^- + 1)c_i^-)y_i^- + ((t_i^+ - 1)c_i^+ + (1 + t_i^+)c_i^-)y_i^+\right) \ge -\mathsf{prob}^{\mathbf{P}}(\mathbf{c})$. Finally, we negate both sides of this inequality, and use the bijective transformation $z = -z'$.

▶ **Lemma 13** (Sufficient conditions for good approximations). *For every $k \geq 2 \in \mathbb{N}$, $0 \leq \gamma, \delta \leq 1$, let $\mathbf{t} = (\delta, 1)$ and $\mathbf{p} = (\frac{1}{2}(1+\gamma))$. The algorithm $\texttt{Obl}_k^{\mathbf{t},\mathbf{p}}$ has approximation ratio $\alpha(\texttt{Obl}_k^{\mathbf{t},\mathbf{p}}) \geq 2^{-(k-1)}\beta$ if the following statement holds: There exist $X, Y \geq 0$ such that:*

$$\begin{cases} (1+\delta)\left(1 - \dfrac{i+j}{k}\right)Y + (1-\delta)\dfrac{j}{k}X \leq \beta^{-1}(1-\gamma)^i(1+\gamma)^j & \forall i, j \in \mathbb{N}, i+j \leq k \\ 2 - (1-\delta)\left(1 - \dfrac{i+j}{k}\right)Y - (1+\delta)\dfrac{i}{k}X \leq \beta^{-1}(1-\gamma)^i(1+\gamma)^j & \forall i, j \in \mathbb{N}, i+j \leq k \end{cases}$$

Proof. Consider applying the dual characterization of the approximation ratio (Lemma 12) with the solution $z = 2\beta/2^k$, $y^+_{-1} = X\beta/(k2^k)$, $y^+_0 = Y\beta/(k2^k)$, and $y^+_{+1} = y^-_{-1} = y^-_0 = y^-_{+1} = 0$; it is sufficient to show that this solution is feasible. Note that $t^+_{-1} = -\delta$ and $t^+_0 = \delta$. Thus, the feasibility constraints in Lemma 12 become

$$\frac{\beta}{2^k}\left(\mathbb{1}[\mathbf{c} \in \mathsf{PosPtn}^L_k] \cdot 2 + ((-\delta - 1)c^+_{-1} + (1-\delta)c^-_{-1})\frac{X}{k} + ((\delta - 1)c^+_0 + (1+\delta)c^-_0)\frac{Y}{k}\right) \leq \mathsf{prob}^{\mathbf{P}}(\mathbf{c}) \quad \forall \mathbf{c} \in \mathsf{Ptn}^L_k. \quad (7)$$

By Equation (6), and since $c^+_{-1} + c^+_0 + c^+_{+1} + c^-_{-1} + c^-_0 + c^-_{+1} = k$, we have

$$\mathsf{prob}^{\mathbf{P}}(\mathbf{c}) = \left(\frac{1}{2} - \frac{\gamma}{2}\right)^{c^+_{-1}+c^-_{+1}}\left(\frac{1}{2}\right)^{c^+_0+c^-_0}\left(\frac{1}{2} + \frac{\gamma}{2}\right)^{c^+_{+1}+c^-_{-1}} = 2^{-k}(1-\gamma)^{c^+_{-1}+c^-_{+1}}(1+\gamma)^{c^+_{+1}+c^-_{-1}}.$$

Thus, dividing through by $\beta/2^k$, Equation (7) becomes

$$\mathbb{1}[\mathbf{c} \in \mathsf{PosPtn}^L_k] \cdot 2 + ((1-\delta)c^-_{-1} - (1+\delta)c^+_{-1})\frac{X}{k} + ((1+\delta)c^-_0 - (1-\delta)c^+_0)\frac{Y}{k} \leq \beta^{-1}(1-\gamma)^{c^+_{-1}+c^-_{+1}}(1+\gamma)^{c^+_{+1}+c^-_{-1}} \quad \forall \mathbf{c} \in \mathsf{Ptn}^L_k. \quad (8)$$

Finally, we claim that Equation (8) is implied by the hypothesis. Indeed, we consider two cases. First, if $\mathbf{c} \in \mathsf{PosPtn}^L_k$, then $c^-_{-1} = c^-_0 = c^-_{+1} = 0$ and $c^+_0 = k - c^+_{-1} - c^+_{+1}$, so Equation (8) becomes

$$2 - (1+\delta)\frac{c^+_{-1}}{k}X - (1-\delta)\frac{k - c^+_{-1} - c^+_{+1}}{k}Y \leq \beta^{-1}(1-\gamma)^{c^+_{-1}}(1+\gamma)^{c^+_{+1}}.$$

This is precisely the second hypothesized inequality, for $c^+_{-1} = i, c^+_{+1} = j$. On the other hand, if $\mathbf{c} \notin \mathsf{PosPtn}^L_k$, then we observe that replacing $(c^+_{-1}, c^+_0, c^+_{+1}, c^-_{-1}, c^-_0, c^-_{+1}) \mapsto (0, 0, 0, c^-_{-1} + c^+_{-1}, c^-_0 + c^+_0, c^-_{+1} + c^+_{+1})$ fixes the RHS of Equation (8), while only increasing the LHS; thus, it suffices to prove Equation (8) only in this extreme case. Hence, we can assume $c^-_0 = k - c^-_{-1} - c^-_{+1}$, so Equation (8) becomes

$$(1-\delta)\frac{c^-_{-1}}{k}X + (1+\delta)\frac{k - c^-_{-1} - c^-_{+1}}{k}Y \leq \beta^{-1}(1-\gamma)^{c^-_{+1}}(1+\gamma)^{c^-_{-1}},$$

which is precisely the first assumed condition for $c^-_{-1} = j, c^-_{+1} = i$. ◀

▶ Remark 14. We chose the specific family of dual solutions used in the proof of Lemma 13 by inspecting an LP solver's output for $k = 2$ and $k = 3$. Our investigation also suggests that this solution is unique in a certain sense: In the simplest case of $k = 2$ and $\delta = \gamma = 0$, it appears that *every* optimal feasible solution requires $y^+_{-1} = 2/9$, and further, the only solution with only two nonzero y entries sets $y^+_0 = 1/9$.

References

1 Paola Alimonti. New local search approximation techniques for maximum generalized satisfiability problems. *Information Processing Letters*, 57(3):151–158, February 1996. Conference version in CIAC 1994. doi:10.1016/0020-0190(95)00196-4.

2 Paola Alimonti. Non-oblivious local search for MAX 2-CCSP with application to MAX DICUT. In Rolf H. Möhring, editor, *Graph-Theoretic Concepts in Computer Science*, Lecture Notes in Computer Science, pages 2–14. Springer, 1997. doi:10.1007/BFb0024483.

3 Sepehr Assadi, Gillat Kol, Raghuvansh R. Saxena, and Huacheng Yu. Multi-Pass Graph Streaming Lower Bounds for Cycle Counting, MAX-CUT, Matching Size, and Other Problems. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS 2020, Virtual, November 16-19, 2020)*, pages 354–364, virtual, November 2020. doi:10.1109/FOCS46700.2020.00041.

4 Sepehr Assadi and Vishvajeet N. Graph streaming lower bounds for parameter estimation and property testing via a streaming XOR lemma. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC 2021, Virtual, June 21-25, 2021)*, pages 612–625, virtual, June 2021. Association for Computing Machinery. doi:10.1145/3406325.3451110.

5 Amotz Bar-Noy and Michael Lampis. Online maximum directed cut. *Journal of Combinatorial Optimization*, 24(1):52–64, July 2012. Conference version in ISAAC 2009. doi:10.1007/s10878-010-9318-6.

6 Nico Bertram, Jonas Ellert, and Johannes Fischer. A Parallel Framework for Approximate Max-Dicut in Partitionable Graphs. In Christian Schulz and Bora Uçar, editors, *20th International Symposium on Experimental Algorithms (SEA 2022, Heidelberg, Germany, July 25-27, 2022)*, volume 233 of *LIPIcs*, pages 10:1–10:15, Heidelberg, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SEA.2022.10.

7 Joanna Boyland, Michael Hwang, Tarun Prasad, Noah Singer, and Santhoshini Velusamy. On sketching approximations for symmetric Boolean CSPs. In Amit Chakrabarti and Chaitanya Swamy, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX 2022, Virtual, September 19-21, 2022)*, volume 245 of *LIPIcs*, pages 38:1–38:23, virtual, July 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX/RANDOM.2022.38.

8 Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Near-optimal algorithms for maximum constraint satisfaction problems. *ACM Transactions on Algorithms*, 5(3):1–14, July 2009. Conference version in SODA 2007. doi:10.1145/1541885.1541893.

9 Lijie Chen, Gillat Kol, Dmitry Paramonov, Raghuvansh Saxena, Zhao Song, and Huacheng Yu. Towards Multi-Pass Streaming Lower Bounds for Optimal Approximation of Max-Cut. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms*, 2023.

10 Chi-Ning Chou, Alexander Golovnev, Amirbehshad Shahrasbi, Madhu Sudan, and Santhoshini Velusamy. Sketching Approximability of (Weak) Monarchy Predicates. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX 2022, Virtual, September 19-21, 2022)*, volume 245 of *LIPIcs*, pages 35:1–35:17, virtual, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX/RANDOM.2022.35.

11 Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, Ameya Velingker, and Santhoshini Velusamy. Linear Space Streaming Lower Bounds for Approximating CSPs. In *Proceedings of the 54th Annual ACM Symposium on Theory of Computing (STOC 2022, Rome, Italy, June 20-24, 2022)*, Rome, Italy, 2022. doi:10.1145/3519935.3519983.

12 Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, and Santhoshini Velusamy. Approximability of all Boolean CSPs with linear sketches. *arXiv*, February 2021. arXiv:2102.12351v7.

13 Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, and Santhoshini Velusamy. Approximability of all finite CSPs with linear sketches. In *Proceedings of the 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2021, Denver, CO, USA, February 7-10, 2022)*, Denver, CO, USA, 2021. IEEE Computer Society. doi:10.1109/FOCS52979.2021.00117.

14 Chi-Ning Chou, Alexander Golovnev, and Santhoshini Velusamy. Optimal Streaming Approximations for all Boolean Max-2CSPs and Max-kSAT. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS 2020, Virtual, November 16-19, 2020)*, pages 330–341, virtual, November 2020. IEEE Computer Society. doi:10.1109/FOCS46700.2020.00039.

15 Lars Engebretsen and Jonas Holmerin. More efficient queries in PCPs for NP and improved approximation hardness of maximum CSP. *Random Structures and Algorithms*, 33(4):497–514, December 2008. Conference version in STACS 2005. doi:10.1002/rsa.20226.

16 Uriel Feige and Michel X. Goemans. Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. In *Proceedings of the 3rd Israel Symposium on the Theory of Computing and Systems (ISTCS 2003, January 4-6, 1995)*, pages 182–189. IEEE Computer Society, 1995. doi:10.1109/ISTCS.1995.377033.

17 Uriel Feige and Shlomo Jozeph. Oblivious Algorithms for the Maximum Directed Cut Problem. *Algorithmica*, 71(2):409–428, February 2015. doi:10.1007/s00453-013-9806-z.

18 Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, November 1995. Conference version in STOC 1994. doi:10.1145/227683.227684.

19 Venkatesan Guruswami and Runzhou Tao. Streaming Hardness of Unique Games. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX 2019, Cambridge, MA, USA, September 20-22, 2019)*, volume 145 of *LIPIcs*, pages 5:1–5:12, Cambridge, MA, USA, September 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.5.

20 Venkatesan Guruswami, Ameya Velingker, and Santhoshini Velusamy. Streaming Complexity of Approximating Max 2CSP and Max Acyclic Subgraph. In Klaus Jansen, José D. P. Rolim, David Williamson, and Santosh S. Vempala, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX 2017, Berkeley, CA, USA, August 16-18, 2017)*, volume 81 of *LIPIcs*, pages 8:1–8:19, Berkeley, CA, USA, August 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX-RANDOM.2017.8.

21 Eran Halperin and Uri Zwick. Combinatorial approximation algorithms for the maximum directed cut problem. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2001, Washington, DC, USA, January 7-9, 2001)*, pages 1–7, Washington, DC, USA, 2001.

22 Gustav Hast. Approximating Max kCSP Using Random Restrictions. In Klaus Jansen, Sanjeev Khanna, José D. P. Rolim, and Dana Ron, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX 2004, Cambridge, MA, USA, August 22-24, 2004)*, volume 3122 of *LNCS*, pages 151–162, Cambridge, MA, USA, 2004. Springer. doi:10.1007/978-3-540-27821-4_14.

23 Gustav Hast. Approximating Max kCSP – Outperforming a Random Assignment with Almost a Linear Factor. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming (ICALP 2005, July 11-15, 2005)*, volume 3580 of *LNCS*, pages 956–968. Springer, 2005. doi:10.1007/11523468_77.

24 Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001. doi:10.1145/502090.502098.

25 Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM*, 50(6):795–824, November 2003. doi:10.1145/950620.950621.

26 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Streaming lower bounds for approximating MAX-CUT. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015, San Diego, California, USA, January 4-6, 2015)*, pages 1263–1282, San Diego, California, USA, January 2015. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611973730.84.

27 Michael Kapralov, Sanjeev Khanna, Madhu Sudan, and Ameya Velingker. $(1 + \omega(1))$-approximation to MAX-CUT requires linear space. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017, Barcelona, Spain, January 16-19, 2017)*, pages 1703–1722, Barcelona, Spain, January 2017. Society for Industrial and Applied Mathematics. doi:10.5555/3039686.3039798.

28 Michael Kapralov and Dmitry Krachun. An optimal space lower bound for approximating MAX-CUT. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC 2019, Phoenix, AZ, USA, June 23-26, 2019)*, pages 277–288, Phoenix, AZ, USA, June 2019. Association for Computing Machinery. doi:10.1145/3313276.3316364.

29 Dmitry Kogan and Robert Krauthgamer. Sketching cuts in graphs and hypergraphs. In *Proceedings of the 6th Annual Conference on Innovations in Theoretical Computer Science (ITCS 2015, Rehovot, Israel, January 11-13, 2015)*, pages 367–376, Rehovot, Israel, 2015. Association for Computing Machinery. doi:10.1145/2688073.2688093.

30 Michael Lewin, Dror Livnat, and Uri Zwick. Improved Rounding Techniques for the MAX 2-SAT and MAX DI-CUT Problems. In William J. Cook and Andreas S. Schulz, editors, *Integer Programming and Combinatorial Optimization*, pages 67–82, 2002. doi:10.1007/3-540-47867-1_6.

31 Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: An approach based on strongly factor-revealing LPs. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC 2011, San Jose, CA, USA, June 6-8, 2011)*, pages 597–606, San Jose, CA, USA, June 2011. Association for Computing Machinery. doi:10.1145/1993636.1993716.

32 Jiří Matoušek and Bernd Gärtner. *Understanding and Using Linear Programming*. Universitext. Springer, Berlin; New York, 2007.

33 Shiro Matuura and Tomomi Matsui. 0.863-Approximation Algorithm for MAX DICUT. In Michel Goemans, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, editors, *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques (APPROX 2001, Berkeley, CA, USA, August 18-20, 2001)*, volume 2129 of *LNCS*, pages 138–146, Berlin, Heidelberg, 2001. Springer. doi:10.1007/3-540-44666-4_17.

34 Alex Samorodnitsky and Luca Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC 2000, Portland, OR, USA, May 21-23, 2000)*, pages 191–199, Portland, OR, USA, 2000. Association for Computing Machinery. doi:10.1145/335305.335329.

35 Alex Samorodnitsky and Luca Trevisan. Gowers Uniformity, Influence of Variables, and PCPs. *SIAM Journal on Computing*, 39(1):323–360, January 2009. Conference version in STOC 2006. doi:10.1137/070681612.

36 Raghuvansh R. Saxena, Noah Singer, Madhu Sudan, and Santhoshini Velusamy. Improved streaming algorithms for Maximum Directed Cut via smoothed snapshots. In *63rd Annual Symposium on Foundations of Computer Science*, Santa Cruz, CA, USA, 2023. IEEE Computing Society. To appear.

37 Raghuvansh R. Saxena, Noah Singer, Madhu Sudan, and Santhoshini Velusamy. Streaming complexity of CSPs with randomly ordered constraints. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2023, Florence, Italy, January 22-25, 2023)*, Florence, Italy, January 2023.

38 Noah Singer. *On Streaming Approximation Algorithms for Constraint Satisfaction Problems*. Undergraduate thesis, Harvard University, Cambridge, MA, March 2022.

39 Noah Singer, Madhu Sudan, and Santhoshini Velusamy. Streaming approximation resistance of every ordering CSP. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX 2021, Virtual, August 16-18, 2021)*, volume 207 of *LIPIcs*, pages 17:1–17:19, virtual, July 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX/RANDOM.2021.17.

40 David Steurer. Fast SDP Algorithms for Constraint Satisfaction Problems. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010, Austin, TX, USA, January 17-19, 2010)*, pages 684–697, Austin, TX, USA, January 2010. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611973075.56.

41 Madhu Sudan. Streaming and Sketching Complexity of CSPs: A survey (Invited Talk). In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming*, volume 229 of *LIPIcs*, pages 5:1–5:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.5.

42 Madhu Sudan and Luca Trevisan. Probabilistically checkable proofs with low amortized query complexity. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (SFCS 1998, Palo Alto, CA, USA, November 8-11, 1998)*, pages 18–27, Palo Alto, CA, USA, 1998. IEEE Computer Society. doi:10.1109/SFCS.1998.743425.

43 Luca Trevisan. Parallel Approximation Algorithms by Positive Linear Programming. *Algorithmica*, 21(1):72–88, May 1998. doi:10.1007/PL00009209.

44 Luca Trevisan. Recycling queries in PCPs and in linearity tests. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC 1998, Dallas, Texas, USA, May 24-26, 1998)*, pages 299–308, Dallas, Texas, USA, 1998. Association for Computing Machinery. doi:10.1145/276698.276769.

45 Luca Trevisan, Gregory B. Sorkin, Madhu Sudan, and David P. Williamson. Gadgets, Approximation, and Linear Programming. *SIAM Journal on Computing*, 29(6):2074–2097, January 2000. Conference version in FOCS 1996. doi:10.1137/S0097539797328847.

46 Zhenning Zhang, Donglei Du, Chenchen Wu, Dachuan Xu, and Dongmei Zhang. A spectral partitioning algorithm for maximum directed cut problem. *Journal of Combinatorial Optimization*, 42(3):373–395, October 2021. Conference version in COCOA 2017. doi:10.1007/s10878-018-0369-4.

47 Uri Zwick. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1998, San Francisco, CA, USA, January 25-27, 1998)*, pages 201–210, San Francisco, CA, USA, 1998. Association for Computing Machinery. doi:10.5555/314613.314701.

A 10/7-Approximation for Discrete Bamboo Garden Trimming and Continuous Trimming on Star Graphs

Felix Höhne ✉
Fraunhofer Institute for Industrial Mathematics ITWM, Kaiserslautern, Germany

Rob van Stee ✉
University of Siegen, Germany

Abstract

In the discrete bamboo garden trimming problem we are given n bamboo that grow at rates $v_1, \dots, v_n$ per day. Each day a robotic gardener cuts down one bamboo to height 0. The goal is to find a schedule that minimizes the height of the tallest bamboo that ever exists.

We present a 10/7-approximation algorithm that is based on a reduction to the pinwheel problem. This is consistent with the approach of earlier algorithms, but some new techniques are used that lead to a better approximation ratio.

We also consider the continuous version of the problem where the gardener travels in a metric space between plants and cuts down a plant each time he reaches one. We show that on the star graph the previously proposed algorithm *Reduce-Fastest* is a 6-approximation and the known *Deadline-Driven Strategy* is a $(3 + 2\sqrt{2})$-approximation. The Deadline-Driven Strategy is also a $(9 + 2\sqrt{5})$-approximation on star graphs with multiple plants on each branch.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases bamboo garden trimming, approximation algorithms, scheduling

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.16

Category APPROX

Supplementary Material *Software*: `https://github.com/Felixhhne/bamboo`
archived at `swh:1:dir:dc21af15b5e2d50d6b27b4b6f8fce8de22b6e9cb`

1 Introduction

In the discrete bamboo garden trimming problem (BGT), first introduced by Gasieniec et al. [8] we are given a set of n bamboo that grow at rates $v_1, \dots, v_n$ per day, i.e. the height of bamboo i extends by v_i each day. We assume that these growth rates are arranged such that $v_1 \geq v_2 \geq \dots \geq v_n$. Initially the height is set to zero. Each day a robotic gardener cuts down one bamboo to height zero. The goal is to design a trimming schedule such that the height of the tallest bamboo is minimized. Gasieniec et al. [8] gave a 2-approximation for discrete BGT which has been improved by van Ee [14] to a $\frac{12}{7}$-approximation.

Both results are obtained by reducing BGT to the pinwheel scheduling problem. The pinwheel scheduling problem is motivated by the communication between a satellite and its ground station. The ground station wants to receive messages from n satellites. Time is slotted and a satellite i sends a message for p_i consecutive timeslots before switching to a different message. Each timeslot the ground station receives a message from a single satellite. This means in order to guarantee that no message is missed we need to find a schedule that allocates at least one timeslot to satellite i in any interval of p_i units of time. We discuss the literature on pinwheel scheduling in Section 2.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 16; pp. 16:1–16:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

If a BGT algorithm maintains a height of K then each bamboo i must be visited at least once in a period of $p_i^* := \lfloor \frac{K}{v_i} \rfloor$ time steps. That is, there is a BGT schedule that maintains height K for the input $(v_1, \ldots, v_n)$ if and only if the pinwheel problem $(p_1^*, \ldots, p_n^*)$ is schedulable.

There are also a number of results that do not use the connection to the pinwheel problem, using various greedy-type algorithms. The *Deadline-Driven Strategy* always cuts the bamboo with the earliest deadline provided that the height of this bamboo has reached a certain threshold. The deadline of a bamboo is the time it reaches the height the algorithm wants to maintain. This algorithm has already been considered for discrete BGT and there it is a 2-approximation as shown by J. Kuszmaul [12].

A second simple algorithm is *Reduce-Fastest* which is a 2.62-approximation for discrete BGT as shown by Bilò et al. [2] This algorithm always cuts the fastest plant which has reached a certain threshold. A similar algorithm is *Reduce-Max* that always cuts the highest bamboo and is a 4-approximation. Both of these algorithms are online algorithms based on simple queries. This means they are flexible and can easily adapt to changes in the input (the set of growth rates).

In the first part of this paper we present an algorithm based on a pinweel reduction that improves the approximation ratio for discrete BGT to $\frac{10}{7}$. This algorithm combines a binary search with the known technique of porous schedules for pinwheel scheduling. This marks the first time that porous schedules are used to approximate the BGT problem.

In the second part, we consider the continuous version of the BGT problem, also introduced by Gasieniec et al. [8]. In this version the bamboo are located in some metric space and the gardener needs to travel between the bamboo to cut them. Cutting is done instantly and the goal is to find a route that minimizes the maximum height of the bamboos. Gasieniec et al. give a $O(\log n)$-approximation algorithm that works in any metric space.

We consider the case where the underlying metric is a star graph. In this context we study some of the simple algorithms above (compared to the relative complicatedness of a pinwheel schedule) that achieve constant approximation ratios for this case.

1.1 Our results

For discrete BGT we propose a 10/7-approximation algorithm that is based on a reduction to the pinwheel-problem. Previous work used only the sum of the speeds H as a lower bound on the optimum value. We improve on this by using binary search in the interval $[H, 2H]$ and by planning to lose a certain factor in advance. By this we mean that all periods are multiplied by our goal ratio after calculating them from the speeds, making it easier to find a pinwheel schedule. We can show that our algorithm always finds a schedule of height at most $10K/7$ if a schedule of height K exists. This is achieved by using the technique of porous schedules which we will discuss in Section 2. Thus, the binary search is used to essentially find the best possible lower bound for the optimal value within this framework.

The result can be improved to $\frac{7}{5}$ using a computer-assisted proof. Unfortunately the proof is too long to fit into the appendix but can be found under the link `https://github.com/Felixhhne/bamboo`

In the continuous version of the problem we consider a star graph. We show that the 2.62-approximate algorithm Reduce-Fastest for discrete BGT, which can be seen as a special instance of a star graph, still works on arbitrary star graphs with one bamboo on each branch and we show that it is a 6-approximation in this case.

Furthermore the deadline driven algorithm gives us a $(3+2\sqrt{2})$-approximation on the star. This result can be extended to the case where there are multiple bamboo on each branch of the star. Here we pay a price in the approximation ratio and achieve a $(9+2\sqrt{5})$-approximation.

1.2 Related work

Both the discrete and continuous version of BGT were first introduced by Gasieniec et al. [8]. They provide a variety of results. The first algorithm they present is Reduce-Fastest, which in each step cuts the fastest growing bamboo above a certain height treshhold. This algorithm is a 2.62-approximation as shown by Bilò el al. [2]. A similar algorithm is Reduce-Max that always cuts the highest bamboo and is a 4-approximation. The final algorithm in this set of algorithms based on simple queries is the Deadline-Driven Strategy which is a 2-approximation. The results for Reduce-Max as well as the Deadline-Driven Strategy are from J. Kuszmaul [12].

Gasieniec et al. also give a fully offline 2-approximation algorithm that preprocesses the input and reduces the problem to a pinwheel-instance. This approach has been improved by van Ee [14] to a $\frac{12}{7}$-approximation.

There are other problems where the goal is to minimize the maximum height or backlog reached on a machine. One example is the Minimum Backlog Problem [3, 1, 13] where an adversary distributes water among a set of cups and the player may empty one or more cups on their turn.

In the problem of Buffer minimization with conflicts [5] there is a set of machines on a graph and load may arrive on these machines at any time. The algorithm runs machines to decrease their load but machines that are adjacent to each other on the graph may not run at the same time.

2 Pinwheel problems

We represent an instance of the pinwheel scheduling problem by the vector $p = (p_1, \ldots, p_n)$ of periods with which each plant should be cut (or: each machine should be scheduled).

▶ **Definition 1.** *The* density *of a pinwheel scheduling instance $p = (p_1, \ldots, p_n)$ is $d(p) = \sum_{i=1}^{n} \frac{1}{p_i}$.*

For example, the instance $(2, 3)$ has density $\frac{5}{6}$ and can be scheduled by repeating the sequence 12. Because 121212 . . . is the *only* feasible schedule, the instance $(2, 3, p_3)$ can not be scheduled regardless of the value of p_3.

The pinwheel problem was first introduced by Holte et al. [11] and then picked up by Chan and Chin [4]. They conjecture that any pinwheel instance with density up to 5/6 can be scheduled. The above example shows that a better guarantee is not possible. This conjecture is supported by a variety of works, including Fishburn and Lagarias [7] who show that any instance with $p_1 = 2$ and density up to 5/6 can be scheduled. Dei Wing [6] shows that the claim also holds for low-dimensional vectors with dimension up to 5 and more recently Gasieniec et al. [9] improve this result further by proving the claim for instances with up to 12 elements. Additionally they give a set of schedules that solve all schedulable instances with at most 5 tasks.

Without loss of generality, the elements of the pinwheel instance are in nondecreasing order. An obvious requirement for schedulability is $d(p) \leq 1$. Hence, aside from the instance (1) an instance can only be scheduled if $p_1 \geq 2$. Consequently we assume $2 \leq p_1 \leq p_2 \leq \cdots \leq p_n$.

Given a pinwheel instance $p = (p_1, \ldots, p_n)$, a successful schedule for p is an infinite sequence over $\{1, \ldots, n\}$ such that any subsequence of length p_i contains at least one i. An important consideration for pinwheel scheduling and BGT is the representation of a solution. This is because in general an explicit representation of the schedule may take exponential space. The solutions for pinwheel scheduling provided by Fishburn and Lagarias as well as

Chan and Chin come in the form of *fast online schedulers*. These are algorithms that can decide whether they can schedule an instance in polynomial time and then generate each symbol of the schedule in constant time.

Porous schedules

Fishburn and Lagarias introduced the concept of porous schedules [7] as a useful tool in the construction of pinwheel schedules.

Let $p = (p_1, \ldots, p_n)$. Let $\mathcal{S}$ be a subset of $\{1, \ldots, n\}$ and define $\mathcal{U} = \{1, \ldots, n\} \setminus \mathcal{S}$. A *porous schedule* for $\mathcal{S}$ (the set of *scheduled* tasks) is a schedule that allocates a slot for plant $i \in \mathcal{S}$ at least once every p_i positions, but also contains slots that are not allocated to any plant. When writing out the schedule the first type of slot is denoted by i and the second type, which we also call a hole, is denoted by _. In this paper we will only consider schedules that consist of infinite repeats of finite lists, and describe such a schedule by giving only that list. Even that part may have exponential size as a function of the size of the input.

Let

$$D_{\mathcal{U}} = 1 - \sum_{k \notin \mathcal{U}} \frac{1}{p_k^*}$$

be the maximum possible density of the unscheduled machines.

For example, given the instance (2,4,8,9), we could set $\mathcal{U} = \{3, 4\}$, which corresponds to the (unscheduled) plants with speed 8 and 9, and $\mathcal{S} = \{1, 2\}$, which corresponds to the plants with speed 2 and 4. The schedule 12_ (meaning 12_12_12_...) is a porous schedule for the scheduled tasks $\mathcal{S}$, while $\mathcal{U}$ is the set of leftover plants that still need to be scheduled.

▶ **Definition 2.** *Let $p = (p_1, \ldots, p_n)$ be a pinwheel instance and let f be a porous schedule for $\mathcal{S} \subseteq \{1, \ldots, n\}$. Then for $i \in \mathcal{U} = \{1, \ldots, n\} \setminus \mathcal{S}$, we define h_i as the minimum number of holes in p_i consecutive positions of f.*

Sometimes we also write the function $h(x)$ to represent the minimum number of holes in x consecutive positions. The set $h = \{h_i | i \in \mathcal{U}\}$ is again a pinwheel instance unless there exists an i with $h_i = 0$. Of course, in general h might not be schedulable. Lemma 2 of [7] explains how solving the pinwheel problem h can lead to a solution of p. A version of it is given below.

▶ **Lemma 3.** *Consider a pinwheel instance p and let a porous schedule f for a subset $\mathcal{S}$ of the tasks. Let $\mathcal{U} := \{1, \ldots, n\} \setminus \mathcal{S}$. If $h = \{h_i | i \in \mathcal{U}\}$ is a schedulable pinwheel instance, then p is schedulable.*

Proof. By assumption, there is a schedule $g : \mathbb{Z} \to \mathcal{U}$ (that we can enumerate using a fast online scheduler) for the input h. Let β be an order preserving map from the set of holes in f to $\mathbb{Z}$. Let $f' = f$ except for the holes in f where $f'(k) = g(\beta(k))$. Then f' is a schedule for p. ◀

Let $b \in [0, 1]$. We say that b is a *density guarantee* for the pinwheel problem if all instances p with $d(p) \leq b$ can be scheduled. This is always taken to mean that there exists a schedule for p and we can efficiently find a fast online scheduler to generate the sequence. Our results rely on the following theorem.

▶ **Theorem 4** ([7]). *The value $3/4$ is a density guarantee for the pinwheel problem. For instances with $p_1 = 2$, the density guarantee is $5/6$. In both instances, schedules can be found in time $O(n^3)$.*

Note that solving or scheduling a pinwheel instance for our purposes means finding a fast online scheduler but does not include explicitly writing out the schedule (not even just the finite part that repeats). Chan and Chin give an algorithm that runs in time $O(n^3)$ and schedules any instance with density at most $\frac{7}{10}$. Their algorithm achieves the following guarantees based on p_1:

p_1	2	3	4	5	6	7	8	≥ 9
guarantee	0.75	0.70	0.708	0.721	0.733	0.738	0.744	> 0.75

In particular this means any instance with $p_1 \geq 9$ and density at most 0.75 can be scheduled in time $O(n^3)$. Fishburn and Lagarias find schedules for the remaining cases with small values for p_1 and density up to 0.75 based on the following idea.

They find a porous schedule for the first few elements and they extend this schedule using the results from Chan and Chin and Lemma 3 to find a schedule for the remaining machines that fits into the holes. For example for the instance $(3, 4, p_3, \dots)$ we can create a porous schedule 12_ as discussed above which has a hole at every third position. The remaining machines after removing machines 1 and 2 have density at most $\frac{3}{4} - \frac{1}{3} - \frac{1}{4} = \frac{1}{6}$, thus $p_i \geq 6$ for $i \geq 3$. For this pattern of holes we get the values for h_i by dividing the remaining periods by 3. For $p_3 = 6, 7, 8$ we get $h_3 = 2$ which increases the density by a factor of not more than 4, thus the density of the resulting instance is at most $\frac{1}{6} \cdot 4 < \frac{3}{4}$ with the first element of the new instance being 2. By the table of Chan and Chin this is schedulable. A similar argument holds for $p_3 \geq 9$ with the density of the new schedule being not more than $\frac{7}{10}$. They further refine this approach in the paper.

3 A 10/7-approximation for discrete BGT

We claim that $H = \sum_{i=1}^{n} v_i$ is a lower bound on the optimal value. As long as all bamboo have height at most $H' < H$ the sum of all bamboo heights increases by at least $H - H' > 0$ each step until it eventually exceeds nH'. Then there must be a bamboo with height more than H'. On the other hand, there is an algorithm that produces a schedule of height $2H$ [8]. Thus, the optimal value is indeed somewhere in the interval $[H, 2H]$.

Our algorithm is based on the following key result.

▶ **Lemma 5.** *Consider a pinwheel instance p and a porous schedule f for a subset $\mathcal{S}$ of the tasks. Let $\mathcal{U} := \{1, \dots, n\} \setminus \mathcal{S}$. If $\frac{h_i}{p_i} \geq D_{\mathcal{U}}$ for each $i \in \mathcal{U}$ in a porous schedule for $\mathcal{S}$ and $\frac{1}{p_i} \leq \frac{3}{4p_i^*}$ for all $i \in \mathcal{U}$, then $(p_1, \dots, p_n)$ is schedulable for* Alg.

Proof. If $\frac{h_i}{p_i} \geq D_{\mathcal{U}}$ then

$$\sum_{i \in \mathcal{U}} \frac{1}{h_i} \leq \frac{1}{D_{\mathcal{U}}} \sum_{i \in \mathcal{U}} \frac{1}{p_i} \leq \frac{1}{D_{\mathcal{U}}} \frac{3}{4} D_{\mathcal{U}} = \frac{3}{4}.$$

Thus the instance is schedulable by Lemma 3 and by Theorem 4. ◀

Our algorithm begins by defining periods for the pinwheel problem based on the growth speeds. These periods are then multiplied by a certain factor to get an instance that is easier to schedule. It would be easiest if we could multiply all periods by 4/3. That way we would get an instance with density at most 3/4 which is therefore schedulable by Theorem 4. However, the input for the pinwheel problem must be a set of natural numbers. This means that we must round down after scaling the periods. Even when choosing a scaling factor larger than 4/3 (in our case, 10/7), after rounding down there can still be some periods that become too small, resulting in a density that is too high. This means that we cannot rely on the previous work as a black box. Typically, there will be a few difficult periods remaining. For instance, the period 4 becomes $\lfloor \frac{10}{7} \cdot 4 \rfloor = 5$ after scaling, and 5 becomes 7. Typically, longer periods are easier to handle as the condition $\lfloor \frac{10}{7} p^* \rfloor \geq \frac{4}{3} p^*$ is often satisfied.

We overcome this difficulty by finding a porous schedule for the few difficult periods by hand and then showing that the remaining periods can be scheduled in the holes. To do this we need to consider some inputs for the pinwheel problem that were not considered in previous works, since (as discussed) the overall density of the instance may still be above 3/4 at this point. Thus we need to find a porous schedule for the cases not covered by the table of Chan and Chin and a schedule for the remaining machines.

Algorithm 1 Binary Search for BGT.

```
Let $H = \sum_{i=1}^{n} v_i$ and $R = \frac{10}{7}$. Using binary search in the interval $[H, 2H]$ find the
  smallest $K$ such that the following procedure returns a valid schedule and return
  this schedule.
1. Given $K$ define $p_i^* = \lfloor \frac{K}{v_i} \rfloor$ and $p_i = \lfloor R p_i^* \rfloor$.
2. Solve the pinwheel problem $(p_1, \ldots, p_n)$.
```

In order to analyze Algorithm 1, we first show that whenever there exists a schedule for the pinwheel problem p^*, then the algorithm can find a schedule for p.

Assume that there exists a schedule for p^*. Then $d(p^*) \leq 1$. Whenever $\frac{p_i^*}{p_i} \leq \frac{3}{4}$ for all i we have $d(p) \leq \frac{3}{4}$ which means p is schedulable.

We can find schedules for cases that contain i with $\frac{p_i}{p_i^*} > \frac{3}{4}$ by assigning those periods to the subset $\mathcal{S}$ and applying Lemma 5. We create a porous schedule for $\mathcal{S}$ and define h_i for $i \in \mathcal{U} := \{1, \ldots, n\} \setminus \mathcal{S}$ as the minimum number of holes in p_i consecutive positions of the porous schedule. In some cases, the porous schedule is completely regular, with holes in every k-th location (and only there). In that case we get

$$\frac{h_i}{p_i} = \frac{\lfloor \frac{1}{k} \lfloor \frac{10}{7} p_i^* \rfloor \rfloor}{\lfloor \frac{10}{7} p_i^* \rfloor}$$

and for Lemma 5 we need to show that $h_i / p_i \geq D_{\mathcal{U}}$. If we write $p_i^* = 7a + b$ for some values a, b then this means showing that

$$\frac{5a + \lfloor \frac{1}{k} \lfloor \frac{10}{7} b \rfloor \rfloor}{10a + \lfloor \frac{10}{7} b \rfloor} \geq D_{\mathcal{U}}.$$

▶ **Lemma 6.** *If there is a schedule of height K then the procedure in Algorithm 1 finds a schedule of height at most $\frac{10}{7}K$.*

Proof. Let $p_i^* = \lfloor \frac{K}{v_i} \rfloor$. Assume there is a schedule of height K. Then there is a solution to the pinwheel problem $p^* = (p_1^*, \ldots, p_n^*)$ and thus $\sum_{i=1}^{n} \frac{1}{p_i^*} \leq 1$. If $p_1^* = 1$ then there is only one plant and the schedule is trivial. Thus we consider $p_1^* \geq 2$. The algorithm calculates the periods $p_i = \lfloor \frac{10 p_i^*}{7} \rfloor$ and solves the resulting pinwheel problem. Because of the definition of p_i^* the resulting schedule has height at most $\frac{10K}{7}$.

We show that if a schedule of height K exists, there is always a schedule for the pinwheel problem of Alg. If $p_i^* \geq 11$ then $\frac{1}{p_i}$ is smaller then $\frac{1}{p_i^*}$ by a factor of at least $\frac{3}{4}$ since

$$\frac{1}{p_i} = \frac{1}{\lfloor 10/7 p_i^* \rfloor} \leq \frac{1}{(10/7 p_i^* - 1)} \leq \frac{3}{4} \frac{1}{p_i^*} \text{ for } p_i^* \geq 11.$$

The same is true for every other value of p_i^* except $p_i^* = 2$ and $p_i^* = 4$ which can easily be verified. Thus if there is no p_i^* with value 2 or 4 the density of the pinwheel problem is at most 3/4 and it can be scheduled. We now consider all cases where these values do occur and use Lemma 5 to show that the pinwheel problem can be solved.

We denote these cases by listing the initial values of the instance p^*. For example, the notation 4,4,5 means that we are considering the case where $p_1^* = p_2^* = 4$ and $p_3^* = 5$. In that case $p = (5, 5, 7, \dots)$ with zero or more plants after the first three.

For each case, we create a porous schedule for a subset $\mathcal{S}$ of periods in p. We then need to show that $\frac{h_i}{p_i} \geq D_{\mathcal{U}}$ as well as $\frac{1}{p_i} \leq \frac{3}{4p_i^*}$ holds for all periods $\mathcal{U}$ that are not yet scheduled by the porous schedule. If that is the case, then p is schedulable by Lemma 5.

2 We have $p_i/p_i^* \geq 6/5$ for all p_i. This can easily be verified for $p_i^* = 2$ and $p_i^* = 4$ and it holds for all other periods because for those periods we have $p_i/p_i^* \geq 4/3$. This means the density in the pinwheel problem of Alg is at most 5/6. Since $p_1 = 2$ this problem can be solved.

$3, 3, 4$ and $3, 4, 4$ In this case there are only three plants and the schedule is 123.

$3, 4, p_3^* \geq 5$ Then $p = (4, 5, p_3 \geq 7, \dots)$. Alg schedules plants 1 and 2 using the schedule 1_2_. Then $D_{\mathcal{U}} = 5/12$ and $\frac{h_i}{p_i} = \frac{\lfloor \frac{1}{2} \lfloor \frac{10}{7} p_i^* \rfloor \rfloor}{\lfloor \frac{10}{7} p_i^* \rfloor}$. Let $p^* = 7a + b$. Then $\frac{h_i}{p_i} = \frac{5a + \lfloor \frac{1}{2} \lfloor \frac{10}{7} b \rfloor \rfloor}{10a + \lfloor \frac{10}{7} b \rfloor}$.

If $\frac{5a + \lfloor \frac{1}{2} \lfloor \frac{10}{7} b \rfloor \rfloor}{10a + \lfloor \frac{10}{7} b \rfloor} \geq D_{\mathcal{U}}$ then $\frac{5(a+1) + \lfloor \frac{1}{2} \lfloor \frac{10}{7} b \rfloor \rfloor}{10(a+1) + \lfloor \frac{10}{7} b \rfloor} \geq D_{\mathcal{U}}$ since $\frac{5}{10} > D_{\mathcal{U}}$.

Therefore if the inequality holds for a particular p_i^*, then it also holds for all subsequent p_i^* that have the same remainder after division by 7. In particular, this means that if there are 7 consecutive periods p_i^* with $h_i/p_i \geq D_{\mathcal{U}}$ then $h_i/p_i \geq D_{\mathcal{U}}$ also holds for all subsequent periods p_i^*.

We can determine that $h_i/p_i \geq D_{\mathcal{U}}$ holds for $p_i^* \geq 5$ by looking at enough periods (in this case periods 5 to 11). This argument is repeated in many of the subsequent cases. Appendix A contains tables that show the values for each case.

Furthermore $\frac{1}{p_i} \leq \frac{3}{4p_i^*}$ holds for $p_i^* \geq 5$. This means we can schedule this case.

$4, 4$ Then $p = (5, 5, \dots)$. Alg schedules plants 1 and 2 using the schedule 1_2__. Then $D_{\mathcal{U}} = 1/2$. For $p^* = 7a + b$ we get $\frac{h_i}{p_i} = \frac{6a + h(\lfloor \frac{10}{7} b \rfloor)}{10a + \lfloor \frac{10}{7} b \rfloor}$. By looking at 7 periods we can determine that $h_i/p_i \geq D_{\mathcal{U}}$ holds for $p_i^* \geq 4$.

Since $\frac{1}{p_i} \leq \frac{3}{4p_i^*}$ holds for $p_i^* \geq 5$ but not $p_i^* = 4$ we still need to consider the case where $p_1^* = p_2^* = p_3^* = 4$ but we can schedule this case for $p_3^* \geq 5$.

$4, 4, 4$ Then $p = (5, 5, 5, \dots)$. Alg schedules plants 1 to 3 using the schedule 123__. Then $D_{\mathcal{U}} = 1/4$. Using the same approach as in the previous case we can determine that $h_i/p_i \geq D_{\mathcal{U}}$ holds for $p_i^* \geq 4$. As in the previous case this means we can schedule all cases except the case where $p_1^* = \dots = p_4^* = 4$ since in this case $\frac{1}{p_i} \leq \frac{3}{4p_i^*}$ is not guaranteed. However in this case there are only four plants and therefore this case can be scheduled as well.

In all subsequent cases, all periods except the first are greater than 4. This means $\frac{1}{p_i} \leq \frac{3}{4p_i^*}$ holds for all periods after the first and we only need to verify $h_i/p_i \geq D_{\mathcal{U}}$. We get the following results.

p^*	p	Schedule	$D_{\mathcal{U}}$	p_i^*	h_i/p_i
$4,5,\geq 9$	$5,7$	1_2__	$11/20$	–	–
$4,5,6$	$5,7,8$	31__2 _13__ 21__3 _12__	$23/60$	$14a+b$	$\frac{10a+h(\lfloor\frac{10}{7}b\rfloor)}{20a+\lfloor\frac{10}{7}b\rfloor}$
$4,5,6,6$	$5,7,8,8$	see 3,4,5			
$4,5,7$	$5,7,10$	see next line			
$4,5,8$	$5,7,11$	1_3__ 12___ 1_32_ 1___2	$17/40$	$14a+b$	$\frac{11a+h(\lfloor\frac{10}{7}b\rfloor)}{20a+\lfloor\frac{10}{7}b\rfloor}$
$4,6,\geq 9$	$5,8$	1____ 1_2__ 1___2	$7/12$	$21a+b$	$\frac{20a+h(\lfloor\frac{10}{7}b\rfloor)}{30a+\lfloor\frac{10}{7}b\rfloor}$
$4,6,6$	$5,8,8$	see next line			
$4,6,7$	$5,8,10$	1__3_ 1_2__ 13__2	$37/84$	$21a+b$	$\frac{16a+h(\lfloor\frac{10}{7}b\rfloor)}{30a+\lfloor\frac{10}{7}b\rfloor}$
4,6,8	5,8,11	1__3_ 1_2__ 1__32 1____ 1_23_ 1___2	$\frac{11}{24}$	$21a+b$	$\frac{17a+h(\lfloor\frac{10}{7}b\rfloor)}{30a+\lfloor\frac{10}{7}b\rfloor}$
4,6,8,8	5,8,11,11	1__3_ 1_24_ 1__32 1__4_ 1_23_ 1__42	$\frac{1}{3}$	$21a+b$	$\frac{14a+h(\lfloor\frac{10}{7}b\rfloor)}{30a+\lfloor\frac{10}{7}b\rfloor}$
$4,7$	$5,10$	see next line			
$4,8$	$5,11$	1_2__1____	$5/8$	$7a+b$	$\frac{7a+h(\lfloor\frac{10}{7}b\rfloor)}{10a+\lfloor\frac{10}{7}b\rfloor}$
$4,\geq 9$	$5,\geq 12$	1____	$3/4$	$7a+b$	$\frac{8a+\lfloor\frac{4}{5}\lfloor\frac{10}{7}b\rfloor\rfloor}{10a+\lfloor\frac{10}{7}b\rfloor}$

◀

▶ **Theorem 7.** *Algorithm 1 is a polynomial-time $\frac{10}{7}$-approximation.*

Proof. The approximation ratio follows directly from Lemma 6. Since any height K that is at least the optimal height is schedulable, the procedure in algorithm 1 finds a valid schedule of height $\frac{10}{7}K$ for any height that is at least OPT. This means the binary search settles on a value that is at most OPT (since it is integer) and the algorithm returns a schedule of height at most $\frac{10}{7}$ times that value.

The porous schedules are given in the paper of Fishburn and Lagarias (in particular see table 1 of [7]) and are thus available in $O(1)$. Then the algorithm in [4] with runtime $O(n^3)$ is used to schedule the remaining machines. This means pinwheel instances with density at most 0.75 can be solved in time $O(n^3)$. In addition, as mentioned above Fishburn and Lagarias show that instances with $p_1 = 2$ and density up to $\frac{5}{6}$ can also be solved in time $O(n^3)$. ◀

4 Continuous BGT on a star

We now examine the continuous case of the BGT-problem. For this problem Gasieniec et al. propose a $O(\log n)$-approximation algorithm that works in any metric space. We consider the star graph and show that many of the constant approximation factor algorithms for discrete BGT can be extended to this case with only small losses in the approximation ratio.

4.1 Notation

Plants $1, \ldots, n$ grow at the end of each of n branches on a star graph. We define d_i as the time it takes the server to travel to plant i cut it and return to the center. The growth rate of plant i is denoted by h_i. Each *round* the server visits a plant and returns back to the center.

A *schedule* for continuous BGT on a star graph, is an infinite sequence over $\{1,\ldots,n\}$, that describes the order plants are visited in. While BGT is an infinite problem, it is sufficient to consider cyclic schedules. This is because any schedule that maintains a finite maximum height of a must visit each plant i at least once in $\frac{a}{h_i b}$ rounds, where b is the minimum distance needed to visit a plant. This means a schedule for BGT (on a star) solves the pinwheel problem with periods $\frac{a}{h_i b}$. Since any pinwheel schedule is cyclic (see Theorem 2.1 in [10]) the same is true for BGT schedules.

4.2 A lower bound for the star

Consider a cyclic schedule of length L. Let m_i be the amount of times plant i is visited in a segment of length L. Then $\frac{L}{m_i}$ is the average period between visits of i.

▶ **Lemma 8.** *The average height of p after a cut is $h_p(d_p + \sum_{i\neq p} \frac{m_i}{m_p} d_i)$.*

Proof. Consider a plant p. Let $\lambda_1,\ldots,\lambda_{m_p}$ be the heights plant p reaches in the schedule. The sum of all heights is the same as the sum of all distances that are traversed times the speed of the plant, i.e. $\sum_{k=1}^{m_p} \lambda_k = h_p \sum_{i=1}^{n} d_i m_i$. The average height is

$$\frac{1}{m_p}\sum_{k=1}^{m_p}\lambda_k = \frac{1}{m_p} h_p \sum_{i=1}^{n} d_i m_i = h_p(d_p + \sum_{i\neq p}\frac{m_i}{m_p} d_i).$$ ◀

Naturally the average height of p is a lower bound on the maximum height of p. This inspires the following model: The algorithm may visit each plant i at a certain fixed period f_i. These periods do not need to match a feasible schedule and may even be fractional. We only require that $\sum_{i=1}^{n} \frac{1}{f_i} = 1$ which means that the frequencies add up to one plant visited per round. The height a plant p reaches in this model is defined as $h_p(d_p + \sum_{i\neq p} \frac{f_p}{f_i} d_i)$.

By setting $f_i = \frac{L}{m_i}$ we get a solution for the new model with a height that is equal to the average height of a schedule in the original model and not more than the maximum height. This means a lower bound on the height in the new model is also a lower bound on the average as well as maximum height for continuous BGT on a star graph.

▶ **Lemma 9.** *$R \coloneqq \sum_{i=1}^{n} h_i d_i$ is a lower bound on the optimal height for continuous BGT on a star graph.*

Proof. Consider the new model. We set $K = \sum_{i=1}^{n} h_i$ and $f_i = \frac{K}{h_i}$. Then $\sum_{i=1}^{n} \frac{1}{f_i} = 1$ and this is a valid solution to the new model. Then the height of plant p is $h_p(d_p + \sum_{i\neq p} \frac{f_p}{f_i} d_i) = h_p(d_p + \sum_{i\neq p} \frac{h_i}{h_p} d_i) = \sum_{i=1}^{n} h_i d_i = R$.

It follows that all plants reach a height of R. Because of the requirement $\sum_{i=1}^{n} \frac{1}{f_i} = 1$ it is not possible to reach a maximum height lower than R, since visiting one plant more often would mean visiting other plants less often. This means R is a lower bound on the maximum height in the new model and therefore also a lower bound on the maximum height for continuous BGT on a star graph. ◀

4.3 Algorithms on the star

The algorithm Reduce-Fastest(x), introduced by Gasieniec et al. [8], cuts the next fastest growing bamboo among those with height at least $x \cdot R$. The proof of the following lemma is structured in a way similar to the proof by J. Kuszmaul for discrete BGT [13]; see the appendix.

▶ **Lemma 10.** *Reduce-Fastest*$(2(R + Dh_{max}))$ *is a 6-approximation on the star.*

Proof. Assume that at some point there is a bamboo b_i that reaches height $3(R + Dh_{max})$. Let t_1 be the most recent time b_i reaches height $2(R + Dh_{max})$ and t_3 the time it reaches height $3(R + Dh_{max})$. Furthermore let t_2 be the first time in between t_1 and t_2 where the gardener visits the center. Since it takes at most distance D to visit a plant and return to the center the height of b_i is at most $2R + 3Dh_{max}$ at this time. We consider the set S of bamboo that are cut at least once during the time interval $[t_2, t_3)$. For bamboo $j \in S$, let m_j be the number of times the bamboo is cut in the interval $[t_2, t_3)$.

Since b_i already has height at least $2(R + Dh_{max})$ when the algorithm decides to cut j we have $h_j \geq h_i$ for all $j \in S$.

For $m_j \geq 2$ we have $h_j(t_3 - t_2) > 2(R + Dh_{max})(m_j - 1)$ or $\frac{h_j(t_3-t_2)}{2(m_j-1)} > R + Dh_{max}$ because bamboo b_j needs to grow to height $2(R+Dh_{max})$ at least $m_j - 1$ times in the interval in order to be cut m_j times. Meanwhile bamboo b_i grows by at most $R + Dh_{max}$ during this interval and therefore $R + Dh_{max} > h_i(t_3 - t_2)$. It follows that $h_j > 2(m_j - 1)h_i \geq m_j h_i$.

During the interval the algorithm visits each plant $b_j \in S$ a total of m_j times and travels distance d_j each time. Additionally distance $\frac{d_i}{2}$ is traversed to reach plant b_i.

$$\begin{aligned} t_3 - t_2 &= \sum_{b_j \in S} m_j d_j + \frac{d_i}{2} \\ &< \sum_{b_j \in S} \frac{h_j d_j}{h_i} + \frac{d_i}{2} = \frac{1}{h_i} \sum_{b_j \in S} h_j d_j + \frac{d_i}{2} \\ &\leq \frac{1}{h_i}(R - h_i d_i) + \frac{d_i}{2} = \frac{R}{h_i} - \frac{d_i}{2} \end{aligned}$$

This however means that the inverval is too short for plant b_i to grow by height R since this takes time $\frac{R}{h_i}$. This is a contradiction and thus no plant can reach height $3(R + Dh_{max}) \leq 6L$. ◀

Define $L = \max(R, Dh_1)$, where D is the diameter of the star and h_1 is the growth rate of the fastest bamboo. This is a lower bound on Opt.

The Deadline-Driven algorithm always cuts the bamboo with the earliest deadline among those with height at least L where the deadline is determined by some height the algorithm wants to maintain. This means that in each round the algorithm chooses the plant with the earliest deadline, travels towards this plant and cuts it before returning to the center. The algorithm does not turn around when a more urgent plant reaches height L.

We say a plant is *requested* when it reaches height $(1 + \sqrt{2})L$ and before that this request is *initialized* at the time the plant has height 0. This happens either after a cut or at the beginning of the process. We choose the *deadline* as the time a bamboo reaches height $(3 + 2\sqrt{2})L$.

This means each request to cut a plant consists of an initialization time, a request time and a deadline.

▶ **Lemma 11.** *The Deadline-Driven algorithm maintains a height of* $(3 + 2\sqrt{2})L$.

Proof. Assume plant i reaches height $(3 + \sqrt{2})L$ at time T and is not cut. We scale the time (and distance) such that $L/h_i = 1$ which means plant i grows by L in one timestep. This is possible because whenever we scale the length of all edges by a factor then the heights of all plants reached in any schedule is scaled by the same factor. This holds for both Alg and Opt which means the approximation ratio is unaffected. We can also see this as changing the timescale using L/h_i as our unit of time.

Let t_1 be the last time the algorithm is idle or busy processing a request with deadline after T. Let time 0 be the most recent time before t_1 where plant i has height 0.

This means between time t_1 and T the algorithm is only processing requests with deadlines at or before T and it is not idle. Let the set of these requests be S and let v be the earliest request time among all request in S. We have $i \in S$, so $v \leq 1 + \sqrt{2}$ and $t_1 \geq v$.

We further divide the set S into old requests S_0 which are initialised before v and new requests S_1 which are initialised after v. The time required to process S_0 is $\sum_{j \in S_0} d_j$ because any bamboo with an old request must be visited once to fulfill the request. Afterwards the bamboo either has a deadline after T and is not visited again or becomes part of the bamboo with new requests. We next show that all bamboo with new requests also have an old request. Consider the earliest new request for a bamboo. This new request gets initialized inbetween v and T. This means there was a previous request for that bamboo with a deadline between v and T. This request is an old request.

The time required to process S_1 is $\sum_{j \in S_1} m_j d_j$ where m_j is the number of new requests of bamboo j which are requests with initialization time after v and a deadline before or at T. (Here $j \in S$ means there is a request for bamboo j in the set of requests S. We may also see S as a multi-set of bamboo instead.)

It is possible that just before v a request with a deadline after T arrives and is processed by the algorithm. The algorithm traverses a distance of r to serve this request if it exists, otherwise $r = 0$. This means $t_1 \leq v + r$.

We now show that $\sum_{j \in S_0} d_j + \sum_{j \in S_1} m_j d_j + v + r < T$ which contradicts the assumption that plant i is not cut before time T.

We begin by finding upper bounds on the time required to process the requests in S.

A bamboo with an old request must grow by at least $(2 + \sqrt{2})L$ in time at most $T - v$ to have a deadline before T which means $h_j(T - v) \geq (2 + \sqrt{2})L$. Meanwhile plant i grows by $(T - v)L$ in time $(T - v)$, that is $h_i(T - v) = (T - v)L$. It follows that

$$\frac{h_j}{h_i} \geq \frac{2 + \sqrt{2}}{T - v} \text{ for } j \in S_0 \tag{1}$$

A bamboo with new requests must reach the threshold $(1 + \sqrt{2})L$ exactly m_j times in time at most $T - v$ in order to be requested m_j times before T. Then $h_j(T - v) \geq m_j(1 + \sqrt{2})$. It follows that

$$\frac{h_j}{h_i} \geq m_j \frac{1 + \sqrt{2}}{T - v} \text{ for } j \in S_1 \tag{2}$$

We can now find upper bounds for the processing times of the requests in S. We first get

$$\sum_{j \in S_0} d_j \overset{(1)}{\leq} \frac{1}{h_i} \frac{T - v}{2 + \sqrt{2}} \sum_{j \in S_0} d_j h_j < \frac{T - v}{2 + \sqrt{2}} \frac{R}{h_i}$$

and then

$$\sum_{j \in S_1} m_j d_j \overset{(2)}{\leq} \frac{1}{h_i} \frac{T - v}{1 + \sqrt{2}} \sum_{j \in S_0} d_j h_j < \frac{T - v}{1 + \sqrt{2}} \frac{R}{h_i}$$

Given the timescale and because $R \leq L$ it takes time less than $\frac{T-v}{2+\sqrt{2}}$ to process the old requests and less than $\frac{T-v}{1+\sqrt{2}}$ to process the new requests.

It remains to show $\frac{T-v}{2+\sqrt{2}} + \frac{T-v}{1+\sqrt{2}} + v + r \leq T$.

We have $r \leq L/h_1 \leq L/h_i = 1$ where the first inequality holds because otherwise plant 1 would grow by more than L in the time it takes to travel distance r and the second follows from the ordering of the plants by their growth rate. Furthermore since $v < 1 + \sqrt{2}$ and the earliest deadline of i is $3 + 2\sqrt{2}$ we have $T - v \geq (2 + \sqrt{2})$.

It follows

$$r \leq 1 = \left(\frac{1}{2+\sqrt{2}}\right)\left(2+\sqrt{2}\right) \leq \left(\frac{1}{2+\sqrt{2}}\right)(T-v)$$

and then

$$\begin{aligned}
& r \leq \left(\frac{1}{2+\sqrt{2}}\right)(T-v) \\
& \Rightarrow \left(\frac{1}{2+\sqrt{2}}\right) v + r \leq \left(\frac{1}{2+\sqrt{2}}\right) T \\
& \Rightarrow \left(\frac{1}{2+\sqrt{2}}\right) v + r \leq \left(\frac{1}{2+\sqrt{2}}\right) T + \left(1 - \frac{1}{2+\sqrt{2}}\right) T - \left(1 - \frac{1}{2+\sqrt{2}}\right) T \\
& \Rightarrow \left(1 - \frac{1}{2+\sqrt{2}}\right) T + \left(\frac{1}{2+\sqrt{2}}\right) v + r \leq T \\
& \Rightarrow \left(1 - \frac{1}{2+\sqrt{2}}\right) T - \left(1 - \frac{1}{2+\sqrt{2}}\right) v + v + r \leq T \\
& \Rightarrow \frac{T-v}{2+\sqrt{2}} + \frac{T-v}{1+\sqrt{2}} + v + r \leq T
\end{aligned}$$

This means we can process all requests (including the request for b_i with deadline at time T) before time T and plant i does not grow above height $(3 + 2\sqrt{2})L$. ◀

4.4 Extending the star graph

It is also possible to extend some of these results to an extended version of a star graph, where several plants are placed on each branch at different distances from the center. We again number the plants from 1 to n and the definition of h_i and d_i remains the same.

This means that d_i is the time it takes the server to travel the path leading up to node i from the center two times (once in both directions). Additionally we denote with δ_i the time it takes the server to travel the singular edge, connected to node i from direction of the center, two times (once in both directions). The definitions of d_i and δ_i are visualized in Figure 1.

We assume that the speeds are decreasing on each branch. (If there is a plant a that is further away than plant b on a branch and has the same speed or more, then plant b gets visited whenever plant a is visited.)

For the star graph with multiple plants on each branch, there is more than one possibility to accurately represent the schedule. One possibility is to list each visit to a plant, but it is also possible to omit plants that are visited by the server in passing, as it travels to a plant further on the branch. The representation we will be using, lists all plants that are visited in a round, but omits the second visit, where the server passes the plant again on its way to the center.

▶ **Lemma 12.** *The value $\frac{1}{2}R$ with $R = \sum_{i=1}^{n} h_i\delta_i$ is a lower bound on the optimal height for continuous BGT on a star graph with multiple plants on each branch.*

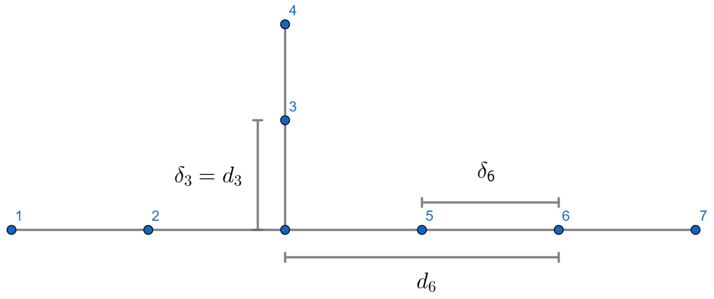


Figure 1 Multiple plants on each branch: example for values d_i and δ_i.

For the star with multiple plants on each branch we use the Deadline-Driven algorithm with a small modification. Whenever the algorithm visits a bamboo b_j on a branch it walks double the distance and cuts all additional bamboo it encounters. These bamboo can be removed from the graph since they are always cut together with b_j. Then the distance between b_j and the next bamboo b_i (i.e δ_i) on the branch is at least as much as the distance from b_j to the center (i.e. $d_i - \delta_i$). In particular it follows that $\delta_i \geq d_i - \delta_i$ and thus $d_i \leq 2\delta_i$.

Furthermore for the bamboo with multiple plants our lower bound is $\frac{1}{2}R$ with $R := \sum_{i=1}^{n} h_i\delta_i$ and therefore we define $L = \max(\frac{1}{2}R, Dh_1)$ to get a lower bound on OPT. For the star with multiple plants on each branch the deadline is the point in time a plant reaches height $(9+2\sqrt{5})L$ and it gets requested at height $(4+2\sqrt{5})L$. The initialization time remains the same.

▶ **Lemma 13.** *The modified Deadline-Driven algorithm maintains a height of* $(9+2\sqrt{5})L$*.*

5 Conclusions

It is possible to achieve a better approximation ratio by setting R in algorithm 1 to a lower value but this requires a much larger case analysis. This is because the periods p_i will be smaller. Using some computer assistance it is for example possible to achieve a ratio of $\frac{7}{5}$ but the length of the proof encourages finding new ideas. Using more cases, we could potentially get even closer to the ratio 4/3, but we could never reach it as long as we can only rely on pinwheel instances with density at most 3/4 being schedulable.

It should be noted that the pinwheel instances that our algorithm encounters are not general ones. For instance, the 10/7-approximation never encounters the period 6. It is conceivable that for the limited set of pinwheel instances that need to be considered, the density guarantee could be improved. Finally, of course the conjecture by Chan and Chin that all instances with density at most 5/6 are schedulable is still open. If that were proved, we could potentially get close to 6/5 (instead of only to 4/3). An approximation scheme currently seems out of reach.

References

1 Michael A. Bender, Sándor P. Fekete, Alexander Kröller, Vincenzo Liberatore, Joseph S. B. Mitchell, Valentin Polishchuk, and Jukka Suomela. The minimum backlog problem. *Theor. Comput. Sci.*, 605:51–61, 2015. doi:10.1016/j.tcs.2015.08.027.

2 Davide Bilò, Luciano Gualà, Stefano Leucci, Guido Proietti, and Giacomo Scornavacca. Cutting bamboo down to size. In Martin Farach-Colton, Giuseppe Prencipe, and Ryuhei Uehara, editors, *10th International Conference on Fun with Algorithms, FUN 2021, May 30 to June 1, 2021, Favignana Island, Sicily, Italy*, volume 157 of *LIPIcs*, pages 5:1–5:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.FUN.2021.5.

3 Marijke H. L. Bodlaender, Cor A. J. Hurkens, Vincent J. J. Kusters, Frank Staals, Gerhard J. Woeginger, and Hans Zantema. Cinderella versus the wicked stepmother. In Jos C. M. Baeten, Thomas Ball, and Frank S. de Boer, editors, *Theoretical Computer Science - 7th IFIP TC 1/WG 2.2 International Conference, TCS 2012, Amsterdam, The Netherlands, September 26-28, 2012. Proceedings*, volume 7604 of *Lecture Notes in Computer Science*, pages 57–71. Springer, 2012. doi:10.1007/978-3-642-33475-7_5.

4 Mee Yee Chan and Francis Y. L. Chin. General schedulers for the pinwheel problem based on double-integer reduction. *IEEE Trans. Computers*, 41(6):755–768, 1992. doi:10.1109/12.144627.

5 Marek Chrobak, János Csirik, Csanád Imreh, John Noga, Jirí Sgall, and Gerhard J. Woeginger. The buffer minimization problem for multiprocessor scheduling with conflicts. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*, volume 2076 of *Lecture Notes in Computer Science*, pages 862–874. Springer, 2001. doi:10.1007/3-540-48224-5_70.

6 Wei Ding. A branch-and-cut approach to examining the maximum density guarantee for pinwheel schedulability of low-dimensional vectors. *Real Time Syst.*, 56(3):293–314, 2020. doi:10.1007/s11241-020-09349-w.

7 Peter C. Fishburn and J. C. Lagarias. Pinwheel scheduling: Achievable densities. *Algorithmica*, 34(1):14–38, 2002. doi:10.1007/s00453-002-0938-9.

8 Leszek Gasieniec, Ralf Klasing, Christos Levcopoulos, Andrzej Lingas, Jie Min, and Tomasz Radzik. Bamboo garden trimming problem (perpetual maintenance of machines with different attendance urgency factors). In Bernhard Steffen, Christel Baier, Mark van den Brand, Johann Eder, Mike Hinchey, and Tiziana Margaria, editors, *SOFSEM 2017: Theory and Practice of Computer Science - 43rd International Conference on Current Trends in Theory and Practice of Computer Science, Limerick, Ireland, January 16-20, 2017, Proceedings*, volume 10139 of *Lecture Notes in Computer Science*, pages 229–240. Springer, 2017. doi:10.1007/978-3-319-51963-0_18.

9 Leszek Gasieniec, Benjamin Smith, and Sebastian Wild. Towards the 5/6-density conjecture of pinwheel scheduling. In Cynthia A. Phillips and Bettina Speckmann, editors, *Proceedings of the Symposium on Algorithm Engineering and Experiments, ALENEX 2022, Alexandria, VA, USA, January 9-10, 2022*, pages 91–103. SIAM, 2022. doi:10.1137/1.9781611977042.8.

10 Robert Holte, Aloysius Mok, Louis Rosier, Igor Tulchinsky, and Donald Varvel. Pinwheel: a real-time scheduling problem. In *Proceedings of the Hawaii International Conference on System Science*, pages 693–702 vol.2, 1989. doi:10.1109/HICSS.1989.48075.

11 Robert Holte, Louis E. Rosier, Igor Tulchinsky, and Donald A. Varvel. Pinwheel scheduling with two distinct numbers. *Theor. Comput. Sci.*, 100(1):105–135, 1992. doi:10.1016/0304-3975(92)90365-M.

12 John Kuszmaul. Bamboo trimming revisited: Simple algorithms can do well too. *CoRR*, abs/2201.07350, 2022. arXiv:2201.07350.

13 William Kuszmaul. Achieving optimal backlog in the vanilla multi-processor cup game. 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 1558–1577. SIAM, 2020. doi:10.1137/1.9781611975994.96.

14 Martijn van Ee. A 12/7-approximation algorithm for the discrete bamboo garden trimming problem. *CoRR*, abs/2004.11731, 2020. arXiv:2004.11731.

A Tables for checking $h_i/p_i \geq D_{\mathcal{U}}$ in the proof of Lemma 6

Case $3, 4, p_3^* \geq 5$ with schedule 1_2_:

p_i^*	p_i	h_i	h_i/p_i	$D_{\mathcal{U}}$
5	7	3	0.429	0.417
6	8	4	0.5	
7	10	5	0.5	
8	11	5	0.45	
9	12	6	0.5	
10	14	7	0.5	
11	15	7	0.47	
12	17	8	0.47	

Case $4, 4$ with schedule 1_2__

p_i^*	p_i	h_i	h_i/p_i	$D_{\mathcal{U}}$
4	5	3	0.6	0.5
5	7	4	0.57	
6	8	4	0.5	
7	10	6	0.6	
8	11	6	0.54	
9	12	7	0.58	
10	14	8	0.57	
11	15	9	0.6	

Case $4, 4, 4$ with schedule 123__

p_i^*	p_i	h_i	h_i/p_i	$D_{\mathcal{U}}$
4	5	2	0.4	0.25
5	7	2	0.28	
6	8	2	0.25	
7	10	4	0.4	
8	11	4	0.36	
9	12	4	0.33	
10	14	5	0.36	
11	15	6	0.4	

Case $4, 5$ with schedule 1_2__

p_i^*	p_i	h_i	h_i/p_i	$D_{\mathcal{U}}$
9	12	7	0.5833333333	0.55
10	14	8	0.5714285714	
11	15	9	0.6	
12	17	10	0.5882352941	
13	18	10	0.5555555556	
14	20	12	0.6	
15	21	12	0.5714285714	
16	22	13	0.5909090909	

Case $4, 5, 6$ with schedule 31__2 _13__ 21__ 3 _12__

p_i^*	p_i	h_i	h_i/p_i	$D_{\mathcal{U}}$
7	10	5	0.5	0.38
8	11	5	0.45	
9	12	5	0.42	
10	14	6	0.43	
11	15	7	0.47	
12	17	8	0.47	
13	18	8	0.44	
14	20	10	0.5	
15	21	10	0.48	
16	22	10	0.45	
17	24	11	0.45	
18	25	12	0.48	
19	27	13	0.48	
20	28	13	0.46	
21	30	15	0.5	

Case $4, 5, 8$ with schedule 1_3__ 12___ 1_32_ 1___2

p_i^*	p_i	h_i	h_i/p_i	$D_{\mathcal{U}}$
7	10	5	0.5	0.425
8	11	5	0.45	
9	12	6	0.5	
10	14	7	0.5	
11	15	7	0.47	
12	17	8	0.47	
13	18	9	0.5	
14	20	11	0.55	
15	21	11	0.52	
16	22	11	0.5	
17	24	12	0.5	
18	25	13	0.52	
19	27	14	0.525	
20	28	14	0.5	
21	30	16	0.53	

Case $4, 6$ with schedule 1____ 1_2__ 1___2

p_i^*	p_i	h_i	h/p	$D_{\mathcal{U}}$
9	12	7	0.58	0.583
10	14	9	0.64	
11	15	10	0.67	
12	17	10	0.59	
13	18	11	0.61	
14	20	13	0.65	
15	21	13	0.62	
16	22	14	0.64	
17	24	15	0.63	
18	25	16	0,64	
19	27	17	0.63	
20	28	18	0.64	
21	30	20	0.67	
22	31	20	0.65	
23	32	20	0.63	
24	34	22	0.65	
25	35	23	0.66	
26	37	24	0.65	
27	38	25	0.66	
28	40	26	0.65	
29	41	26	0.63	
30	42	27	0.649	

Case $4, 6, 7$ with schedule 1__3_ 1_2__ 13__2

p^*	p	h	h/p	D_A
7	10	6	0.6	0.440
8	11	6	0.54	0.458
9	12	7	0.58	0.458
10	14	8	0.57	
11	15	8	0.53	
12	17	10	0.58	
13	18	10	0.55	
14	20	11	0.55	
15	21	12	0.57	
16	22	12	0.55	
17	24	14	0.58	
18	25	14	0.56	
19	27	15	0.56	
20	28	16	0.57	
21	30	16	0.53	
22	31	17	0.55	
23	32	18	0.56	
24	34	19	0.56	
25	35	19	0.54	
26	37	20	0.54	
27	38	21	0.55	
28	40	22	0.55	

Case $4, 6, 8$ with schedule 1__3_ 1_2__ 1__32 1____ 1_23_ 1___2

p^*	p	h	h/p	D_A
9	12	6	0.5	0.458
10	14	7	0.5	
11	15	8	0.53	
12	17	8	0.47	
13	18	9	0.5	
14	20	11	0.55	
15	21	11	0.52	
16	22	12	0.55	
17	24	12	0.5	
18	25	13	0.52	
19	27	14	0.52	
20	28	15	0.54	
21	30	17	0.57	
22	31	17	0.55	
23	32	17	0.53	
24	34	18	0.53	
25	35	19	0.54	
26	37	20	0.54	
27	38	21	0.55	
28	40	22	0.55	
29	41	22	0.53	
30	42	23	0.546	

Case $4, 6, 8, 8$ with schedule 1__3_ 1_24_ 1__32 1__4_ 1_23_ 1__42

p^*	p	h	h/p	D_A
4	5	2	0.4	0.333
5	7	3	0.43	
6	8	3	0.375	
7	10	4	0.4	
8	11	4	0.36	
9	12	5	0.42	
10	14	6	0.43	
11	15	7	0.47	
12	17	7	0.41	
13	18	7	0.39	
14	20	9	0.45	
15	21	9	0.43	
16	22	10	0.45	
17	24	10	0.42	
18	25	11	0.44	
19	27	12	0.44	
20	28	12	0.43	
21	30	14	0.47	
22	31	14	0.45	
23	32	14	0.44	
24	34	15	0.44	
25	35	16	0.46	

Case $4, 8$ with schedule 1_2__ 1____

p^*	p	h	h/p	D_A
6	8	5	0.625	0.625
7	10	7	0.7	
8	11	7	0.64	
9	12	8	0.67	
10	14	9	0.64	
11	15	10	0.67	
12	17	11	0.65	
13	18	12	0.67	

B Multiple plants on each branch

We now find a lower bound for continuous BGT on a star graph with multiple plants on each branch. In this scenario, we make a distinction between cuts where the server comes from the center, and cuts where the server comes from the direction opposite the center. For both types of cuts, we find a lower bound on the maximum height a plant reaches in a particular schedule.

Consider plant p, and let A be the set of plants that are on the same branch but further from the center, and B the set of plants that are closer to the center on the same branch or on another branch.

Since the algorithm needs to visit the end of the branch at some point, p reaches a height of at least $h_p \sum_{i \in A} \delta_i$. This happens as the server is coming from the direction opposite the center and this is a lower bound on the maximum height.

Next, we develop a lower bound based on the cuts where the server comes from the direction of the center. Consider a cyclic schedule of length L, where only the first cut of each round, where the server comes from the center, is listed in this schedule. Each round the algorithm visits a branch and possibly multiple plants. We define m_i as the amount of rounds plant i is visited in. Then $\frac{L}{m_i}$ is the average period between visits of plant i, but only counting the first cut each round where the server comes from the direction of the center.

▶ **Lemma 14.** *The average height of p after a cut, with the server coming from the direction of the center, is $h_p(\delta_p + \sum_{i \in B} \frac{m_i}{m_p} \delta_i)$.*

Proof. Consider a plant p. Let $\lambda_1, \ldots, \lambda_{m_p}$ be the heights plant p reaches in the schedule, while only considering cuts coming from the center. The sum of these heights is the same as the sum of all distances travelled to visit plants in set $B \cup \{p\}$ multiplied by the speed of the plant. This means $\sum_{k=1}^{m_p} \lambda_k = h_p \sum_{i \in B \cup \{p\}} \delta_i m_i$. The average height is

$$\frac{1}{m_p} \sum_{k=1}^{m_p} \lambda_k = \frac{1}{m_p} h_p \sum_{i \in B \cup \{p\}} \delta_i m_i = h_p(\delta_p + \sum_{i \in B} \frac{m_i}{m_p} \delta_i) \qquad ◀$$

We now adapt our model. The algorithm may visit each plant i at a certain fixed period f_i. Let C be the set of all plants, that are closest to the center on their branch. We require $\sum_{i \in C} \frac{1}{f_i} = 1$, which means that the frequencies add up to one branch visited each round. Furthermore, the periods of plants on the same branch should be non decreasing as distance to the center increases. We define the height of a plant p as $\max(h_p(\sum_{i \in A} \delta_i, h_p(\delta_p + \sum_{i \in B} \frac{f_p}{f_i} \delta_i))$.

By setting $f_i = \frac{L}{m_i}$ we get a solution for the new model with a height that is equal to the average height of a schedule in the original model and not more than the maximum height. A lower bound on the maximum height in this new model is then a lower bound on the maximum height for continuous BGT on a star graph with multiple plants on each branch.

▶ **Lemma 12.** *The value $\frac{1}{2}R$ with $R = \sum_{i=1}^{n} h_i \delta_i$ is a lower bound on the optimal height for continuous BGT on a star graph with multiple plants on each branch.*

Proof. Consider the new model. We set $K = \sum_{i \in C}^{n} h_i$ and $f_i = \frac{K}{h_i}$. Let H_p be the maximum height that plant p reaches in the new model. By the definition of height we get $H_p \geq h_p \sum_{i \in A} \delta_i$ as well as $H_p \geq h_p(d_p + \sum_{i \in B} \frac{f_p}{f_i} d_i)$. Then

$$2H_p \geq h_p \sum_{i \in A} \delta_i + h_p(d_p + \sum_{i \in B} \frac{f_p}{f_i} \delta_i) = \sum_{i \in A} h_p \delta_i + h_p \delta_p + \sum_{i \in B} h_i \delta_i \geq \sum_{i=1}^{n} h_i \delta_i$$

The last inequality follows from the fact that the speeds are decreasing with distance to the center and therefore $h_p \geq h_i$ for $i \in A$. It follows that all plants reach a height of at least $\frac{1}{2}R$ and again it is not possible to achieve a lower height on all plants, since visiting one plant more often increases the height on other plants. ◀

▶ **Lemma 13.** *The modified deadline driven algorithm maintains a height of $(9 + 4\sqrt{5})L$.*

Proof. Assume plant i reaches height $(9 + 4\sqrt{5})L$ at time T. We scale the time (and distance) such that $L/h_i = 1$ which means plant i grows by L in one timestep. The old and new requests S_0 and S_1 as well as v and r are defined analogously to the proof for the star.

The time required to process S_0 is at most $\sum_{j\in S_0} d_j$ because any bamboo with an old request must be visited once to fulfill the request and in the worst case we start from the center traversing distance d_j to fulfill the request. It is possible that the algorithm is more efficient whenever plants with consecutive deadlines lie behind each other on the same branch.

Similarly the time required to process S_1 is at most $\sum_{j\in S_1} m_j d_j$ where m_j is the number of new requests on plant b_j.

Therefore in order to arrive at a contradiction we again need to show $\sum_{j\in S_0} d_j + \sum_{j\in S_1} m_j d_j + v + r < T$ which means plant i can be cut before time T.

Again we find upper bounds on the time required to process the old and new requests. These bounds are slightly different due to the changes in the algorithm as well as the lower bound.

A bamboo with an old request must grow by at least $(5+2\sqrt{5})L$ in time at most $T-v$ to have a deadline before T which means $h_j(T-v) \geq (5+2\sqrt{5})L$. Meanwhile plant i grows by $(T-v)L$ in time $(T-v)$, that is $h_i(T-v) = (T-v)L$. It follows that

$$\frac{h_j}{h_i} \geq \frac{5+2\sqrt{5}}{T-v} \text{ for } j \in S_0 \tag{3}$$

A bamboo with new requests must grow by $(4+2\sqrt{5})L$ exactly m_j times in time at most $T-v$ in order to have m_j deadlines before T. Then $h_j(T-v) \geq m_j(4+2\sqrt{5})$. It follows that

$$\frac{h_j}{h_i} \geq m_j\frac{4+2\sqrt{5}}{T-v} \text{ for } j \in S_1 \tag{4}$$

We now get

$$\sum_{j\in S_0} d_j \overset{(3)}{\leq} \frac{1}{h_i}\frac{T-v}{5+2\sqrt{5}}\sum_{j\in S_0} d_j h_j \leq \frac{1}{h_i}\frac{T-v}{5+2\sqrt{5}}\sum_{j\in S_0} 2\delta_j h_j < \frac{T-v}{5+2\sqrt{5}}\frac{2R}{h_i} \leq \frac{T-v}{5+2\sqrt{5}}\frac{4L}{h_i}$$

where we are using $d_i \leq 2\delta_i$ in the second inequality and $R \leq 2L$ in the last inequality. Analogously the following holds for the time required to process the new requests

$$\sum_{j\in S_1} m_j d_j \overset{(4)}{\leq} \frac{1}{h_i}\frac{T-v}{4+2\sqrt{5}}\sum_{j\in S_0} d_j h_j \leq \frac{1}{h_i}\frac{T-v}{4+2\sqrt{5}}\sum_{j\in S_0} 2\delta_j h_j < \frac{T-v}{4+2\sqrt{5}}\frac{2R}{h_i} \leq \frac{T-v}{4+2\sqrt{5}}\frac{4L}{h_i}$$

Given the timescale this means it takes time at most $4\frac{T-v}{5+2\sqrt{5}}$ to process the old requests and at most $4\frac{T-v}{4+2\sqrt{5}}$ to process the new requests. Furthermore since $v < 4+2\sqrt{5}$ and the earliest deadline of i is $9+4\sqrt{5}$ we have $T-v \geq (5+2\sqrt{5})$. It follows

$$\begin{aligned}
&r \leq 1 = \left(\frac{1}{5+2\sqrt{5}}\right)(5+2\sqrt{5}) \leq \left(\frac{1}{5+2\sqrt{5}}\right)(T-v) \\
&\Rightarrow \left(\frac{1}{5+2\sqrt{5}}\right)v + r \leq \left(\frac{1}{5+2\sqrt{5}}\right)T \\
&\Rightarrow \left(\frac{1}{5+2\sqrt{5}}\right)v + r \leq \left(\frac{1}{5+2\sqrt{5}}\right)T + \left(1-\frac{1}{5+2\sqrt{5}}\right)T - \left(1-\frac{1}{5+2\sqrt{5}}\right)T \\
&\Rightarrow \left(1-\frac{1}{5+2\sqrt{5}}\right)T + \left(\frac{1}{5+2\sqrt{5}}\right)v + r \leq T \\
&\Rightarrow \left(1-\frac{1}{5+2\sqrt{5}}\right)T - \left(1-\frac{1}{5+2\sqrt{5}}\right)v + v + r \leq T \\
&\Rightarrow 4\frac{T-v}{5+2\sqrt{5}} + 4\frac{T-v}{4+2\sqrt{5}} + v + r \leq T
\end{aligned}$$

This means we can process all requests (including the request for b_i with deadline at time T) before time T and plant i does not grow above height $(9+4\sqrt{5})L$. ◀

Online Matching with Set and Concave Delays

Lindsey Deryckere ✉
School of Computer Science, The University of Sydney, Australia

Seeun William Umboh ✉ 🏠
School of Computing and Information Systems, The University of Melbourne, Australia

Abstract

We initiate the study of online problems with *set delay*, where the delay cost at any given time is an arbitrary function of the set of pending requests. In particular, we study the online min-cost perfect matching with set delay (MPMD-Set) problem, which generalises the online min-cost perfect matching with delay (MPMD) problem introduced by Emek et al. (STOC 2016). In MPMD, m requests arrive over time in a metric space of n points. When a request arrives the algorithm must choose to either match or delay the request. The goal is to create a perfect matching of all requests while minimising the sum of distances between matched requests, and the total delay costs incurred by each of the requests. In contrast to previous work we study MPMD-Set in the *non-clairvoyant* setting, where the algorithm does not know the future delay costs. We first show no algorithm is competitive in n or m. We then study the natural special case of *size-based* delay where the delay is a non-decreasing function of the number of unmatched requests. Our main result is the first non-clairvoyant algorithms for online min-cost perfect matching with size-based delay that are competitive in terms of m. In fact, these are the first non-clairvoyant algorithms for any variant of MPMD. A key technical ingredient is an analog of the symmetric difference of matchings that may be useful for other special classes of set delay. Furthermore, we prove a lower bound of $\Omega(n)$ for any deterministic algorithm and $\Omega(\log n)$ for any randomised algorithm. These lower bounds also hold for clairvoyant algorithms. Finally, we also give an m-competitive deterministic algorithm for uniform concave delays in the clairvoyant setting.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases online algorithms, matching, delay, non-clairvoyant

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.17

Category APPROX

Related Version *Full Version*: https://arxiv.org/abs/2211.02394

1 Introduction

Studying online problems with delay is a line of work that has recently gained traction in online algorithms (e.g. [4, 19, 21, 23]). In such problems, request arrive over time requiring service. Delaying the service of a request accumulates a delay cost given by a delay function associated with the request. The total cost of a solution is the cost of servicing all requests plus the sum of all delay costs incurred by each request.

We initiate the study of online problems with *set delay*. In this model, we generalize the notion of delay to one where the instantaneous delay cost at any point in time is determined by an arbitrary monotone non-decreasing function of the set of pending requests, rather than the sum of individual delay functions associated with each request. In particular, we study the online min-cost perfect matching with set delay (MPMD-Set) problem, which generalizes of the min-cost perfect matching with delays (MPMD) problem introduced by Emek et al. [19].

In MPMD, m requests arrive over time in a metric space of n points. Upon arrival of a request the algorithm must choose to either match the request, incurring a cost equal to the distance between the two requests, or to delay the request, incurring a cost given by a delay function associated with the request. Prior results for MPMD have mostly focused on each request sharing the same delay function (in particular, linear, concave, and convex) and achieve competitive ratios that solely depend either on n or m. Moreover, existing algorithms rely on clairvoyance, where the algorithm has full knowledge of future delay costs. Furthermore, existing randomised algorithms rely on metric embeddings which require knowledge of the metric space in advance.

In this paper, our main contribution is to study the more general MPMD-Set in the least restrictive setting where the algorithm does not know the metric space in advance and has no knowledge of future delay costs. We begin by showing that, in contrast to prior results, the MPMD-Set problem does not admit a deterministic competitive ratio that solely depends on n or m.

▶ **Theorem 1.** *Every deterministic algorithm for MPMD-Set has competitive ratio $\Omega(\Phi)$, where Φ is the aspect ratio of the metric space.*

Our lower bound holds even for simple instances where n and m are constants. Thus, we restrict our attention to designing a competitive solution for the MPMD-Set problem where the instantaneous delay cost at any point in time is a monotone non-decreasing function of the number of unmatched requests at that time. We call such a delay cost function *size-based* (See Section 2 for a formal definition). MPMD-Set with size-based delay (MPMD-Size) has natural applications in practical settings with service-level agreements such as cloud computing.[1]

Our main result is the first competitive algorithms for MPMD-Size, where the competitive ratio is a function of the number of requests. At the core of our result is a reduction from MPMD-Size to the well-known Metrical Task System (MTS) problem (defined in Section 1.1).

▶ **Theorem 2.** *For any $f(N)$-competitive algorithm for MTS with N states, there is an $f(2^m)$-competitive algorithm for MPMD-Size.*

We obtain our main result by applying state-of-the-art algorithms for MTS with some modifications.

▶ **Corollary 3.** *For MPMD-Size, there is an $O(2^m)$-competitive deterministic algorithm and an $O(m^4)$-competitive randomised algorithm.*

We emphasise that our algorithms are non-clairvoyant and do not need to know the metric space in advance. To the best of our knowledge, this is the first non-clairvoyant online algorithm for this problem. Non-clairvoyant algorithms nevertheless have been designed for other online problems such as the Set Cover problem [4], the k-server problem [25], and multi-level aggregation [26]. We also remark that every deterministic algorithm for known variants of online matching with delays has a competitive ratio that depends on m.

We complement Corollary 3 with the following lower bounds.

▶ **Theorem 4.** *Every deterministic algorithm for MPMD-Size has competitive ratio $\Omega(n)$.*

▶ **Theorem 5.** *Every randomised algorithm for MPMD-Size has competitive ratio $\Omega(\log n)$.*

[1] In these settings, the service level agreement requires the cloud provider to provide a certain level of service and the provider incurs penalties if the level is not met.

Finally, we consider MPMD with uniform concave delay in the clairvoyant setting and give the first deterministic algorithm for it. In this problem, we are given a non-negative, non-decreasing concave function f. The delay cost incurred by a request r is $f(w_r)$ where w_r is the time between r's arrival and when it was matched. The total delay cost is the sum of the delay cost of each request.

▶ **Theorem 6.** *There exists an $O(m)$-competitive deterministic algorithm for MPMD with concave delay.*

The correctness and competitiveness of our algorithm only relies on the fact that the time-augmented space satisfies the properties of a metric space. Similar to previous deterministic solutions for uniform linear delay, our algorithm does not need the metric space to be finite, and does not need to know it in advance.

The proofs of theorems 4, 5 and 6 can be found in the full version.

1.1 Our Techniques

Our main technical contribution is an online reduction from the MPMD-Set problem to MTS, which constitutes the proof of Theorem 2. The Metrical Task System (MTS) problem, introduced by Borodin et al. [15], is a cost minimisation problem defined by a set of states $S = \{s_1, s_2, ..., s_k\}$ and a cost matrix c that defines the cost of moving between states. The input consists of an initial state S_0 and a sequence of tasks $T = (t_1, ..., t_\ell)$. Each task t_j is associated with a k-dimensional cost vector C_j whose i-th coordinate defines the cost of servicing task t_j in state s_i. For a given input task sequence T, a solution is a sequence of states (called a *schedule*) $\sigma = (S_1, S_2, ..., S_\ell)$, where S_j is the state that task j is processed in. The total cost of a schedule consists of the costs associated with moving states (*transition cost*), as well as the cost of processing the tasks (*processing cost*).The aim is to produce a schedule of minimum cost.

We briefly outline the three main parts of the reduction below.

Step 1: MPMD-Set to MTS. The first part of the reduction transforms an instance of MPMD-Set into an instance of MTS. A natural approach at a reduction to MTS is to use the set of all possible matchings of requests as the set of states for the MTS instance. The transition cost between two states is then the total length of the edges in the symmetric difference of the corresponding matchings. Finally, there is a task for each timestep in the matching problem and the cost of processing the task in a state is the instantaneous delay incurred by the set of unmatched requests. Unfortunately, the number of states is equal to the number of possible matchings between the requests which is $\sim (\frac{m}{e})^{m/2} \frac{e^{\sqrt{m}}}{(4e)^{1/4}}$ for m requests.

Instead, we use the set of all possible even-sized subsets of the requests as the set of MTS states. Each state represents a set of requests that are matched. The set of input states thus develops over time as more requests arrive. The initial state is the empty set.

We now define the transition costs of the MTS. We define a *transition graph* $G(V, E)$ where V is the set of even-sized subsets of requests. The transition graph G has an edge between two states S and S' if $S \subset S'$ and $|S'| = |S| + 2$. In other words, S_2 consists of the same requests as S_1 with 2 additional requests, say p, q. The cost of the edge between the two states is $d(p, q)$, the distance between p and q in the original MPMD-Set instance. The transition cost between any two states in the MTS instance is defined to be the minimum cost path between the two corresponding nodes in G. The delay cost is translated into the vector costs associated with serving tasks: for any timestep t, the cost of servicing a task in state is simply the instantaneous delay cost accumulated by the set of requests that have arrived so far in the original MPMD-Set instance, that are not in that state.

Henceforth, we refer to an instance of MTS that is reduced from MPMD-Set as *MPMD-Set-MTS*, and an MTS instance that is reduced from MPMD-Set with size-based delay as *MPMD-Size-MTS*.

▶ **Definition 7** (Monotone). *A schedule $\sigma = (S_1, \ldots, S_\ell)$ is* monotone *if every transition only adds requests to the current state, i.e. $S_{i-1} \subseteq S_i$ for every i. In other words, the path never involves moving to a strictly smaller state. An algorithm for MPMD-Set-MTS is* monotone *if it always produces a* monotone *schedule.*

It is easy to see that a monotone schedule can be converted into a solution for the MPMD-Set instance without any increase in cost: when the schedule adds a set of requests S' to its current state, we add a min-cost perfect matching on S' to our matching. However, when we run existing MTS algorithms on the MPMD-Set-MTS instance, there is no guarantee that they will return a monotone schedule.

Step 2: Converting to Monotone. Fortunately, it is possible to convert, in an online manner, an arbitrary MPMD-Size-MTS solution to a monotone solution at no extra cost. We do this by designing an online algorithm which, given an online sequence of states $\sigma = (S_1, \ldots, S_\ell)$, produces for each state S_i a corresponding state S_i' such that the resulting schedule produced by the algorithm is *monotone*. We refer to an algorithm that *transforms* a given state as a *state conversion algorithm*.

Since the instantaneous delay in the MPMD-Size instance is a non-increasing function of currently matched requests, for every task in the MPMD-Size-MTS instance created by the reduction, the processing cost is a monotone non-increasing function of the state size. Our algorithm will exploit this by maintaining the invariant that our state is always at least large as that of σ. The technical crux here is to show that when our state is smaller, we can augment our state in a cost-efficient manner. If the MTS states were matchings, we can consider augmenting paths in the symmetric difference of our matching and that of σ. Motivated by this, we define analogs of the symmetric difference and augmenting paths to augment our state. We believe these ideas are useful for other interesting special classes of set delay functions.

Step 3: Applying MTS algorithms. There are two issues that prevent us from applying existing MTS algorithms directly. First, the cost bounds of all known algorithms for MTS have an additive term that is equal to the diameter of the MTS state space, and the MTS instance created by our reduction has state space with diameter much larger than the optimal. The second issue is that our reduction creates an MTS instance whose state space is constructed online, i.e. the states arrive over time. At a high level, the first issue can be overcome by a guess-and-double approach. The second issue is a problem for randomised MTS algorithms that rely on embedding the state space into a tree as a pre-processing step. This is overcome by using the online embedding of [10]. See Section 4.3 for a more detailed discussion.

Designing a deterministic algorithm for MPMD with Concave Delay. We use the (offline) moat-growing framework (generally used for constrained connectivity problems) by Goemans and Williamson [22], to design an online deterministic linear programming-based algorithm for MPMD with uniform concave delay. Our algorithm is a modification of an existing linear programming-based algorithm due to Bienkowski et al. [12], who give a deterministic competitive algorithm for MPMD with uniform linear delay. Their algorithm heavily relies on the fact that, when the delay function is linear, requests accumulate delay cost at the same rate at all times, regardless of their arrival time. The main challenge in applying the framework to concave delay is that, unlike in the case of linear delay, the requests can accumulate delay at different rates at any point in time, depending on their arrival time. See the full version of the paper on ArXiv for a detailed discussion.

1.2 Related Work

MPMD was introduced by Emek et al. [19] where the delay functions associated with each request are uniform linear. They designed a randomised algorithm that achieves a competitive ratio of $O(\log^2 n + \log \Delta)$, where n is the number of points in the metric space and Δ is its aspect ratio. Azar et al. [3] used a randomised HST embedding to provide a $O(\log n)$-competitive almost-deterministic algorithm, improving Emek et al.'s bound and removing the dependency on the aspect ratio of the metric space. Furthermore, they provided a lower bound of $\Omega(\sqrt{\log n})$ for any randomised algorithm in the case of linear delay. Ashlagi et al. [1] improved this lower bound to $\Omega(\frac{\log n}{\log \log n})$ and $\Omega(\sqrt{\frac{\log n}{\log \log n}})$ for the bipartite case, which are the best known so far. Liu et al. furthermore adapted the algorithm by Azar et al. to the bipartite setting and improved the analysis of Emek et al.'s algorithm to $O(\log n)$. The next deterministic algorithm for simple metrics was by Emek et al. [20] who proved a competitive ratio of 3 for the simple metric space of 2 points. The first deterministic algorithm for general metric spaces was by Bienkowski et al. [13] and their analysis resulted in a competitive ratio of $O(m^{2.46})$. Bienkowski et al. [12] and Azar et al. [6] concurrently and independently improved this bound to $O(m)$ and $O(m^{0.59})$ respectively, introducing the first linear and sub-linear deterministic solutions to the problem. The algorithms above assumed the delay cost to be given by a uniform linear delay function associated with each individual request.

Liu et al. [27] was the first to consider convex delay functions and demonstrated an interesting gap between the solutions for the case with linear delay and convex delay on a uniform metric space by giving a deterministic asymptotically optimal $O(m)$-competitive algorithm for the uniform metric space.

Azar et al. [8] subsequently considered the problem with concave delay and achieved an $O(1)$-competitive deterministic algorithm for the single point metric space and an $O(\log n)$ randomised algorithm for general metric spaces.

The above algorithms assumed all requests incurred delay in accordance with uniform delay functions and regarded the delay function to be associated with each individual request. Furthermore, all prior solutions to MPMD assumed clairvoyance. To the best of our knowledge, no one has considered the non-clairvoyant generalisation of the problem where the delay function depends on the set of unmatched requests.

Non-clairvoyant algorithms nevertheless have been designed for other online problems such as the Set Cover problem [4, 26], the k-server problem [25], and multi-level aggregation [26].

The notion of introducing delay to online problems originated well before it was applied to online metric matching and finds applications in amongst others aggregating messages in computer networks, aggregating orders in supply-chain management, and operating systems. See [4, 5, 7, 9, 11, 14, 17, 18, 21, 23, 26, 28] for further reading. All problems above define the cost of delay as a function associated with each request. To the best of our knowledge, no online problems with delay have so far defined the cost of delay as an arbitrary function of the set of unmatched requests.

2 Preliminaries

In this section we introduce our notation and give formal definitions for set delay and size-based delay functions, as well as MPMD-Set.

2.1 Min-cost Perfect Matching with Delay

The min-cost perfect matching with delays (MPMD) problem, introduced by Emek et. al [19] is defined on a metric space (V, d), which consists of a set of points V and distance function $d : V \times V \to \mathbb{R}^+$. An online input instance over (V, d) is a sequence of requests $R = (r_1, ..., r_m)$ that arrive at points in the metric space over time. Each $r_k \in R$ has an associated position and arrival time. We assume, without loss of generality, that time is divided into discrete timesteps.

Upon the arrival of a request, the algorithm must choose to either match the request, incurring a cost equal to the distance between the two requests in the metric space, or to delay the request, incurring a cost given by a delay function associated with the request, in the hope of finding a more suitable match in the near future.

A solution produced by an online matching algorithm is a sequence of matchings $\mathbb{M} = (M_0...M_{final})$, where M_i is the matching associated with the ith timestep. Note that we assume that requests only arrive at the start of a timestep. A solution $\mathbb{M}$ must satisfy the following properties:

- $M_0 = \emptyset$
- M_{final} is a perfect matching
- For all i, $M_i \subseteq M_{i+1}$

We refer to the third property as *monotonicity*. The cost associated with a solution $\mathbb{M}$ consists of the sum of the distances between matched requests in M_{final} plus the sum of the delay costs incurred by all requests. The aim of an online matching algorithm is to produce a sequence of matchings that satisfies the above properties with minimal cost.

In the original MPMD problem, the delay cost incurred by a request is the time between its arrival and when it was matched. In MPMD-Set, the instantaneous delay incurred at a timestep t can be an arbitrary function of the set of currently unmatched requests U_t. The total delay cost is the sum over timesteps t of $f_t(U_t)$ where f_t is a set delay function as defined below.

▶ **Definition 8** (Set delay function). *Let U be a set of requests. We define a delay function $f_t : 2^U \to \mathbb{R}^{\geq 0}$, for any timestep t, to be a* set delay *function if it satisfies the following properties:*

- $f_t(\emptyset) = 0$
- $A \subseteq B \Rightarrow f_t(A) \leq f_t(B)$
- *For all $\emptyset \neq U \in 2^V$, we have $\sum_{t=0}^{\infty} f_t(U) = \infty$*

The last property implies that all requests must eventually be matched.

In MPMD-Size, the set delay function is a size-based delay function, as defined below.

▶ **Definition 9** (Size-based delay function). *We define a delay function $f_t : 2^U \to \mathbb{R}^{\geq 0}$ to be* size-based *if, for any timestep t, it satisfies all properties of a set delay function and is monotone non-decreasing as a function of the size of the set of requests U.*

3 A Lower Bound for MPMD-Set

In this section, we prove Theorem 1.

Proof of Theorem 1. Consider a four-point metric space (as depicted in figure 1) with three points at distance ϵ from one another, and the fourth point (p_4) at distance D from the other points, where D is the diameter of the metric space.

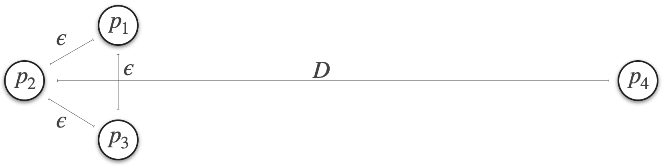


Figure 1 A visualisation of the four-point metric space.

We define a request sequence of six requests $R = (r_1, r_2, r_3, r_4, r_5, r_6)$ where the first four requests arrive at time $t = 0$ and the latter two arrive at time $t = 2$. For each $i \in \{1...4\}$, we place request r_i on point p_i. At $t = 2$, we then place request r_5 on p_3 and r_6 on p_4. In terms of delay, we are working with the special case of deadline functions. A *deadline function* is a delay function that is 0 up until some time d, called the deadline, and ∞ afterwards. When the deadline of a request is reached, the algorithm must ensure that the request is matched.

At $t = 0$, request r_1 reaches its deadline and hence the algorithm will need to match two requests. Since the algorithm is non-clairvoyant, the algorithm has no knowledge of the deadlines of future requests. We therefore assume without loss of generality that it matches r_1 to r_3 and pays a distance cost of ϵ. At $t = 1$, r_2 reaches its deadline and the algorithm is forced to match it to r_4 at a distance cost of D. At $t = 2$, the final two requests will arrive and instantly reach their deadline. The algorithm will consequently need to match r_5 to r_6 at a distance cost of D. The total cost of ALG is $2D + \epsilon$.

The optimal offline solution OPT is to match r_1 to r_2 and match locally at $t = 2$ on points p_3 and p_4. The total cost of OPT is therefore ϵ. The competitive ratio of the algorithm is $\Omega(D/\epsilon)$ and D/ϵ is the aspect ratio of the metric space, as desired. ◀

4 An Online Reduction from MPMD-Set to MTS

In this section, we prove Theorem 2 by defining a reduction from MPMD-Set to MTS. We start by translating an arbitrary instance of MPMD-Set into an instance of MTS in Section 4.1. In Section 4.2, we show that we can transform an arbitrary MPMD-Size-MTS solution into a monotone solution of the same or less cost. As observed in the Introduction, a monotone schedule directly corresponds to a solution for the online matching with delay problem of equal cost. This completes the proof of Theorem 2. We finish this section with a proof of Corollary 3.

4.1 Translating an instance of MPMD-Set into and instance of MTS

We define the set of internal states of the MTS instance to be the set of all possible subsets of the requests that have arrived so far in the MPMD-Set instance. In order to define the transition cost associated with moving between states, we define the following graph.

▶ **Definition 10** (Transition graph G). *The nodes of G are the states of the MPMD-Set-MTS instance. We define an undirected edge $(S, S') \in E$ between states S and S' if $S = S' \cup \{r, r'\}$ and define its cost $c(S, S')$ to be $d(r, r')$. Paths in the transition graph are called* transition paths.

We define the transition cost $c(S, S')$ of moving between arbitrary states S and S' in the MTS instance to be the cost of the shortest path between them in the transition graph G. Note that a path in G between states S, S' corresponds to a sequence of transitions from S to S', where each edge in the path corresponds to a transition that either adds two requests or removes two requests. Also, note that any schedule σ, produced by an online MTS algorithm, corresponds to a path in the metric completion of G and the total transition cost incurred by σ is simply the total cost of the path. Each task in the MTS instance is associated with a given timestep in the MPMD-Set problem. The cost vector associated with each task is, for every state S, the instantaneous delay cost accumulated by the set of requests not in S.

▶ **Definition 11** (R_i). *We define R_i to be the set of requests that have arrived up to and including timestep i in the original MPMD-Set instance.*

The total cost of a schedule $\sigma = (S_0, ..., S_T)$ can be expressed as follows.

$$\text{cost}(\sigma) = \sum_{i=0}^{T-1} c(S_i, S_{i+1}) + f_i(R_i \setminus S_i).$$

By construction, the cost associated with processing the tasks represent the delay cost incurred by the requests, while the distance cost is represented by the transition costs associated with moving between states.

4.2 Size-based delay functions admit monotone scheduling algorithms

In this subsection, we prove the existence of an online algorithm that converts an arbitrary MPMD-Size-MTS solution into a monotone solution, without incurring any extra cost.

▶ **Lemma 12.** *There exists an online algorithm that converts an arbitrary MPMD-Size-MTS solution into a monotone solution of the same or less cost.*

Proof. To prove Lemma 12, we define an online *state conversion algorithm* (Sensible-ALG) which, for every state S_i in a schedule σ, produced by an *arbitrary* online scheduling algorithm (OSA), produces a state S'_i such that the cost of the schedule $\sigma' = (S'_0, \ldots, S'_{|\sigma|})$ is at most the cost of the original schedule σ, and σ' is monotone.

The state conversion algorithm aims to maintain the invariant that $|S'_i| \geq |S_i|$ and the main property of a monotone schedule, which is that $S'_{i-1} \subseteq S'_i$ for every i. At a high level, it does this by adding requests to its current state when the invariant is violated that allows it to also move closer to the current state of σ. The algorithm Sensible-ALG uses the following analog of the symmetric difference of two matchings to augment its current state in a cost-efficient manner.

▶ **Definition 13** ((A, B)-difference graph). *Let H be a (multi-)graph with vertex set R, and A, B be states. The graph H is an (A, B)-*difference graph *if it is a T-join with $T = A \triangle B$: for every $r \in R$, the degree of r in H is odd if and only if $r \in A \triangle B$.*

For example, one way to obtain an (A, B)-difference graph is by taking a perfect matching M_A on A and a perfect matching M_B on B and then taking the symmetric difference of M_A and M_B.

▶ **Definition 14** (P-difference graph and realisable difference graphs)**.** *Let A, B be states and P be a transition path between A and B. The P-*difference graph $\mathrm{diff}(P)$ *is a (multi-)graph on vertex set R where the multiplicity of the edge (p, q) is equal to the number of transitions along the path P that adds or removes the $\{p, q\}$. An (A, B)-difference graph H is* realisable *if it is the P-difference graph for a transition path P between A, B.*

Note that if P is a transition path between states A, B, then a P-difference graph is an (A, B)-difference graph. Moreover, if P is a shortest path between S and S', then the total cost of the edges in $\mathrm{diff}(P)$ is exactly equal to $c(S, S')$. We also note that not all difference graphs are realisable[2]

We now characterise the structure of difference graphs that correspond to shortest paths in the transition graph.

▶ **Definition 15** (Canonical difference graphs)**.** *Let H be an (A, B)-difference graph. We say that H is* canonical *if it can be decomposed into a collection of $\ell := |A \triangle B|/2$ edge-disjoint paths $Q_1, \ldots, Q_\ell$ between disjoint pairs (p_i, q_i) of $A \triangle B$ such that for each i, Q_i consists of a single edge (p_i, q_i) if either both p, q are in $A \setminus B$ or both in $B \setminus A$, and Q_i consists of two edges $(p, s), (s, q)$ for some other request s if exactly one of p, q is in $A \setminus B$ and the other in $B \setminus A$.*

▶ **Proposition 16.** *If H is a canonical (A, B)-difference graph, then H is realisable.*

Proof. We show by induction on $\ell = |A \triangle B|/2$ that there is a transition path P from A to B. Consider the base case $\ell = 1$. Suppose Q_1 consists of a single edge (p, q). If $p, q \in A \setminus B$, then we can transition directly from A to B by removing p, q; otherwise, we add p, q. Next, suppose that Q_1 consists of two edges $(p, s), (s, q)$. Consider the case that $p \in A \setminus B$ and $q \in B \setminus A$. If $s \in A \cap B$, then we can transition from A to B by first removing p, s and then adding s, q; otherwise we first add s, q and then remove p, s. The case when $q \in A \setminus B$ and $p \in B \setminus A$ follows similarly. In all of these cases, we have that H is exactly the P-difference graph where P is the transition path corresponding to the sequence of transitions used.

Suppose the statement is true for ℓ up to $k - 1$ and let H be a canonical difference graph with k edge-disjoint paths. Let A' be the state after executing the above procedure on Q_1 and P_1 be the transition path corresponding to the sequence of transitions from A to A'. As argued above, Q_1 is the P_1-difference graph. Since $H \setminus Q_1$ is a canonical (A', B)-difference graph with $k - 1$ edge-disjoint paths, we can apply induction to get a transition path P_2 from A' to B such that $H \setminus Q_1$ is the P_2-difference graph. By concatenating P_1 and P_2, we get a transition path P from A to B. Moreover, H is the P-difference graph, as desired. ◀

▶ **Lemma 17.** *Let A, B be states. There exists a shortest path P in the transition graph between A, B such that* $\mathrm{diff}(P)$ *is canonical.*

Proof. Since $\mathrm{diff}(P)$ is an $(A \triangle B)$-join, it contains a collection of $\ell := |A \triangle B|/2$ edge-disjoint paths $Q_1, \ldots, Q_\ell$ connecting disjoint pairs of requests $p_i, q_i \in A \triangle B$.

We now shortcut these paths to produce a canonical (A, B)-difference graph H. In particular, for each i, if $p_i, q_i \in A \setminus B$ or $p_i, q_i \in B \setminus A$, add the edge (p_i, q_i) to H; otherwise, $p_i \in A \setminus B$ and $q_i \in B \setminus A$ (or vice versa), there must be an intermediate node s_i on the path

[2] For example, consider the difference graph H consisting of requests p, q, r and edges (p, q) and (q, r). Observe that H is an $(\{p, r\}, \emptyset)$-difference graph but there is no transition path P from $\{p, r\}$ to $\emptyset$ such that $\mathrm{diff}(P) = H$.

Q_i and we add the edges $(p_i, s_i), (s_i, q_i)$. The existence of s_i is because if Q_i only consists of a single edge (p_i, q_i) then p_i and q_i must be both in $A \setminus B$ or $B \setminus A$, otherwise the transition in P that adds or removes p_i, q_i is invalid.

By triangle inequality, the total cost of the edges in H is at most that of $Q_1 \cup \cdots \cup Q_\ell$ which in turn is contained in $\mathrm{diff}(P)$. Since H is a canonical (A, B)-difference graph, there exists a transition path P' with $\mathrm{diff}(P') = H$. Since $c(P')$ is equal to the total cost of the edges in $\mathrm{diff}(P') = H$, we get the lemma. ◀

▶ **Proposition 18.** *Let A, B be states such that $|A| = |B| + 2$. Let P be a shortest path in the transition graph between A, B such that $\mathrm{diff}(P)$ is canonical and $Q_1, \ldots, Q_\ell$ be the corresponding collection of $|A \triangle B|/2$ edge-disjoint paths connecting disjoint pairs of $A \triangle B$. Then, one such path Q_i is a single edge (p, q) with $p, q \in A \setminus B$.*

This proposition follows from the fact that $|A \setminus B| = |B \setminus A| + 2$ and so there must be a path Q_i connecting $p, q \in A \setminus B$. By definition of canonical difference graphs, Q_i is a single edge (p, q).

We are now ready to formally define Sensible-ALG.

Description of Sensible-ALG. Our online algorithm takes as input a sequence of states $\sigma = (S_1, \ldots, S_{|\sigma|})$ produced by an online scheduling algorithm. We assume that σ satisfies that for all $0 \leq i < |\sigma| - 1$, S_i and S_{i+1} are neighbours in G. This is without loss of generality: if the input schedule does not satisfy these properties then we can add intermediate timesteps and all states on the shortest path between S_i and S_{i+1} such that it satisfies the above property, and the cost remains the same. When a new state S_i arrives, if $|S_i| \leq |S'_{i-1}|$, the algorithm sets $S'_i = S'_{i-1}$; otherwise, it computes a shortest transition path P between S_i and S'_{i-1} such that $\mathrm{diff}(P)$ is canonical, and sets $S'_i = S'_{i-1} \cup \{p, q\}$ where $p, q \in S_i \setminus S'_{i-1}$ and (p, q) is a path in the path decomposition of $\mathrm{diff}(P)$.

Since Sensible-ALG always transitions to a state that is a superset of its previous state, its solution σ' is monotone.

Next, we analyse the transition cost of σ'.

▶ **Lemma 19.** $\sum_{i=0}^{|\sigma|-1} c(S'_{i-1}, S'_i) \leq \sum_{i=0}^{|\sigma|-1} c(S_{i-1}, S_i)$.

Proof. We prove this lemma by introducing the potential $\phi_i = c(S_i, S'_i)$.

We claim that in each iteration i, the transition cost of σ' is at most the transition cost of σ plus the decrease in the potential.

▷ Claim 20. For all i, we have $c(S'_{i-1}, S'_i) \leq c(S_{i-1}, S_i) - (\phi_i - \phi_{i-1})$.

Proof of Claim 20. By triangle inequality, we have

$$c(S_i, S'_{i-1}) \leq c(S_{i-1}, S_i) + c(S_{i-1}, S'_{i-1}). \tag{1}$$

Next, we will show that

$$c(S'_{i-1}, S'_i) \leq c(S_i, S'_{i-1}) - c(S_i, S'_i). \tag{2}$$

Combining these two inequalities yields the claim.

Observe that Inequality (2) holds when $S'_i = S'_{i-1}$. Suppose $S'_i \neq S'_{i-1}$. In this case, the algorithm computes the shortest path P between S_i and S'_{i-1} that is canonical and set $S'_i = S'_{i-1} \cup \{p, q\}$ where $p, q \in S_i \setminus S'_{i-1}$ and (p, q) is a path in the path decomposition of $\mathrm{diff}(P)$. We have that $\mathrm{diff}(P) \setminus (p, q)$ is a canonical (S_i, S'_i)-difference graph and thus, by

Proposition 16, corresponds to a transition path between S_i, S'_i with length $c(S_i, S'_{i-1}) - c(p,q) = c(S_i, S'_{i-1}) - c(S'_{i-1}, S'_i)$. Thus, $c(S_i, S'_i) \leq c(S_i, S'_{i-1}) - c(S'_{i-1}, S'_i)$. Rearranging this inequality yields Inequality (2). This completes the proof of the claim. ◁

Using Claim 20, we determine the total transition cost incurred by σ' as follows.

$$\begin{aligned}\sum_{i=1}^{|\sigma|} c(S'_{i-1}, S'_i) &\leq \sum_{i=1}^{|\sigma|} c(S_{i-1}, S_i) - \sum_{i=1}^{|\sigma|}(\phi_i - \phi_{i-1})\\ &= \sum_{i=1}^{|\sigma|} c(S_{i-1}, S_i) - \phi_{|\sigma|} + \phi_0\\ &\leq \sum_{i=1}^{|\sigma|} c(S_{i-1}, S_i).\end{aligned}$$

The last inequality holds because $\phi_0 = 0$ and the potential is non-negative. ◀

The cost of an MPMD-Set-MTS solution consists of the transitions cost, as well as the processing cost. Since in every iteration i, we have $|S'_i| \geq |S_i|$ and the processing cost of a state is a monotone non-increasing function of the size of the state, we get that the total processing cost of σ' is at most that of σ. Together with Lemma 19, we get that the total cost of σ' is at most that of σ. This concludes the proof of Lemma 12. ◀

4.3 Applying MTS Algorithms to MPMD-Set-MTS

In this section, we prove Corollary 3. Consider an instance of MPMD-Set with m requests in a metric space of n points and the instance of MPMD-Set-MTS created by applying Theorem 2. Let N be the number of states of the MPMD-Set-MTS instance.

There are two issues that arise when applying MTS algorithms to MPMD-Set-MTS directly.

4.3.1 Eliminating the Diameter

The first issue is that all known MTS algorithms have a cost bound of the form $f(N) \cdot cost(OPT) + D$ where OPT is the optimal MTS solution and D is the diameter of the MTS state space. Observe that D is at least the distance between the empty matching and the max-cost perfect matching, i.e. the cost of the max-cost perfect matching. Unfortunately, the cost of the max-cost perfect matching can be much larger than that of the optimal solution. To overcome this, one could restrict the MTS solution to only use states whose distance from the initial state is at most $cost(OPT)$. This can be achieved by setting the costs of the other states to be infinite. This effectively reduces the diameter of the state space to at most $2 \cdot cost(OPT)$, and would give us a cost bound of $O(f(N)) \cdot cost(OPT) + 2 \cdot cost(OPT)$. The issue is that, since the MTS tasks arrive in an online fashion, the optimal solution remains unknown until all tasks have arrived. To address this issue we use the Guess-and-Double Method, which, maintains a guess of the value of the optimal solution as the tasks are processed. This guess is used to determine the diameter of the state space used by the algorithm. When the guess becomes too small, the value of the guess is increased and the algorithm is simulated on the input that has already been processed, only this time on the larger state space, to determine the state it would now be in, and processes the new tasks accordingly. For the benefit of the reader, we give a high level overview of the method below.

The Guess-and-Double Method. Let $R = [r_1, r_2, ...r_T]$ be the sequence of tasks given by the MTS problem. Let OPT_t be the cost of the optimal solution for processing the tasks that have arrived up to and including time t. At any time t we maintain a *guess* j such that

$$2^{j-1} < OPT_t \leq 2^j. \tag{3}$$

When our guess j no longer satisfies (3) we increase it's value (thus doubling the radius of the metric space) until it is back within our bounds. Each new guess instantiates a new *phase*. At the start of each phase, we *restart* the MTS algorithm with a superset of the previous state space, which now consists of all states within distance 2^j from the original start state. Note that by *restart* we mean that we simulate the MTS algorithm with the new diameter on all tasks that have arrived so far, and process the new tasks in accordance with the decisions made by the algorithm with the new diameter.

Our initial guess j satisfies $2^{j-1} < OPT_1 \leq 2^j$. We define MTS_j to be the MTS algorithm that operates on the given state space with diameter 2^j. Let $R_t = [r_1, ..., r_t]$ be the sequence of tasks that have arrived up to and including timestep t. We define $MTS_j(R_t)$ to be the state MTS_j would end up in after processing R_t. The Guess-and-Double method, for every timestep t, maintains a guess j that satisfies (3), and moves to $MTS_j(R_t)$.

Algorithm 1 Updating the guess and splitting the tasks into phases.

```
Initialise the first phase.
Choose $j$ such that $2^{j-1} < OPT_1 \leq 2^j$.
for Every timestep $t$ do
    if $2^{j-1} < OPT_t \leq 2^j$ then
        Move to state $MTS_j(R_t)$
    end
    else
        End the previous phase and initialise a new phase.
        Update the value of $j$ such that $2^{j-1} < OPT_t \leq 2^j$.
        Move to state $MTS_j(R_t)$
    end
end
```

Note that each time we update the value of our guess, the value of the optimal solution has at least doubled since we made our last guess.

To analyse the total cost of the resulting schedule we look at two separate costs. The first is the cost incurred within each phase. This includes the transition costs incurred during the phase (from moving between states), plus the cost associated with servicing all tasks that arrived during the phase in the states the algorithm moved through during the phase. We refer to this cost as the *internal* phase cost. The second cost consist of the transition costs incurred in moving between the last state of a phase i, and the first state of the consecutive phase $i+1$. We refer to this cost as the *external* phase cost.

We start by bounding the internal phase cost. Let $cost(MTS_j)$ denote the internal phase cost of the phase associated with guess j. Because $cost(MTS_j)$ can be upper bounded by the cost the algorithm would have incurred for processing all tasks that arrived prior to and during the phase associated with guess j, we can bound $cost(MTS_j)$ as follows:

$$cost(MTS_j) \leq (O(f(N)) + 2) \cdot cost(OPT_j).$$

Therefore, the total internal phase cost incurred over all k phases is

$$\begin{aligned}\sum_{i=1}^{k} ALG_i &\leq (O(f(N)) + 2) \cdot \sum_{i=1}^{k} cost(OPT_i)\\ &\leq 2 \cdot (O(f(N)) + 2) \cdot cost(OPT_k).\end{aligned}$$

Next, we bound the external phase cost. Let S_i be the last state of a given phase i, associated with a guess j, and let S'_{i+1} be the first state of phase $i+1$, associated with the next guess j'. We bound $c(S_i, S'_{i+1})$[3] as follows:

Let S_0 be the empty start state.

$$c(S_i, S'_{i+1}) \leq c(S_i, S_0) + c(S'_{i+1}, S_0).$$

For all consecutive phases i and $i+1$, associated with guesses j and j' respectively, it holds that

$$c(S_i, S_0) \leq (O(f(N)) + 2) \cdot cost(OPT_i)$$

and

$$c(S'_{i+1}, S_0) \leq (O(f(N)) + 2) \cdot cost(OPT_{i+1}).$$

We thus bound the cost over all $k-1$ phase transitions as follows

$$\begin{aligned}\sum_{i=1}^{k-1} c(S_i, S'_{i+1}) &\leq 2 \cdot \sum_{i=1}^{k-1} (O(f(N)) + 2) \cdot cost(OPT_{i+1})\\ &\leq 4 \cdot (O(f(N)) + 2) \cdot cost(OPT_k).\end{aligned}$$

The total cost of the solution produced by the Guess-and-Double method can thus be bounded by $6 \cdot (O(f(N)) + 2) \cdot cost(OPT_k)$.

We can now use the $O(N)$-competitive deterministic algorithm of [15] to obtain our deterministic algorithm for MPMD-Size.

4.3.2 The Need for an Online Embedding

The second issue stems from the fact that the reduction in Theorem 2 creates an MTS instance where the states are arriving over time. This is because the states correspond to matchings of requests and the requests are arriving online. This does not pose a problem for the deterministic $O(N)$-competitive Work Function Algorithm of [15]. However, we cannot directly apply the current-best randomised algorithm for MTS of [16] as it pre-computes a probabilistic embedding of the MTS metric space into a hierarchically separated tree (HST). Instead, we need to use a probabilistic online embedding into a HST together with the $O(\log N)$-competitive randomized algorithm for MTS on HSTs of [16]. Using the online embedding of [24] adds a factor of $O(\log N \log \Phi)$ where Φ is the ratio of the largest distance to the smallest distance in the MTS state space, i.e. the aspect ratio. However, Φ can be arbitrarily large. We deal with this by proving that the Abstract Network Design Framework of [10] can be extended to apply to MTS. For the benefit of the reader, we give a short overview of the Abstract Network Design Framework.

[3] Recall that we denote the transition cost of going from state S_i to S'_{i+1} by $c(S_i, S'_{i+1})$.

The Abstract Network Design Framework. In an instance of abstract network design, the algorithm is given a connected graph $G(V, E)$ with edge lengths $d : E \rightarrow \mathbb{R}_+$. At each timestep i, a requests that consists of a set of points in the graph, called *terminals*, arrive in an online fashion. The algorithm provides a response $R_i = (G_i, C_i)$, which consists of a subgraph $G_i \subseteq E$, and a connectivity list C_i, which is an ordered subset of terminal pairs from the terminals that have arrived so far, and determines what will become (and remain) connected from this timestep onwards. The algorithm is given, at each timestep, a feasibility function $F_i : (C_1, ..., C_i) \rightarrow \{0, 1\}$ that maps a sequence of connectivity lists to either 0 (infeasible) or 1 (feasible). A solution S_i, which consists of a sequence of responses for every time step up to and including timestep i, is feasible if $F_i(C_1, ..., C_i) = 1$ and all pairs in C_j are connected in all G_i for all $i \geq j$. To determine the cost of a solution to the first i requests S_i, the framework uses a load function $\rho_i : 2^{\{1,...,i\}} \rightarrow \mathbb{R}_+$, which takes as input the sequence of timesteps in which the edge was used, and outputs a corresponding multiplier to the cost of an edge $d(e)$. It aims to model how the cost of using an edge grows as the edge is used multiple times. The function must be subadditive, monotone non-decreasing, and satisfy $\rho_i(I) = 0$ if and only if $I = \emptyset$. The total cost of a solution S_i is defined as follows.

$$cost(S_i) = \sum_{e \in E} d(e) \cdot \rho(\{j \leq i : e \in G_j\})$$

Extending the Abstract Network Design Framework. Though we can express the transition cost of an instance of general MTS using this framework, we cannot express the cost vectors associated with processing the tasks in MTS in the current state of the framework. In order to address this issue, we propose the following alterations to generalise the framework.

We replace the feasibility function with a function $F'_i : (C_1, ..., C_i) \rightarrow \mathbb{R}_+$, where $F'_i(C_1, ..., C_i)$ is the processing cost of the algorithm during timestep i if the solution $S_i = ((G_1, C_1), \ldots, (G_i, C_i))$ is feasible, and ∞ otherwise.

We now re-define the cost of a solution to the first i requests S_i to incorporate the total processing cost incurred by the algorithm.

$$cost(S_i) = \sum_{e \in E} d(e) \cdot \rho(\{j \leq i : e \in G_j\} + \sum_{l=1}^{i} F'(C_1, \ldots, C_l)$$

Since the processing cost provided to the algorithm is independent of the metric space, it follows that the processing cost remains unaffected by the online embedding. It therefore does not affect the overhead due to the online embedding.

Note that the extension of this framework means it can now be used to model online problems with delay, where the delay cost can be modelled as the processing cost.

Expressing MTS in the Abstract Network Design Framework. It remains to formulate the general MTS problem in the Extended Abstract Network Design Framework defined above.

We define the terminal set of the ith request to be the set of states that have arrived so far. We define the cost of an edge $d((u, v))$ to be the transition cost between states u, v. Let $\mathbb{T}_i$ be the cost vector associated with processing task i, and $T_i(w)$ be the cost of processing task i in state w. Let v be the last terminal in C_i. The extended feasibility function F_i is defined by $F_i(C_1, \ldots, C_i) = T_i(v)$ if there is only a single ordered pair in each C_j for all $j \leq i$ and the sequence of $(C_1, \ldots, C_i)$ is a valid path. The load function is simply the cardinality function because we pay the transition cost associated with the edge each time we transition to a different state.

Min-operator. Bartal et al. [10] show that if the problem can be captured by the Abstract Network Design Framework and admits a *min-operator*, it is possible to reduce the overhead due to the online embedding to $O(\log^3 N)$.

▶ **Definition 21** (Min-operator). *An algorithm admits a min-operator with factor $\mu \geq 1$ if there exists a competitive algorithm*[4] *for the problem, and for any two deterministic online algorithms A and B*[5]*, there exists a third online deterministic algorithm C such that the cost of C satisfies $cost(C) \leq \mu \cdot \min\{cost(A), cost(B)\}$, where $cost(A)$ and $cost(B)$ are the respective costs of algorithms A and B. If either algorithms A or B are randomised, the expected cost of C must satisfy $E[cost(C)] \leq \mu \cdot \min\{E[cost(A)], E[cost(B)]\}$.*

We have shown that the results by Bartal et al. [10] also hold for the extended Abstract Network Design Framework. Since the MTS problem in general admits a min-operator [2], using the framework of [10] allows us to reduce the overhead due to the online embedding to $O(\log^3 N)$ for an overall competitive ratio of $O(\log^4 N)$.

References

1 Itai Ashlagi, Yossi Azar, Moses Charikar, Ashish Chiplunkar, Ofir Geri, Haim Kaplan, Rahul Makhijani, Yuyi Wang, and Roger Wattenhofer. Min-cost bipartite perfect matching with delays. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017*, volume 81 of *LIPIcs*, pages 1:1–1:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.APPROX-RANDOM.2017.1.

2 Yossi Azar, Andrei Z. Broder, and Mark S. Manasse. On-line choice of on-line algorithms. In *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms*, pages 432–440. ACM/SIAM, 1993. URL: http://dl.acm.org/citation.cfm?id=313559.313847.

3 Yossi Azar, Ashish Chiplunkar, and Haim Kaplan. Polylogarithmic bounds on the competitiveness of min-cost perfect matching with delays. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 1051–1061. SIAM, 2017. doi:10.1137/1.9781611974782.67.

4 Yossi Azar, Ashish Chiplunkar, Shay Kutten, and Noam Touitou. Set cover with delay – Clairvoyance is not required. In *Proceedings of the 28th Annual European Symposium on Algorithms, ESA 2020*, volume 173 of *LIPIcs*, pages 8:1–8:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

5 Yossi Azar, Yuval Emek, Rob van Stee, and Danny Vainstein. The price of clustering in bin-packing with applications to bin-packingwith delays. In *Proceedings of the 31st ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2019*, pages 1–10. ACM, 2019. doi:10.1145/3323165.3323180.

6 Yossi Azar and Amit Jacob Fanani. Deterministic min-cost matching with delays. *Theory Comput. Syst.*, 64(4):572–592, 2020. doi:10.1007/s00224-019-09963-7.

7 Yossi Azar, Arun Ganesh, Rong Ge, and Debmalya Panigrahi. Online service with delay. *ACM Trans. Algorithms*, 17(3):23:1–23:31, 2021. doi:10.1145/3459925.

8 Yossi Azar, Runtian Ren, and Danny Vainstein. The min-cost matching with concave delays problem. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 301–320. SIAM, 2021. doi:10.1137/1.9781611976465.20.

[4] This algorithm may be randomised.

[5] Note that these algorithms need not be competitive on all instances of the problem.

9 Yossi Azar and Noam Touitou. Beyond tree embeddings – A deterministic framework for network design with deadlines or delay. In Sandy Irani, editor, *Proceedings of the 61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 1368–1379. IEEE, 2020. doi:10.1109/FOCS46700.2020.00129.

10 Yair Bartal, Nova Fandina, and Seeun William Umboh. Online probabilistic metric embedding: A general framework for bypassing inherent bounds. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020*, pages 1538–1557. SIAM, 2020. doi:10.1137/1.9781611975994.95.

11 Marcin Bienkowski, Martin Böhm, Jaroslaw Byrka, Marek Chrobak, Christoph Dürr, Lukáš Folwarczný, Lukasz Jez, Jirí Sgall, Kim Thang Nguyen, and Pavel Veselý. Online algorithms for multilevel aggregation. *Oper. Res.*, 68(1):214–232, 2020. doi:10.1287/opre.2019.1847.

12 Marcin Bienkowski, Artur Kraska, Hsiang-Hsuan Liu, and Pawel Schmidt. A primal-dual online deterministic algorithm for matching with delays. In *Approximation and Online Algorithms – Proceedings of the 16th International Workshop, WAOA 2018, Helsinki, Finland, August 23-24, 2018, Revised Selected Papers*, volume 11312 of *Lecture Notes in Computer Science*, pages 51–68. Springer, 2018. doi:10.1007/978-3-030-04693-4_4.

13 Marcin Bienkowski, Artur Kraska, and Pawel Schmidt. A match in time saves nine: Deterministic online matching with delays. In *Approximation and Online Algorithms – Proceedings of the 15th International Workshop, WAOA 2017*, volume 10787 of *Lecture Notes in Computer Science*, pages 132–146. Springer, 2017. doi:10.1007/978-3-319-89441-6_11.

14 Marcin Bienkowski, Artur Kraska, and Pawel Schmidt. Online service with delay on a line. In Zvi Lotker and Boaz Patt-Shamir, editors, *Structural Information and Communication Complexity – Proceedings of the 25th International Colloquium, SIROCCO 2018, Ma'ale HaHamisha, Israel, June 18-21, 2018, Revised Selected Papers*, volume 11085 of *Lecture Notes in Computer Science*, pages 237–248. Springer, 2018. doi:10.1007/978-3-030-01325-7_22.

15 Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal on-line algorithm for metrical task system. *J. ACM*, 39(4):745–763, 1992. doi:10.1145/146585.146588.

16 Sébastien Bubeck, Michael B. Cohen, James R. Lee, and Yin Tat Lee. Metrical task systems on trees via mirror descent and unfair gluing. *SIAM J. Comput.*, 50(3):909–923, 2021. doi:10.1137/19M1237879.

17 Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Ohad Talmon. O(depth)-competitive algorithm for online multi-level aggregation. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 1235–1244. SIAM, 2017. doi:10.1137/1.9781611974782.80.

18 Ryder Chen, Jahanvi Khatkar, and Seeun William Umboh. Online weighted cardinality joint replenishment problem with delay. In *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming, ICALP 2022*, volume 229 of *LIPIcs*, pages 40:1–40:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

19 Yuval Emek, Shay Kutten, and Roger Wattenhofer. Online matching: haste makes waste! In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016*, pages 333–344. ACM, 2016. doi:10.1145/2897518.2897557.

20 Yuval Emek, Yaacov Shapiro, and Yuyi Wang. Minimum cost perfect matching with delays for two sources. *Theor. Comput. Sci.*, 754:122–129, 2019. doi:10.1016/j.tcs.2018.07.004.

21 Leah Epstein. On bin packing with clustering and bin packing with delays. *Discret. Optim.*, 41:100647, 2021. doi:10.1016/j.disopt.2021.100647.

22 Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995. doi:10.1137/S0097539793242618.

23 Anupam Gupta, Amit Kumar, and Debmalya Panigrahi. Caching with time windows. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1125–1138. ACM, 2020. doi:10.1145/3357713.3384277.

24 Piotr Indyk, Avner Magen, Anastasios Sidiropoulos, and Anastasios Zouzias. Online embeddings. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 13th International Workshop, APPROX 2010, and proceedings of the 14th International Workshop, RANDOM 2010*, volume 6302 of *Lecture Notes in Computer Science*, pages 246–259. Springer, 2010.

25 Predrag Krnetic, Darya Melnyk, Yuyi Wang, and Roger Wattenhofer. The k-server problem with delays on the uniform metric space. In Yixin Cao, Siu-Wing Cheng, and Minming Li, editors, *31st International Symposium on Algorithms and Computation, ISAAC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 181 of *LIPIcs*, pages 61:1–61:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ISAAC.2020.61.

26 Ngoc Mai Le, Seeun William Umboh, and Ningyuan Xie. The power of clairvoyance for multi-level aggregation and set cover with delay. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2023. To appear.

27 Xingwu Liu, Zhida Pan, Yuyi Wang, and Roger Wattenhofer. Impatient online matching. In *Proceedings of the 29th International Symposium on Algorithms and Computation, ISAAC 2018*, volume 123 of *LIPIcs*, pages 62:1–62:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ISAAC.2018.62.

28 Noam Touitou. Nearly-tight lower bounds for set cover and network design with deadlines/delay. In *Proceedings of the 32nd International Symposium on Algorithms and Computation, ISAAC 2021*, volume 212 of *LIPIcs*, pages 53:1–53:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ISAAC.2021.53.

An Approximation Algorithm for the Exact Matching Problem in Bipartite Graphs

Anita Dürr
Department of Computer Science, ETH Zürich, Switzerland

Nicolas El Maalouly
Department of Computer Science, ETH Zürich, Switzerland

Lasse Wulf
Institute of Discrete Mathematics, TU Graz, Austria

Abstract

In 1982 Papadimitriou and Yannakakis introduced the Exact Matching problem, in which given a red and blue edge-colored graph G and an integer k one has to decide whether there exists a perfect matching in G with exactly k red edges. Even though a randomized polynomial-time algorithm for this problem was quickly found a few years later, it is still unknown today whether a deterministic polynomial-time algorithm exists. This makes the Exact Matching problem an important candidate to test the **RP**=**P** hypothesis.

In this paper we focus on approximating Exact Matching. While there exists a simple algorithm that computes in deterministic polynomial-time an *almost* perfect matching with exactly k red edges, not a lot of work focuses on computing perfect matchings with *almost* k red edges. In fact such an algorithm for bipartite graphs running in deterministic polynomial-time was published only recently (STACS'23). It outputs a perfect matching with k' red edges with the guarantee that $0.5k \leq k' \leq 1.5k$. In the present paper we aim at approximating the number of red edges without exceeding the limit of k red edges. We construct a deterministic polynomial-time algorithm, which on bipartite graphs computes a perfect matching with k' red edges such that $\frac{k}{3} \leq k' \leq k$.

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

Keywords and phrases Perfect Matching, Exact Matching, Red-Blue Matching, Approximation Algorithms, Bounded Color Matching

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.18

Category APPROX

Related Version *Full Version*: http://arxiv.org/abs/2307.02205

Funding *Lasse Wulf*: Supported by the Austrian Science Fund (FWF): W1230.

1 Introduction

In the Exact Matching problem (denoted as EM) one is given a graph G whose edges are colored in red or blue and an integer k and has to decide whether there exists a perfect matching M in G containing exactly k red edges. This problem was first introduced in 1982 by Papadimitriou and Yannakakis [20] who conjectured it to be **NP**-complete. However a few years later, Mulmuley et al. [19] showed that the problem can be solved in randomized polynomial-time, making it a member of the class **RP**. This makes it unlikely to be **NP**-hard. In fact, while it is commonly believed that **RP**=**P**, the proof of this statement is a major open problem in complexity theory. Since EM is contained in **RP** but a deterministic polynomial-time algorithm for it is not known, the problem is a good candidate for testing the **RP**=**P** hypothesis. Actually Mulmuley et al. [19] even showed that EM is contained in **RNC**, which is the class of decision problems that can be solved by a polylogarithmic-time algorithm using polynomially many parallel processors while having access to randomness. As studied in [22], derandomizing the perfect matching problem from this complexity class is also a big open problem. This makes EM even more intriguing as randomness allows it to be efficiently parallelizable but we don't even know how to solve it sequentially without randomness.

Due to this, numerous works cite EM as an open problem. This includes the above mentioned seminal work on the parallel computation complexity of the matching problem [22], planarizing gadgets for perfect matchings [14], multicriteria optimization [13], matroid intersection [5], DNA sequencing [4], binary linear equation systems with small hamming weight [2], recoverable robust assignment [10] as well as more general constrained matching problems [3, 17, 21, 18]. Progress in finding deterministic algorithms for EM has only been made for very restricted classes of graphs, thus illustrating the difficulty of the problem. There exists a deterministic polynomial-time algorithm for EM in planar graphs and more generally in $K_{3,3}$-minor free graphs [25], as well as for graphs of bounded genus [11]. Additionally, standard dynamic programming techniques on the tree decomposition of the graph can be used to solve EM on graphs of bounded treewidth [7, 23]. Contrary to those classes of sparse graphs, EM on dense graphs seems to be harder to solve. At least 4 articles study solely complete and complete bipartite graphs [16, 12, 24, 15]. More recently those results were generalized in [8] with a polynomial-time algorithm solving EM for constant independence number[1] α and constant bipartite independence number[2] β. These algorithm have **XP**-time, i.e. they run in time $O(g(\alpha)n^{f(\alpha)})$ and $O(g(\beta)n^{f(\beta)})$. This was further improved for bipartite graphs in [9] where the authors showed an FPT algorithm parameterized by the bipartite independence number β solving EM on bipartite graphs.

Approximating EM

Given the unknown complexity status of EM, it is natural to try approximating it. In order to approximate EM, we need to allow for non-optimal solutions. Here two directions can be taken: either we consider non-perfect matchings, or we consider perfect matchings with not exactly k red edges. Yuster [25] took the first approach and showed an algorithm that computes an *almost perfect exact matching* (a matching with exactly k red edges and either $\frac{n}{2}$ or $\frac{n}{2} - 1$ total edges[3]). This is as close as possible for this type of approximation. However, there are two things to consider with Yuster's algorithm: First, the algorithm itself is very simple. Secondly, as long as a trivial condition on k is met, such an almost perfect exact matching always exists. It is surprising that this approximation is achieved by such simple methods in such generality. For the closely related *budgeted matching* problem, more sophisticated methods have been used recently to achieve a PTAS [3] and efficient PTAS [1]. These methods also do not guarantee to return a perfect matching.

The alternative is to find approximations which always return a perfect matching, but with possibly the wrong number of red edges. These kinds of approximations seem considerably more difficult to tackle. It is puzzling how little is known about this type of approximation, while the other type of approximation admits much stronger results. Very recently, El

[1] The independence number of a graph G is defined as the largest number α such that G contains an independent set of size α.

[2] The bipartite independence number of a bipartite graph G is defined as the largest number β such that G contains a balanced independent set of size 2β, i.e. an independent set using exactly β vertices from each partition.

[3] n is the number of vertices in the graph.

Maalouly [7] presented a polynomial-time algorithm which for bipartite graphs outputs a perfect matching containing k' red edges with $0.5k \leq k' \leq 1.5k$. To the best of our knowledge, this is the only result in this direction, despite the original EM problem being from 1982. Moreover, note that this algorithm can return both a perfect matching with too many or too few red edges. Hence El Maalouly's algorithm is not an approximation algorithm in the classical sense, as it only guarantees a two-sided, but not a one-sided, approximation.

Our contribution

We consider the problem of approximating EM, such that a perfect matching is always returned, and such that the approximation is one-sided. We show the first positive result of this kind. Formally, we consider the following optimization variant of EM:

Exact Matching Optimization (EM-opt)
Input: A graph G whose edges are colored red or blue; and an integer k.
Task: Maximize $|R(M)|$ subject to the constraint that M is a perfect matching in G and $|R(M)| \leq k$, where $R(M)$ is the set of red edges in M.

We consider bipartite graphs and prove:

▶ **Theorem 1.** *There exists a deterministic polynomial-time 3-approximation for EM-opt in bipartite graphs.*

We remark that there are three kinds of input instances to EM-opt : First, instances which are also YES-instances of EM, i.e. a perfect matching with k red edges exists. For those instances, our algorithm returns a perfect matching with k' red edges and $\frac{1}{3}k \leq k' \leq k$. Second, there are instances, where EM-opt is infeasible, in the sense that all perfect matchings have $k+1$ or more red edges. For such instances, our algorithm returns "infeasible". Finally, there are instances which are feasible, but the optimal solution to EM-opt has k^* red edges with $k^* < k$. For such instances, our algorithm returns a perfect matching with k' red edges and $\frac{1}{3}k^* \leq k' \leq k^*$.

The main idea behind our approximation algorithm is to start with an initial perfect matching, and then repeatedly try a small improvement step using dynamic programming. Whenever a small improvement step is not possible, we prove that the graph is "rigid" in a certain sense. We can prove that as a result of this rigidity, it is a good idea to guess a single edge e and to enforce that this edge e is contained in the solution matching. This paper contains concepts and lemmas, which make these ideas formal and may help for the development of future approximation algorithms. Our new ideas are based on a geometric intuition about the cycles appearing in such a "rigid" graph.

2 Preliminaries

2.1 Definitions and notations

All graphs considered are simple. Given a graph $G = (V, E)$, a perfect matching M of G is a subset of E such that every vertex in V is adjacent to exactly one edge in M. We introduce two tools related to M: the weight function w_M and the weighted directed graph G_M. These tools were first introduced in [8, 7]. They can be used to reason about EM, as is explained further below.

▶ **Definition** (Weight function w_M). *Given a graph $G = (V, E)$ whose edges are colored red or blue and a perfect matching M on G, we define the weight function w_M on the edges of G as follows:*

$$w_M(e) = \begin{cases} 0 & \textit{if } e \textit{ is a blue edge} \\ -1 & \textit{if } e \in M \textit{ is a red edge} \\ +1 & \textit{if } e \notin M \textit{ is a red edge} \end{cases}$$

▶ **Definition** (Oriented and edge-weighted graph G_M). *Given a bipartite graph $G = (A \sqcup B, E)$ whose edges are colored red or blue and a perfect matching M on G, the graph G_M is the oriented and edge-weighted graph such that:*

- *the vertex set of G_M is the vertex set $A \sqcup B$ of G;*
- *the edge set of G_M is the edge set of G such that edges in M are oriented from A to B and edges not in M are oriented from B to A;*
- *edges in G_M are weighted according to the weight function w_M.*

For ease of notation, we identify subgraphs of G and G_M with their edge sets. Any reference to their vertices will be made explicit. Since every edge in G_M has an undirected unweighted version in G, every subgraph H_M in G_M can be associated to the subgraph H in G of corresponding undirected and unweighted edge set, and vice versa. With a slight abuse of notation, we therefore sometimes do not differentiate H of H_M and denote them by the same object. If E_1 and E_2 are two edge sets, then we denote by $E_1 \Delta E_2 := (E_1 \setminus E_2) \cup (E_2 \setminus E_1)$ their symmetric difference. Let H be a subgraph of G_M. We use the following notations:

- $R(H)$ denotes the set of red edges in H.
- $E^+(H)$ denotes the set of positive edges in H (i.e. the red edges in H not contained in M).
- $E^-(H)$ denotes the set of negative edges in H (i.e. the red edges in H contained in M).
- $w_M(H)$ is the weight of H with respect to the edge weight function w_M, i.e. $w_M(H) = |E^+(H)| - |E^-(H)|$.

Unless stated otherwise, all considered cycles and paths in G_M are directed. Finally if C is a directed cycle in G_M, and $P_1 \subseteq C$ and $P_2 \subseteq C$ are sub-paths on C, we denote by $C[P_1, P_2]$ (respectively $C(P_1, P_2)$) the sub-path in C from P_1 to P_2 included (resp. excluded). If $w_M(C) > 0$ then we call C a *positive* cycle and if $w_M(C) \leq 0$ we call C a *non-positive* cycle. A *walk* in a directed graph is a sequence of edges $(e_1, \dots, e_t)$ such that the end vertex of e_i is the start vertex of e_{i+1} for $i = 1, \dots, t-1$. In contrast to a path, a walk may visit vertices and edges twice. A walk is *closed*, if its start and end vertex are equal.

2.2 Observations on G_M

In this subsection, we explain some simple observations, which we use later to reason about our approximation algorithm. Let G be a bipartite graph and M be a perfect matching of G. A cycle in G is said to be *M-alternating* if for any two adjacent edges in the cycle, one of them is in M and the other is not. It is a well-known fact that if M and M' are two perfect matchings, then $M \Delta M'$ is a set of vertex-disjoint cycles that are both M-alternating and M'-alternating. We now make the following important observations on the bipartite graphs G and G_M and the weight function w_M.

▶ **Observation 2.** *A cycle C in G is M-alternating if and only if the corresponding cycle C_M in G_M is directed.*

Proof. This follows directly from the definition of G_M. ◀

▶ **Observation 3.** *Let C be a directed cycle in G_M. Then $M' := M \Delta C$ is a perfect matching whose number of red edges is $|R(M')| = |R(M)| + w_M(C)$.*

Proof. Since C is M-alternating, it is a well-known fact that M' is again a perfect matching. The equation $|R(M')| = |R(M)| + w_M(C)$, follows directly from the definition of w_M. If an edge in C is blue, it does not change the amount of red edges of M. If an edge in C is red, it changes the number of red edges of M by ±1, depending on whether or not it is in M. ◀

▶ **Observation 4.** *Let C_1 and C_2 be two directed cycles in G_M that intersect at a vertex v. Then C_1 and C_2 also intersect on an edge adjacent to v.*

Proof. By Observation 2, C_1 and C_2 are M-alternating so they both contain exactly one adjacent edge to v that is also contained in M. Since M is a perfect matching, there exists a single adjacent edge to v in M. Thus C_1 and C_2 both contain this edge. ◀

2.3 Previous ideas

The first part of our algorithm for Theorem 1 presented in Section 3 is strongly inspired by the algorithm in [7, Theorem 1] that computes a perfect matching with between $0.5k$ and $1.5k$ red edges. We summarize the main ideas here.

One can compute a perfect matching of minimum number of red edges in polynomial time by using a maximum weight perfect matching algorithm (e.g. [6]) on the graph G where red edges have weight 0 and blue edges have weight +1. The idea in [7] is to start with a perfect matching of minimum number of red edges M. If $0.5k \leq |R(M)| \leq 1.5k$ then we are already done and can output M. Otherwise $|R(M)| < 0.5k$ (it cannot have more than k red edges by minimality). In that case the algorithm iteratively improves M using positive cycles. Indeed, assume for a moment that we could find a directed cycle C in G_M such that both $w_M(C) > 0$ and $|E^+(C)| \leq k$, then the perfect matching $M' := M \Delta C$ is such that

$$\begin{aligned} |R(M)| < |R(M')| &= |R(M)| + w_M(C) \\ &\leq |R(M)| + |E^+(C)| < 0.5k + k = 1.5k. \end{aligned}$$

Thus we can iteratively compute such cycles C and replace M by $M \Delta C$. We make true progress every time until $0.5k \leq |R(M)| \leq 1.5k$.

An important observation proven in [7] is that we can determine in polynomial time if such a cycle exists. We recall this result in Proposition 5.

▶ **Proposition 5** (Adapted from [7, Proposition 11]). *Let $G := (V, E)$ be an edge-weighted directed graph and $t \in \mathbb{N}$ be a parameter. There exists a deterministic polynomial-time algorithm that, given G, determines whether or not there exists a directed cycle C in G with $w(C) > 0$ and $|E^+(C)| \leq t$. If such a cycle C exists then the algorithm also outputs a cycle C' with the same properties as C (i.e. $w(C') > 0$ and $|E^+(C')| \leq t$).*

The idea of Proposition 5 is to flip the sign of all weights so that we are looking for a negative cycle. Those can be found by shortest path algorithms: for example the Bellman-Ford algorithm which relies on a dynamic program (DP). Each entry of this DP contains the shortest distance between two vertices. The DP can be adapted so that it contains an additional budget constraint corresponding to the number of negative edges (i.e. positive before we flip the sign of the weight) a path is allowed to use. We also add in the entry of the DP the last edge used in the path. This way, when looking at the entries corresponding to the shortest path from a vertex v to the same vertex v we are able to determine whether or not there exists a negative cycle that uses no more than t negative edges (i.e. positive before we flip the sign of the weight). If it is the case then the algorithm can output such a cycle. We redirect the reader for a detailed proof of Proposition 11 in [7, Section 3.1]. We finally remark that the proof in [7] is completed by showing that as long as M does not have between $0.5k$ and $1.5k$ red edges, a cycle C with $w_M(C) > 0$ and $|E^+(C)| \leq k$ always exists. However, to get a one-sided approximation, we need to guarantee the existence of a cycle C with $w_M(C) > 0$ and $|E^+(C)| \leq k - |R(M)|$. This might not be possible for certain graphs if $|R(M)| > 0$. This means that the above result is not enough for our purpose, as it only leads to a two-sided approximation.

2.4 Assumptions

In order to reduce the technical details needed to present our 3-approximation algorithm of EM-OPT, we show in this subsection that several simplifying assumptions can be made about the input instance.

▷ Claim 6. We can assume without loss of generality that the input instance to EM-OPT is also a YES-instance of EM, i.e. there exists a perfect matching with exactly k red edges.

Proof. Fix an instance $\langle G, k\rangle$ of EM-OPT. If the instance is feasible, let $k^* \leq k$ be the number of red edges in an optimal solution of EM-OPT. Suppose $\mathcal{A}$ is an algorithm for EM-OPT, which given an instance $\langle G, k\rangle$ returns a 3-approximate solution to EM-OPT if $k = k^*$, and returns some arbitrary perfect matching otherwise. We devise an algorithm $\mathcal{A}'$, which is a 3-approximation of EM-OPT for the input $\langle G, k\rangle$. First, $\mathcal{A}'$ computes in polynomial time the perfect matchings $M_{\min}$ and $M_{\max}$ with the minimum and maximum number of red edges. If it is not the case that $|R(M_{\min})| \leq k \leq |R(M_{\max})|$, then $\mathcal{A}'$ returns "infeasible". Afterwards, $\mathcal{A}'$ calls $\mathcal{A}$ as a subroutine for all $k' \in \{|R(M_{\min})|, \dots, k\}$ and receives a perfect matching $M_{k'}$ each time. Finally, $\mathcal{A}'$ returns the best among all the matchings $M_{k'}$ that are feasible for EM-OPT. It is easy to see that $\mathcal{A}'$ is a correct 3-approximation, since there will be an iteration with $k' = k^*$. If $\mathcal{A}$ has running time $O(f)$, then $\mathcal{A}'$ has running time $O(nf + f_{\text{Mat}})$, where f_{Mat} denotes the time to deterministically compute a maximum weight perfect matching. ◁

▷ Claim 7. We can assume without loss of generality that for every edge $e \in E$ there exists a perfect matching of G containing e.

Proof. If an edge is not contained in any perfect matching, it is irrelevant and it can be deleted. We can therefore pre-process the graph and check for each edge in time $O(f'_{\text{Mat}})$ if it is irrelevant. Here f'_{Mat} denotes the time to deterministically check if a graph has a perfect matching, e.g. by running a maximum weight perfect matching algorithm. ◁

3 A 3-approximation of Exact Matching

In this section we are proving our main result, Theorem 1. Let $G = (A \sqcup B, E)$ be a red-blue edge-colored bipartite graph and k be an integer, such that they together form an instance of EM-OPT. We work under the assumptions of Section 2.4. In Algorithm 1 we show an algorithm that given G and k outputs a perfect matching containing between $\frac{1}{3}k$ and k red edges. Let us discuss the general ideas before formally analysing the algorithm.

We reuse the idea from [7] to iteratively improve a perfect matching M by a small amount until it has the right amount of red edges. This improvement is done by finding positive cycles in G_M with no more than $\frac{2}{3}k$ positive edges (using the algorithm of Proposition 5). As

Algorithm 1 The 3-approximation algorithm for EM-OPT of Theorem 1.

```
Input: A bipartite graph $G = (A \sqcup B, E)$ and an integer $k \geq 0$, such that they fulfill
       the assumptions from Section 2.4.
Output: $M$, a perfect matching such that $\frac{1}{3}k \leq |R(M)| \leq k$.
1 Compute a perfect matching $M$ of minimal number of red edges.
2 If $|R(M)| \geq \frac{1}{3}k$ then output $M$.
3 Otherwise compute a cycle $C$ in $G_M$ with $w_M(C) > 0$ and $|E^+(C)| \leq \frac{2}{3}k$, or
    determine that no such cycle exists.
4 If such a cycle $C$ exists, set $M \leftarrow M \Delta C$ and go to Step 2.
5 Otherwise there are no positive cycles with $|E^+(C)| \leq \frac{2}{3}k$ in $G_M$. For every red edge
    $e \in R(G)$ do the following:
6     Compute the perfect matching $M^e$ containing $e$ of minimal number of red edges.
7     If $\frac{1}{3}k \leq |R(M^e)| \leq k$ then output $M^e$.
8 Output $\perp$
```

opposed to the algorithm in [7], we do not want to "overshoot" the number of red edges, i.e. the obtained perfect matching cannot have more than k red edges. We therefore constrain the cycle to have $|E^+(C)| \leq \frac{2}{3}k$. This implies $|R(M \Delta C)| \leq k$. However the issue with that approach is that unlike in [7], we cannot guarantee that such a cycle always exists. So what do we do if we are not able find a good cycle? The answer to this question turns out to be the key insight behind our algorithm. For every edge e we define

$$M^e \in \arg\min\{|R(M)| : M \text{ is a perfect matching with } e \in M\}$$

to be a perfect matching of G containing e with minimal number of red edges. Note that by Claim 7, M^e is properly defined. We also note that M^e can be computed in polynomial time.

We claim that in the case that no good cycle is found, the set of matchings M^e "magically" fixes our problems. Specifically, we claim that at least one of the matchings M^e is a sufficient approximation. The rough intuition behind this is the following. The fact that no good cycle exists means that the graph does not contain small substructures, which can be used to modify the current solution M towards a better approximation. In a sense, the graph is rigid. This rigidity means that maybe there could exist some edge e in the optimal solution M^*, such that every perfect matching including e is already quite similar to M^*. Our approximation algorithm simply tries to guess this edge e. After proving Lemma 8, the remaining part of the paper is devoted to quantify and prove this structural statement.

▶ **Lemma 8.** *Algorithm 1 runs in polynomial time and either outputs $\perp$ or a correct 3-approximation of EM-OPT.*

Proof. Note that the first inner loop is executed at most $O(|V|)$ times, since $w_M(C) > 0$ in each iteration, thus $|R(M)|$ is monotonously increasing. Furthermore, every iteration of the loop is executed in polynomial time due to Proposition 5. The second loop is executed at most $O(|E|)$ times. The matching in *Step 1*, as well as the matchings M^e can be computed in polynomial time using a minimum weight perfect matching algorithm. In total, this takes polynomial time. If the algorithm does not output $\perp$, then it outputs either in *Step 7* (which is obviously a 3-approximation), or in *Step 2*, which is a 3-approximation because $k/3 \leq |R(M \Delta C)| = |R(M)| + w_M(C) \leq |R(M)| + |E^+(C)| \leq k$. ◀

All it remains to prove is that Algorithm 1 never outputs $\perp$. This happens if the conditions in *Step 2* and in *Step 7* are not satisfied. In that regard, define the following *critical tuple*.

▶ **Definition** (Critical tuple). *Let (G, k, M^*, M) be such that G is a bipartite graph whose edges are colored red or blue, $k \geq 0$ is an integer and M^* and M are perfect matchings of G. The tuple (G, k, M^*, M) is said to be* critical *if:*

- *All the assumptions in Section 2.4 hold.*
- *M^* is a perfect matching of G with exactly k red edges.*
- *$|R(M)| < \frac{1}{3}k$.*
- *If C is a directed cycle in G_M such that $w_M(C) > 0$ then $|E^+(C)| > \frac{2}{3}k$.*
- *For every edge $e \in R(M^* \setminus M)$ we have $|R(M^e)| < \frac{1}{3}k$.*

Note that if Algorithm 1 outputs $\perp$, we must have a critcal tuple. Indeed, for every edge $e \in R(M^* \setminus M)$, by minimality of M^e, we have $|R(M^e)| \leq k$. So if the algorithm returns $\perp$, then it must be the case that $|R(M^e)| < k/3$ for all $e \in R(G)$ so in particular also for all $e \in R(M^* \setminus M)$. We will prove the following: critical tuples do not exist. This means that Algorithm 1 never outputs $\perp$ and is therefore a correct 3-approximation algorithm.

▶ **Lemma 9.** *A tuple (G, k, M^*, M) can never be critical.*

3.1 Overview of the main proof

In this section, we explain the idea behind the proof of our main lemma, Lemma 9. From this point onward, consider a fixed tuple (G, k, M^*, M). We prove Lemma 9 by contradiction and therefore assume that (G, k, M^*, M) is a critical tuple. The main strategy of the proof is to find a set of cycles with contradictory properties. Such a set of cycles is called a *target set*. We first introduce a special cycle, which we call the cycle C^+.

▶ **Definition** (Cycle C^+). *C^+ is the positive cycle in G_M such that $C^+ \subseteq M\Delta M^*$.*

We claim that C^+ is well-defined and unique. Indeed, since $|R(M)| < |R(M^*)|$, the set of cycles $M\Delta M^*$ has to contain at least one positive cycle. Furthermore, there are at most $k = |R(M^*)|$ positive edges in $M\Delta M^*$ and by the definition of a critical tuple, every positive cycle contains at least $\frac{2}{3}k$ of them. Hence there is no second positive cycle in $M\Delta M^*$.

We list the following simple observations. The last two state that there only exist two different types of cycles in a critical tuple. These two types can be distinguished by examining the number of positive edges in a cycle C. We will repeatedly make use of this key observation.

▶ **Observation 10.** *Let (G, k, M^*, M) be a critical tuple. Then the following properties hold.*

1. *$|E^-(G_M)| < \frac{1}{3}k$.*
2. *$\frac{2}{3}k < |E^+(C^+)| \leq k$*
3. *If C is a directed cycle in G_M then $w_M(C) > 0$ if and only if $|E^+(C)| > \frac{2}{3}k$.*
4. *If C is a directed cycle in G_M then $w_M(C) \leq 0$ if and only if $|E^+(C)| < \frac{1}{3}k$.*

Proof. (G, k, M^*, M) is a critical tuple so in particular $|R(M)| < \frac{1}{3}k$, thus the graph G_M contains at most $\frac{1}{3}k$ negative edges. As a consequence, if C is a non-positive cycle then $|E^+(C)| \leq |E^-(C)|$ and thus $|E^+(C)| < \frac{1}{3}k$. Furthermore, in a critical tuple every positive cycle contains at least $\frac{2}{3}k$ positive edges, hence the third observation follows directly. In particular, since C^+ is a positive cycle, we have $\frac{2}{3}k < |E^+(C^+)|$. Finally the upper bound on $|E^+(C^+)|$ comes from the fact that positive edges in $M\Delta M^*$ are red edges in M^*, hence $M\Delta M^*$ contains at most $k = |R(M^*)|$ positive edges. In particualr $|E^+(C^+)| \leq k$. ◀

▶ **Definition** (Target set). *We call a set $\mathcal{C}$ of non-positive cycles in G_M a* target set *if:*

1. *$\forall e \in E^+(C^+)$ there exists a cycle in $\mathcal{C}$ containing e. (Condition 1)*
2. *$\forall C \in \mathcal{C}$, $C \cap C^+$ is a single path. (Condition 2)*
3. *$\forall e \in E^-(G_M)$ there are at most two cycles in $\mathcal{C}$ containing e. (Condition 3)*

The following Lemma 11 shows that in order to prove our main lemma, Lemma 9, it suffices to find a target set.

▶ **Lemma 11.** *Let (G, k, M^*, M) be a critical tuple and $\mathcal{C}$ be a target set in G_M. Then a contradiction arises.*

Proof. We can lower bound the sum of the weights of cycles in $\mathcal{C}$.

$$\begin{aligned}\sum_{C\in\mathcal{C}} w_M(C) &\geq 1\cdot|E^+(C^+)| - 2\cdot|E^-(G_M)| \\ &> \frac{2}{3}k - 2\cdot\frac{1}{3}k && \text{(by Observation 10)} \\ &= 0\end{aligned}$$

However $\sum_{C\in\mathcal{C}} w_M(C) \leq 0$ since all cycles $C \in \mathcal{C}$ have non-positive weight $w_M(C) \leq 0$, so this is a contradiction. ◀

Note that Condition 2 is not needed in the proof of Lemma 11. However Condition 2 will be useful to achieve Condition 3.

The rest of the paper is structured as follows: First, we construct a set $\mathcal{C}$ of non-positive cycles satisfying only Condition 1. This is explained in Section 3.2. After that, we explain in Section 3.3 how to modify the initial set to additionally satisfy Condition 2. In Section 3.4 we show how to modify this set again to also satisfy Condition 3. Finally, the proof of Lemma 9 is summarized in Section 3.5. To help visualize the reasonings on G_M we provide various illustrations. We focus only on the orientation of paths and cycles and not on actual vertices and edges of G_M so we omit them and draw paths and cycles using simple lines whose orientation is indicated by an arrow. Individual edges and vertices will be indicated in red.

3.2 Obtaining an initial set of cycles satisfying Condition 1

We explain how to obtain an initial set of non-positive cycles in G_M which satisfies only Condition 1. To this end, for a fixed critical tuple (G, k, M^*, M) consider the following definition:

▶ **Definition** (Cycle C_e)**.** *Let $e \in E^+(C^+)$. We define $C_e \subseteq M\Delta M^e$ as the unique directed cycle in $M\Delta M^e$ which contains the edge e.*

Observe that C_e is well-defined, since $e \in M\Delta M^e$. We claim that for all possible $e \in E^+(C^+)$ we have $w_M(C_e) \leq 0$. Indeed, positive edges in $M\Delta M^e$ are red edges in M^e. By the definition of a critical tuple and $E^+(C^+) \subseteq R(M^* \setminus M)$, we have $|R(M^e)| < \frac{1}{3}k$, thus $|E^+(C_e)| < \frac{1}{3}k$. Hence by Observation 10, C_e has non-positive weight in G_M. As a direct consequence, the set of cycles $\{C_e : e \in E^+(C^+)\}$ is a set of non-positive cycles satisfying Condition 1.

3.3 Modifying the set to satisfy Condition 2

We now show how to modify the initial set $\{C_e : e \in E^+(C^+)\}$ into a set of non-positive cycles such that both Conditions 1 and 2 hold. Consider a fixed critical tuple (G, k, M^*, M). If every cycle C_e intersects C^+ in a single path, then we would directly have Condition 2. However, in reality the interaction between C_e and C^+ can be a great amount more complicated. One example is depicted in Figure 1. In order to analyse the interaction between these two cycles, we define multiple notions around them. First, we decompose the cycle C_e into *jumps* and *interjumps* where jumps are the sub-paths of C_e outside of C^+ and interjumps are the sub-paths of C_e intersecting C^+ between jumps.

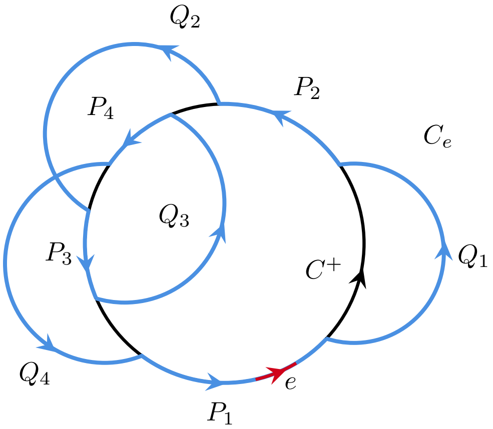


Figure 1 Decomposition of the cycle C_e (in blue) into 4 jumps Q_1, Q_2, Q_3, Q_4 and 4 interjumps P_1, P_2, P_3, P_4. The first interjump P_1 contains the edge e.

▶ **Definition** (Jumps and interjumps). *Let $e \in E^+(C^+)$. A jump of C_e is a sub-path $Q \subseteq C_e$ such that its endpoints are vertices of C^+ but none of its inner vertices are in C^+. Note that $Q \subseteq G_M \setminus C^+$. An interjump of C_e is a sub-path of $P \subseteq C_e$ contained in C^+ such that its endpoints are the endpoints of jumps of C_e. Note that $P \subseteq C^+$.*

▶ **Definition** (Decomposition of C_e into alternating jumps and interjumps). *Let $e \in E^+(C^+)$. Decompose C_e into jumps $Q_1, \ldots, Q_\ell$ and interjumps $P_1, \ldots P_\ell$ such that $e \in P_1$ and C_e is the concatenation of alternating jumps and interjumps:*

$$C_e = P_1 Q_1 \ldots P_j Q_j \ldots P_\ell Q_\ell$$

Let $\mathcal{J}_e = \{Q_1, Q_2, \ldots, Q_\ell\}$ be the set of jumps in C_e and $\mathcal{I}_e = \{P_1, P_2, \ldots, P_\ell\}$ be the set of interjumps in C_e.

Note that every jump in C_e must be followed by an interjump since all cycles considered are M-alternating cycles which cannot intersect in a single vertex (by Observation 4). Figure 1 shows a cycle C_e and its decomposition into jumps and interjumps.

To understand the interplay between C_e and C^+, we introduce the notion of forward and backward motion based on a visual intuition. Consider the directed cycle C^+ and take the convention of always drawing it with an anticlockwise orientation. We say that moving along C^+ in the direction of its directed edges equals to an *anticlockwise* or *forward* motion, while moving on the cycle C^+ against the direction of its directed edges equals to a *clockwise* or *backward* motion. We also want to interpret the direction a jump $Q \in \mathcal{J}_e$ is taking. However since Q is not a path in C^+ there is no obvious rule whether it should be seen as moving forward or backward. In fact, Figure 2 shows that the same jump can in principle be interpreted either as following a forward or a backward motion. We introduce the following rule which classifies all jumps as either a forward or a backward jump. We follow the convention to draw forward jumps outside of C^+, while we draw backward jumps inside of C^+. With this interpretation, in Figure 1 the jumps Q_1, Q_2 and Q_4 are forward and the jump Q_3 is backward.

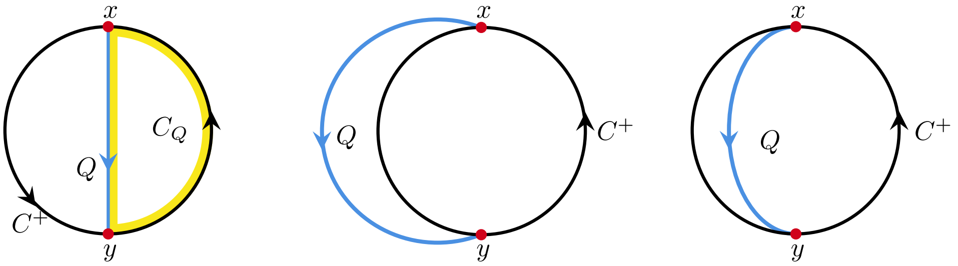


Figure 2 Consider a jump Q (blue) from the vertex x to the vertex y in C^+. Depending on the weight of C_Q (yellow highlight), the jump Q is either interpreted as following a forward (middle) or a backward (right) motion.

▶ **Definition** (Cycle C_Q and forward/backward jump). *Let $e \in E^+(C^+)$ and let Q be a jump of C_e. We define C_Q to be the unique directed cycle in the graph $C^+ \cup Q$ containing Q. Then:*

- *if $w_M(C_Q) > 0$, we call Q a* forward *jump;*
- *if $w_M(C_Q) \leq 0$, we call Q a* backward *jump.*

Let $\mathcal{J}_e^f$ be the set of forward jumps in C_e and $\mathcal{J}_e^b$ the set of backward jumps in C_e.

Observe that this is a valid definition, since the cycle C_Q is unique and independent of whether Q is a forward or a backward jump. We give a short informal explanation of why we use the weight of C_Q to distinguish between forward and backward jumps. A positive cycle must contain a large number of positive edges (at least $\frac{2}{3}k$ many, by Observation 10) and most of the positive edges of C_Q are contained in C^+ (since C_e has at most $\frac{1}{3}k$ positive edges). So for a forward jump Q_f, the cycle C_{Q_f} must have a large intersection with C^+ to be positive, i.e. Q_f only skips a small portion of the cycle. Similarly, for a backward jump Q_b, the cycle C_{Q_b} must be a non-positive cycle so it cannot contain too many positive edges and therefore cannot intersect a large part of C^+.

Finally, we introduce the notion of the *reach* of a jump Q, which is intuitively the sub-path of C^+ the jump Q is "jumping over". Figure 3 illustrates the reach of a forward jump and a backward jump. We say that the jump Q *covers* the edges in the reach of Q.

▶ **Definition** (Reach of a jump). *Let $e \in E^+(C^+)$ and let Q be a jump of C_e.*

- *If Q is a forward jump: the reach of Q is the sub-path of C^+ defined as $r(Q) := C^+ \setminus C_Q$.*
- *If Q is a backward jump: the reach of Q is the sub-path of C^+ defined as $r(Q) := C_Q \setminus Q$.*

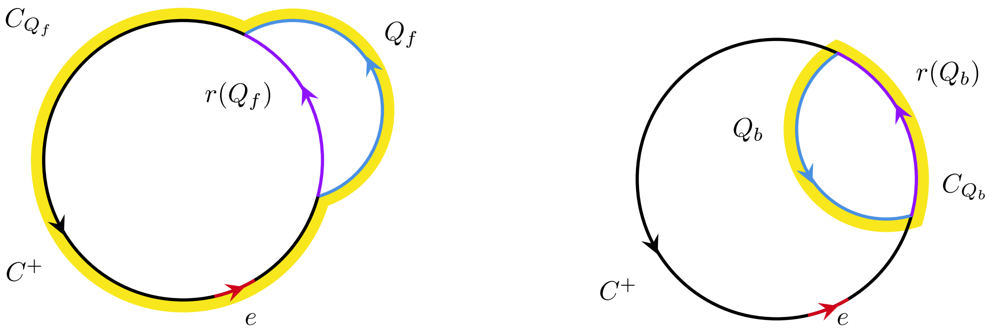


Figure 3 A forward jump Q_f and a backward jump Q_b (in blue). The respective reaches $r(Q_f)$ and $r(Q_b)$ of the jumps are colored in purple and the cycles C_{Q_f} and C_{Q_b} are highlighted in yellow.

We also extend the definition of the reach of a jump to the reach of C_e. Intuitively the reach of C_e is the union of sub-paths of C^+ the cycle C_e is jumping over. Here we differentiate between the sub-paths that are jumped over using *forward motions* (forward jumps or interjumps) and *backward motions* (backward jumps).

▶ **Definition** (Reach of C_e). *Let $e \in E^+(C^+)$. The forward reach $r_f(C_e)$ of C_e is defined as the union of every interjump of C_e and the reach of every forward jump of C_e.*

$$r_f(C_e) := \bigcup_{Q \in \mathcal{J}_e^f} r(Q) \cup \bigcup_{P \in \mathcal{I}_e} P$$

The backward reach $r_b(C_e)$ of C_e is defined as the union of the reaches of every backward jump of C_e.

$$r_b(C_e) := \bigcup_{Q \in \mathcal{J}_e^b} r(Q)$$

The reach $r(C_e)$ of C_e is defined as the union of the forward reach and the backward reach of C_e.

$$r(C_e) := r_f(C_e) \cup r_b(C_e)$$

All above definitions still hold when replacing C_e by a sub-path $\mathcal{P}$ of C_e. Since the reach of a truncated jump is undefined, we will only consider sub-paths which endpoints are vertices of C^+. If $\mathcal{P}$ contains a sub-path of an interjump then the forward reach of $\mathcal{P}$ contains this sub-path and not the entire interjump.

With those newly defined notions, we note that to achieve Condition 2, it suffices to prove the following Lemma 12. Due to space contraints, we leave the proof to Appendix A.

▶ **Lemma 12.** *Let (G, k, M^*, M) be a critical tuple. Every edge $e \in E^+(C^+)$ is contained in the backward reach of C_e, i.e. $\forall e \in E^+(C^+) : e \in r_b(C_e)$.*

Indeed, we can then construct a set of non-positive cycles satisfying Conditions 1 and 2 as explained in Lemma 13.

▶ **Lemma 13.** *Let (G, k, M^*, M) be a critical tuple. There exists a set $\mathcal{C}$ of non-positive cycles in G_M satisfying Conditions 1 and 2, i.e. such that the following holds:*

- *$\forall e \in E^+(C^+)$ there exists a cycle in $\mathcal{C}$ containing e.*
- *$\forall C \in \mathcal{C}$, $C \cap C^+$ is a single path.*

Proof. By Lemma 12, for each edge $e \in E^+(C^+)$ there exists a backward jump Q_e in C_e that covers e. Since it is a backward jump, the cycle C_{Q_e} formed by Q_e and its reach is a non-positive cycle. It also contains e and intersects C^+ in a single path $r(Q_e)$. Hence the set $\mathcal{C} := \{C_{Q_e} : e \in E^+(C^+), Q_e \in \mathcal{J}_e^b, e \in r(Q_e)\}$ satisfies Conditions 1 and 2. ◀

3.4 Modifying the set to satisfy Condition 3

In the previous section we obtained a set $\mathcal{C}$ of non-positive cycles satisfying Conditions 1 and 2. To additionally satisfy Condition 3, which is that for every negative edge $e \in E^-(G_M)$ there exist at most two cycles in $\mathcal{C}$ containing e, we proceed in two steps. First we focus in Lemma 15 on bounding the number of cycles containing a negative edge on the cycle C^+. Then we prove in Lemma 16 how to bound the number of cycles containing a negative edge outside the cycle C^+.

Observation 14 shows with a simple argument that if a set of paths in C^+ covers the whole cycle C^+, then it is enough to keep at most two paths covering the same edge in C^+. This is the key idea behind Lemma 15.

▶ **Observation 14.** *If $\mathcal{P}'$ is a set of paths in C^+, then there exists a subset $\mathcal{P} \subseteq \mathcal{P}'$ such that $\bigcup_{P' \in \mathcal{P}'} P' = \bigcup_{P \in \mathcal{P}} P$ and every $e \in C^+$ is contained in at most two paths of $\mathcal{P}$.*

Proof. Let $X = \bigcup_{P' \in \mathcal{P}'} P'$. Take $\mathcal{P}$ to be a minimal subset of $\mathcal{P}'$ such that $\bigcup_{P \in \mathcal{P}} P = X$. Suppose there exists an edge $e \in C^+$ such that $\exists P_1, P_2, P_3 \in \mathcal{P}$ with $e \in P_1 \cap P_2 \cap P_3$. W.l.o.g. suppose P_1 contains an edge that is furthest before e on C^+ among all edges of P_1, P_2 and P_3, and P_2 contains an edge furthest after e. Then $P_3 \subseteq P_1 \cup P_2$, which contradicts the minimality of $\mathcal{P}$. So every edge is contained in at most two paths from $\mathcal{P}$. ◀

▶ **Lemma 15.** *Let (G, k, M^*, M) be a critical tuple and $\mathcal{C}$ be a set of non-positive cycles in G_M satisfying Conditions 1 and 2, i.e. such that:*

- *$\forall e \in E^+(C^+)$ there exists a cycle in $\mathcal{C}$ containing e.*
- *$\forall C \in \mathcal{C}$, $C \cap C^+$ is a single path.*

Then there exists a set $\mathcal{C}' \subseteq \mathcal{C}$ of non-positive cycles satisfying the above Conditions 1 and 2 as well as the following property:

- *$\forall e \in E^-(C^+)$ there are at most two cycles in $\mathcal{C}$ containing e.*

Proof. By Condition 2, we can associate to each cycle C of $\mathcal{C}$ the sub-path $P_C := C \cap C^+$ intersecting with C^+. Let $\mathcal{P} := \{P_C : C \in \mathcal{C}\}$ be the set of such intersecting paths. By Condition 1 for every edge $e \in E^+(C^+)$ there exists a path $P \in \mathcal{P}$ containing e. We can thus apply Observation 14 to $\mathcal{P}$ and get a set $\mathcal{P}' \subseteq \mathcal{P}$ such that for every edge $e \in E^+(C^+)$ there exists a path $P \in \mathcal{P}'$ containing e and every $e \in C^+$ is contained in at most two paths from $\mathcal{P}'$. Let $\mathcal{C}' := \{C : P_C \in \mathcal{P}'\}$ be the set of cycles associated to each path in $\mathcal{P}'$. Then $\mathcal{C}' \subseteq \mathcal{C}$ still satisfies Conditions 1 and 2 and additionally for every edge $e \in C^+$ at most two cycles in $\mathcal{C}'$ contain e. ◀

▶ **Lemma 16.** *Let (G, k, M^*, M) be a critical tuple and $\mathcal{C}$ be a set of non-positive cycles in G_M satisfying Conditions 1 and 2, i.e. such that:*

- *$\forall e \in E^+(C^+)$ there exists a cycle in $\mathcal{C}$ containing e.*
- *$\forall C \in \mathcal{C}$, $C \cap C^+$ is a single path.*

Then there exists a set $\mathcal{C}' \subseteq \mathcal{C}$ of non-positive cycles satisfying the above Conditions 1 and 2 as well as the following property:

- *$\forall e \in E^-(G_M \setminus C^+)$ there are at most two cycles in $\mathcal{C}$ containing e.*

Proof. We show how to reduce three non-positive cycles of $\mathcal{C}$ containing an edge $e \in E^-(G_M \setminus C^+)$ into two non-positive cycles such that, when replacing the three cycles by the two new cycles, Conditions 1 and 2 still hold. By repeating this transformation, the number of cycles containing a negative edge $e \in E^-(G_M \setminus C^+)$ decreases until eventually at most two cycles in $\mathcal{C}$ contain e.

Fix an edge $e \in E^-(G_M \setminus C^+)$ and three non-positive cycles C_{Q_1}, C_{Q_2} and C_{Q_3} in $\mathcal{C}$ all containing e. For $i \in \{1, 2, 3\}$, by Condition 2, C_{Q_i} intersects C^+ in a single path. Hence let $r(Q_i) = C_{Q_i} \cap C^+$ be this path and let $Q_i := C_{Q_i} \setminus C^+$ be the remaining path of C_{Q_i} not intersecting C^+. Let also s_i and f_i be the first and last edge of Q_i. We can assume w.l.o.g. that the first edge of $r(Q_i)$ appears anti-clockwise on C^+ in the order $1, 2, 3$.

For $(i, j) \in \{(1, 2), (2, 3), (3, 1)\}$ consider the walk $W_{ij} = Q_j[s_j, e] \cup Q_i(e, f_i]$ that goes from s_j to f_i. This is a valid walk since $e \in Q_1 \cap Q_2 \cap Q_3$. W_{ij} can use edges multiple times so we identify W_{ij} to its multi-edge set. As illustrated in Figure 4, we can extract a simple

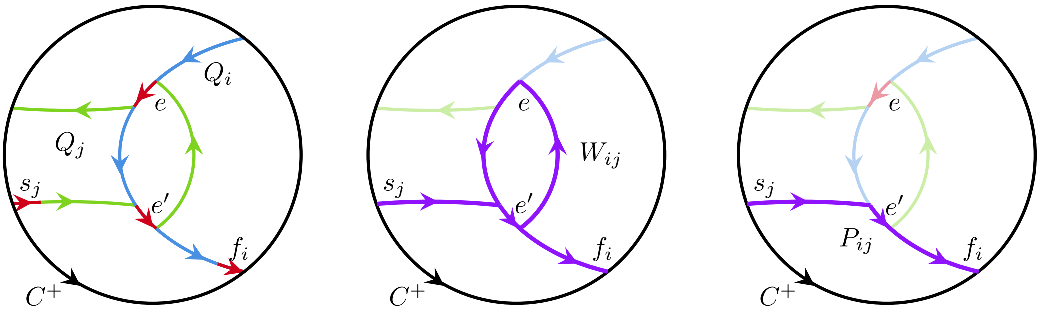


Figure 4 Possible configuration appearing in the proof of Lemma 16. Two jumps Q_i (blue) and Q_j (green) intersect in two different edges $e \in Q_1 \cap Q_2 \cap Q_3$ and $e' \in Q_i \cap Q_j$. The first edge of Q_j is denoted s_j and the last edge of Q_i is denoted f_i (red). The walk W_{ij} (purple, middle) from s_j to f_i is defined as $W_{ij} = Q_j[s_j, e] \cup Q_i(e, f_i]$ and uses the edge e' twice. The simple path P_{ij} (purple, right) is the simple sub-path from W_{ij} starting at s_j and ending at f_i.

path $P_{ij} \subseteq W_{ij}$ that goes from s_j to f_i. Let $C_{ij} = P_{ij} \cup C^+[r(Q_i), r(Q_j)]$. C_{ij} is a simple directed cycle since P_{ij} is a simple path from s_j to f_i outside of C^+ and $C^+[r(Q_i), r(Q_j)]$ is a simple path on C^+ from the endpoint of f_i on C^+ to the endpoint of s_j on C^+. An example of this construction is shown in Figure 5. If C_{ij} is a non-positive cycle, then we can replace C_{Q_i} and C_{Q_j} by C_{ij} in $\mathcal{C}$ and Conditions 1 and 2 would still hold. Indeed, by construction C_{ij} intersects C^+ on the single path $C^+[r(Q_i), r(Q_j)]$ so Condition 2 holds. Condition 1 holds because every edge $e \in E^+(C^+)$ that is contained in C_{Q_i} or C_{Q_j} is in fact in $r(Q_i)$ or $r(Q_j)$, which are both contained in C_{ij}. Hence it remains to show that there exists a pair $(i, j) \in \{(1, 2), (2, 3), (3, 1)\}$ such that C_{ij} is a non-positive cycle.

Suppose there exists a pair $(i, j) \in \{(1, 2), (2, 3), (3, 1)\}$ such that $r(Q_i)$ and $r(Q_j)$ are intersecting. Then we have:

$$\begin{aligned}
|E^+(C_{ij})| &= |E^+(P_{ij})| + |E^+(C^+[r(Q_i), r(Q_j)])| \\
&= |E^+(P_{ij})| + |E^+(r(Q_i) \cup r(Q_j))| \\
&\leq |E^+(W_{ij})| + |E^+(r(Q_i))| + |E^+(r(Q_j))| && \text{(because } P_{ij} \subseteq W_{ij}\text{)} \\
&= |E^+(Q_j[s_j, e])| + |E^+(Q_i(e, f_i])| + |E^+(r(Q_i))| + |E^+(r(Q_j))| \\
&\leq |E^+(Q_j)| + |E^+(Q_i)| + |E^+(r(Q_i))| + |E^+(r(Q_j))| \\
&= |E^+(C_{Q_i})| + |E^+(C_{Q_j})| \\
&< 2 \cdot \frac{1}{3}k && \text{(by Observation 10)}
\end{aligned}$$

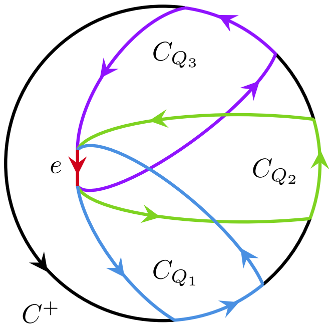
where the second equality follows from $C^+[r(Q_i), r(Q_j)] \subseteq r(Q_i) \cup r(Q_j)$. By Observation 10, C_{ij} cannot be positive and hence is a non-positive cycle, so we get the desired result.

If none of the paths $r(Q_1), r(Q_2), r(Q_3)$ are intersecting then assume for the sake of contradiction that C_{12}, C_{23} and C_{31} are all positive. By Observation 10 this means that

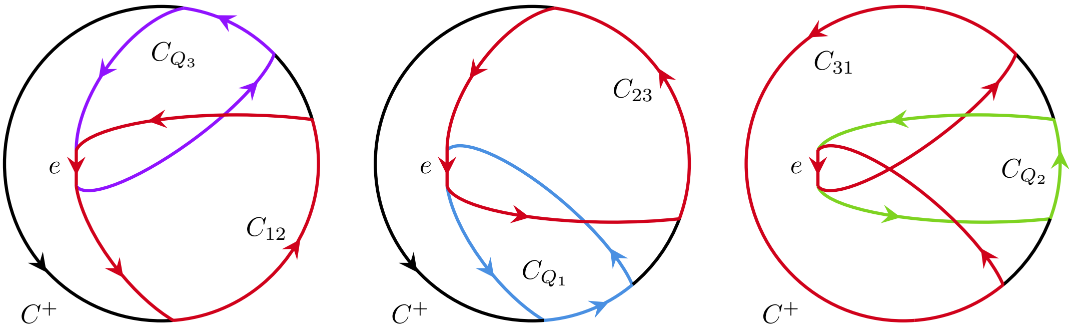
$$|E^+(C_{12})| + |E^+(C_{23})| + |E^+(C_{31})| > 3 \cdot \frac{2}{3}k = 2k.$$

However we also have for any pair $(i, j) \in \{(1, 2), (2, 3), (3, 1)\}$ that

$$\begin{aligned}
|E^+(C_{ij})| &= |E^+(P_{ij})| + |E^+(C^+[r(Q_i), r(Q_j)])| \\
&\leq |E^+(W_{ij})| + |E^+(C^+[r(Q_i), r(Q_j)])| && \text{(because } P_{ij} \subseteq W_{ij}\text{)} \\
&= |E^+(Q_j[s_j, e])| + |E^+(Q_i(e, f_i])| + |E^+(C^+[r(Q_i), r(Q_j)])|
\end{aligned}$$



(a) Three non-positive cycles C_{Q_1} (blue) C_{Q_2} (green) and C_{Q_3} (purple) intersecting in a single negative edge $e \notin C^+$.



(b) The newly constructed cycles C_{12}, C_{23} and C_{31} (in red from left to right). To construct C_{12} we combine the two cycles C_{Q_1} and C_{Q_2}.

Figure 5 Construction of three cycles C_{12}, C_{23} and C_{31} as done in the proof of Lemma 16. Note that this is a simplified configuration where the pairwise intersection of two non-positive cycles corresponds to the intersection of the three non-positive cycles C_{Q_1}, C_{Q_2} and C_{Q_3}.

Hence the total number of positive edges in the three cycles is:

$$\begin{aligned}
&|E^+(C_{12})| + |E^+(C_{23})| + |E^+(C_{31})| \\
&\quad \le |E^+(Q_2[s_2, e])| + |E^+(Q_1(e, f_1])| + |E^+(C^+[r(Q_1), r(Q_2)])| \\
&\qquad + |E^+(Q_3[s_3, e])| + |E^+(Q_2(e, f_2])| + |E^+(C^+[r(Q_2), r(Q_3)])| \\
&\qquad + |E^+(Q_1[s_1, e])| + |E^+(Q_3(e, f_3])| + |E^+(C^+[r(Q_3), r(Q_1)])| \\
&\quad = |E^+(Q_1)| + |E^+(Q_2)| + |E^+(Q_3)| \\
&\qquad + |E^+(C^+)| + |E^+(r(Q_1))| + |E^+(r(Q_2))| + |E^+(r(Q_3))| \\
&\quad = |E^+(C^+)| + |E^+(C_{Q_1})| + |E^+(C_{Q_2})| + |E^+(C_{Q_3})| \\
&\quad < k + 3 \cdot \frac{1}{3}k = 2k \qquad\qquad \text{(by Observation 10)}
\end{aligned}$$

where we can apply Observation 10 because C_{Q_i} is non-positive for any $i \in \{1, 2, 3\}$. Hence there has to exists a pair $(i, j) \in \{(1, 2), (2, 3), (3, 1)\}$ such that C_{ij} is non-positive. ◀

3.5 Constructing a target set

We can now prove Lemma 9.

Proof of Lemma 9. For the sake of contradiction let (G, k, M^*, M) be a critical tuple. Apply Lemmas 13, 16 and 15 in that order consecutively and get a target set $\mathcal{C}$. By Lemma 11, this leads to a contradiction. Thus (G, k, M^*, M) cannot be a critical tuple. ◀

As a consequence, Lemmas 8 and 9 together show that Algorithm 1 is a polynomial-time algorithm that always outputs a perfect matching M containing between $\frac{1}{3}k$ and k red edges. This completes the proof of Theorem 1.

4 Conclusion

In this paper we formally define EM-OPT, an optimization variant of the EXACT MATCHING problem and show a deterministic polynomial-time approximation algorithm for EM-OPT which achieves an approximation ratio of 3 on bipartite graphs. Although the algorithm is fairly simple to present, the proof of its correctness is more complex. In the second part of the algorithm we iterate over all edges e and compute a perfect matching of minimum number of red edges containing e. Most of our work is put into proving that there exists such a matching that approximates the optimal solution of EM-OPT by a factor 3. As our calculations are tight for an approximation ratio of 3, a natural continuation of our work would be to improve this ratio, e.g. to 2. We speculate that the algorithm could be improved by iterating over larger subsets of edges (of constant size) instead of one edge at a time and that such an algorithm could give a better approximation and maybe even a PTAS. The analysis of such an algorithm, however, remains quite challenging and new techniques and insights might be needed. Another open problem is to find an approximation algorithm for general graphs.

References

1 Ilan Doron Arad, Ariel Kulik, and Hadas Shachnai. An EPTAS for budgeted matching and budgeted matroid intersection. *CoRR*, abs/2302.05681, 2023. doi:10.48550/arXiv.2302.05681.

2 Vikraman Arvind, Johannes Köbler, Sebastian Kuhnert, and Jacobo Torán. Solving linear equations parameterized by hamming weight. *Algorithmica*, 75(2):322–338, 2016. doi:10.1007/s00453-015-0098-3.

3 André Berger, Vincenzo Bonifaci, Fabrizio Grandoni, and Guido Schäfer. Budgeted matching and budgeted matroid intersection via the gasoline puzzle. *Math. Program.*, 128(1-2):355–372, 2011. doi:10.1007/s10107-009-0307-4.

4 Jacek Blazewicz, Piotr Formanowicz, Marta Kasprzak, Petra Schuurman, and Gerhard J. Woeginger. A polynomial time equivalence between DNA sequencing and the exact perfect matching problem. *Discret. Optim.*, 4(2):154–162, 2007. doi:10.1016/j.disopt.2006.07.004.

5 Paolo M. Camerini, Giulia Galbiati, and Francesco Maffioli. Random pseudo-polynomial algorithms for exact matroid problems. *J. Algorithms*, 13(2):258–273, 1992. doi:10.1016/0196-6774(92)90018-8.

6 Jack Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of research of the National Bureau of Standards B*, 69(125-130):55–56, 1965.

7 Nicolas El Maalouly. Exact matching: Algorithms and related problems. In Petra Berenbrink, Patricia Bouyer, Anuj Dawar, and Mamadou Moustapha Kanté, editors, *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023, March 7-9, 2023, Hamburg, Germany*, volume 254 of *LIPIcs*, pages 29:1–29:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.STACS.2023.29.

8 Nicolas El Maalouly and Raphael Steiner. Exact matching in graphs of bounded independence number. In Stefan Szeider, Robert Ganian, and Alexandra Silva, editors, *47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022, August 22-26, 2022, Vienna, Austria*, volume 241 of *LIPIcs*, pages 46:1–46:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.MFCS.2022.46.

9 Nicolas El Maalouly, Raphael Steiner, and Lasse Wulf. Exact matching: Correct parity and FPT parameterized by independence number. *CoRR*, abs/2207.09797, 2022. doi:10.48550/arXiv.2207.09797.

10 Dennis Fischer, Tim A. Hartmann, Stefan Lendl, and Gerhard J. Woeginger. An investigation of the recoverable robust assignment problem. In Petr A. Golovach and Meirav Zehavi, editors, *16th International Symposium on Parameterized and Exact Computation, IPEC 2021, September 8-10, 2021, Lisbon, Portugal*, volume 214 of *LIPIcs*, pages 19:1–19:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.IPEC.2021.19.

11 Anna Galluccio and Martin Loebl. On the theory of pfaffian orientations. I. perfect matchings and permanents. *Electron. J. Comb.*, 6, 1999. doi:10.37236/1438.

12 Hans-Florian Geerdes and Jácint Szabó. A unified proof for Karzanov's exact matching theorem. *quick proof QP-2011-02, Egerváry Research Group, Budapest*, 2011.

13 Fabrizio Grandoni and Rico Zenklusen. Optimization with more than one budget. *CoRR*, abs/1002.2147, 2010. arXiv:1002.2147.

14 Rohit Gurjar, Arpita Korwar, Jochen Messner, Simon Straub, and Thomas Thierauf. Planarizing gadgets for perfect matching do not exist. *ACM Trans. Comput. Theory*, 8(4):14:1–14:15, 2016. doi:10.1145/2934310.

15 Rohit Gurjar, Arpita Korwar, Jochen Messner, and Thomas Thierauf. Exact perfect matching in complete graphs. *ACM Trans. Comput. Theory*, 9(2):8:1–8:20, 2017. doi:10.1145/3041402.

16 AV Karzanov. Maximum matching of given weight in complete and complete bipartite graphs. *Cybernetics*, 23(1):8–13, 1987.

17 Monaldo Mastrolilli and Georgios Stamoulis. Constrained matching problems in bipartite graphs. In Ali Ridha Mahjoub, Vangelis Markakis, Ioannis Milis, and Vangelis Th. Paschos, editors, *Combinatorial Optimization – Second International Symposium, ISCO 2012, Athens, Greece, April 19-21, 2012, Revised Selected Papers*, volume 7422 of *Lecture Notes in Computer Science*, pages 344–355. Springer, 2012. doi:10.1007/978-3-642-32147-4_31.

18 Monaldo Mastrolilli and Georgios Stamoulis. Bi-criteria and approximation algorithms for restricted matchings. *Theor. Comput. Sci.*, 540:115–132, 2014. doi:10.1016/j.tcs.2013.11.027.

19 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Comb.*, 7(1):105–113, 1987. doi:10.1007/BF02579206.

20 Christos H. Papadimitriou and Mihalis Yannakakis. The complexity of restricted spanning tree problems. *J. ACM*, 29(2):285–309, April 1982. doi:10.1145/322307.322309.

21 Georgios Stamoulis. Approximation algorithms for bounded color matchings via convex decompositions. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014 – 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, volume 8635 of *Lecture Notes in Computer Science*, pages 625–636. Springer, 2014. doi:10.1007/978-3-662-44465-8_53.

22 Ola Svensson and Jakub Tarnawski. The matching problem in general graphs is in quasi-NC. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 696–707. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.70.

23 Moshe Y Vardi and Zhiwei Zhang. Quantum-inspired perfect matching under vertex-color constraints. *arXiv preprint arXiv:2209.13063*, 2022.

24 Tongnyoul Yi, Katta G. Murty, and Cosimo Spera. Matchings in colored bipartite networks. *Discret. Appl. Math.*, 121(1-3):261–277, 2002. doi:10.1016/S0166-218X(01)00300-6.

25 Raphael Yuster. Almost exact matchings. *Algorithmica*, 63(1-2):39–50, 2012. doi:10.1007/s00453-011-9519-0.

A Proof of Lemma 12

This section is devoted to the proof of Lemma 12. The proof technique is a combination of reasonings about forward and backward motion, shown in Lemmas 17 and 18, and an interpolation argument.

Let's start with a basic fact about sub-paths of C_e. Let e_s, e_f be two distinct edges on C^+. The following lemma captures the intuitive fact that any path from e_s to e_f must go either in a clockwise motion, or in an anticlockwise motion. While doing so, it must traverse all the edges between e_s and e_f in that direction.

▶ **Lemma 17.** *Let (G, k, M^*, M) be a critical tuple and fix an edge $e \in E^+(C^+)$. If e_s, e_f are distinct edges in C^+ and $\mathcal{P} = C_e[e_s, e_f]$ is a sub-path of C_e from e_s to e_f, then either $C^+[e_s, e_f] \subseteq r_f(\mathcal{P})$ or $C^+[e_f, e_s] \subseteq r_b(\mathcal{P})$.*

Proof. As a helpful tool, we consider the following definition of *shadow*. Let Q be a jump of C_e from x to y, where x and y are vertices in C^+. If Q is a forward jump, then the shadow $s(Q)$ of Q is the walk in C^+, which starts at x and goes anticlockwise, until it encounters y. If Q is a backward jump, then $s(Q)$ is the walk in C^+, which starts at x and goes clockwise, until it encounters y. We remark that formally, since $s(Q)$ can travel edges in reverse direction, $s(Q)$ is a walk in the undirected analogue of C^+ in G. For an interjump P from vertex x to y, the shadow of P is defined to be the path P itself. For a sub-path $\mathcal{P} \subseteq C_e$ consisting of the concatenation of alternating jumps and interjumps the shadow $s(\mathcal{P})$ of $\mathcal{P}$ is the concatenation of the corresponding shadows of jumps and interjumps in the same order.

By definition, the following basic observations hold. For any sub-path $\mathcal{P} \subseteq C_e$, its shadow $s(\mathcal{P})$ is a walk contained in C^+ such that it has the same start and end vertex as $\mathcal{P}$. Furthermore, the walk $s(\mathcal{P})$ can be thought of as a sequence of steps, where every step traverses one single edge of C^+ either clockwise or anticlockwise. The set $r_f(\mathcal{P})$ is exactly the set of edges which are traveled by the walk $s(\mathcal{P})$ with an anticlockwise step. The set $r_b(\mathcal{P})$ is exactly the set of edges which are traveled by the walk $s(\mathcal{P})$ with a clockwise step. The set $r(\mathcal{P})$ is exactly the set of edges which are traveled by the walk $s(\mathcal{P})$.

Now return to the statement of the lemma. The walk $s(\mathcal{P})$ is a walk contained in C^+, starting with the edge e_s and ending with the edge e_f. It is obvious that $s(\mathcal{P})$ must either traverse all of $C^+[e_s, e_f]$ with anticlockwise steps, or all of $C^+[e_f, e_s]$ with clockwise steps. This implies that either $C^+[e_s, e_f] \subseteq r_f(\mathcal{P})$ or $C^+[e_f, e_s] \subseteq r_b(\mathcal{P})$. ◀

In the proof of Lemma 12, we will consider a set of forward jumps and interjumps in C_e that cover all the cycle C^+. However we want to avoid forward jumps covering the edge e. The following Lemma 18 shows that this is possible under certain assumptions. Intuitively, since e is already contained in an interjump of C_e, there is no need to include a forward jump covering e in the set of forward jumps and interjumps covering C^+. However one needs to carefully select the forward jumps of the set, which explains the technicallity of the proof below.

▶ **Lemma 18.** *Let (G, k, M^*, M) be a critical tuple and fix an edge $e \in E^+(C^+)$. If $e \notin r_b(C_e)$, then there exists a set of forward jumps $\mathcal{Q} \subseteq \mathcal{J}_e^f$ such that*

$$C^+ = \bigcup_{Q \in \mathcal{Q}} r(Q) \cup \bigcup_{P \in \mathcal{I}_e} P$$

and $\nexists Q_f \in \mathcal{Q}$ with $e \in r(Q_f)$.

Proof. We prove the statement by constructing a tuple $(\mathcal{P}, \mathcal{R}, e')$, where $e' \in C^+$ is an edge of C^+, $\mathcal{P} = C_e[e, e']$ is a sub-path of C_e and $\mathcal{R}$ is a set of jumps and interjumps of C_e. We want $(\mathcal{P}, \mathcal{R}, e')$ to satisfy the following invariant.

▶ **Invariant.** *A tuple $(\mathcal{P}, \mathcal{R}, e')$ satisfies the invariant if the following properties hold:*

1. *There is no forward jump Q in $\mathcal{R}$ such that $e \in r(Q)$.*
2. *$C^+[e, e'] \subseteq r_f(\mathcal{P})$.*
3. *$C^+(e', e) \subseteq r_f(\mathcal{R})$*

We also want to ensure that there is no forward jump Q in $\mathcal{P}$ with $e \in r(Q)$. Indeed, in that case, define $\mathcal{Q}$ to be the set of forward jumps in $\mathcal{P} \cup \mathcal{R}$. Then by Invariant 1, $\mathcal{Q}$ doesn't contain any forward jump Q with $e \in r(Q)$. Also, by Invariants 2 and 3 of $(\mathcal{P}, \mathcal{R}, e')$:

$$\begin{aligned} C^+ &= C^+[e, e'] \cup C^+(e', e) \\ &\subseteq r_f(\mathcal{P}) \cup r_f(\mathcal{R}) \\ &\subseteq \bigcup_{Q \in \mathcal{Q}} r(Q) \cup \bigcup_{P \in \mathcal{I}_e} P \end{aligned}$$

which is the conclusion of Lemma 18.

We start with a tuple $(\mathcal{P}, \mathcal{R}, e')$ satisfying the invariant properties and then iteratively modify the tuple in order to achieve the additional property that no forward jump in $\mathcal{P}$ covers e. At first, e' is the edge before e in C_e. Note that $e' \in C^+$ because $e \notin M$ and C_e is M-alternating so $e' \in M$, and since C^+ is also M-alternating, edges of M adjacent to e must be in C^+. Let $\mathcal{P} = C_e[e, e']$ be a path starting in e and covering all edges of C_e and let $\mathcal{R} = \emptyset$ be an empty set. Clearly Invariants 1 and 3 holds for $(\mathcal{P}, \mathcal{R}, e')$. By Lemma 17 applied on $\mathcal{P}$, either $C^+[e, e'] \subseteq r_f(\mathcal{P})$ or $C^+[e', e] \subseteq r_b(\mathcal{P})$. However $e \notin r_b(C_e)$ so the latter is not possible. Hence $(\mathcal{P}, \mathcal{R}, e')$ satisfies all invariant properties. Let t be the number of forward jumps Q in $\mathcal{P}$ with $e \in r(Q)$. If $t = 0$, we are done.

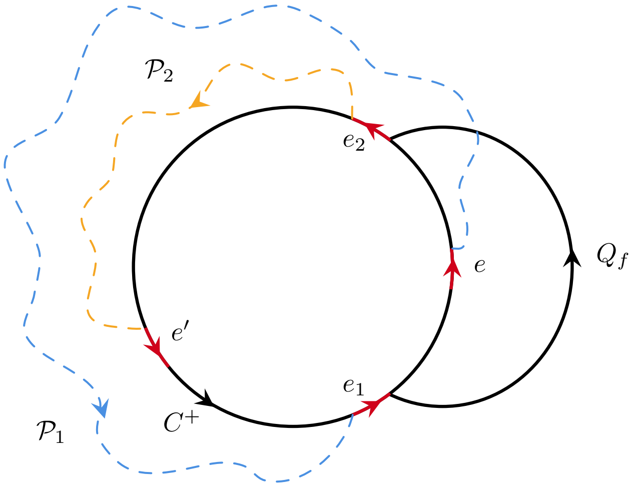


Figure 6 To prove Lemma 18 we consider a path $\mathcal{P} = C_e[e, e']$ that contains a jump Q_f with $e \in r(Q_f)$. e_1 and e_2 are the edges before and after the jump Q_f. We split $\mathcal{P}$ into two paths $\mathcal{P}_1$ (blue) and $\mathcal{P}_2$ (orange) that are respectively before and after the jump Q_f.

If $t > 0$, we replace $(\mathcal{P}, \mathcal{R}, e')$ by another tuple $(\mathcal{P}', \mathcal{R}', e_1)$ where $\mathcal{P}'$ contains $t - 1$ forward jumps covering e. We ensure that $(\mathcal{P}', \mathcal{R}', e_1)$ also satisfies the invariant properties, so that we can repeat the process until $t = 0$. Let Q_f be the last forward jump in $\mathcal{P}$ covering e

and let e_1 and e_2 be the first edges in C^+ before and after Q_f (see Figure 6). Consider the sub-paths $\mathcal{P}_1 = C_e[e, e_1]$ and $\mathcal{P}_2 = C_e[e_2, e']$ and define $\mathcal{P}' := \mathcal{P}_1$ and $\mathcal{R}' := \mathcal{R} \cup \mathcal{P}_2$. By definition, $\mathcal{P}'$ is the sub-path of $\mathcal{P}$ before Q_f (excluded), hence it contains $t-1$ forward jumps covering e. It remains to show that $(\mathcal{P}', \mathcal{R}', e_1)$ satisfies the invariant. By definition of Q_f, there is no forward jump in $\mathcal{P}_2$ covering e. Thus, since $(\mathcal{P}, \mathcal{R}, e')$ satisfies Invariant 1, $(\mathcal{P}', \mathcal{R}', e_1)$ also satisfies Invariant 1. We can apply Lemma 17 on $\mathcal{P}_1$ and get that either $C^+[e, e_1] \subseteq r_f(\mathcal{P}_1)$ or $C^+[e_1, e] \subseteq r_b(\mathcal{P}_1)$. However $e \notin r_b(C_e)$ so the latter case is not possible, hence $(\mathcal{P}', \mathcal{R}', e_1)$ satisfies Invariant 2. Finally, to prove that $C^+(e_1, e) \subseteq r_f(\mathcal{R}')$, consider two cases. If $C^+(e_1, e) \subseteq C^+(e', e)$ (as in Figure 6) then $C^+(e_1, e) \subseteq r_f(\mathcal{R}) \subseteq r_f(\mathcal{R}')$ by Invariant 3 of $(\mathcal{P}, \mathcal{R}, e')$. Otherwise if $C^+(e', e) \subseteq C^+(e_1, e)$, then the edges e_2, e_1, e', e appear in that order on C^+. We can thus write $C^+(e_1, e) = C^+(e_1, e'] \cup C^+(e', e)$. By Invariant 3 of $(\mathcal{P}, \mathcal{R}, e')$, we know that $C^+(e', e) \subseteq r_f(\mathcal{R})$. Note that $C^+(e_1, e'] \subseteq C^+[e_2, e']$. Applying Lemma 17 on $\mathcal{P}_2$ we get that either $C^+[e_2, e'] \subseteq r_f(\mathcal{P}_2)$ or $C^+[e', e_2] \subseteq r_b(\mathcal{P}_2)$. But $e \in C^+[e', e_2]$ and $e \notin r_b(C_e)$ so the latter case is not possible. Hence $C^+(e_1, e'] \subseteq \mathcal{P}_2$ and we thus have $C^+(e_1, e) \subseteq r_f(\mathcal{P}_2) \cup r_f(\mathcal{R}) = r_f(\mathcal{R}')$. So in both cases Invariant 3 holds for $(\mathcal{P}', \mathcal{R}', e_1)$. Therefore, we can safely replace $(\mathcal{P}, \mathcal{R}, e')$ by $(\mathcal{P}', \mathcal{R}', e_1)$ and maintain the invariant. ◀

We now have collected all the ingredients to prove the main result of this subsection. While the previous lemmas were proven using an intuition on forward and backward motion, Lemma 12 is proven using an interpolation argument.

Proof of Lemma 12. For the sake of contradiction, assume that there exists an edge $e \in E^+(C^+)$ such that $e \notin r_b(C_e)$. Apply Lemma 18 and let $\mathcal{Q}'$ be the resulting set of forward jumps. By identifying every jump in $\mathcal{Q}'$ with its reach, we can apply Observation 14 and get the resulting set $\mathcal{Q} \subseteq \mathcal{Q}'$. Consequently, $C^+ = \bigcup_{Q \in \mathcal{Q}} r(Q) \cup \bigcup_{P \in \mathcal{I}_e} P$, there is no jump $Q \in \mathcal{Q}$ covering e and every edge $e' \in C^+$ is contained in the reach of at most two jumps of $\mathcal{Q}$. Order the jumps in $\mathcal{Q}$ by the distance between e and their reach and write $\mathcal{Q} = \{Q_1, Q_2, \dots, Q_t\}$. Define $\mathcal{Q}_{odd} = \{Q_1, Q_3, \dots\}$ and $\mathcal{Q}_{even} = \{Q_2, Q_4, \dots\}$ to be the set of jumps of respectively odd and even index. Since there is no forward jump in $\mathcal{Q}$ covering e, the first and the last jump in $\mathcal{Q}$ are non-intersecting (otherwise the reach of one of them would contain e). Additionally, the reach of a jump Q_i can only intersect the reach of Q_{i-1} and Q_{i+1} (by the property obtained by Observation 14). Hence the reach of any two jumps in $\mathcal{Q}_{odd}$ as well as the reach of any two jumps in $\mathcal{Q}_{even}$ are not intersecting.

We can thus define two cycles $C^+_{odd} = C^+ \cup \mathcal{Q}_{odd} \setminus \{r(Q_1), r(Q_3), \dots\}$ and $C^+_{even} = C^+ \cup \mathcal{Q}_{even} \setminus \{r(Q_2), r(Q_4), \dots\}$. We show that at least one of them is non-positive using Observation 10. Indeed, the number of positive edges in C^+_{odd} is

$$\begin{aligned}|E^+(C^+_{odd})| &= |E^+(C^+)| + \sum_{Q \in \mathcal{Q}_{odd}} |E^+(Q)| - \sum_{Q \in \mathcal{Q}_{odd}} |E^+(r(Q))| \\ &= |E^+(C^+)| - \sum_{Q \in \mathcal{Q}_{odd}} m_Q\end{aligned}$$

where we define $m_Q := |E^+(r(Q))| - |E^+(Q)|$. Assume for now that $\sum\limits_{Q \in \mathcal{Q}_{odd}} m_Q \geq \sum\limits_{Q \in \mathcal{Q}_{even}} m_Q$ so that $\sum\limits_{Q \in \mathcal{Q}} m_Q = \sum\limits_{Q \in \mathcal{Q}_{odd}} m_Q + \sum\limits_{Q \in \mathcal{Q}_{even}} m_Q \leq 2 \sum\limits_{Q \in \mathcal{Q}_{odd}} m_Q$. So we get

$$|E^+(C^+_{odd})| \leq |E^+(C^+)| - \frac{1}{2} \sum_{Q \in \mathcal{Q}} m_Q$$

and we also have

$$\begin{aligned}\sum_{Q\in\mathcal{Q}} m_Q &= \sum_{Q\in\mathcal{Q}} |E^+(r(Q))| - \sum_{Q\in\mathcal{Q}} |E^+(Q)| \\ &= \left(\sum_{Q\in\mathcal{Q}} |E^+(r(Q))| + \sum_{P\in\mathcal{I}_e} |E^+(P)|\right) - \left(\sum_{P\in\mathcal{I}_e} |E^+(P)| + \sum_{Q\in\mathcal{Q}} |E^+(Q)|\right) \\ &\geq |E^+(C^+)| - \left(\sum_{P\in\mathcal{I}_e} |E^+(P)| + \sum_{Q\in\mathcal{Q}} |E^+(Q)|\right) \qquad \text{(by the construction of } \mathcal{Q}) \\ &\geq |E^+(C^+)| - |E^+(C_e)|\end{aligned}$$

where the last inequality comes from the fact that C_e can contain jumps outside of $\mathcal{Q}$. We finally get the following bound on the number of positive edges in C^+_{odd}.

$$\begin{aligned}|E^+(C^+_{odd})| &\leq |E^+(C^+)| - \frac{1}{2}(|E^+(C^+)| - |E^+(C_e)|) \\ &= \frac{1}{2}|E^+(C^+)| + \frac{1}{2}|E^+(C_e)| \\ &< \frac{1}{2}k + \frac{1}{2}\cdot\frac{1}{3}k \qquad \text{(by Observation 10)} \\ &= \frac{2}{3}k\end{aligned}$$

By Observation 10, we can conclude that C^+_{odd} is a non-positive cycle. If $\sum_{Q\in\mathcal{Q}_{odd}} m_Q \leq \sum_{Q\in\mathcal{Q}_{even}} m_Q$ then the same argument on C^+_{even} shows that C^+_{even} is a non-positive cycle. In the following we assume that C^+_{odd} is a non-positive cycle, but the reasoning can be adapted for the case that C^+_{even} is non-positive by interchanging *odd* by *even*.

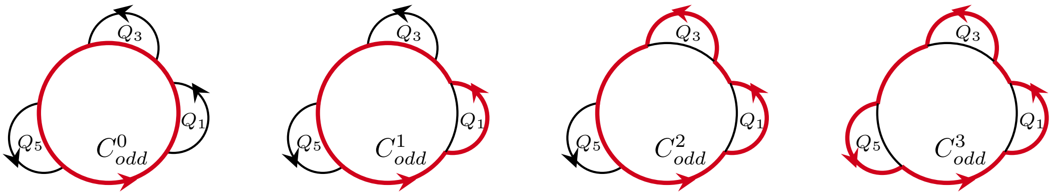


Figure 7 A sequence of cycles $C^0_{odd}, C^1_{odd}, C^2_{odd}, C^3_{odd}$ (in red from left to right) defined by sequentially adding the jumps in $\mathcal{Q}_{odd} = \{Q_1, Q_3, Q_5\}$ to the cycle C^+ and removing the corresponding reach, as done in the proof of Lemma 12. Note that $C^0_{odd} = C^+$.

Define a sequence of cycles $C^{i+1}_{odd} := C^i_{odd} \cup Q_{2i+1} \setminus r(Q_{2i+1})$ for $i \in |\mathcal{Q}_{odd}|$ starting at $C^0_{odd} := C^+$ and ending at C^+_{odd}. See Figure 7 for an example. The constructed C^i_{odd}'s are simple cycles since the reaches in $\mathcal{Q}_{odd}$ are non-intersecting. The sequence starts with a positive cycle C^+ and ends with a non-positive cycle C^+_{odd} so there must exists an index i such that C^i_{odd} is positive and C^{i+1}_{odd} is non-positive. However the number of positive edges in the non-positive cycle C^{i+1}_{odd} is

$$\begin{aligned}|E^+(C^{i+1}_{odd})| &= |E^+(C^i_{odd})| + |E^+(Q_{2i+1})| - |E^+(r(Q_{2i+1})| \\ &= |E^+(C^i_{odd})| + |E^+(C_{Q_{2i+1}})| - |E^+(C^+)| \\ &> \frac{2}{3}k + \frac{2}{3}k - k \qquad \text{(by Observation 10)} \\ &= \frac{1}{3}k\end{aligned}$$

By Observation 10, this implies that C^{i+1}_{odd} is positive, but we previously argued that it is non-positive. Hence it is not possible that $e \notin r_b(C_e)$. ◀

Tighter Approximation for the Uniform Cost-Distance Steiner Tree Problem

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

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

Yannik Kyle Dustin Spitzley
Research Institute for Discrete Mathematics and Hausdorff Center for Mathematics, University of Bonn, Germany

Abstract

Uniform cost-distance Steiner trees minimize the sum of the total length and weighted path lengths from a dedicated root to the other terminals. They are applied when the tree is intended for signal transmission, e.g. in chip design or telecommunication networks. They are a special case of general cost-distance Steiner trees, where different distance functions are used for total length and path lengths.

We improve the best published approximation factor for the uniform cost-distance Steiner tree problem from 2.39 [15] to 2.05. If we can approximate the minimum-length Steiner tree problem arbitrarily well, our algorithm achieves an approximation factor arbitrarily close to $1 + \frac{1}{\sqrt{2}}$. This bound is tight in the following sense. We also prove the gap $1 + \frac{1}{\sqrt{2}}$ between optimum solutions and the lower bound which we and all previous approximation algorithms for this problem use.

Similarly to previous approaches, we start with an approximate minimum-length Steiner tree and split it into subtrees that are later re-connected. To improve the approximation factor, we split it into components more carefully, taking the cost structure into account, and we significantly enhance the analysis.

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

Keywords and phrases cost-distance Steiner tree, approximation algorithm, uniform

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.19

Category APPROX

Related Version *Full Version*: https://arxiv.org/abs/2305.03381

1 Introduction

Steiner trees can be found in numerous applications, in particular in chip design and telecommunications. In these applications, both the total tree length and the signal speed are important. We consider Steiner trees that do not only minimize the total cost, but also the weighted path lengths from a dedicated root r to the other terminals. Formally, the problem is defined as follows.

An instance (M, c, T, r, p, w) consists of a metric space (M, c), a root r, a finite set T of sinks, a map $p : T\dot{\cup}\{r\} \to M$, and sink delay weights $w\colon T \to \mathbb{R}_{\geq 0}$. The task is to compute a Steiner tree A for $T \cup \{r\}$ with an extension $p : V(A) \setminus (T \cup \{r\}) \to M$ minimizing

$$\sum_{\{x,y\}\in E(A)} c(p(x),p(y)) + \sum_{t\in T}\left(w(t)\sum_{\{x,y\}\in E(A_{[r,t]})} c(p(x),p(y))\right), \tag{1}$$

where $A_{[r,t]}$ is the unique r-t-path in A. We call (1) the **(total) cost** of (A,p).

Given a Steiner tree A, we call

$$\sum_{\{x,y\}\in E(A)} c(p(x),p(y)) \quad \text{and} \quad \sum_{t\in T}\left(w(t)\sum_{\{x,y\}\in E(A_{[r,t]})} c(p(x),p(y))\right)$$

its **connection cost** and its **delay cost**.

Usually the position p of vertices is clear from the context. Then, we simply write $c(x,y)$ instead of $c(p(x),p(y))$ and $c(e)$ instead of $c(x,y)$ for edges $e=\{x,y\}$. To shorten the notation, we often also omit the underlying metric space from the notation and write only (T,r,w) to denote an instance. A simple lower bound for the objective function, is given by

$$C_{SMT}(T\cup\{r\}) + D(T,r,w), \tag{2}$$

where $C_{SMT}(T\cup\{r\})$ is the connection cost of a minimum-length Steiner tree for $T\cup\{r\}$, i.e. a Steiner tree A for $T\cup\{r\}$ minimizing $\sum_{e\in E(A)} c(e)$, and $D(T,r,w) := \sum_{t\in T} w(t)c(r,t)$ is the sum of weighted root-sink distances.

The Uniform Cost-Distance Steiner Tree Problem was first mentioned by [17], who considered the (general) cost-distance Steiner tree problem, where the connection cost may be unrelated to the delay cost. Cost-distance Steiner trees are heavily used in VLSI routing and interconnect optimization [11, 7]. Here, the weights arise as Lagrangean multipliers when optimizing global signal delay constraints on an integrated circuit [11]. Uniform cost-distance Steiner trees are computed as a first step of a Steiner tree oracle in global routing [11, 7].

The general cost-distance Steiner tree problem does not permit an approximation factor better than $\Omega(\log\log|T|)$ unless NP $\subseteq$ DTIME$(|T|^{\mathcal{O}(\log\log\log|T|)})$ [6], while a randomized $\mathcal{O}(\log|T|)$-factor approximation algorithm was given by [17] and [4]. Meyerson, Munagala and Plotkin [17] observed that a constant factor approximation algorithm for the Uniform Cost-Distance Steiner Tree Problem can be obtained using the shallow-light spanning tree algorithm from [16]. The resulting factor is 3.57. Using shallow-light Steiner trees [12] instead of spanning trees, the factor was improved to 2.87 independently by [10] and [18]. The first algorithm using the delay weights algorithmically was given by Khazraei and Held [15]. They achieve an approximation factor of $1+\beta$, where β is the approximation factor for computing a minimum-length Steiner tree. All these approaches compare against the lower bound in (2).

Similarly to algorithms for shallow-light trees, the algorithm in [15] starts from an approximately minimum Steiner tree, which is cut into a forest whose components are connected to the root r individually. While [16] cut the tree whenever the path length is too large, [15] cut off a subtree if its delay weight exceeds a certain threshold. Each cut-off tree is later re-connected through a direct connection from the root through one of its terminals, minimizing the resulting objective function.

The special case where we require a spanning tree instead of a Steiner tree and $w(t)$ is identical for all $t\in T$ is known as the *cable-trench problem.* It does not permit a PTAS unless P = NP [2].

The Uniform Cost-Distance Steiner Tree Problem is related to the *single-sink buy-at-bulk problem* where a set of demands needs to be routed from a set of sources to a single sink using a pipe network that has to be constructed from a finite set of possible pipe types with different costs and capacities [9, 19, 14]. The best known approximation factor for this problem is 40.82 due to [8], who also achieve a factor of 20.41 for the splittable case. If there is only one pipe type this problem is equivalent to the Uniform Cost-Distance Steiner Tree Problem. In fact, the threshold-based tree cutting used in the proof of [15] is similar to the algorithm in [9], but the re-connect to the root/sink differs.

1.1 Our contribution

In this paper, we improve the approximation algorithm in [15] for the Uniform Cost-Distance Steiner Tree Problem.

▶ **Theorem 1.** *The Uniform Cost-Distance Steiner Tree Problem can be approximated in polynomial time with an approximation factor of*

$$\beta + \frac{\beta}{\sqrt{\beta^2+1}+\beta-1},$$

where $\beta \geq 1$ is the approximation guarantee for the minimum-length Steiner tree problem.

With the best known approximation factor for the minimum Steiner tree problem $\beta = \ln(4)+\epsilon$ [3, 20], this results in an approximation factor < 2.05 and for $\beta = 1$ this gives the factor $1 + \frac{1}{\sqrt{2}} < 1.71$, clearly improving upon the previously best factors 2.39 and 2.0 in [15]. The polynomial-time approximation scheme by [1] allows choosing β arbitrarily close to one in the Euclidean and the Manhattan planes. However, general metric spaces do not allow $\beta \leq \frac{96}{95}$ unless $\mathrm{P} = \mathrm{NP}$ [5].

Assuming an ideal Steiner tree approximation factor of $\beta = 1$, our new approximation factor is tight with respect to the lower bound (2). We prove the following result

▶ **Theorem 2.**

$$\sup_{T,c,w} \frac{\mathrm{OPT}(T,r,w)}{C_{SMT}(T\cup\{r\}) + D(T,r,w)} = 1 + \frac{1}{\sqrt{2}},$$

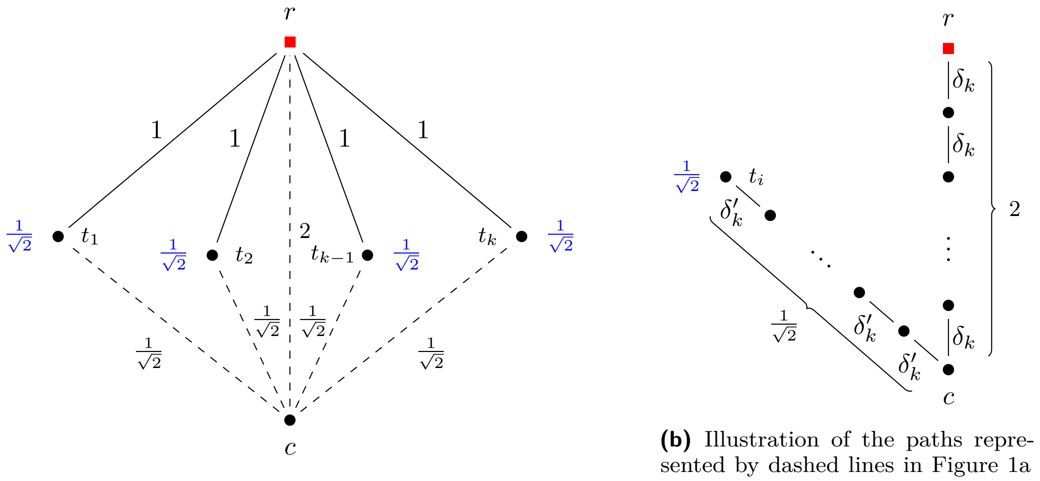
where $\mathrm{OPT}(T,r,w)$ denotes the optimum solution value for (T,r,w).

The algorithm in [15] starts from a short Steiner tree and iteratively splits off subtrees whose delay weight exceeds a given threshold. We proceed similarly, but we also take the structure of the subtrees into account and split off subtrees once they can be reconnected efficiently.

While [15] obtain a running time of $\mathcal{O}(\Lambda + |T|^2)$, where Λ is the time to compute an initial β-approximate minimum Steiner tree, our running time is $\mathcal{O}(\Lambda + |T|)$ (assuming that the metric c can be evaluated in constant time). Thus, it is very fast and applicable for large chip design instances.

We would like to mention that in a preliminary unpublished paper we achieved worse factors of 2.15 ($\beta = \ln(4)$) and 1.80 ($\beta = 1$) [13] using a more complicated algorithm and analysis. That paper also shows that the factor $1+\beta$ in [15] is tight for that algorithm.

The remainder of this paper is structured as follows. In Section 2, we show that the supremum in Theorem 2 is at least $1 + \frac{1}{\sqrt{2}}$. Then, in Section 3 we will briefly summarize the algorithm and proof from [15], as our work enhances it.



(a) Illustration of the complete instance.

(b) Illustration of the paths represented by dashed lines in Figure 1a by a single t_i.

Figure 1 Instance defined in the proof of Theorem 3. Solid lines represent edges, each dashed line represents a path. Black figures denote edge/path lengths, blue figures denote terminal weights.

Our improved splitting algorithm and analysis is presented in Section 4. The proof of Theorem 1 is presented in Section 4.2. It also shows that the supremum in Theorem 2 is at most $1 + \frac{1}{\sqrt{2}}$. We finish with conclusions in Section 5.

2 Optimality Gap of Lower Bound

In this section, we will show that the gap between an optimum solution and the lower bound in (2) can be as large as $1+\frac{1}{\sqrt{2}}$. Together with the approximation factor of our new algorithm for $\beta = 1$ (Theorem 1), the gap is asymptotically $1 + \frac{1}{\sqrt{2}}$.

▶ **Theorem 3.** *There are instances $I^{(k)}_{k\in\mathbb{N}}$ with $I^{(k)} = (T^{(k)}, r^{(k)}, w^{(k)})$ ($k \in \mathbb{N}$) for the uniform cost-distance Steiner tree problem such that*

$$\lim_{k\to\infty} \frac{\mathrm{OPT}^{(k)}}{C^{(k)} + D^{(k)}} = 1 + \frac{1}{\sqrt{2}},$$

where $\mathrm{OPT}^{(k)}$ is the optimum value for the instance $I^{(k)}$, while $C^{(k)} = C_{SMT}(T^{(k)} \cup \{r\})$, $D^{(k)} := D(T^{(k)}, r^{(k)}, w^{(k)})$ denote minimum possible connection cost and the minimum possible delay cost of $I^{(k)}$.

Proof. We will construct instances with underlying graph metrics induced by graphs indicated in Figure 1a. For $k \in \mathbb{N}$, we define the graph $G^{(k)} = (V^{(k)}, E^{(k)})$ by

$$V^{(k)} = \{r^{(k)}, c^{(k)}, t_1^{(k)}, \dots, t_k^{(k)}, v_1^{(k)}, \dots, v_q^{(k)}\}$$
$$\text{and } E^{(k)} = E_{r,c}^{(k)} \,\dot\cup\, E_{c,t}^{(k)} \,\dot\cup\, E_{r,t}^{(k)},$$

where q is chosen sufficiently large to provide sufficiently many inner path vertices $v_i^{(k)}$ ($1 \le i \le q$) in the following definitions of $E_{r,c}^{(k)}$, $E_{c,t}^{(k)}$ and $E_{r,t}^{(k)}$: For $0 < \delta_k < \delta'_k < \frac{1}{k}$, $E_{r,c}^{(k)}$ contains edges of length δ_k forming a path of total length 2 between $r^{(k)}$ and $c^{(k)}$, and $E_{c,t}^{(k)}$ contains edges of length δ'_k, forming paths of length $\frac{1}{\sqrt{2}}$ between $c^{(k)}$ and each $t_i^{(k)}$. Lastly,

$$E_{r,t}^{(k)} = \{\{r^{(k)}, t_i^{(k)}\} \mid i = 1, \dots, k\}$$

connects each $t_i^{(k)}$ directly to $r^{(k)}$ with an edge of length 1. The terminals are given by $T^{(k)} = V^{(k)} \setminus \{r^{(k)}\}$, and the delay weights $w^{(k)} \colon T^{(k)} \to \mathbb{R}_{\geq 0}$ are defined as

$$w^{(k)}(t) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } t \in \{t_1^{(k)}, \dots, t_k^{(k)}\} \\ 0 & \text{else.} \end{cases}$$

Now, the lower bound becomes:

$$C^{(k)} + D^{(k)} = 2 + \frac{k}{\sqrt{2}} + \sum_{i=1}^{k} \frac{1}{\sqrt{2}} \operatorname{dist}_{G^{(k)}}(r^{(k)}, t_i^{(k)}) = 2 + 2\,\frac{k}{\sqrt{2}} = 2 + \sqrt{2}\,k. \tag{3}$$

We claim that every optimum solution contains all edges of the form $\{r^{(k)}, t_i^{(k)}\}$. Additionally, we claim that all optimum solutions contain all edges of length δ_k and all but k edges of length δ'_k. This determines the structure of an optimum solution up to the choice of the k ommitted edges. The length of an optimum solution is $(1 + \frac{1}{\sqrt{2}})k + 2 - \delta'_k\ k$, and its objective is

$$\mathrm{OPT}^{(k)} = \left(1 + \frac{1}{\sqrt{2}}\right) k + 2 - \delta'_k k + \frac{k}{\sqrt{2}} = (1 + \sqrt{2})k + 2 - \delta'_k k. \tag{4}$$

Combining (3) and (4), we see that

$$\lim_{k\to\infty} \frac{\mathrm{OPT}^{(k)}}{C^{(k)} + D^{(k)}} = \lim_{k\to\infty} \frac{(1 + \sqrt{2})k + 2 - \delta'_k k}{2 + \sqrt{2}\ k} = \frac{1 + \sqrt{2}}{\sqrt{2}} = 1 + \frac{1}{\sqrt{2}}$$

as stated in the theorem.

To prove the first claim, assume there is an optimum solution Y^* not containing an edge $\{r^{(k)}, t_i^{(k)}\}$ for some $i \in \{1, \dots, k\}$. First, observe that any path from $r^{(k)}$ to $t_i^{(k)}$ not using the edge $\{r^{(k)}, t_i^{(k)}\}$ contains c, so we have

$$\operatorname{dist}_{Y^*}(r^{(k)}, t_i^{(k)}) \geq \operatorname{dist}_{G^{(k)}}(r^{(k)}, c^{(k)}) + \operatorname{dist}_{G^{(k)}}(c^{(k)}, t_i^{(k)}) = 1 + \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}} = 1 + \sqrt{2}.$$

Let e be the edge of length δ'_k adjacent to $t_i^{(k)}$. Then $e \in E(Y^*)$, as otherwise $t_i^{(k)}$ would be isolated in Y^*. Now define Y' from Y^* by adding $\{r^{(k)}, t_i^{(k)}\}$ and removing e. This increases the connection cost by $1 - \delta'_k$. The delay cost decreases by at least

$$w(t_i^{(k)}) \left(\operatorname{dist}_{Y^*}(r^{(k)}, t_i^{(k)}) - \operatorname{dist}_{Y'}(r^{(k)}, t_i^{(k)})\right) \geq \frac{1 + \sqrt{2} - 1}{\sqrt{2}} = 1,$$

where we use $\operatorname{dist}_{Y'}(r^{(k)}, t_j^{(k)}) \leq \operatorname{dist}_{Y^*}(r^{(k)}, t_j^{(k)})$ for $j \neq i$. Thus, the total cost decreases by at least $1 - (1 - \delta'_k) = \delta'_k$, a contradiction to the optimality of Y^*.

Now we prove the second claim: By the first claim all optimum solutions have the same delay cost $\frac{k}{\sqrt{2}}$. Hence, only the connection cost for the remaining terminals is relevant. From each maximal path ending in c consisting only of short edges of length either δ_k or δ'_k, any solution must contain either all edges or all but one. Furthermore, there must be such a path from which the solution contains all edges, otherwise there would be no r-c-path. Since $\delta_k < \delta'_k$, the shortest such configuration is to take all edges of length δ_k and all but k edges of length δ'_k (namely all but one from each path). ◀

Algorithm 1 $(1+\beta)$-approximation algorithm by [15] using a parameter $\mu > 0$.

Step 1 (initial arborescence):
First, compute a β-approximate minimum cost Steiner r-arborescence A_0 for $T \cup \{r\}$ with outdegree 0 at all sinks in T and outdegree 2 at all Steiner vertices in $V(A_0) \setminus (T \cup \{r\})$.

Step 2 (split into branching):
Traverse A_0 bottom-up. For each traversed edge $(x,y) \in E(A_0)$, if $W_{(A_0)_y} > \mu$, remove the edge (x,y) creating a new arborescence $(A_0)_y$ in the branching.

Let $\mathcal{A}$ denote the set of all arborescences that were cut off from A_0 this way.

Step 3 (Re-connect arborescences):
Re-connect each sub-arborescence A' that was cut off in Step 2 as follows: Select a vertex $t \in T' := T_{A'}$ that minimizes the cost for serving the sinks in T' through the r-arborescence $A' + (r,t)$, i.e. select a vertex $t \in T'$ as a **port** for T' that minimizes

$$c(r,t) + C_{A'} + \sum_{t' \in T'} w(t') \cdot (c(r,t) + c(E(A'_{[t,t']}))).$$

Let $t_1, \ldots, t_{|\mathcal{A}|} \in T$ be the set of selected port vertices. Return the union of the final branching and the port connections $A_0 + \{(r,t_i) : i \in \{1,\ldots,|\mathcal{A}|\}\ \}$.

3 The $(1+\beta)$-approximation algorithm

For shorter formulas, we will use the following notation in the remainder of this paper. Let A be an arborescence. By A_v we denote the sub-arborescence rooted at v. Furthermore, $T_A := V(A) \cap T$ is the set of terminals in A, $W_A := w(T_A)$ is the sum of delay weights in A, $C_A := c(E(A))$ is the **connection cost** of A and $D_A := D_{T_A} := \sum_{t \in T_A} w(t)c(r,t)$ the **minimum possible delay cost** for connecting the sinks in T_A (independent of the structure of A).

Recall that $\beta \geq 1$ is the approximation guarantee for the minimum-length Steiner tree problem. The algorithm in [15] is described in Algorithm 1. After orienting its edges, we can consider any solution A as an r-arborescence. We use arborescences instead of trees to simplify the algorithmic notation.

3.1 Essential steps for a $1+\beta$ approximation

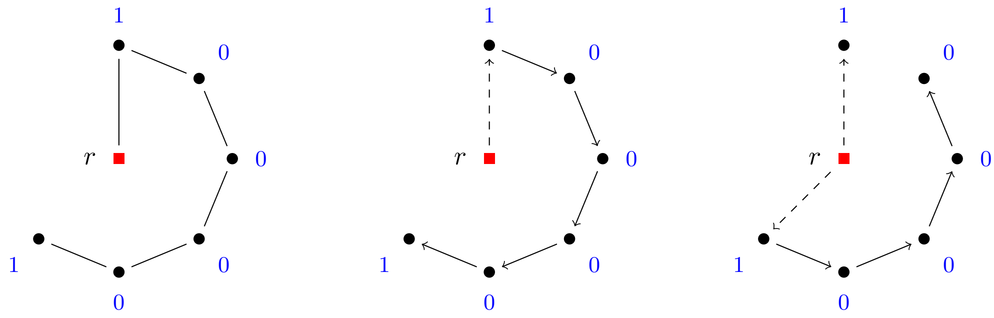
We quickly recap the essential steps in the analysis of [15], which we will use in our analysis. The cost to connect an arborescence $A' \in \mathcal{A}$ to the root r can be estimated as follows:

▶ **Lemma 4** (Khazraei and Held [15], Lemma 1). *Let $A' \in \mathcal{A}$ with corresponding terminal set T'. By the choice of the port $t \in T'$, the r-arborescence $(A' + \{r,t\})$ has a total cost at most*

$$C_{A'} + \sum_{e=(x,y) \in E(A')} \frac{2W_{A'_y}(W_{A'} - W_{A'_y})}{W_{A'}} c(e) + \left(1 + \frac{1}{W_{A'}}\right) D_{T'} \tag{5}$$

$$\leq \left(1 + \frac{W_{A'}}{2}\right) C_{A'} + \left(1 + \frac{1}{W_{A'}}\right) D_{T'} \tag{6}$$

$$\leq (1+\mu)\, C_{A'} + \left(1 + \frac{1}{\mu}\right) D_{T'}. \tag{7}$$



(a) Minimum-length Steiner tree. **(b)** Cost: $6 + (1 + 6) = 13$. **(c)** Cost: $6 + (1 + 1) = 8$.

Figure 2 Weakness of Algorithm 1: (M, c) is induced by a complete graph with seven vertices and unit weights. Delay weights are indicated by the blue node labels and $\mu = 1$. Algorithm 1 might start with the minimum-length Steiner tree on the left. Then the algorithm will cut the edge incident to r and reconnect the sub-arborescence resulting possibly in the solution in the middle. On the right a better splitting is shown.

We sketch the proof in Appendix A, because we use an inequality that was not stated explicitly in [15], Lemma 1. Note (from the proof) that only the bound (7) uses how the arborescences $A' \in \mathcal{A}$ were cut off during Step 2. In particular the bounds (5)-(6) will also apply for our improvement.

A similar cost bound can be shown easily for the arborescence A_r containing the root r after Step 2. Summing up the resulting cost bounds and choosing $\mu = \frac{1}{\beta}$, [15] obtain the approximation factor $(1 + \beta)$.

4 Improving the approximation ratio

Algorithm 1 suffers from the following weakness. Assume that after splitting we are given a sub-arborescence $A' \in \mathcal{A}$ with a high delay weight $W_{A'}$, a high connection cost $C_{A'}$, but a low minimum possible delay cost $D_{A'}$, e.g. as shown in Figure 2.

In this section, we propose a refined splitting criterion that provides a better approximation ratio. Instead of using a fixed threshold μ, we allow to split off sub-arborescences earlier if their expected re-connection cost (5) is sufficiently cheap. The precise criterion is specified in (8) (inside Algorithm 2). Observe that (8) provides cheaper solutions than (7), as one occurrence of μ is replaced by $\frac{\mu}{2}$.

Then we show in Lemma 8 that every sub-arborescence of the remaining root component has delay weight at most μ. This allows us to prove a similar improved cost bound for the root component in Lemma 9.

In Section 4.2, we simply combine all sub-arborescences and choose μ to prove Theorem 1. Theorem 2 follows as an immediate consequence.

4.1 Improving the splitting routine

Algorithm 2 shows our improved splitting step, which cuts off a sub-arborescence if we can re-connect it cheaply, i.e. if (8) holds. With Lemma 4 we immediately get the following result for the cut-off sub-arborescences:

Algorithm 2 Modifying Step 2 of Algorithm 1.

Step 2 (split into branching):
Traverse A_0 bottom-up. For each traversed edge $(v,z) \in E(A_0)$ consider $A_z := (A_0)_z$:
If $W_{A_z} > 0$ and

$$\sum_{e=(p,q)\in E(A_z)} \frac{2W_{(A_z)_q}(W_{A_z} - W_{(A_z)_q})}{W_{A_z}} c(e) + \frac{D_{A_z}}{W_{A_z}} \leq \frac{\mu}{2}\left(C_{A_z} + c(v,z)\right) + \frac{D_{A_z}}{\mu}, \quad (8)$$

remove (v,z) creating a new arborescence A_z.

▶ **Lemma 5.** *Let $A' \in \mathcal{A}$ be an arborescence that was cut off in Algorithm 2 and let $e_{A'}$ be the incoming edge in the root of the arborescence A' which was deleted during this step. Then the corresponding terminals in $T_{A'}$ can be connected to the root r with total cost at most*

$$\left(1 + \frac{\mu}{2}\right)\left(C_{A'} + c(e_{A'})\right) + \left(1 + \frac{1}{\mu}\right) D_{A'}.$$

After the original Step 2 of Algorithm 1, it is clear that for all edges $(r,x) \in \delta^+_{A_0}(r)$ of the root component the total delay weight $W_{(A_0)_x}$ is at most μ. We show that this also holds after the modified Step 2 in Algorithm 2. However, the analysis is more complicated and uses the following two functions.

▶ **Definition 6.** *Let $\mu > 0$ and $X^\mu := \{(a,b,c) \in (\mu, 2\mu) \times (0,\mu)^2 : c \leq a - b < \mu\}$. We define the functions $f, g\colon X^\mu \to \mathbb{R}$ as*

$$f(a,b,c) := \frac{2(a-c)c}{a} - \frac{\mu}{2} + \left(\frac{1}{a} - \frac{1}{\mu}\right) \cdot \frac{1}{\frac{1}{a-b} - \frac{1}{\mu}} \cdot \left(\frac{\mu}{2} - \frac{2((a-b)-c)c}{a-b}\right)$$

$$g(a,b,c) := \frac{2(a-c)c}{a} - \frac{\mu}{2} + \left(\frac{1}{a} - \frac{1}{\mu}\right) \cdot \frac{1}{\frac{1}{a-b} - \frac{1}{\mu}} \cdot \frac{\mu}{2}.$$

▶ **Lemma 7.** *For all $(a,b,c) \in X^\mu$, $f(a,b,c) \leq 0$ and $g(a,b,c) \leq 0$.*

A proof of Lemma 7 based on algebraic transformations can be found in Appendix B.

▶ **Lemma 8.** *After cutting off sub-arborescences with Algorithm 2, every child $x \in \Gamma^+_{A_r}(r)$ of r in the remaining root component $A_r := (A_0)_r$ satisfies $W_{(A_r)_x} \leq \mu$.*

Proof. Assume the opposite would be true. Let z be a vertex in $A_r - r$ such that the weight of the sub-arborescence $A_z := (A_r)_z$ exceeds μ and the weight of every child arborescence $(A_z)_x$ is at most μ for all edges $(z,x) \in \delta^+_{A_z}(z)$. We distinguish between two cases:

Case 1. z is a terminal. Then z is also a leaf and the left-hand side of (8) simplifies to

$$\frac{1}{W_{A_z}} D_{A_z} \leq \frac{1}{\mu} D_{A_z}$$

since A_z does not contain any edges. But then A_z would have been cut-off in Step 2, a contradiction.

Case 2. z is a Steiner vertex. Then z has two outgoing edges $e_x := (z,x), e_y := (z,y) \in \delta^+_{A_z}(z)$ as shown in Figure 3. A single outgoing edge would contradict the choice of z. With $A_x := (A_z)_x$ or $A_y := (A_z)_y$ this implies $0 < W_{A_x}, W_{A_y} \leq \mu$. If $W_{A_x} = \mu$, Lemma 4, (6)

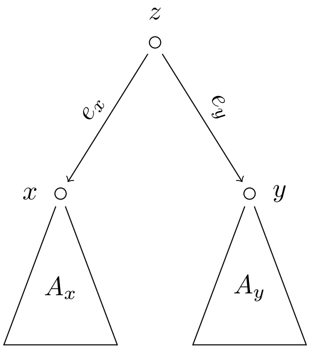


Figure 3 Setting in the proof of Lemma 8 if z is not a terminal.

shows that A_x satisfied the bound (8) when it was considered in Step 2 and would have been cut off. Analogously, $W_{A_y} \neq \mu$. Thus, $W_{A_x}, W_{A_y} < \mu$. Since (8) does not hold for A_x, we get (by transforming its negation)

$$\underbrace{\left(\frac{1}{W_{A_x}} - \frac{1}{\mu}\right)}_{>0} D_{A_x} > \sum_{e=(u,v)\in E(A_x)} \left(\frac{\mu}{2} - \frac{2(W_{A_x} - W_{(A_x)_v})W_{(A_x)_v}}{W_{A_x}}\right) c(e) + \frac{\mu}{2} c(e_x)$$

Combining this with the analogue inequality for A_y and using $D_{A_z} = D_{A_x} + D_{A_y}$, we get

$$\begin{aligned}
&\underbrace{\left(\frac{1}{W_{A_z}} - \frac{1}{\mu}\right)}_{<0} D_{A_z} \\
&\quad < \left(\frac{1}{W_{A_z}} - \frac{1}{\mu}\right) \Bigg(\sum_{e=(u,v)\in E(A_x)} \frac{1}{\frac{1}{W_{A_x}} - \frac{1}{\mu}} \left(\frac{\mu}{2} - \frac{2(W_{A_x} - W_{(A_x)_v})W_{(A_x)_v}}{W_{A_x}}\right) c(e) \\
&\qquad\qquad + \frac{\frac{\mu}{2}}{\frac{1}{W_{A_x}} - \frac{1}{\mu}} c(e_x) \\
&\qquad\qquad + \sum_{e=(u,v)\in E(A_y)} \frac{1}{\frac{1}{W_{A_y}} - \frac{1}{\mu}} \left(\frac{\mu}{2} - \frac{2(W_{A_y} - W_{(A_y)_v})W_{(A_y)_v}}{W_{A_y}}\right) c(e) \\
&\qquad\qquad + \frac{\frac{\mu}{2}}{\frac{1}{W_{A_y}} - \frac{1}{\mu}} c(e_y) \Bigg).
\end{aligned}$$

This inequality together with

$$\begin{aligned}
&\sum_{e=(u,v)\in E(A_z)} \left(\frac{2(W_{A_z} - W_{(A_z)_v})W_{(A_z)_v}}{W_{A_z}} - \frac{\mu}{2}\right) c(e) \\
&= \sum_{e=(u,v)\in E(A_x)} \left(\frac{2(W_{A_z} - W_{(A_z)_v})W_{(A_z)_v}}{W_{A_z}} - \frac{\mu}{2}\right) c(e) \\
&\quad + \left(\frac{2(W_{A_z} - W_{(A_z)_x})W_{(A_z)_x}}{W_{A_z}} - \frac{\mu}{2}\right) c(e_x) \\
&\quad + \sum_{e=(u,v)\in E(A_y)} \left(\frac{2(W_{A_z} - W_{(A_z)_v})W_{(A_z)_v}}{W_{A_z}} - \frac{\mu}{2}\right) c(e) \\
&\quad + \left(\frac{2(W_{A_z} - W_{(A_z)_y})W_{(A_z)_y}}{W_{A_z}} - \frac{\mu}{2}\right) c(e_y)
\end{aligned}$$

yields

$$\begin{aligned}
&\sum_{e=(u,v)\in E(A_z)} \left(\frac{2(W_{A_z} - W_{(A_z)_v})W_{(A_z)_v}}{W_{A_z}} - \frac{\mu}{2}\right) c(e) + \left(\frac{1}{W_{A_z}} - \frac{1}{\mu}\right) D_{A_z} \\
&\quad < \sum_{e=(u,v)\in E(A_x)} f(W_{A_z}, W_{A_y}, W_{(A_z)_v})c(e) + \sum_{e=(u,v)\in E(A_y)} f(W_{A_z}, W_{A_x}, W_{(A_z)_v})c(e) \\
&\quad + g(W_{A_z}, W_{A_y}, W_{(A_z)_v})c(e_x) + g(W_{A_z}, W_{A_x}, W_{(A_z)_v})c(e_y).
\end{aligned}$$

By Lemma 7 and $0 < W_{A_x}, W_{A_y} < \mu$, the last term is non-positive. Therefore A_z satisfied the bound (8) when it was considered in Step 2 and would have been cut off, a contradiction. ◀

In [15] the final root arborescence A_r, which was not cut off in Step 2 of Algorithm 1, was kept unaltered. Using Lemma 8, we show how to connect it in a better way.

▶ **Lemma 9.** *Let A_r be the sub-arborescence of A_0 rooted at r after the modified Step 2 of Algorithm 1. The terminal set T_{A_r} can be connected to the root r with total cost at most*

$$\left(1 + \frac{\mu}{2}\right) C_{A_r} + \left(1 + \frac{1}{\mu}\right) D_{A_r}.$$

Proof. Let $(r, x) \in \delta^+_{A_r}(r)$ be arbitrary and A_x the arborescence of $A_r - r$ rooted at x. We show that the terminal set T_{A_x} can be connected to the root r with total cost at most

$$\left(1 + \frac{\mu}{2}\right) (C_{A_x} + c(r, x)) + \left(1 + \frac{1}{\mu}\right) D_{A_x}.$$

Adding this cost for all edges in $\delta^+_{A_r}(r)$, we obtain the claim.

We distinguish between two cases:

Case 1.

$$W_{A_x}(C_{A_x} + c(r, x)) \le \frac{\mu}{2}(C_{A_x} + c(r, x)) + \frac{1}{\mu} D_{A_x}.$$

By keeping the arborescence A_x connected through (r, x), the connection cost is $C_{A_x} + c(r, x)$. In particular, for each terminal $t \in T_{A_x}$, the r-t-path in $A_x + (r, x)$ has a length of at most $C_{A_x} + c(r, x)$. We therefore obtain a total cost of at most

$$(1 + W_{A_x})(C_{A_x} + c(r, x)) \le \left(1 + \frac{\mu}{2}\right) (C_{A_x} + c(r, x)) + \frac{1}{\mu} D_{A_x}.$$

Case 2.

$$W_{A_x}(C_{A_x} + c(r, x)) > \frac{\mu}{2}(C_{A_x} + c(r, x)) + \frac{1}{\mu} D_{A_x}. \tag{9}$$

Therefore we have $W_{A_x} > 0$ and obtain from (9) an upper bound on the minimum possible delay cost of A_x

$$D_{A_x} < \left(W_{A_x} - \frac{\mu}{2}\right) \mu (C_{A_x} + c(r, x)). \tag{10}$$

We remove the edge (r, x) and connect the arborescence A_x to the root r. By Lemma 8, $W_{A_x} \leq \mu$. As in Lemma 4 we obtain total cost of at most

$$\begin{aligned}
&\left(1+\frac{W_{A_x}}{2}\right)C_{A_x}+\left(1+\frac{1}{W_{A_x}}\right)D_{A_x}\\
&=\left(1+\frac{W_{A_x}}{2}\right)C_{A_x}+\left(1+\frac{1}{\mu}\right)D_{A_x}+\underbrace{\frac{\mu-W_{A_x}}{\mu W_{A_x}}}_{\geq 0}D_{A_x}\\
&\overset{(10)}{\leq}\left(1+\frac{W_{A_x}}{2}\right)C_{A_x}+\left(1+\frac{1}{\mu}\right)D_{A_x}+\frac{\mu-W_{A_x}}{\mu W_{A_x}}\left(W_{A_x}-\frac{\mu}{2}\right)\mu(C_{A_x}+c(r,x))\\
&\leq\left(1-\frac{W_{A_x}}{2}+\frac{3}{2}\mu-\frac{\mu^2}{2W_{A_x}}\right)(C_{A_x}+c(r,x))+\left(1+\frac{1}{\mu}\right)D_{A_x}.
\end{aligned}$$

With the following estimation we obtain the claimed bound

$$-\frac{W_{A_x}}{2}+\frac{3}{2}\mu-\frac{\mu^2}{2W_{A_x}}=\frac{\mu}{2}-\frac{1}{2}\left(\sqrt{W_{A_x}}-\frac{\mu}{\sqrt{W_{A_x}}}\right)^2\leq\frac{\mu}{2}.$$ ◀

▶ **Theorem 10.** *Algorithm 2 and the re-connect in Step 3 as well as of the root component can be implemented to run in time $\mathcal{O}(|T|)$.*

Proof. A naïve implementation would immediately result in a quadratic running time. We can achieve a linear running time by computing all relevant information incrementally in constant time per node during the bottom-up traversal. Details can be found in Appendix C. ◀

4.2 Proving Theorem 1 and Theorem 2

We start by analyzing the combination of all sub-arborescences.

▶ **Theorem 11.** *Given an instance (T, r, w) of the Uniform Cost-Distance Steiner Tree Problem, we can compute in $\mathcal{O}(\Lambda + |T|)$ time a Steiner tree with objective value at most*

$$\left(1+\frac{\mu}{2}\right)C+\left(1+\frac{1}{\mu}\right)D, \tag{11}$$

where C is the cost of a β-approximate minimum-length Steiner tree and $D := D(T, r, w)$. Here, Λ is the running time for computing a β-approximate minimum Steiner tree for $T \cup \{r\}$.

Proof. We run Algorithm 1 with two modifications:

1. The cut-off routine (Step 2) is modified according to Algorithm 2.
2. The arborescence A_r containing the root r after Step 2 may be re-connected to the root r according to Lemma 9.

The total cost of the computed solution is upper bounded by the sum of the cost bounds for these r-arborescences, which is (11). For the running time analysis, we consider the individual steps of the algorithm:

In Step 1, a β-approximate minimum Steiner tree for $T \cup \{r\}$ is computed in time $\mathcal{O}(\Lambda)$ and transformed into the arborescence A_0 obeying the degree constraints in linear time as in [15]. The linear running time of Step 2 and Step 3 follows from Theorem 10. ◀

Finally, we choose the threshold μ based on the quantities C and D to prove Theorem 1:

Table 1 Comparison of approximation factors for the Uniform Cost-Distance Steiner Tree Problem with different approximation factors β for the minimum-length Steiner tree problem.

Parameter β	1	$\ln(4)+\epsilon$	$\frac{3}{2}$	2
Algorithm 1 [15]	2.00000	2.38630	2.50000	3.00000
Theorem 1	1.70711	2.04782	2.15139	2.61804

Proof of Theorem 1. We make the following modification of the algorithm in Theorem 11:

If $C = c(E(A_0)) = 0$, each r-t-path, $t \in T$, has length 0 in A_0. So this is already an optimal solution and we just return A_0.

Otherwise, set $\mu := \sqrt{\frac{2D}{C}}$ and the algorithm from Theorem 11 provides us with a solution with total cost at most

$$C + D + \sqrt{2}\sqrt{CD} \leq \beta C_{SMT}(T \cup \{r\}) + D + \sqrt{2}\sqrt{\beta C_{SMT}(T \cup \{r\}) \cdot D}.$$

We divide this by the lower bound $C_{SMT}(T \cup \{r\}) + D$ in (2). Now, the approximation factor is at most the maximum of the function $h\colon \mathbb{R}_{>0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ given by

$$h(x,y) := \frac{\beta x + y + \sqrt{2}\sqrt{\beta x y}}{x+y}.$$

As we proof in Appendix D using algebraic reformulations,

$$h(x,y) \leq \beta + \frac{\beta}{\sqrt{\beta^2+1}+\beta-1},$$

proving the claimed approximation ratio. ◀

Using Theorem 1 we obtain the approximation factors shown in Table 1 (rounded to five decimal digits) for some interesting values of β in Table 1.

Proof of Theorem 2. This is a direct consequence of Theorem 3 and Theorem 1 for $\beta = 1$. ◀

5 Conclusion

We significantly improve the approximation factor for the Uniform Cost-Distance Steiner Tree Problem. For the lower bound (2) it is best possible if the minimum-length Steiner tree problem can be solved optimally.

This is achieved by an enhancement of the cut-off routine, where we do not simply cut off by delay weight, but take the (cost) structure of the sub-arborescences into account. Furthermore, the root component will be reconnected in a smarter way.

Our algorithm is very fast. After computing an approximate minimum-length Steiner tree, the remaining cutting and re-assembling takes linear time, which previously took a quadratic running time.

Based on the lower bound gap result, further attempts to improve the approximation ratio should rather focus on improving the lower bound. However, this does not appear to be easy.

References

1 Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998.

2 Marcelo P.L. Benedito, Lehilton L.C. Pedrosa, and Hugo K.K. Rosado. On the inapproximability of the cable-trench problem. *Procedia Computer Science*, 195:39–48, 2021.

3 Jarosław Byrka, Fabrizio Grandoni, Thomas Rothvoss, and Laura Sanità. Steiner tree approximation via iterative randomized rounding. *Journal of the ACM*, 60(1):article 6, 2013.

4 Chandra Chekuri, Sanjeev Khanna, and Joseph Naor. A deterministic algorithm for the cost-distance problem. In *Proc. ACM-SIAM symposium on Discrete Algorithms (SODA '01), 2001*, pages 232–233. SIAM, 2001.

5 Miroslav Chlebík and Janka Chlebíková. The steiner tree problem on graphs: Inapproximability results. *Theoretical Computer Science*, 406(3):207–214, 2008.

6 Julia Chuzhoy, Anupam Gupta, Joseph Naor, and Amitabh Sinha. On the approximability of some network design problems. *ACM Transactions on Algorithms*, 4(2):article 23, 2008.

7 Siad Daboul, Stephan Held, Bento Natura, and Daniel Rotter. Global interconnect optimization. *ACM Transactions on Design Automation of Electronic Systems*, 2023. (to appear, conference paper in Proc. ICCAD '19). doi:10.1145/3587044.

8 Fabrizio Grandoni and Thomas Rothvoß. Network design via core detouring for problems without a core. In *International Colloquium on Automata, Languages, and Programming*, pages 490–502, 2010.

9 Sudipto Guha, Adam Meyerson, and Kamesh Munagala. A constant factor approximation for the single sink edge installation problem. *SIAM Journal on Computing*, 38(6):2426–2442, 2009.

10 Longkun Guo, Nianchen Zou, and Yidong Li. Approximating the shallow-light steiner tree problem when cost and delay are linearly dependent. In *Proc. International Symposium on Parallel Architectures, Algorithms and Programming*, pages 99–103, 2014.

11 Stephan Held, Dirk Müller, Daniel Rotter, Rudolf Scheifele, Vera Traub, and Jens Vygen. Global routing with timing constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(2):406–419, 2018.

12 Stephan Held and Daniel Rotter. Shallow-light steiner arborescences with vertex delays. In *Proc. International Conference on Integer Programming and Combinatorial Optimization (IPCO '13)*, pages 229–241, 2013.

13 Stephan Held and Yannik.K.D. Spitzley. Further improvements on approximating the uniform cost-distance steiner tree problem. Technical report, Research Institute for Discrete Mathematics, University of Bonn, 2022. arXiv:2211.03830.

14 Raja Jothi and Balaji Raghavachari. Improved approximation algorithms for the single-sink buy-at-bulk network design problems. *Journal of Discrete Algorithms*, 7(2):249–255, 2009.

15 Ardalan Khazraei and Stephan Held. An improved approximation algorithm for the uniform cost-distance Steiner tree problem. In *Proc. Workshop on Approximation and Online Algorithms (WAOA 2020)*, pages 189–203. Springer, 2021.

16 Samir Khuller, Balaji Raghavachari, and Neal E. Young. Balancing minimum spanning trees and shortest-path trees. *Algorithmica*, 14(4):305–321, 1995.

17 Adam Meyerson, Kamesh Munagala, and Serge Plotkin. Cost-distance: Two metric network design. *SIAM Journal on Computing*, 38(4):1648–1659, 2008.

18 Daniel Rotter. *Timing-constrained global routing with buffered Steiner trees*. PhD thesis, Universitäts-und Landesbibliothek Bonn, 2017.

19 Kunal Talwar. The single-sink buy-at-bulk lp has constant integrality gap. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 475–486, 2002.

20 Vera Traub and Rico Zenklusen. Local search for weighted tree augmentation and steiner tree. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA '22)*, pages 3253–3272, 2022.

A Proof of Lemma 4

Proof (Lemma 4). Note that Lemma 1 in [15] states only the bound (7). The bounds (5) and (6) follow immediately from their proof, which we will briefly sketch: We choose a terminal $t \in T'$ randomly with probability $p_t := \frac{w(t)}{W_{A'}}$ as the "port" vertex (only for the analysis). Then we obtain:

- The expected cost of (r,t) is $\mathbb{E}(c(r,t)) = \sum_{t \in T'} p_t c(r,t) = \frac{1}{W_{A'}} D_{T'}$.
- The (deterministic) connection cost within A' is $C_{A'}$.
- The expected effective delay cost of (r,t) is
$$\mathbb{E}(W_{A'} \cdot c(r,t)) = W_{A'} \cdot \sum_{t \in T'} p_t c(r,t) = D_{T'}.$$
- The expected delay weight served by an edge $(x,y) \in E(A')$ is
$$\frac{W_{A'_y}}{W_{A'}} \cdot (W_{A'} - W_{A'_y}) + \frac{W_{A'} - W_{A'_y}}{W_{A'}} \cdot W_{A'_y} = \frac{2W_{A'_y}(W_{A'} - W_{A'_y})}{W_{A'}} \le \frac{W_{A'}}{2},$$
where A'_y is the sub-arborescence of $A' - (x,y)$ containing y. The formula reflects the expected component of $(A' - (x,y))$ in which the port vertex is located. Summation over all edges in A' yields the following expected delay cost contribution of $E(A')$:
$$\sum_{e=(x,y) \in E(A')} \frac{2W_{A'_y}(W_{A'} - W_{A'_y})}{W_{A'}} c(e) \le \frac{W_{A'}}{2} C_{A'}.$$

The addition of these four terms gives the expected total cost of connecting A' to the root r, and provides the bound in (5). The bounds (6)-(7) follow as (5) is maximized for $W_{A'_y} = \frac{1}{2} W_{A'}$ and $W_{A'} \le 2\mu$ or A' is a (heavy) singleton. The deterministic best choice of the "port" vertex in Algorithm 1 cannot be more expensive. ◀

B Proof of Lemma 7

Proof. Note that the functions f and g differ only in the last factor. Actually, because of $\frac{1}{a} - \frac{1}{\mu} < 0$, $\frac{1}{\frac{1}{a-b} - \frac{1}{\mu}} > 0$ and $\frac{\mu}{2} - \frac{2((a-b)-c)c}{a-b} \le \frac{\mu}{2}$ we get $f(a,b,c) \ge g(a,b,c)$ for all $(a,b,c) \in X^\mu$, so it is sufficient to show $f(a,b,c) \le 0$.

We have

$$\begin{aligned} f(a,b,c) &= \frac{4ca - 4c^2 - \mu a}{2a} + \frac{(\mu - a)(a-b)}{a(\mu + b - a)} \left(\frac{\mu(a-b) - 4c(a-b-c)}{2(a-b)} \right) \\ &= \frac{(\mu - a)(4ca - 4c^2 - \mu a) + b(4ca - 4c^2 - \mu a)}{2a(\mu + b - a)} \\ &\quad + \frac{(\mu - a)\left(\mu(a-b) - 4c(a-b-c)\right)}{2a(\mu + b - a)} \\ &= \frac{b(4ca - 4c^2 - \mu a) + b(\mu - a)(4c - \mu)}{2a(\mu + b - a)} \\ &= \frac{b(4ca - 4c^2 - \mu a + 4\mu c - \mu^2 - 4ca + \mu a)}{2a(\mu + b - a)} \\ &= -\frac{b(2c - \mu)^2}{2a(\mu + b - a)} \\ &\le 0, \end{aligned}$$

where $\mu + b - a > 0$ by $(a,b,c) \in X^\mu$. ◀

C Detailed Proof of Theorem 10

Proof (Theorem 10). We will proof two claims:

1. Checking whether a branch should be cut off at the traversed vertex can be done in $\mathcal{O}(1)$ time.
2. Choosing the ports can be done in linear time.

Then, we just observe that Step 2 traverses the initial tree once, which also needs linear time.

Proof of Claim 1. We keep track of five values for each node v and its corresponding sub-arborescence $A_v := (A_0)_v$: $W_v := W_{A_v}$, the weight inside A_v, $D_v := D_{A_v}$, the minimum possible delay cost of A_v, $C_v := C_{A_v}$, the connection cost of A_v, $S_v^1 := \sum_{e=(p,q)\in E(A_v)} W_q(W_v - W_q)c(e)$ and $S_v^2 := \sum_{e=(p,q)\in E(A_v)} W_q c(e)$.

For leaves, we can compute these in constant time. For a node v with only one child x (because the other has been cut off), we can compute the values as follows: $W_v = W_x$, $D_v = D_x$, $C_v = C_x + c(v,x)$, $S_v^1 = S_x^1$, and $S_v^2 = S_x^2 + W_x c(v,x)$.

Whenever we consider a node v with children x and y, and $A_x := (A_v)_x$ and $A_y := (A_v)_y$, we can compute the values for v like so: $W_v = W_x + W_y$, $D_v = D_x + D_y$, $C_v = C_x + c(v,x) + C_y + c(v,y)$,

and

$$\begin{aligned}
S_v^1 &= \sum_{e=(p,q)\in E(A_v)} W_q(W_v - W_q)c(e)\\
&= \left(\sum_{e=(p,q)\in E(A_x)} W_q(W_v - W_q)c(e)\right) + W_x(W_v - W_x)c(v,x)\\
&\quad + \left(\sum_{e=(p,q)\in E(A_y)} W_q(W_v - W_q)c(e)\right) + W_y(W_v - W_y)c(v,y)\\
&= \left(\sum_{e=(p,q)\in E(A_x)} W_q(W_x - W_q)c(e)\right) + \left(\sum_{e=(p,q)\in E(A_x)} W_q W_y c(e)\right)\\
&\quad + W_x(W_v - W_x)c(v,x)\\
&\quad + \left(\sum_{e=(p,q)\in E(A_y)} W_q(W_y - W_q)c(e)\right) + \left(\sum_{e=(p,q)\in E(A_y)} W_q W_x c(e)\right)\\
&\quad + W_y(W_v - W_y)c(v,y)\\
&= S_x^1 + W_y S_x^2 + W_x(W_v - W_x)c(v,x) + S_y^1 + W_x S_y^2 + W_y(W_v - W_y)c(v,y)
\end{aligned}$$

and

$$S_v^2 = S_x^2 + W_x c(v,x) + S_y^2 + W_y c(v,y).$$

Proof of Claim 2. For each node $v \in V(A')$, the cost when using v as the port is

$$\text{cost}_v = c(r,v) + C_{A'} + \sum_{t\in T_{A'}} w(t)\cdot(c(r,v) + c(E(A'_{[v,t]}))).$$

So for an edge $e = (x, y) \in E(A')$ we have

$$\begin{aligned}\mathrm{cost}_x - \mathrm{cost}_y &= c(r,x) - c(r,y) + \sum_{t\in T_{A'}} w(t)(c(r,x) - c(r,y)) + \sum_{t\in T_{A'_y}} w(t)c(e)\\ &\quad - \sum_{t\in T_{A'}\setminus T_{A'_y}} w(t)c(e)\\ &= (c(r,x) - c(r,y))(1 + W_{A'}) + c(e)W_{A'_y} - c(e)(W_{A'} - W_{A'_y}).\end{aligned}$$

This allows us to compute in constant time the cost for choosing y as the port from the cost for choosing its parent x as the port. We take advantage of this property and first compute the cost for using the root of A' as the port in $\mathcal{O}(|E(A')| + |T_{A'}|)$. Then, we find the find the best "port" vertex in a top-down traversal in the claimed linear time. ◀

D Upper Bound on $h(x, y)$

We prove that for $\beta \geq 1$, $x, y \geq 0, x + y > 0$

$$\begin{aligned}h(x,y) &:= \frac{\beta x + y + \sqrt{2}\sqrt{\beta x y}}{x+y}\\ &= \beta + \frac{(1-\beta)y + \sqrt{2}\sqrt{\beta x y}}{x+y}\\ &\leq \beta + \frac{\beta}{\sqrt{\beta^2+1}+\beta-1}.\end{aligned}$$

Proof. For shorter notation, we set

$$a := \frac{\beta}{\sqrt{\beta^2+1}+\beta-1} > 0$$

and get

$$\begin{aligned}\frac{\beta}{2a} + 1 - \beta - a &= \frac{1}{2}\sqrt{\beta^2+1} - \frac{1}{2}(\beta-1) - \frac{\beta}{\sqrt{\beta^2+1}+\beta-1}\\ &= \frac{(\sqrt{\beta^2+1} - (\beta-1))(\sqrt{\beta^2+1} + (\beta-1)) - 2\beta}{2(\sqrt{\beta^2+1}+\beta-1)}\\ &= 0.\end{aligned}$$

Therefore,

$$\begin{aligned}h(x,y) &= \beta + \frac{(1-\beta)y + \sqrt{2}\sqrt{\beta x y}}{x+y} - \frac{\left(\frac{\beta}{2a} + 1 - \beta - a\right)y}{x+y}\\ &= \beta + \frac{\sqrt{2}\sqrt{\beta x y}}{x+y} - \frac{\left(\frac{\beta}{2a} - a\right)y}{x+y}\\ &= \beta + a - \frac{ax}{x+y} + \frac{\sqrt{2}\sqrt{\beta x y}}{x+y} - \frac{\frac{\beta}{2a}y}{x+y}\\ &= \beta + a - \frac{a}{x+y}\left(\sqrt{x} - \frac{\sqrt{\beta}}{\sqrt{2}a}\sqrt{y}\right)^2.\end{aligned}$$

As $a > 0$, we obtain

$$h(x,y) \leq \beta + a = \beta + \frac{\beta}{\sqrt{\beta^2+1}+\beta-1}.$$ ◀

Round and Bipartize for Vertex Cover Approximation

Danish Kashaev
Centrum Wiskunde & Informatica, Amsterdam, The Netherlands

Guido Schäfer
Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
ILLC, University of Amsterdam, The Netherlands

Abstract

The vertex cover problem is a fundamental and widely studied combinatorial optimization problem. It is known that its standard linear programming relaxation is integral for bipartite graphs and half-integral for general graphs. As a consequence, the natural rounding algorithm based on this relaxation computes an optimal solution for bipartite graphs and a 2-approximation for general graphs. This raises the question of whether one can interpolate the rounding curve of the standard linear programming relaxation in a beyond the worst-case manner, depending on how close the graph is to being bipartite. In this paper, we consider a round-and-bipartize algorithm that exploits the knowledge of an induced bipartite subgraph to attain improved approximation ratios. Equivalently, we suppose that we work with a pair $(\mathcal{G}, S)$, consisting of a graph with an odd cycle transversal.

If S is a stable set, we prove a tight approximation ratio of $1 + 1/\rho$, where $2\rho - 1$ denotes the odd girth (i.e., length of the shortest odd cycle) of the contracted graph $\tilde{\mathcal{G}} := \mathcal{G}/S$ and satisfies $\rho \in [2, \infty]$, with $\rho = \infty$ corresponding to the bipartite case. If S is an arbitrary set, we prove a tight approximation ratio of $(1 + 1/\rho)(1 - \alpha) + 2\alpha$, where $\alpha \in [0, 1]$ is a natural parameter measuring the quality of the set S. The technique used to prove tight improved approximation ratios relies on a structural analysis of the contracted graph $\tilde{\mathcal{G}}$, in combination with an understanding of the weight space where the fully half-integral solution is optimal. Tightness is shown by constructing classes of weight functions matching the obtained upper bounds. As a byproduct of the structural analysis, we also obtain improved tight bounds on the integrality gap and the fractional chromatic number of 3-colorable graphs. We also discuss algorithmic applications in order to find good odd cycle transversals, connecting to the MinUncut and Colouring problems. Finally, we show that our analysis is optimal in the following sense: the worst case bounds for ρ and α, which are $\rho = 2$ and $\alpha = 1 - 4/n$, recover the integrality gap of $2 - 2/n$ of the standard linear programming relaxation, where n is the number of vertices of the graph.

2012 ACM Subject Classification Theory of computation → Rounding techniques

Keywords and phrases Combinatorial optimization, approximation algorithms, rounding algorithms, beyond the worst-case analysis

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.20

Category APPROX

Related Version *Full Version*: https://arxiv.org/abs/2211.01699

1 Introduction

In the vertex cover problem we are given a weighted graph $\mathcal{G} = (V, E, w)$, where $w : V \mapsto \mathbb{R}_+$ is a non-negative weight function on the vertices, and the goal is to find the minimal weight subset of vertices $U \subset V$ that covers every edge of the graph, i.e.,

$$\min\{w(U) \mid U \subset V,\ |U \cap \{i, j\}| \geq 1\ \forall (i, j) \in E\}.$$

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 20; pp. 20:1–20:20
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We denote by OPT an optimal subset of vertices for this problem, and by $w(OPT)$ the total weight of that solution. The vertex cover problem is known to be NP-complete [27] and APX-complete [38]. Moreover, it was shown to be NP-hard to approximate within a factor of 7/6 in [23], a factor later improved to 1.36 in [16]. It is in fact NP-hard to approximate within a factor of $2-\varepsilon$ for any fixed $\varepsilon > 0$ if the unique games conjecture is true [28].

A natural linear programming relaxation, as well as its dual, is given by:

$$\begin{aligned} &\min \sum_{v\in V} w_v x_v &\qquad& \max \sum_{e\in E} y_e \\ &x_u + x_v \geq 1 \quad \forall (u,v)\in E && y(\delta(v)) \leq w_v \quad \forall v \in V \\ &x_v \geq 0 \quad \forall v \in V && y_e \geq 0 \quad \forall e \in E \end{aligned}$$

For a given graph $\mathcal{G}$, we denote the primal linear program by $P(\mathcal{G})$ and the dual by $D(\mathcal{G})$. The integrality gap of the standard linear relaxation $P(\mathcal{G})$ on a graph of n vertices is upper bounded by $2-2/n$, a bound which is attained on the complete graph. In fact, a more fine-grained analysis shows that it is equal to $2-2/\chi^f(\mathcal{G})$, where $\chi^f(\mathcal{G})$ is the fractional chromatic number of the graph [45]. An integrality gap of $2-\varepsilon$ is proved for a large class of linear programs in [4]. It is also known that any linear program which approximates vertex cover within a factor of $2-\varepsilon$ requires super-polynomially many inequalities [10].

An important property of $P(\mathcal{G})$ is the fact that any extreme point solution x^* is half-integral, i.e., $x^*_v \in \{0, \frac{1}{2}, 1\}$ for any vertex $v \in V$ [35]. This gives rise to a straightforward rounding algorithm by solving $P(\mathcal{G})$ and outputting all vertices whose LP variable is at least a half, i.e., $U := \{v \in V \mid x^*_v \geq \frac{1}{2}\}$. It is easy to see that this a 2-approximation, because $w(U) \leq 2w(OPT)$, see [24]. Moreover, it is known that $P(\mathcal{G})$ is integral for any bipartite graph [30]. As a consequence, the rounding algorithm returns an optimal solution if the graph is bipartite. This raises the question of whether we can interpolate the rounding curve of the standard linear program, depending on how close the graph is to being bipartite.

Set-up and algorithm

We consider the following setup. We are given a weighted non-bipartite graph $\mathcal{G} = (V, E, w)$ and an optimal solution $x^* \in \{0, \frac{1}{2}, 1\}$ to $P(\mathcal{G})$. We denote by $V_\alpha := \{v \in V \mid x^*_v = \alpha\}$ the vertices taking value α and by $\mathcal{G}_\alpha = \mathcal{G}[V_\alpha]$ the subgraph of $\mathcal{G}$ induced by the vertices V_α for any $\alpha \in \{0, \frac{1}{2}, 1\}$. By a standard preprocessing step, we may assume that we only work on the graph $\mathcal{G}_{1/2}$, since any c-approximate solution on this reduced graph can be lifted to a c-approximate solution on the original graph by adding the nodes V_1 to the solution [35]. In addition, we suppose that we have knowledge of an odd cycle transversal S of $\mathcal{G}_{1/2}$, meaning that $\mathcal{G}_{1/2} \setminus S$ is a bipartite graph. Equivalently, S intersects every odd cycle of $\mathcal{G}_{1/2}$. The question of finding a good such odd cycle transversal is also tackled later in the paper.

We consider the following simple round and bipartize algorithm, detailed in Algorithm 1. It first solves $P(\mathcal{G})$, takes the vertices assigned value one by the linear program to the solution and removes all the integral nodes from the graph to arrive at $\mathcal{G}_{1/2}$. The algorithm then takes all the vertices in the set S to the solution, removes them from the graph and solves another (now integral) linear program to get the optimal solution on the bipartite remainder. These vertices are then also added to the solution.

The question studied is the following. What is the worst-case approximation ratio of the algorithm and which weight functions are attaining it? Our motivation to study this question comes from the structural difference between the polyhedron of $P(\mathcal{G})$ for bipartite and non-bipartite graphs. In particular, we are interested in identifying parameters of the

Algorithm 1 Round and bipartize.

```
Input: Weighted graph $\mathcal{G} = (V, E, w)$, odd cycle transversal $S \subset V_{1/2}$
Output: Vertex cover $U \subset V$
Solve the linear program $P(\mathcal{G})$ to get $V_0$ , $V_{1/2}$ and $V_1$
Solve the integral linear program $P(\mathcal{G}_{1/2} \setminus S)$ to get $W \subset V_{1/2}$
return $V_1 \cup S \cup W$
```

problem that enable us to derive more fine-grained bounds determining the approximation ratio of the algorithm, and allow to interpolate the rounding curve of the standard linear program from 1 to 2, depending on how far the graph is from being bipartite. As it turns out, the *odd girth*, i.e., the length of the shortest odd cycle, is a key parameter determining tight bounds on the approximation ratio. It is also a natural parameter, since a graph is bipartite if and only if it does contain an odd cycle. The larger the odd girth, the closer the graph is to being bipartite. It is also shown in [21] that graphs with a large odd girth admit a small cardinality odd cycle transversal.

Contributions and high-level view

We first do a pre-processing step and show that we may without loss of generality focus on weighted graphs $\mathcal{G} = (V, E, w)$ where the weights come from a certain weight space Q^W. Each edge has a dual weight $y_e \geq 0$ with a total sum of $y(E) = 1$, and the weight on each node is then determined by $w_v = y(\delta(v))$. This follows from the Nemhauser-Trotter theorem, complementary slackness and an appropriate normalization.

We then do the analysis under the assumption that S is a stable set, highlighting the main ideas of the analysis and the proof techniques. We show that the approximation ratio is upper bounded by $1 + 1/\rho$, where $2\rho - 1$ denotes the odd girth of the graph $\tilde{\mathcal{G}} := \mathcal{G}/S$, where all the vertices in S are contracted into a single node. Note that the parameter range is $\rho \in [2, \infty]$, with $\rho = \infty$ naturally corresponding to the case where $\tilde{\mathcal{G}}$ is bipartite. The proof technique involves a key concept, that we call *pairwise edge-separate* feasible vertex covers. Constructing k such covers allows to bound the approximation ratio by $1 + 1/k$. The construction of ρ such covers to get the result follows from a structural understanding of the contracted graph $\tilde{\mathcal{G}}$. As a byproduct, this structural understanding also allows to get improved bounds on the integrality gap and the fractional chromatic number of 3-colorable graphs. In particular, it even manages to compute an exact formula, depending on the odd girth, for the integrality gap and the fractional chromatic number of the contracted graph $\tilde{\mathcal{G}}$.

We then construct a class of weight functions $\mathcal{W} \subset Q^W$ where this upper bound holds with equality, thus showing that this proof technique obtains tight bounds and might have additional applications. This result can then be lifted to the case where S is a general set, by introducing an additional parameter α counting the total dual sum of the weights on the edges inside S, i.e. $\alpha = y(E[S])$. This then leads to an approximation ratio interpolating the rounding curve of the standard linear program with a tight bound of $(1 + 1/\rho)(1 - \alpha) + 2\alpha$ for any values of $\rho \in [2, \infty]$ and $\alpha \in [0, 1]$.

We then discuss algorithmic applications to find good such sets S, connecting to the MinUnCut and Colouring problems. Finally, we show that our analysis is optimal in the following sense: the worst case bounds for ρ and α, which are $\rho = 2$ and $\alpha = 1 - 4/n$, recover the integrality gap of $2 - 2/n$ of the standard linear programming relaxation for a graph on n vertices.

Implications and related work

Our analysis falls into the framework of beyond the worst-case analysis [41]. In particular, note that an odd cycle transversal always exists: we may simply take $S = V_{1/2}$, which recovers the standard 2-approximation algorithm for vertex cover. Depending on how S is chosen, our algorithm can however admit significantly better approximation ratios.

Our algorithm also connects to *learning-augmented algorithms*, which have access to some prediction in their input (e.g., obtained for instance through machine learning). This prediction is assumed to come without any worst-case guarantees, and the goal is then to take advantage of it by making the algorithm perform better when this prediction is good, while still keeping robust worst-case guarantees when it is off [7, 33, 39, 31, 2, 3]. In our case, assuming a prediction on the set S, robustness is guaranteed since we are never worse than a 2-approximation. In fact, even if the predicted set is not an odd cycle transversal, one may simply greedily add vertices to it until it becomes one, while still guaranteeing a 2-approximation. Otherwise, our results provide a precise understanding of how the approximation ratio improves depending on the predicted set S. In addition, once such a set S is found, the parameters α and ρ can easily be computed to see the improved guarantee on the approximation ratio. One may thus re-run the machine learning algorithm if the parameters give a bound very close to the worst-case of 2.

The odd cycle transversal number $\operatorname{oct}(\mathcal{G})$ is defined as the minimum number of vertices needed to be removed in order to make the graph bipartite. The minimum odd cycle transversal problem has been studied in terms of fixed-parameter tractability [40, 29]. In particular, it is the first problem where the iterated compression technique has been applied [40], now a classical tool in the design of fixed-parameter tractable algorithms. The best known approximation algorithm for it has a ratio of $O(\sqrt{\log(n)})$ [1]. Another relevant concept is the *odd cycle packing number* $\operatorname{ocp}(\mathcal{G})$, defined as the maximum number of vertex-disjoint odd cycles of $\mathcal{G}$ and satisfying $\operatorname{ocp}(\mathcal{G}) \leq \operatorname{oct}(\mathcal{G})$. The related maximum stable set problem has been studied in terms of $\operatorname{ocp}(\mathcal{G})$ in [11, 5, 15, 19].

A key property implying the integrality of a polyhedron is the *total unimodularity* (TU) of the constraint matrix describing the underlying problem, meaning that all the square subdeterminants of the matrix are required to lie in $\{-1, 0, 1\}$ (see for instance [43, 44]). In general we believe it is an interesting question to study whether one may exploit the knowledge of a TU substructure in an integer program to obtain improved approximation guarantees through rounding algorithms. In our case, the knowledge of an odd cycle transveral S is equivalent to the knowledge of an induced bipartite subgraph $\mathcal{G}' = \mathcal{G} \setminus S$, for which $P(\mathcal{G}')$ is TU. We hope the techniques introduced for the pair $(\mathcal{G}, S)$ can help tackle other problems.

One technique which might also benefit from our analysis is iterative rounding, which requires solving a linear program at each iteration [32]. Having a better analysis for the case where the linear program becomes integral could potentially be used to reduce the number of iterations and allow for better guarantees, since iterative rounding can terminate at this step without losing solution quality.

Several different algorithms achieving approximation ratios of $2 - o(1)$ have been found for the weighted and unweighted versions of the vertex cover problem: [26, 22, 9, 34, 8, 25]. Another large body of work is interested in exact fixed parameter tractable algorithms for the decision version: [12, 6, 13, 14, 17, 36, 37, 46, 18].

2 Preliminaries

We define $\mathbb{R}_+$ to be the non-negative real numbers and $[k] := \{1, \dots, k\}$ to be the natural numbers up to $k \in \mathbb{N}$. For a vector $x \in \mathbb{R}^n$, we denote the sum of the coordinates on a subset by $x(A) := \sum_{i \in A} x_i$. A key property of $P(\mathcal{G})$ was introduced by Nemhauser and Trotter in [35]. It essentially allows to reduce an optimal solution $x^* \in \{0, \frac{1}{2}, 1\}^V$ to a fully half-integral solution by looking at the subgraph induced by the half-integral vertices. As before, we denote by $V_\alpha := \{v \in V \mid x^*_v = \alpha\}$ the vertices taking value α and by $\mathcal{G}_\alpha = \mathcal{G}[V_\alpha]$ the subgraph induced by the vertices V_α.

▶ **Theorem 1** (Nemhauser, Trotter [35]). *Let $x^* \in \{0, \frac{1}{2}, 1\}^V$ be an optimal extreme point solution to $P(\mathcal{G})$. Then, $w(OPT(\mathcal{G}_{1/2})) = w(OPT(\mathcal{G})) - w(V_1)$.*

▶ **Corollary 2.** *Let $x^* \in \{0, \frac{1}{2}, 1\}^V$ be an optimal solution to $P(\mathcal{G})$. If $S \subset V_{1/2}$ is a feasible vertex cover on $\mathcal{G}_{1/2}$ with approximation ratio at most ϕ, i.e., $w(S) \leq \phi\, w(OPT(\mathcal{G}_{1/2}))$, then $w(S) + w(V_1) \leq \phi\, w(OPT(\mathcal{G}))$.*

The previous corollary thus implies that we may restrict our attention to the graph $\mathcal{G}_{1/2}$, since any ϕ-approximate solution on this reduced instance can be lifted to a ϕ-approximate solution on the original graph by adding V_1 to the solution. Note that on the weighted graph $\mathcal{G}_{1/2}$, the solution $(\frac{1}{2}, \dots, \frac{1}{2})$ is optimal.

For a given set $S \subset V$, we define $\mathcal{G}' := \mathcal{G} \setminus S = (V', E')$ to be the graph obtained by removing the set S and all the incident edges to it. Hence, $E = E' \cup \delta(S) \cup E[S]$ where $\delta(S) = \{(u, v) \in E \mid u \in S, v \notin S\}$ and $E[S] := \{(u, v) \in E \mid u \in S, v \in S\}$. We also denote by $\tilde{\mathcal{G}} := \mathcal{G}/S = (\tilde{V}, \tilde{E})$ the graph obtained by contracting all the vertices in S into a single new node $v^S \in \tilde{V}$. We allow for multiple edges, but no self-loops. The only edges present in E but not in $\tilde{E}$ are thus the ones with both endpoints in S, i.e., $E[S]$.

3 Weight Space

By Corollary 2, we may assume that every weighted graph $\mathcal{G}$ we work with has the property that the fully half-integral solution $x = (\frac{1}{2}, \dots, \frac{1}{2})$ is an optimal solution to the linear program $P(\mathcal{G})$. In this section, we characterize the weight functions satisfying this assumption. The following lemma is a simple application of complementary slackness and the proof is omitted.

▶ **Lemma 3.** *Let $\mathcal{G} = (V, E)$ be a graph and let $w : V \to \mathbb{R}_+$ be a weight function. The feasible solution $(\frac{1}{2}, \dots, \frac{1}{2})$ to the linear program $P(\mathcal{G})$ is optimal if and only if there exists $y \in \mathbb{R}^E_+$ satisfying $y(\delta(v)) = w_v$ for every $v \in V$.*

Such instances have been called *edge-induced* in [15, 19], in the sense that the dual values on the edges are free parameters, and the weights on the nodes are determined once the dual values are fixed. Such instances also satisfy:

$$w(V) = \sum_{v \in V} w_v = \sum_{v \in V} y(\delta(v)) = \sum_{v \in V} \sum_{e \in E} y_e\, 1_{\{e \in \delta(v)\}} = \sum_{e \in E} y_e \sum_{v \in V} 1_{\{e \in \delta(v)\}} = 2\, y(E).$$

Observe that the approximation ratio of a feasible solution $U \subset V$ is defined as $w(U)/w(OPT(\mathcal{G}))$ and is invariant under scaling of the weights. We thus make a normalization ensuring that the optimal LP solution has objective value one, i.e., $w(V)/2 = y(E) = 1$, to get the following weight space polytope:

$$Q^W := \left\{ w \in \mathbb{R}^V_+ \mid \exists y \in [0, 1]^E \text{ such that } y(E) = 1 \text{ and } w_v = y(\delta(v)) \quad \forall v \in V \right\}.$$

We end this section by showing that this normalization of the weight space allows us to get a convenient lower bound on $w(OPT(\mathcal{G}))$.

▶ **Lemma 4.** *Let $\mathcal{G} = (V, E)$ be a graph. For any $w \in Q^W$, $w(OPT(\mathcal{G})) \geq 1$.*

Proof. Since $w \in Q^W$, we know that the fully half-integral solution is an optimal linear programming solution, showing $1 = w(V)/2 \leq w(OPT(\mathcal{G}))$, by feasibility of $OPT(\mathcal{G})$. ◀

4 Round and Bipartize

4.1 Algorithm

This section is devoted to the analysis of the approximation ratio of the algorithm and is the main contribution of the paper. We assume that we are given as input a pair $(\mathcal{G}, S)$ consisting of a weighted graph and an odd cycle transversal $S \subset V$. By Corollary 2, we may assume that the weight function satisfies $w \in Q^W$. By the previous section, there are dual edge weights $y_e \geq 0$ such that $w_v = y(\delta(v))$ for every $v \in V$ and which satisfy $\sum_{e \in E} y_e = 1$.

The algorithm is now very simple. First, take the vertices in $S \subset V$ to the cover and remove them from the graph. Then solve the integral linear program $P(\mathcal{G} \setminus S)$ and take the vertices having LP value one to the cover. The *approximation ratio*, given a weight function w, is thus defined as

$$R(w) := \frac{w(S) + w(OPT(\mathcal{G} \setminus S))}{w(OPT(\mathcal{G}))}. \tag{1}$$

For simplicity of notation, we omit the dependence on w of $OPT(\mathcal{G})$ and $OPT(\mathcal{G} \setminus S)$. As a reminder, the bipartite graph $\mathcal{G} \setminus S$ is denoted by $\mathcal{G}' = (V', E')$. The vertex contracted graph $\mathcal{G}/S$ is denoted by $\tilde{\mathcal{G}} = (\tilde{V}, \tilde{E})$ and the contracted node is denoted by v_S.

4.2 Stable Set to Bipartite

We assume in this section that S is a stable set. We will then generalize the results obtained here in a natural way to the most general setting of an arbitrary set S. We now state our main theorem of this section.

▶ **Theorem 5.** *Let $(\mathcal{G}, S)$ be the input to the rounding algorithm, with S being a stable set. For any $w \in Q^W$, the approximation ratio satisfies $R(w) \leq 1 + 1/\rho$, where $2\rho - 1$ is the odd girth of the contracted graph $\tilde{\mathcal{G}}$ and satisfies $\rho \in [2, \infty]$. Moreover, this bound is tight and is attained for a class of weight functions $\mathcal{W} \subset Q^W$.*

▶ Remark 6. We define the odd girth of a bipartite graph as being ∞.

▶ **Definition 7.** *Let $(\mathcal{G}, S)$ be a pair consisting of a graph with an odd cycle transversal S. For a feasible vertex cover $U \subset V \setminus S$ of the bipartite graph $\mathcal{G}' = \mathcal{G} \setminus S$, we define*

$$E_U := \Big\{(u, v) \in E \;\Big|\; u \in U,\ v \in U \text{ or } u \in U,\ v \in S\Big\}.$$

In words, these are the edges with either both endpoints in the cover U, or with one endpoint in U and one in S.

▶ **Definition 8.** *Let $(\mathcal{G}, S)$ be a pair consisting of a graph with an odd cycle transversal S. Feasible vertex covers $U_1, \ldots, U_k$ of the bipartite graph $\mathcal{G}' = \mathcal{G} \setminus S$ are defined to be* pairwise edge-separate *if the edge sets $\{E_{U_1}, \ldots, E_{U_k}\}$ are pairwise disjoint.*

▶ Remark 9. We will often say that covers are pairwise edge-separate for the pair $(\mathcal{G}, S)$. It is however worth emphasizing that these covers are defined on the bipartite graph $\mathcal{G}' = \mathcal{G} \setminus S$.

This definition turns out to be the key concept for us in order to prove improved bounds on the approximation ratio of the algorithm, as shown by the next lemma.

▶ **Lemma 10.** *Let $(\mathcal{G}, S)$ be the input to the rounding algorithm, with S being a stable set. If there exists k pairwise edge-separate feasible vertex covers of the bipartite graph $\mathcal{G}' = \mathcal{G} \setminus S$, then, for every $w \in Q^W$, the approximation ratio of the algorithm satisfies $R(w) \leq 1 + 1/k$.*

Proof. Let $w \in Q^W$ and let $y \in \mathbb{R}_+^E$ be the corresponding dual solution satisfying $w_v = y(\delta(v))$ and $y(E) = 1$. We denote by $\{U_1, \ldots, U_k\}$ the pairwise edge-separate covers of $\mathcal{G}' = (V', E')$. We can now write down the weights of S and every feasible cover U_i with the help of the dual variables:

$$\begin{aligned} w(S) &= \sum_{v \in S} w_v = \sum_{v \in S} y(\delta(v)) = y(\delta(S)) \\ w(U_i) &= \sum_{v \in U_i} w_v = \sum_{v \in U_i} y(\delta(v)) = y(E') + y(E_{U_i}) \qquad \forall i \in [k] \end{aligned}$$

The first equality holds because S is a stable set and thus only has edges crossing the set. The second equality holds because every U_i counts the dual value y_e of every $e \in E'$ at least once, by feasibility of the cover, and thus giving a contribution of $y(E')$. The edges in E_{U_i} then give an additional contribution of $y(E_{U_i})$.

By Lemma 4, the approximation ratio satisfies:

$$\begin{aligned} R(w) &= \frac{w(S) + w(OPT(\mathcal{G} \setminus S))}{w(OPT(\mathcal{G}))} \leq w(S) + w(OPT(\mathcal{G}')) \leq w(S) + \min_{i \in [k]} w(U_i) \\ &= y(\delta(S)) + y(E') + \min_{i \in [k]} y(E_{U_i}) = 1 + \min_{i \in [k]} y(E_{U_i}) \leq 1 + \frac{1}{k}. \end{aligned}$$

The last equality follows from the fact that $E = E' \cup \delta(S)$ and $y(E) = 1$. The last inequality follows from the fact that the edge sets $\{E_{U_i}\}_{i \in [k]}$ are pairwise disjoint and have a dual sum of at most one, since the total sum of the edges of the graph is $y(E) = 1$. This minimum can thus be upper bounded by $1/k$. ◀

In order to prove the upper bound in Theorem 5, we thus need to construct ρ pairwise edge-separate covers of $\mathcal{G}' = \mathcal{G} \setminus S$. The key for being able to do that is to analyze the structure of the contracted graph $\tilde{\mathcal{G}} = \mathcal{G}/S$, where S is contracted into a single node v_S.

▶ **Lemma 11.** *Let $(\mathcal{G}, S)$ be a graph with an odd cycle transversal S. If the contracted graph $\tilde{\mathcal{G}}$ contains an odd cycle, then there exists ρ edge-separate feasible covers for the pair $(\tilde{\mathcal{G}}, v_S)$, where $2\rho - 1$ is the odd girth of $\tilde{\mathcal{G}}$.*

Proof. We now dive deeper into the structure of the bipartite graph $\mathcal{G} \setminus S = \tilde{\mathcal{G}} \setminus v_S$. By assumption, this graph admits a bipartition $A \cup B$ of the vertices. Let us assume that it has k connected components $A_1 \cup B_1, \ldots, A_k \cup B_k$, all of which are bipartite as well, where $A = \bigcup_i A_i$ and $B = \bigcup_i B_i$. We now fix an arbitrary such component $A_j \cup B_j$.

- If v_S has an incident edge to both A_j and B_j, then this component contains (if including v_S) an odd cycle of $\tilde{\mathcal{G}}$. This holds since any path between a node in A_j and a node in B_j has odd length.
- If v_S has incident edges with only one side, we assume without loss of generality that this side is A_j. One could simply switch both sides in the other case while still keeping a valid bipartition of the graph $\tilde{\mathcal{G}} \setminus v_S$.

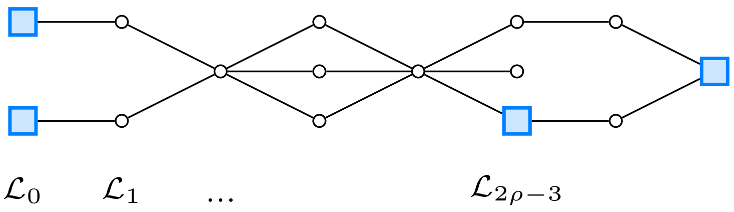


Figure 1 The layers of a bipartite graph $\tilde{\mathcal{G}} \setminus v_S = (A \cup B, E')$ with $\rho = 4$. The blue square vertices correspond to $N(v_S)$, where the two left ones are $\mathcal{L}_0 = N_A(v_S)$ and the two right ones are $N_B(v_S)$.

- If v_S does not have incident edges with either of the two sides, then $A_j \cup B_j$ is a connected component of $\tilde{\mathcal{G}}$. We call such components *dummy* components and denote by $A_d \cup B_d$ the bipartite graph formed by taking the union of all the dummy components.

We denote $N_A(v_S) = N(v_S) \cap A$ and $N_B(v_S) = N(v_S) \cap B$. We now split the graph into layers, where each layer corresponds to the nodes at the same shortest path distance from $N_A(v_S)$. More precisely, we define

$$\mathcal{L}_i := \Big\{ v \in A \cup B \mid d(N_A(v_S), v) = i \Big\} \quad \text{for } i \in \{0, \dots, q\} \tag{2}$$

where $d(N_A(v_S), v)$ represents the unweighted shortest path distance between v and a vertex in $N_A(v_S)$. The parameter q is defined to be the maximal finite distance from $N_A(v_S)$ in the graph $\tilde{\mathcal{G}}$. An important observation is the fact that these layers are alternatingly included in one side of the bipartition, see Figure 1 for an illustration of the construction.

If the graph $\tilde{\mathcal{G}}$ is not connected, note that $d(N_A(v_S), v) = \infty$ for the vertices v lying in dummy components. In order to add the dummy components to the layers and keep alternation between the two sides of the bipartition, we define the last two layers to either be $\{\mathcal{L}_{q+1} := A_d,\ \mathcal{L}_{q+2} := B_d\}$ or $\{\mathcal{L}_{q+1} := B_d,\ \mathcal{L}_{q+2} := A_d\}$, depending on which side of the bipartition the last connected layer $\mathcal{L}_q$ lies. We now have that $\mathcal{L}_i \subset A$ if i is even, and $\mathcal{L}_i \subset B$ if i is odd. In fact,

$$A = \bigcup_{i=0}^{\lfloor l/2 \rfloor} \mathcal{L}_{2i} \quad \text{and} \quad B = \bigcup_{i=1}^{\lceil l/2 \rceil} \mathcal{L}_{2i-1},$$

where the parameter $l \in \mathbb{N}$ represents the index of the last layer: if $\tilde{\mathcal{G}}$ is connected, then $l = q$, otherwise $l = q + 2$. Notice also that $\mathcal{L}_0 = N_A(v_S)$. However, $N_B(v_S)$ may now have several different vertices in different layers, see Figure 1.

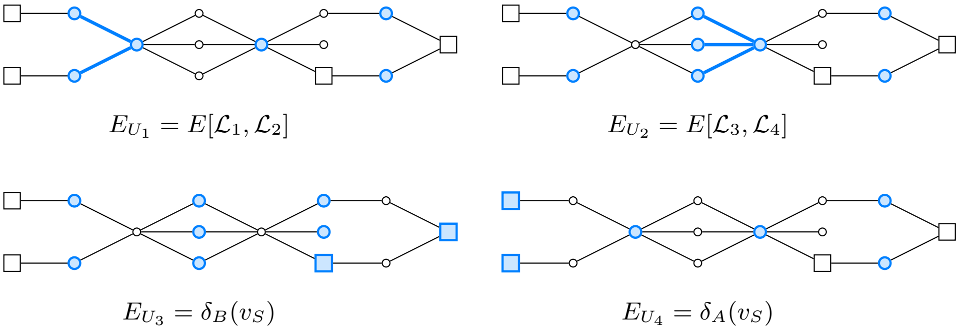
Let $C \subset V$ be an arbitrary odd cycle of $\tilde{\mathcal{G}}$. Notice that this cycle contains v_S, a vertex from $N_A(v_S)$ and a vertex from $N_B(v_S)$, since $\tilde{\mathcal{G}} \setminus v_S$ is bipartite and therefore does not contain an odd cycle. Any odd cycle C in $\tilde{\mathcal{G}}$ thus corresponds to an odd path between a vertex in $N_A(v_S) = \mathcal{L}_0$ and a vertex in $N_B(v_S)$. By the assumption that the shortest odd cycle length of $\tilde{\mathcal{G}}$ is $2\rho - 1$, the first layer having a non-empty intersection with $N_B(v_S)$ is $\mathcal{L}_{2\rho-3}$. A shortest odd cycle of length $2\rho - 1$ therefore corresponds to an odd path of length $2\rho - 3$ between $\mathcal{L}_0$ and a vertex in $\mathcal{L}_{2\rho-3} \cap N_B(v_S)$, see Figure 1 for an illustration. We now define edges connecting two consecutive layers $\mathcal{L}_i$ and $\mathcal{L}_{i+1}$ as follows:

$$E[\mathcal{L}_i, \mathcal{L}_{i+1}] := \{(u, v) \in E' \mid u \in \mathcal{L}_i, v \in \mathcal{L}_{i+1}\} \qquad \forall i \in \{0, \dots, l-1\}.$$

We also denote by

$$\delta_A(v_S) = \{(v_S, u) \in \tilde{E} \mid u \in A\}, \qquad \delta_B(v_S) = \{(v_S, u) \in \tilde{E} \mid u \in B\}$$

the incident edges to v_S respectively connecting to A and B.



$E_{U_1} = E[\mathcal{L}_1, \mathcal{L}_2]$

$E_{U_2} = E[\mathcal{L}_3, \mathcal{L}_4]$

$E_{U_3} = \delta_B(v_S)$

$E_{U_4} = \delta_A(v_S)$

Figure 2 The ρ feasible covers of $\mathcal{G}'$ constructed in the proof of Lemma 11.

We are now ready to construct our desired ρ pairwise edge-separate covers of $\tilde{\mathcal{G}} \setminus v_S$, that we denote by $U_1, \dots, U_\rho$ and illustrated in Figure 2. Firstly, notice that taking one side of the bipartition is a feasible vertex cover. We thus define $U_\rho = A$ and $U_{\rho-1} = B$. Observe that we then have $E_{U_\rho} = \delta_A(v_S)$ and $E_{U_{\rho-1}} = \delta_B(v_S)$. We now construct $\rho - 2$ additional covers with the help of the layers. If $\rho \neq 2$, fix a $j \in [\rho - 2]$, and start the cover U_j by taking the two consecutive layers $\mathcal{L}_{2j-1}$ and $\mathcal{L}_{2j}$. Complete this cover by taking remaining layers alternatingly (hence always skipping one) until covering every edge of the graph. Notice that this cover has an empty intersection with $N(v_S)$. We then have that

$$E_{U_\rho} = \delta_A(v_S), \qquad E_{U_{\rho-1}} = \delta_B(v_S), \qquad E_{U_j} = E[\mathcal{L}_{2j-1}, \mathcal{L}_{2j}] \quad \forall j \in [\rho - 2],$$

which are all pairwise disjoint edge sets, finishing the proof. ◀

We now have all the tools to prove the upper bound of Theorem 5.

Proof. Asume first that $\rho < \infty$, meaning that $\tilde{\mathcal{G}}$ contains an odd cycle. By Lemma 11, there exists ρ pairwise edge-separate covers for the pair $(\tilde{\mathcal{G}}, v_S)$. These covers are then still edge-separate for the pair $(\mathcal{G}, S)$, since the bipartite graph is the same in both cases, i.e. $\mathcal{G}' = \tilde{\mathcal{G}} \setminus v_S = \mathcal{G} \setminus S$. This finishes the proof by Lemma 10.

If $\rho = \infty$, then $\tilde{\mathcal{G}}$ is bipartite, with a bipartition $\tilde{A} \cup \tilde{B}$. Assume without loss of generality that $v^S \in \tilde{A}$. Note that $\tilde{E} = E' \cup \delta(v^S)$ and thus $1 = y(\tilde{E}) = y(E') + y(\delta(v^S))$. Any feasible cover of $\mathcal{G}' = \mathcal{G} \setminus S$ needs to count the dual value of every edge in E' at least once. Taking the cover $\tilde{A} \setminus v^S$ counts every edge in E' exactly once, showing that $w(OPT(\mathcal{G} \setminus S)) = y(E')$. Hence, $R(w) \leq w(S) + w(OPT(\mathcal{G} \setminus S)) = y(\delta(v_S)) + y(E') = 1$. ◀

We now show that this bound is tight and is attained for a class of weight functions $w \in \mathcal{W}$ for any such graph $\mathcal{G}$ and stable set S. For the case where $\rho = \infty$, it is clear that the approximation ratio always satisfies $R(w) \geq 1$, showing that the bound in Theorem 5 is tight for any $w \in Q^W$. We thus assume that $\rho < \infty$. Let $\mathcal{C}$ be all the shortest odd cycles (of length $2\rho - 1$) of the graph $\tilde{\mathcal{G}}$, each of which is containing v_S. For every such cycle $C \in \mathcal{C}$, we define the following dual function on the edges $y^C : \tilde{E} \to \mathbb{R}_+$: set both dual edges incident to v_S to $1/\rho$ and then alternatingly set the dual edges to 0 and $1/\rho$ along the odd cycle. For any edge outside of C, set its dual value to 0. Such a solution clearly satisfies $y^C(\tilde{E}) = 1$. We now take the convex hull of all these functions:

$$\mathcal{Y} := \Big\{ y : \tilde{E} \to \mathbb{R}_+ \mid y = \sum_{C \in \mathcal{C}} \lambda^C y^C, \quad \sum_{C \in \mathcal{C}} \lambda^C = 1, \quad \lambda^C \geq 0 \quad \forall C \in \mathcal{C} \Big\}.$$

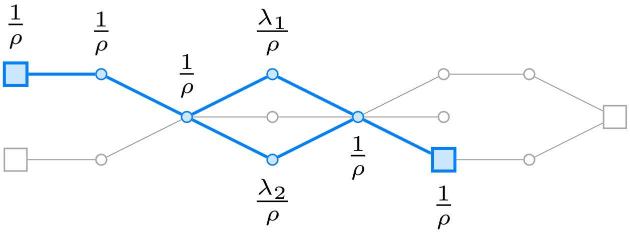


Figure 3 An example of a weight function $w \in \mathcal{W}$ obtained by a convex combination of two basic weight functions of shortest odd cycles.

Because of the one-to-one correspondence between the edge sets $\tilde{E}$ and E, due to the fact that S is a stable set, we can naturally define a weight function on the original vertex set once we fix a $y \in \mathcal{Y}$ by setting $w_v := y(\delta(v))$ for every $v \in V$. We define the space of all such weight functions as $\mathcal{W} := \{w : V \to \mathbb{R}_+ \mid w_v = y(\delta(v)) \ \forall v \in V, \ y \in \mathcal{Y}\}$.

▶ **Theorem 12.** *For any weight function $w \in \mathcal{W}$, the approximation ratio satisfies $R(w) = 1 + 1/\rho$, where $2\rho - 1$ is the odd girth of $\tilde{\mathcal{G}}$ and satisfies $\rho \in [2, \infty)$.*

Proof. Let $\mathcal{C}$ be the set of all the shortest odd cycles (of length $2\rho - 1$) of the graph $\tilde{\mathcal{G}}$ and let $w \in \mathcal{W}$ with the corresponding $y = \sum_{C \in \mathcal{C}} \lambda^C y^C$. Notice that, for any subset of vertices $U \subset V'$ of the bipartite graph $\mathcal{G}'$, we can count its weight as

$$w(U) = \sum_{v \in U} w_v = \sum_{v \in U} y(\delta(v)) = \sum_{v \in U} \sum_{C \in \mathcal{C}} \lambda^C y^C(\delta(v))$$
$$= \sum_{v \in U} \sum_{C \in \mathcal{C}} \frac{\lambda^C}{\rho} 1_{\{v \in C\}} = \frac{1}{\rho} \sum_{C \in \mathcal{C}} \lambda^C \sum_{v \in U} 1_{\{v \in C\}} = \frac{1}{\rho} \sum_{C \in \mathcal{C}} \lambda^C |U \cap C|. \tag{3}$$

The end of the proof now heavily uses the decomposition of $\tilde{\mathcal{G}}$ into layers described in (2). Notice that every odd cycle $C \in \mathcal{C}$ intersects each layer $\mathcal{L}_i$ for $i \in \{0, \dots, 2\rho - 3\}$ exactly once. Therefore, by (3), $w(\mathcal{L}_i) = 1/\rho$ for every $i \in \{0, \dots, 2\rho - 3\}$. We now claim that

$$w(OPT(\mathcal{G})) = 1, \quad w(OPT(\mathcal{G}')) = \frac{\rho - 1}{\rho} \quad \text{and} \quad w(S) = \frac{2}{\rho}.$$

The fact that $w(OPT(\mathcal{G})) \geq 1$ follows from Lemma 4. For the reverse inequality, notice that it is possible to take a feasible cover by taking exactly ρ layers in addition to the zero weight vertices, for instance $\mathcal{L}_0 \cup \mathcal{L}_2 \cup \mathcal{L}_3 \cup \mathcal{L}_5 \cdots \cup \mathcal{L}_{2\rho-3}$, showing $w(OPT(\mathcal{G})) \leq 1$.

Observe now that $w(OPT(\mathcal{G}')) = w(OPT(\mathcal{G} \setminus S)) = w(OPT(\tilde{\mathcal{G}} \setminus v_S))$. After removal of v_S, every cycle $C \in \mathcal{C}$ becomes a path of length $2\rho - 3$ (and thus consisting of $2\rho - 2$ vertices), with one vertex in each layer $\mathcal{L}_i$ for $i \in \{0, \dots, 2\rho - 3\}$. By feasibility, $OPT(\mathcal{G}')$ has to contain at least $\rho - 1$ vertices for every such path. Using (3), we infer $w(OPT(\mathcal{G}')) \geq (\rho - 1)/\rho$. For the reverse inequality, taking $\rho - 1$ layers alternatively, such as $\mathcal{L}_0 \cup \mathcal{L}_2 \cup \mathcal{L}_4 \cdots \cup \mathcal{L}_{2\rho-4}$, as well as the zero weight vertices, builds a feasible cover of weight exactly $(\rho - 1)/\rho$.

Finally, notice that $w(S) = w(v_S) = 2/\rho$ because every $C \in \mathcal{C}$ contains v_S. By combining the three equalities, we get the desired result $R(w) = 1 + 1/\rho$. ◀

Integrality gap and fractional chromatic number

Observe that the graphs we work with in this section have chromatic number $\chi(\mathcal{G}) = 3$. We focus now on the integrality gap and show again that the odd girth plays a key role. A result that we use here is given by Singh in [45], which relates the integrality gap with the fractional chromatic number of a graph: $IG(\mathcal{G}) = 2 - 2/\chi^f(\mathcal{G})$.

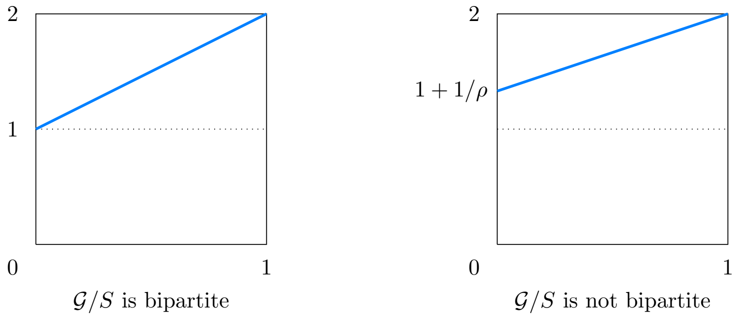


Figure 4 The plot of the approximation ratio $R(w)$ with respect to the parameter $\alpha \in [0,1]$.

▶ **Theorem 13.** *Let $\mathcal{G}$ be a 3-colorable graph with color classes $V = V_1 \cup V_2 \cup V_3$.*

$$\chi^f(\mathcal{G}) \leq 2 + \min_{i\in\{1,2,3\}} \frac{1}{\rho_i - 1}, \quad IG(\mathcal{G}) \leq 1 + \min_{i\in\{1,2,3\}} \frac{1}{2\rho_i - 1}$$

where $2\rho_i - 1$ is the odd girth of the contracted graph $\mathcal{G}/V_i$ for each $i \in \{1,2,3\}$. Moreover, equality holds if one color class only contains one vertex.

In particular, we manage to compute exact formulas of

$$\chi^f(\tilde{\mathcal{G}}) = 2 + \frac{1}{\rho - 1}, \quad IG(\tilde{\mathcal{G}}) = 1 + \frac{1}{2\rho - 1}$$

for the contracted graph $\tilde{\mathcal{G}}$. These statements generalize a result shown for the (odd) cycle graph in [4, 42]. The proofs are left to the appendix due to space constraints and heavily use the structural decomposition of the contracted graph $\tilde{\mathcal{G}}$ into the layers (2), implying that it may have further applications.

4.3 Arbitrary Set to Bipartite

We now consider the setting where S is now an arbitrary set. Our guarantee on the approximation ratio will now also depend on the total sum of the dual variables on the edges inside of the set S. We denote this sum by $\alpha := y(E[S]) \in [0,1]$.

▶ **Theorem 14.** *For any $w \in Q^W$, the approximation ratio satisfies*

$$R(w) \leq \left(1 + \frac{1}{\rho}\right)(1-\alpha) + 2\alpha \qquad \text{with } \alpha \in [0,1] \text{ and } \rho \in [2,\infty]$$

where $2\rho - 1$ denotes the odd girth of the contracted graph $\tilde{\mathcal{G}}$. Moreover, these bounds are tight and are attained for any $\alpha \in [0,1]$ and any $\rho \in [2,\infty]$.

Proof. We only prove here the upper bound with $\rho < \infty$ and leave the remaining statements to the appendix. The proof essentially follows the same arguments as the one of Lemma 10 with the α parameter incorporated, and we thus only highlight the main differences. We decompose the weight of the set S with respect to the dual variables. The edges in $E[S]$ are counted twice, whereas the edges in $\delta(S)$ are counted once:

$$w(S) = 2\alpha + y(\delta(S)). \tag{4}$$

Consider the contracted graph $\tilde{\mathcal{G}} = \mathcal{G}/S = (\tilde{V}, \tilde{E})$ and denote by v^S the contracted node. The edge set of this graph is now $\tilde{E} = \delta(S) \cup E'$, since the edges in $E[S]$ have been collapsed. By Lemma 11, we can construct ρ edge-separate covers $U_1, \dots, U_\rho \subset \tilde{V} \setminus v^S$ for the pair $(\tilde{\mathcal{G}}, v_S)$. These covers are still edge-separate for the pair $(\mathcal{G}, S)$, implying

$$w(OPT(\mathcal{G} \setminus S)) \leq \min_{i \in [\rho]} w(U_i) = y(E') + \min_{i \in [\rho]} y(E_{U_i}) \leq y(E') + \frac{1-\alpha}{\rho}. \tag{5}$$

The first inequality holds since every U_i is a feasible cover of $\mathcal{G} \setminus S$. The second equality holds by counting the weight of a cover U_i in terms of the dual edges. The last inequality holds because the edge sets $\{E_{U_i}\}_{i \in [\rho]}$ are pairwise disjoint, and their total dual sum is at most $1 - \alpha$. Combining Lemma 4, (4) and (5),

$$R(w) \leq 2\alpha + y(\delta(S)) + y(E') + \frac{1-\alpha}{\rho} = 1 + \alpha + \frac{1-\alpha}{\rho} = \left(1 + \frac{1}{\rho}\right)(1-\alpha) + 2\alpha. \qquad \blacktriangleleft$$

5 Algorithmic applications

We focus in this section on efficient ways to find odd cycle transversals with a low value for the α parameter. In fact, once such a set S is found, there can also be freedom in the choice of the dual solution in order to optimize the α parameter. This motivates the following definition.

▶ **Definition 15.** *Let $(\mathcal{G}, S, y, w)$ be a graph with an odd cycle transversal $S \subset V$, weights $w \in Q^W$ and a dual solution $y \in \mathbb{R}_+^E$. A tuple $(\mathcal{G}', S', y', w')$ is* approximation preserving *if*

$$w(S) + w(OPT(\mathcal{G} \setminus S)) \leq w'(S') + w'(OPT(\mathcal{G}' \setminus S')).$$

Moreover, we say that $\alpha \in [0,1]$ is valid *for the pair $(\mathcal{G}, S)$ if there exists an approximation preserving $(\mathcal{G}', S', y', w')$ such that $\alpha = y'(E[S'])$.*

Finding a valid $\alpha \in [0,1]$ would directly allow us to use it in the bound of Theorem 14, where the ρ parameter would correspond to the one of the approximation preserving graph. We present here an application if a coloring of a graph can be found efficiently.

▶ **Theorem 16.** *Let $\mathcal{G} = (V, E)$ be a graph with weights $w \in Q^W$ that can be k-colored in polynomial time for $k \geq 4$. There exists an efficiently findable set $S \subset V$ bipartizing the graph and a valid α such that $\alpha \leq 1 - 4/k$, leading to an approximation ratio of*

$$R(w) \leq 2 - \frac{4}{k}\left(1 - \frac{1}{\rho}\right)$$

Proof. Let us denote by $V_1, \dots, V_k$ the k independent sets defining the color classes of the graph $\mathcal{G}$. We assume without loss of generality that they are ordered by weight $w(V_1) \leq w(V_2) \cdots \leq w(V_k)$. Since $w(V) = 2$, the two color classes with the largest weights satisfy $w(V_{k-1}) + w(V_k) \geq 4/k$. We define the bipartizing set to be the remaining color classes: $S := V_1 \cup \cdots \cup V_{k-2}$. We denote by $y \in \mathbb{R}_+^E$ the dual solution satisfying complementary slackness and $y(E) = 1$.

We now define an approximation preserving $(\mathcal{G}', S', y', w')$ in the following way. Let $\mathcal{G}' = K_k$ be the complete graph on k vertices, denoted by $\{v_1, \dots, v_k\}$. The weights are defined to be

$$w'(v_i) := w(V_i) \quad \text{and} \quad y'(v_i, v_j) := y(E[V_i, V_j])$$

for every $i, j \in [k]$. These clearly satisfy the complementary slackness condition $y'(\delta(v_i)) = w'(v_i)$ for every $i \in [k]$. The bipartizing set is defined to be $S' := \{v_1, \dots, v_{k-2}\}$. This tuple is approximation preserving since $w(S) = w'(S')$ and $w(OPT(\mathcal{G} \setminus S)) \le w'(OPT(\mathcal{G}' \setminus S'))$. In order to prove the theorem, we still need to tweak the dual solution y' to ensure $\alpha := y'(E[S']) \le 1 - 4/k$. Observe that $w'(v_{k-1}) + w'(v_k) \ge 4/k$.

(i) If $y'(v_{k-1}, v_k) = 0$, then the result follows since in that case $y'(\delta(S')) \ge 4/k$ and thus $y'(E[S']) \le 1 - 4/k$.
(ii) If $y'(E[S']) = 0$, then the result trivially follows as well.

Suppose thus that $y'(E[S']) > 0$ and $y'(v_{k-1}, v_k) > 0$. Pick an arbitrary edge $(v_i, v_j) \in E[S']$ satisfying $y'(v_i, v_j) > 0$ and consider the 4-cycle (v_i, v_j, v_{k-1}, v_k). Notice that alternatively increasing and decreasing the dual values on the edges of this cycle by a small amount $\epsilon > 0$ gives another feasible dual solution satisfying the complementary slackness condition. More formally, we set $\epsilon := \min\{y'(v_i, v_j), y'(v_{k-1}, v_k)\}$, decrease $y'(v_i, v_j)$ and $y'(v_{k-1}, v_k)$ by ϵ, while increasing $y'(v_j, v_{k-1})$ and $y'(v_k, v_i)$ by the same amount. Observe that this leads to either (v_i, v_j) or (v_{k-1}, v_k) dropping to dual value zero. We can repeat this procedure until either $y'(E[S']) = 0$ or $y'(v_{k-1}, v_k) = 0$, finishing the proof of the theorem. ◀

We now claim that this result is optimal in the following sense. Consider an n-vertex graph. It is known that the integrality gap of the standard linear programming relaxation for vertex cover is upper bounded by $2 - 2/n$, a bound which is attained on the complete graph. This implies that any approximation algorithm lower bounding $w(OPT)$ by comparing it to the optimal LP solution, as we do in Lemma 4, cannot do better than $2 - 2/n$ in the worst case. Setting $\rho = 2$ in Theorem 16, which corresponds to the worst case since $\rho \in [2, \infty]$, recovers this bound and a result of Hochbaum in [25].

We now make another connection with the MinUncut problem, which is defined on a graph $\mathcal{G} = (V, E)$ with weights y_e for every edge $e \in E$. The goal of the problem is to find a cut of the graph which minimizes the total weight of uncut edges. This problem is NP-hard and admits a $O(\sqrt{\log(n)})$ approximation [1]. We call a MinUncut instance *light* if its optimal solution is bounded above by $y(E)/\Omega(\log(n))$.

▶ **Theorem 17.** *For any light MinUncut instance, combining the $O(\sqrt{\log(n)})$ approximation in [1] with Algorithm 1 outputs a vertex cover with approximation ratio at most $R(w) \le 1 + 1/\rho + o(1)$, where $2\rho - 1$ is the odd girth of the contracted graph $\tilde{\mathcal{G}}$ and satisfies $\rho \in [2, \infty]$.*

Proof. The key observation is that any feasible solution to the MinUncut problem of value α gives an odd cycle transversal S satisfying $y(E[S]) = \alpha$. Indeed, let $(C, V \setminus C)$ be a feasible solution, where the total weight of uncut edges is α. Observe that removing all the uncut edges makes C and $V \setminus C$ become stable sets, implying that the remaining graph is bipartite. We then define $S \subset V$ to be all the nodes incident to the uncut edges. Since removing all the nodes in S also removes all the uncut edges, the remaining graph is bipartite. Moreover, $y(E[S]) = \alpha$.

By the lightness assumption and the weight space normalization, the optimal solution α^* satisfies $\alpha^* \le y(E)/\Omega(\log(n)) = 1/\Omega(\log(n))$. Running the $O(\sqrt{\log(n)})$ approximation algorithm then outputs a solution with value $\alpha \le O(\sqrt{\log(n)})\ \alpha^* \le 1/\Omega(\sqrt{\log(n)}) = o(1)$. This therefore leads to an approximation guarantee of

$$R(w) = \left(1 + \frac{1}{\rho}\right)(1 - \alpha) + 2\alpha = 1 + \frac{1}{\rho} + \alpha\left(1 - \frac{1}{\rho}\right) \le 1 + \frac{1}{\rho} + o(1).$$ ◀

References

1 Amit Agarwal, Moses Charikar, Konstantin Makarychev, and Yury Makarychev. $o(\sqrt{\log(n)})$ approximation algorithms for min uncut, min 2cnf deletion, and directed cut problems. In *Symposium on the Theory of Computing*, 2005.

2 Antonios Antoniadis, Christian Coester, Marek Elias, Adam Polak, and Bertrand Simon. Online metric algorithms with untrusted predictions. In *International Conference on Machine Learning*, pages 345–355. PMLR, 2020.

3 Antonios Antoniadis, Themis Gouleakis, Pieter Kleer, and Pavel Kolev. Secretary and online matching problems with machine learned advice. *Advances in Neural Information Processing Systems*, 33:7933–7944, 2020.

4 Sanjeev Arora, Béla Bollobás, and László Lovász. Proving integrality gaps without knowing the linear program. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 313–322. IEEE, 2002.

5 Stephan Artmann, Robert Weismantel, and Rico Zenklusen. A strongly polynomial algorithm for bimodular integer linear programming. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1206–1219, 2017.

6 R Balasubramanian, Michael R Fellows, and Venkatesh Raman. An improved fixed-parameter algorithm for vertex cover. *Information Processing Letters*, 65(3):163–168, 1998.

7 Étienne Bamas, Andreas Maggiori, and Ola Svensson. The primal-dual method for learning augmented algorithms. *Advances in Neural Information Processing Systems*, 33:20083–20094, 2020.

8 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.

9 Reuven Bar-Yehuda and Shimon Even. A local-ratio theorm for approximating the weighted vertex cover problem. Technical report, Computer Science Department, Technion, 1983.

10 Abbas Bazzi, Samuel Fiorini, Sebastian Pokutta, and Ola Svensson. No small linear program approximates vertex cover within a factor 2- ε. *Mathematics of Operations Research*, 44(1):147–172, 2019.

11 Adrian Bock, Yuri Faenza, Carsten Moldenhauer, and Andres Jacinto Ruiz-Vargas. Solving the stable set problem in terms of the odd cycle packing number. In *34th International Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS 2014)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.

12 Jonathan F Buss and Judy Goldsmith. Nondeterminism within p. *SIAM Journal on Computing*, 22(3):560–572, 1993.

13 Jianer Chen, Iyad A Kanj, and Weijia Jia. Vertex cover: further observations and further improvements. *Journal of Algorithms*, 41(2):280–301, 2001.

14 Jianer Chen, Lihua Liu, and Weijia Jia. Improvement on vertex cover for low-degree graphs. *Networks: An International Journal*, 35(4):253–259, 2000.

15 Michele Conforti, Samuel Fiorini, Tony Huynh, Gwenaël Joret, and Stefan Weltge. The stable set problem in graphs with bounded genus and bounded odd cycle packing number. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2896–2915. SIAM, 2020.

16 Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of mathematics*, pages 439–485, 2005.

17 Rodney G Downey and Michael R Fellows. Fixed-parameter tractability and completeness. In *Complexity Theory: Current Research*, pages 191–225, 1992.

18 Rodney G Downey and Michael Ralph Fellows. *Parameterized complexity*. Springer Science & Business Media, 2012.

19 Samuel Fiorini, Gwenael Joret, Stefan Weltge, and Yelena Yuditsky. Integer programs with bounded subdeterminants and two nonzeros per row. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 13–24. IEEE, 2022.

20 Wayne Goddard and Honghai Xu. Fractional, circular, and defective coloring of series-parallel graphs. *Journal of Graph Theory*, 81(2):146–153, 2016.

21 Ervin Györi, Alexandr V Kostochka, and Tomasz Łuczak. Graphs without short odd cycles are nearly bipartite. *Discrete Mathematics*, 163(1-3):279–284, 1997.

22 Eran Halperin. Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. *SIAM Journal on Computing*, 31(5):1608–1623, 2002.

23 Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.

24 Dorit S Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on computing*, 11(3):555–556, 1982.

25 Dorit S Hochbaum. Efficient bounds for the stable set, vertex cover and set packing problems. *Discrete Applied Mathematics*, 6(3):243–254, 1983.

26 George Karakostas. A better approximation ratio for the vertex cover problem. In *International Colloquium on Automata, Languages, and Programming*, pages 1043–1050. Springer, 2005.

27 Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

28 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2- ε. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

29 Stefan Kratsch and Magnus Wahlström. Compression via matroids: a randomized polynomial kernel for odd cycle transversal. *ACM Transactions on Algorithms (TALG)*, 10(4):1–15, 2014.

30 Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

31 Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1859–1877. SIAM, 2020.

32 Lap Chi Lau, Ramamoorthi Ravi, and Mohit Singh. *Iterative methods in combinatorial optimization*, volume 46. Cambridge University Press, 2011.

33 Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *Journal of the ACM (JACM)*, 68(4):1–25, 2021.

34 Burkhard Monien and Ewald Speckenmeyer. Ramsey numbers and an approximation algorithm for the vertex cover problem. *Acta Informatica*, 22(1):115–123, 1985.

35 George L Nemhauser and Leslie Earl Trotter. Vertex packings: structural properties and algorithms. *Mathematical Programming*, 8(1):232–248, 1975.

36 Rolf Niedermeier and Peter Rossmanith. Upper bounds for vertex cover further improved. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 561–570. Springer, 1999.

37 Rolf Niedermeier and Peter Rossmanith. On efficient fixed-parameter algorithms for weighted vertex cover. *Journal of Algorithms*, 47(2):63–77, 2003.

38 Christos Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 229–234, 1988.

39 Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ml predictions. *Advances in Neural Information Processing Systems*, 31, 2018.

40 Bruce Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004.

41 Tim Roughgarden. *Beyond the worst-case analysis of algorithms*. Cambridge University Press, 2021.

42 Edward R Scheinerman and Daniel H Ullman. *Fractional graph theory: a rational approach to the theory of graphs*. Courier Corporation, 2011.

43 Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

44 Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

45 Mohit Singh. Integrality gap of the vertex cover linear programming relaxation. *Operations Research Letters*, 47(4):288–290, 2019.

46 Ulrike Stege and Michael Ralph Fellows. An improved fixed parameter tractable algorithm for vertex cover. *Technical report/Departement Informatik, ETH Zürich*, 318, 1999.

A Integrality Gap and Fractional Chromatic Number

We focus in this section on proving tight bounds for the integrality gap of 3-colorable graphs. A key result that we use in this section is given by Singh in [45], which relates the integrality gap with the fractional chromatic number of a graph. The latter is denoted as $\chi^f(\mathcal{G})$ and is defined as the optimal solution of the following primal-dual linear programming pair. We denote by $\mathcal{I} \subset 2^V$ the set of all independent sets of the graph $\mathcal{G}$. Solving these linear programs is however NP-hard because of the possible exponential number of independent sets.

$$\begin{aligned} \min \sum_{I\in\mathcal{I}} y_I & \\ \sum_{I\in\mathcal{I}, v\in I} y_I \geq 1 & \quad \forall v \in V \\ y_I \geq 0 & \quad \forall I \in \mathcal{I} \end{aligned} \qquad\qquad \begin{aligned} \max \sum_{v\in V} z_v & \\ \sum_{v\in I} z_v \leq 1 & \quad \forall I \in \mathcal{I} \\ z_v \geq 0 & \quad \forall v \in V \end{aligned}$$

Note that $\chi^f(\mathcal{G}) = 2$ if and only if $\mathcal{G}$ is bipartite.

▶ **Theorem 18** (Singh, [45]). *Let $\mathcal{G} = (V, E)$ be a graph. The integrality gap of the vertex cover linear programming relaxation $P(\mathcal{G})$ is:*

$$IG(\mathcal{G}) = 2 - \frac{2}{\chi^f(\mathcal{G})},$$

where $\chi^f(\mathcal{G})$ is the fractional chromatic number of the graph $\mathcal{G}$.

We first focus on graphs with the existence of a single vertex whose removal produces a bipartite graph. The following theorem generalizes the result given for the cycle graph in [4, 42] and turns out to be the same formula as for series-parallel graphs [20].

▶ **Theorem 19.** *Let $\mathcal{G} = (V, E)$ be a non-bipartite graph and $v_p \in V$ such that $\mathcal{G} \setminus v_p = (A \cup B, E')$ is bipartite. Then,*

$$\chi^f(\mathcal{G}) = 2 + \frac{1}{\rho - 1},$$

where $2\rho - 1$ is the odd girth of $\mathcal{G}$.

Proof of Theorem 19. We prove this theorem by constructing feasible primal and dual solutions of objective value $2 + 1/(\rho - 1)$. By strong duality, these two solutions are then optimal for their respective linear programs, hence proving the theorem.

We first construct the dual solution. Let $\mathcal{C}$ be the set of all the shortest odd cycles of $\mathcal{G}$. For any such cycle $C \in \mathcal{C}$, define the dual solution $z^C \in \mathbb{R}^V$ by

$$z_v^C = \begin{cases} 1/(\rho - 1) & \text{if } v \in C \\ 0 & \text{if } v \in V \setminus C \end{cases}$$

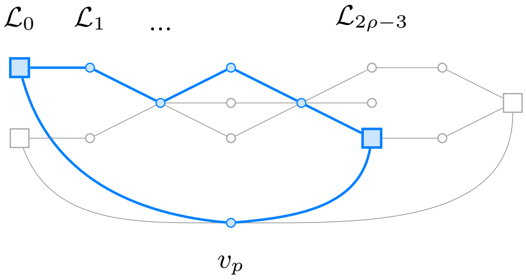


Figure 5 An optimal dual solution constructed in the proof of Theorem 19. Each node on a shortest odd cycle is assigned a fractional value of $1/(\rho-1)$.

This solution is feasible since any independent set in an odd cycle of length $2\rho-1$ has size at most $\rho-1$. Indeed, fix an independent set $I \in \mathcal{I}$, then:

$$\sum_{v\in I} z_v^C = \sum_{v\in I\cap C} \frac{1}{\rho-1} = \frac{|I\cap C|}{\rho-1} \leq 1.$$

Moreover, the objective value of this solution is:

$$\sum_{v\in V} z_v^C = \sum_{v\in C} \frac{1}{\rho-1} = \frac{2\rho-1}{\rho-1} = 2 + \frac{1}{\rho-1}.$$

Let us now construct the primal solution. We will do so by constructing $2\rho-1$ independent sets $I_k \in \mathcal{I}$ and assigning to each of them a fractional value of $y(I_k) = 1/(\rho-1)$. All the other independent sets are assigned value zero. We split the bipartite graph $\mathcal{G} \setminus v_p$ into the layers

$$\mathcal{L}_i := \{v \in A \cup B \mid d(N_A(v_p), v) = i\} \quad \text{for } i \in \{0, \dots, l\}.$$

as explained in (2). As a reminder, any shortest odd cycle corresponds to a path between $\mathcal{L}_0 = N_A(v_p)$ and $\mathcal{L}_{2\rho-3} \cap N_B(v_p)$. The original vertex set V is thus decomposed into $\{v_p\} \cup \mathcal{L}_0 \cup \cdots \cup \mathcal{L}_l$, where each layer is an independent set and only has edges going out to v_p or its two neighbouring layers.

Let us first focus on the subgraph consisting of the vertices in $\{v_p\} \cup \bigcup_{i=0}^{2\rho-3} \mathcal{L}_i$, where any shortest odd cycle has exactly one vertex per layer (per abuse of notation, we say that $\{v_p\}$ is also a layer in this situation). For convenience of indexing, we rename these layers as $\tilde{\mathcal{L}}_1, \dots, \tilde{\mathcal{L}}_{2\rho-1}$ where $\tilde{\mathcal{L}}_1 = v_p$ and $\tilde{\mathcal{L}}_i = \mathcal{L}_{i-2}$ for $i > 1$. We now create $2\rho-1$ independent sets on this subgraph in the following way. The first independent set is defined as $U_1 = \tilde{\mathcal{L}}_1 \cup \tilde{\mathcal{L}}_4 \cup \tilde{\mathcal{L}}_6 \ \cdots \cup \tilde{\mathcal{L}}_{2\rho-2}$, where we take the first layer $\tilde{\mathcal{L}}_1$, skip two before taking the next one and then continue by taking the remaining layers alternatingly (hence always skipping one), see Figure 6. Note that the layer following $\tilde{\mathcal{L}}_{2\rho-1}$ is assumed to be $\tilde{\mathcal{L}}_1$. This procedure generates in fact a distinct independent set by starting at $\tilde{\mathcal{L}}_k$ for any $k \in [2\rho-1]$ and we denote the corresponding independent set by U_k. Notice that each layer is contained in exactly $\rho-1$ of the constructed independent sets $\{U_k \mid k \in [2\rho-1]\}$.

We now focus on the subgraph consisting of the vertices in $\bigcup_{i>2\rho-3} \mathcal{L}_i$. We can construct two different independent sets there by taking either the odd or even indexed layers, i.e.

$$R_1 := \bigcup_{i \text{ odd},\, i>2\rho-3} \mathcal{L}_i \quad \text{and} \quad R_2 := \bigcup_{i \text{ even},\, i>2\rho-3} \mathcal{L}_i.$$

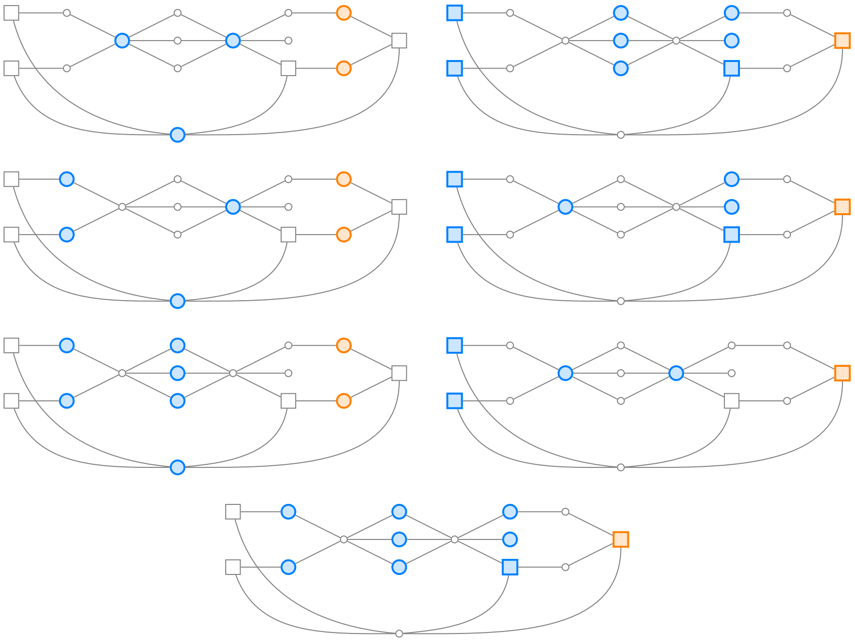


Figure 6 The $2\rho - 1$ independent sets I_k constructed in the optimal primal solution. The blue nodes correspond to $\{U_k \mid k \in [2\rho - 1]\}$, whereas the orange nodes correspond to $\{R_1, R_2\}$.

We now define our final $2\rho - 1$ independent sets on the full graph as:

$$I_k := \begin{cases} U_k \cup R_1 & \text{if } v_p \notin U_k \\ U_k \cup R_2 & \text{if } v_p \in U_k \end{cases} \qquad \forall k \in [2\rho - 1].$$

These are in fact independent sets: in the first case, the first layer in R_1 is $\mathcal{L}_{2\rho-1}$ whereas the last layer in U_k has index at most $2\rho - 3$, meaning that there are no two neighbouring layers. In the second case, since $v_p \in U_k$, we have that $\mathcal{L}_{2\rho-3} \notin U_k$, by construction of U_k. The last layer in U_k thus has index at most $2\rho - 4$, whereas the first layer in R_2 is $\mathcal{L}_{2\rho-2}$, meaning again that there are no two neighbouring layers. In addition, there is no edge between v_p and R_2, because the only even indexed layer having edges sent to v_p is $\mathcal{L}_0 = N_A(v_p)$.

We now define our primal solution as

$$y(I_k) = \frac{1}{\rho - 1} \qquad \forall k \in [2\rho - 1],$$

and $y(I) = 0$ for every other independent set $I \in \mathcal{I}$. We now show this is a feasible solution, i.e. that every vertex $v \in V$ belongs to at least $\rho - 1$ independent sets in $\{I_k \mid k \in [2\rho - 1]\}$. For $v \in \{v_p\} \cup \bigcup_{i=0}^{2\rho-3} \mathcal{L}_i$, such a vertex lies by construction in exactly $\rho - 1$ independent sets $\{U_k \mid k \in [2\rho - 1]\}$, and thus also of $\{I_k \mid k \in [2\rho - 1]\}$. For $v \in \bigcup_{i > 2\rho-3} \mathcal{L}_i$, if v belongs to an even indexed layer, then it is contained in $\rho - 1$ of the desired independent sets. If it belongs to an odd indexed layer, then it is contained in ρ of them. Therefore,

$$\sum_{I \in \mathcal{I}, v \in I} y_I = \sum_{k=1}^{2\rho-1} y(I_k)\, 1_{\{v \in I_k\}} = \frac{1}{\rho - 1} \sum_{k=1}^{2\rho-1} 1_{\{v \in I_k\}} \geq 1.$$

The objective value of this primal solution is clearly $2 + 1/(\rho - 1)$. We have constructed feasible primal and dual solutions with the same objective value. By strong duality, this finishes the proof of the theorem. ◀

We now consider the case where $\mathcal{G} = (V, E)$ is a graph with chromatic number $\chi(\mathcal{G}) = 3$.

▶ **Theorem 20.** *Let $\mathcal{G} = (V, E)$ be a 3-colorable graph with color classes $V = V_1 \cup V_2 \cup V_3$. Then,*

$$\chi^f(\mathcal{G}) \le 2 + \min_{i \in \{1,2,3\}} \frac{1}{\rho_i - 1}$$

where $2\rho_i - 1$ is the odd girth of the contracted graph $\mathcal{G}/V_i$ for each $i \in \{1, 2, 3\}$. Moreover, equality holds if one color class only contains one vertex.

Proof. We prove this theorem by constructing three feasible solutions of value $2 + 1/(\rho_i - 1)$ for each $i \in \{1, 2, 3\}$ to the primal linear program of the fractional chromatic number on the graph $\mathcal{G}$.

Fix an $i \in \{1, 2, 3\}$ and consider the graph $\tilde{\mathcal{G}} := \mathcal{G}/V_i = (\tilde{V}, \tilde{E})$ with odd girth $2\rho_i - 1$. We denote the contracted node by $\tilde{v} \in \tilde{V}$. Since this graph is bipartite if we were to remove $\tilde{v}$, we know that its fractional chromatic number is equal to $2 + 1/(\rho_i - 1)$ by Theorem 19. Let $\{\tilde{I}_k, k \in [2\rho_i - 1]\}$ be the independent sets in the support of the optimal primal solution of the graph $\tilde{\mathcal{G}}$ constructed in the proof of this theorem. For each of these independent sets, we extend them to the original graph in the following way:

$$I_k = \begin{cases} \tilde{I}_k & \text{if } \tilde{v} \notin \tilde{I}_k \\ (\tilde{I}_k \setminus \tilde{v}) \cup V_i & \text{if } \tilde{v} \in \tilde{I}_k. \end{cases}$$

In words, if $\tilde{v}$ happens to belong to $\tilde{I}_k$, we replace it by V_i to get a valid independent set in the original graph. Assigning fractional value $y(I_k) = 1/(\rho_i - 1)$ for every $k \in [2\rho_i - 1]$ yields a feasible primal solution with objective value $2 + 1/(\rho_i - 1)$. Since we can do this for every $i \in \{1, 2, 3\}$, and the optimal minimum value of the primal linear program is at most the objective value of any of these feasible solutions, the proof is finished.

Moreover, this upper bound is in fact tight, since it holds with equality when one of the color classes only contains one vertex by Theorem 19. ◀

B Arbitrary Set to Bipartite: Omitted Proofs

▶ **Theorem 21.** *Let $\mathcal{G} = (V, E)$ be a graph and $S \subset V$ such that $\mathcal{G} \setminus S = (A \cup B, E')$ is bipartite. For any $w \in Q^W$, the approximation ratio satisfies*

$$R(w) \le 1 + \alpha \qquad \textit{with } \alpha \in [0, 1]$$

if the contracted graph $\mathcal{G}/S$ is bipartite.

Proof. The only change with respect to the previous proof is the bound on $w(OPT(\mathcal{G} \setminus S))$ in (5). We denote the contracted graph by $\tilde{\mathcal{G}} = \mathcal{G}/S$ and by v^S the contracted vertex. Suppose $\tilde{\mathcal{G}}$ admits the bipartition $(\tilde{A} \cup \tilde{B}, \tilde{E})$ and assume without loss of generality that $v^S \in \tilde{A}$. Note that $\tilde{E} = E' \cup \delta(v^S)$.

Any feasible cover of $\mathcal{G} \setminus S$ needs to count the dual value of every edge in E' at least once. Taking the cover $\tilde{A} \setminus v^S$ counts every edge in E' exactly once, showing that $w(OPT(\mathcal{G} \setminus S)) = y(E')$. Hence, using $y(E) = \alpha + y(\delta(S)) + y(E') = 1$, we get

$$R(w) \le 2\alpha + y(\delta(S)) + y(E') = 1 + \alpha.$$ ◀

▶ **Theorem 22.** *Let $\alpha \in [0,1]$ and let $\rho \geq 2$. There exists a non-bipartite graph $\mathcal{G} = (V, E)$, with weights $(y, w) \in Q^{Y,W}$, and a set $S \subset V$ with $y(E[S]) = \alpha$, where $\mathcal{G}/S$ has odd girth $2\rho - 1$ and which satisfies*

$$R(w) = \left(1 + \frac{1}{\rho}\right)(1 - \alpha) + 2\alpha.$$

Proof. An example of such a graph can be constructed as follows. We first construct $\mathcal{G}/S$: take an odd cycle of length $2\rho - 1$ with a distinguished node v^S and assign dual value $(1-\alpha)/\rho$ to both edges incident to it. Alternatively assign dual values 0 and $(1-\alpha)/\rho$ along the odd cycle for the remaining edges. In order to construct $\mathcal{G}$, replace v^S by a triangle S with dual edges set to $\alpha, 0$ and 0, where the two previous incident edges to v^S are adjacent to the endpoints of the edge with value α. Note that we replace it with a triangle instead of a single edge in order to avoid $\mathcal{G}$ becoming bipartite. Similarly to the proof of Theorem 12, one can check that

$$w(S) = 2\alpha + \frac{2\,(1-\alpha)}{\rho}; \quad w(OPT(\mathcal{G} \setminus S)) = \frac{(1-\alpha)(\rho-1)}{\rho}; \quad w(OPT(\mathcal{G})) = 1.$$

Therefore,

$$R(w) = 2\alpha + \frac{2\,(1-\alpha)}{\rho} + \frac{(1-\alpha)(\rho-1)}{\rho} = \left(1 + \frac{1}{\rho}\right)(1 - \alpha) + 2\alpha.$$ ◀

▶ **Theorem 23.** *Let $\alpha \in [0,1]$. There exists a non-bipartite graph $\mathcal{G} = (V, E)$, with weights $(y, w) \in Q^{Y,W}$, and a set $S \subset V$ with $y(E[S]) = \alpha$, where $\mathcal{G}/S$ is bipartite and which satisfies*

$$R(w) = 1 + \alpha.$$

Proof. Let $\mathcal{G}$ be an arbitrary odd cycle. Consider an arbitrary edge $(u, v) \in E$ and assign it dual value α. The set S is defined to be $S = \{u, v\}$. Assign dual value zero to the edge $(u, w) \in E$, where w is the second neighbour of u in the cycle. For the remaining edges, arbitrarily assign dual values, while ensuring that they sum up to $1 - \alpha$. The fact that one edge is equal to zero is necessary in order to get the exact formula $w(OPT(\mathcal{G})) = 1$, a feasible cover showing $w(OPT(\mathcal{G})) \leq 1$ being the following: take both endpoints of the edge (u, w) and take remaining vertices alternatively (hence always skipping one) along the odd cycle. All the edges are counted once, except for (u, w), which is counted twice but has value zero. Moreover, $w(S) = 2\alpha + y(\delta(S))$ and $w(OPT(\mathcal{G} \setminus S)) = y(E')$, where E' is the edge set of the bipartite graph $\mathcal{G} \setminus S$. Therefore,

$$R(w) = 2\alpha + y(\delta(S)) + y(E') = 1 + \alpha.$$ ◀

On Minimizing Generalized Makespan on Unrelated Machines

Nikhil Ayyadevara ✉
University of Michigan, Ann Arbor, MI, USA

Nikhil Bansal ✉
University of Michigan, Ann Arbor, MI, USA

Milind Prabhu ✉
University of Michigan, Ann Arbor, MI, USA

Abstract

We consider the *Generalized Makespan Problem* (GMP) on unrelated machines, where we are given n jobs and m machines and each job j has arbitrary processing time p_{ij} on machine i. Additionally, there is a general symmetric monotone norm ψ_i for each machine i, that determines the load on machine i as a function of the sizes of jobs assigned to it. The goal is to assign the jobs to minimize the maximum machine load.

Recently, Deng, Li, and Rabani [8] gave a 3 approximation for GMP when the ψ_i are top-k norms, and they ask the question whether an $O(1)$ approximation exists for general norms ψ? We answer this negatively and show that, under natural complexity assumptions, there is some fixed constant $\delta > 0$, such that GMP is $\Omega(\log^{\delta} n)$ hard to approximate. We also give an $\Omega(\log^{1/2} n)$ integrality gap for the natural configuration LP.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases Hardness of Approximation, Generalized Makespan

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.21

Category APPROX

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

1 Introduction

We consider a question by Deng, Li and Rabani [8] about scheduling jobs to minimize makespan on unrelated machines in the setting of more general norms. Recall that in the unrelated machines setting, we are given n jobs and m machines where each job $j \in [n]$ has some arbitrary processing time p_{ij} on machine $i \in [m]$. Given an assignment of jobs to machines, the load on a machine is the sum of the processing times of all the jobs assigned to it. In a seminal result, Lenstra, Shmoys and Tardos [14] gave a 2-approximation for minimizing the makespan (the maximum machine load). These results were later extended to the problem of minimizing the ℓ_p-norm of the machine loads [1, 4, 13, 16].

In a breakthrough work [5], Chakrabarty and Swamy introduced a substantial generalization of the problem to general symmetric monotone norms. Recall that a function $\phi : \mathbb{R}^m \to \mathbb{R}_{\geq 0}$ is a norm if it satisfies (i) $\phi(u) = 0$ iff $u = 0^n$, (ii) $\phi(\alpha u) = |\alpha| u$ for all u and $\alpha \in R$ and, (iii) $\phi(u + v) \leq \phi(u) + \phi(v)$. We say that ϕ is *symmetric* if $\phi(u) = \phi(u')$ where u' is some permutation of u, and it is monotone if $\psi(u) \leq \psi(v)$ for all u, v satisfying $0 \leq u(i) \leq v(i)$ for all $i \in [m]$.

Surprisingly, they showed that for any arbitrary symmetric monotone norm $\phi : \mathbb{R}^m \to \mathbb{R}_{\geq 0}$, that determines the overall objective as a function of the individual machine loads, there is an $O(1)$ approximation. The approximation ratio was subsequently improved to $4 + \epsilon$ [6]

and more recently to $2+\epsilon$ [11], which remarkably almost matches the bound for makespan. These results introduce several new ideas to handle general norms by relating them to the special class of top-k norms[1], and algorithmic techniques to work with them.

General inner and outer norms. An even further generalization was considered by Deng, Li and Rabani [8], which we refer to as the generalized load-balancing (GLB) problem. Here, additionally, each machine i also has an arbitrary symmetric monotone norm $\psi_i : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ referred to as the *inner-norm*, that determines the machine's load as a function of the sizes of the jobs assigned to it (note that all the results described previously have inner norm ℓ_1). Formally, given an assignment $\rho : [n] \to [m]$ of jobs to machines, the load on machine i is $\mathsf{load}(i) = \psi_i(p_i^\rho)$ where p_i^ρ is the vector of sizes of jobs assigned to i, i.e., $p_i^\rho(j) = p_{ij}$ if $\rho(j) = i$ and 0 otherwise. The overall objective is $\phi(\mathsf{load})$, where $\mathsf{load} = (\mathsf{load}(1), \dots, \mathsf{load}(m))$ is the vector of machine loads (where $\phi : \mathbb{R}^m \to \mathbb{R}_{\geq 0}$ is the norm as in [5]). We shall refer to ϕ as the *outer-norm.* Throughout this paper a general norm always means a symmetric monotone norm.

In its full generality (when the ψ_i and ϕ are general), GLB becomes $\Omega(\log n)$ hard to approximate, as it generalizes[2] the Set Cover problem when $\psi = \ell_\infty$ and $\phi = \ell_1$. Interestingly, [8] gave a matching $O(\log n)$ approximation for general ψ and ϕ, based on solving and rounding a novel configuration LP.

Generalized Makespan Problem (GMP). Given the $\Omega(\log n)$ hardness for the general case, [8] also consider the interesting and natural special case where the outer-norm ℓ_∞, but the inner norm is general (i.e., the goal is to minimize makespan but where the machine loads are given by general inner-norms ψ_i). We refer to this problem as the *Generalized Makepsan Problem* (GMP). In a sense, this problem can be considered as a "dual" of the problem considered by Chakrabarty and Swamy (where the inner-norm is ℓ_1, but the outer-norm is general).

For GMP, Deng, Li and Rabani gave a 3-approximation for the special case when each ψ_i is a top-k norm. The main open question they ask is whether an $O(1)$ approximation is achievable for general inner-norms. Apriori this seems quite plausible as there is close connection between top-k norms and general norms (see e.g. [5, 6]). Moreover, the $O(1)$ approximation of Chakrabarty and Swamy [5] for the "dual" problem, also suggests that GMP may have an $O(1)$ approximation.

Our Results

Our main result is that GMP does not admit an $O(1)$-approximation under standard complexity-theoretic assumptions, answering the main open problem in [8].

Our starting point is an integrality gap instance for the natural configuration LP for GMP.

▶ **Theorem 1.** *There is an instance of GMP with a symmetric monotone norm ψ, for which the natural configuration LP has an integrality gap of $\Omega((\log n)^{1/2})$.*

[1] The top-k norm of a non-negative vector v is the sum of its largest k entries.

[2] Given subsets $S_1, \dots, S_m$ of $[n]$, consider the scheduling instance on m machines (one per set) and n jobs (one per element), with $p_{ij} = 1$ iff element $j \in S_i$ and $p_{ij} = \infty$ otherwise. That is, only jobs in S_i can be assigned to i. The point is that as $\psi = \ell_\infty$, we have $\mathsf{load}(i) = 0$ if no job is assigned to i, and exactly 1 otherwise (even if all jobs in S_i are assigned to i). As $\phi = \ell_1$, the objective is exactly the number of machines (sets) needed to cover all the jobs.

The gap instance is based on a probabilistic construction and a key idea is to work with a suitably chosen norm ψ, defined as the sum of top-k norms at various different scales, that interacts nicely with properties of random set cover instances. This construction is described in Section 3, and it forms a key gadget in our following hardness result.

▶ **Theorem 2.** *There is a universal constant $\delta > 0$, such that any polynomial-time approximation algorithm for GMP has approximation ratio $\Omega(\log^{\delta} n)$ provided that* $\mathsf{NP} \not\subseteq \mathsf{ZTIME}(n^{O(\log\log n)})$.

Our construction for this hardness result builds on the ideas of Lund and Yannakakis [15], who showed the $\Omega(\log n)$ hardness of Set Cover by a gap reduction from the Label Cover problem. However, we require some additional ideas as reducing the Label Cover problem to a scheduling instance and exploiting the properties of the norm ψ requires much more care. Specifically, even though our gadget in Theorem 1 has the $\Omega((\log n)^{1/2})$ gap, embedding it into Label Cover imposes extra constraints on the number of labels in the Label Cover instance, and only leads to an $\Omega(\log^{\delta} n)$ hardness, for some constant $\delta > 0$. Interestingly, this constant δ depends on the soundness parameter of label cover as a function of the number of labels, based on Raz's parallel repetition theorem [18] and its subsequent improvements [10, 17].

▶ Remark. The constant δ can be computed explicitly, but we do not attempt to do this here. This construction is described in Section 4. We remark that there are extensive work on improving the soundness in PCP constructions as a function of the number of labels. The best known result in this direction is due to Chan [7] that achieves soundness roughly $L^{-1/2}$ for L labels. However, Chan's result does not have perfect completeness and hence cannot be used in our constructions. Roughly speaking, this is because each job must be assigned to some machine (this is similar to the reason that one requires perfect completeness in reducing label cover to set cover, as each element must be covered).

▶ Remark. In personal communication, Amey Bhangale has pointed out that in an unpublished manuscript, they can show that assuming the 2-to-2 conjecture with perfect completeness, there is a label cover instance with L labels that has perfect completeness and soundness $L^{-1/2}$. Using a such a label cover, our construction in Section 4 would imply a hardness of $\Omega(\log^{1/8} n)$ in Theorem 2. We note that currently, proving the 2-to-2 conjecture with perfect completeness remains open, and in particular the breakthrough results of Khot, Minzer and Safra [12] on the 2-to-2 conjecture assume imperfect completeness.

2 Preliminaries

2.1 (n, m, ℓ, β) Set-System

The constructions of the integrality gap instance in Section 3 and the reduction from label cover to GMP presented in Section 4 both use the following set system as a building block.

▶ **Definition 3** ((n, m, ℓ, β) Set-system). *Let n, m, ℓ be positive integers, $\beta \in (0, 1)$, U be a set with $|U| = n$, and $A_1, \dots, A_m$ be subsets of U. The sets $(U; A_1, \dots, A_m)$ form an (n, m, ℓ, β) set-system if for every set I of at most ℓ indices from $[m]$, $|\cup_{i\in I} B_i| \leq (1 - \beta)|U|$, where B_i is either A_i or $\overline{A}_i$.*

Intuitively, an (n, m, ℓ, β) set-system has the property that any set cover which uses at most ℓ subsets necessarily uses a complementary pair of subsets A_i and $\overline{A}_i$. Moreover, any collection of at most ℓ subsets that do not contain any complementary pair can cover at most a $(1-\beta)$ fraction of the elements in U. The following lemma shows that for a particular choice of parameters n, m, ℓ, and β, there is a simple and efficient randomized construction of an (n, m, ℓ, β) set-system.

▶ **Lemma 4.** *For a sufficiently large positive integer n and a positive integer $m \in [\sqrt{\log n}, 2\sqrt{\log n}]$ there exists an (n, m, ℓ, β) set-system $(U; A_1, \ldots, A_m)$ with $\ell = m/10$ and $\beta = \exp(-m) = \exp(-O(\sqrt{\log n}))$. There is a polynomial-time algorithm that constructs such a set system with high probability.*

Proof. Let U be a set of n elements, and initialize m empty sets $A_1, \ldots, A_m$. For each element $e \in U$, sample a random index set $J \subset [m]$ of size exactly $m/2$ and add e to sets A_j for $j \in J$.

We show that this construction gives an (n, m, ℓ, β) set-system with high probability. Consider an index set $I \subset [m]$ with $|I| = \ell$ and a collection of sets B_i for $i \in I$ such that each B_i is either A_i or $\overline{A_i}$. For a fixed $e \in U$, let p denote the probability that e is not contained in $\cup_{i\in I} B_i$, i.e., $p = \Pr[e \notin \cup_{i\in I} B_i]$. We have,

$$p \geq \binom{m-\ell}{m/2} \Big/ \binom{m}{m/2} \geq \left(\frac{m-\ell}{m/2}\right)^{m/2} \Big/ \left(\frac{em}{m/2}\right)^{m/2} \geq \left(\frac{m-\ell}{em}\right)^{m/2} \geq \exp(-0.6m),$$

where the second inequality uses that $\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \left(\frac{en}{k}\right)^k$.

The probability that $\cup_{i\in I} B_i$ contains a fixed subset of cardinality greater than or equal to $(1-\beta)n$ is at most $(1-p)^{(1-\beta)n}$. By a union bound over at most $\binom{n}{\beta n} n$ possible subsets with cardinality at least $(1-\beta)n$,

$$\begin{aligned}\Pr\left[|\cup_{i\in I} B_i| \geq (1-\beta)n\right] &\leq \binom{n}{\beta n} n \cdot (1-p)^{(1-\beta)n} \leq (e/\beta)^{\beta n}\, n \cdot e^{-(1-\beta)np} \\ &\leq \exp\left(3n\beta \log(1/\beta) - np/2\right) \leq \exp(-np/4) \leq \exp(-n^{0.9}),\end{aligned}$$

where in the second to last inequality we use that $3\beta \log(1/\beta) < p/4$.

A union bound over the at most $2^\ell \cdot \binom{m}{\ell} \leq \exp(\sqrt{\log n})$ possible choices to pick the ℓ sets B_i, gives that with high probability, the union of any ℓ sets B_i has cardinality less than $(1-\beta)n$. ◀

2.2 Label Cover

In Section 4, we prove the hardness of approximation of GMP via a reduction from the standard label cover problem as defined below.

▶ **Definition 5.** *A* label cover *instance $\mathcal{L}$ is defined by a tuple $((U, V, E), L, \Pi)$. Here (U, V, E) is a bipartite graph with vertices $U \cup V$ and edges $E \subseteq U \times V$; L is a positive integer and Π is a set of functions one for each edge $e \in E$ i.e., $\Pi = \{\pi_e : [L] \to [L] \mid e \in E\}$. A labeling of the vertices $\sigma : U \cup V \to [L]$ is said to satisfy an edge $e = (u, v)$ if $\pi_e(\sigma(u)) = \sigma(v)$. Given $\mathcal{L}$, the goal of the label cover problem is to find a labeling σ^* that satisfies the maximum number of edges in E. We use $OPT(\mathcal{L})$ to denote the fraction of the edges in E satisfied by σ^*.*

As we will need the explicit dependence between the number of labels and the soundness, for completeness we sketch below the precise gap version of the label cover problem that we will use.

▶ **Lemma 6** (Hardness of Gap Label Cover)**.** *Given a label cover instance $\mathcal{L} = ((U, V, E), L, \Pi)$ satisfying:*

(i) $|U| = |V| = N$
(ii) *The degree of every vertex in* $U \cup V$ *is* $d = O((\log N)^{c_1})$ *for some constant* c_1.
(iii) $L = \sqrt{\log N}$

There is some constant $c > 0$*, for which there is no polynomial-time algorithm to decide if* $OPT(\mathcal{L}) = 1$ *or* $OPT(\mathcal{L}) \leq (\log N)^{-c}$ *provided that* $\mathsf{NP} \not\subseteq \mathsf{DTIME}(N^{O(\log\log N)})$.

Proof. Using a standard argument (see for ex., [9, 2]) one can obtain a reduction from a 3SAT-5 instance ϕ with t variables to a Label Cover instance $\mathcal{L}_1 = ((U_1, V_1, E_1), 8, \Pi)$, where $|U_1| = |V_1| = O(t)$ and the graph (U_1, V_1, E) is 15-regular. The instance $\mathcal{L}_1$ has the following property: if ϕ has a satisfying assignment then $OPT(\mathcal{L}_1) = 1$; else if any assignment satisfies at most $(1 - \epsilon)$ fraction of clauses in ϕ, then $OPT(\mathcal{L}_1) \leq (1 - \Theta(\epsilon))$. By the PCP-theorem [3] it follows that, for some constant $\epsilon_0 > 0$, deciding if $OPT(\mathcal{L}_1) = 1$ or $OPT(\mathcal{L}_1) \leq 1 - \epsilon_0$ is NP-hard.

The following well-known construction [2] gives stronger inapproximability results for label cover. We define the kth power of the label cover instance $\mathcal{L}_k = ((U_k, V_k, E_k), 8^k, \Pi^k)$, where U_k, V_k are k-tuples of vertices in U_1, V_1 respectively, E_k is the set of all k-tuples of edges in E_1. The resulting graph has $N = t^{O(k)}$ vertices and is $(15)^k$-regular. The new set of labels[3] consist of k-tuples of $\{1, \dots, 8\}$. For an edge $e = (e_1, \dots, e_k) \in E_k$, we define the function $\pi_e^k(a_1, \dots, a_k) = (\pi_{e_1}(a_1), \dots, \pi_{e_k}(a_k))$. Raz's Parallel Repetition Theorem [18], shows that for the label cover instance constructed above, there exists a constant α such that $OPT(\mathcal{L}_k) \leq (OPT(\mathcal{L}_1))^{\alpha k}$.

We now pick k so that $L = \sqrt{\log N}$. Since $L = 8^k$ and $N = t^{O(k)}$, this gives $k = \Theta(\log\log t)$. This choice of k ensures that $d = (15)^k = (\log N)^{c_1}$ for some constant c_1. Moreover, if $OPT(\mathcal{L}_1) \leq (1 - \epsilon_0)$, then $OPT(\mathcal{L}_k) \leq (1 - \epsilon_0)^{\alpha k} \leq (\log t)^{-c'} \leq (\log N)^{-c}$ for some positive constants c, c'. ◀

3 Integrality Gap for Configuration LP

We begin by describing the configuration LP for GMP. We then explain the high-level ideas behind the gap construction and the properties we need from the norm ψ. We then describe the norm ψ and the integrality gap instance formally and then prove Theorem 1. As mentioned earlier, this gap instance and the norm ψ form the key gadget in our hardness construction in Section 4.1, and understanding it is crucial to the results in Section 4.

Configuration LP. The most natural LP relaxation of GMP is to consider assignment variables $x_{ij} \in [0, 1]$ that determine the fraction of job j assigned to machine i, and impose natural constraints. However, simple examples show that such an LP is too weak to handle general norms ψ. A stronger relaxation is the configuration LP, where we have exponentially many variables $x_{i,C}$, one for each machine i, and a possible subset of jobs C that can be feasibly assigned to machine i.

Let J denote the set of jobs, and $M = [m]$ be the set of machines. For a machine $i \in M$, and subset $C \subseteq J$, let $p_i[C] = (p_{ij} \cdot \mathbb{1}[j \in C])_{j \in J}$ denote the vector of sizes of jobs in C. Let T be a guess on the optimum makespan (we can do a binary search on T). Call a configuration C *valid* for machine i if the load $\psi_i(p_i[C])$ of C on i is at most T. We will slightly abuse notation and denote $\psi_i(p_i[C])$ by $\psi_i(C)$. Consider the following feasibility LP.

[3] The labels are essentially numbers from 1 to 8^k.

$$\sum_{C \subseteq J} x_{i,C} \leq 1 \qquad \forall i \in M \tag{1a}$$

$$\sum_{i \in M, C: j \in C} x_{i,C} = 1 \qquad \forall j \in J \tag{1b}$$

$$x_{i,C} = 0 \qquad \text{if } \psi_i(C) > T \tag{1c}$$

The first constraint says that each machine has at most one configuration. The second constraint says that each job is assigned to a machine, and the last constraint ensures that only valid configurations are considered and hence the generalized makespan on any machine is at most T. This LP can be solved efficiently for any desired accuracy $\epsilon > 0$ (so the configurations satisfy $\psi_i(C) \leq (1+\epsilon)T$), see e.g., [8].

The main idea. We start with a simple instructive example, that motivates our choice of the norm and the choice of parameters that lead to the $\Omega(\sqrt{\log n})$ integrality gap. Consider a random set system on a universe U of n elements and m sets $A_1, \ldots, A_m$ where each element independently lies in $m/2$ randomly chosen sets. We will set $m \ll \log n$. Create an unrelated machine (in fact restricted assignment) scheduling instance I with m machines, where only the jobs in A_i can be assigned to machine i (all other jobs have infinite size on i). As each element lies in $m/2$ sets, the LP solution that picks the configuration A_i on each machine i with $x_{i,A_i} = 2/m$ is feasible, and uses only 2 machines in total.

However, in any integral solution, we claim that several machines must pick a non-trivial fraction of jobs from their sets A_i. Roughly, this is because the union of any $\ell \ll m$ sets A_i still leaves about $2^{-\ell}|U|$ uncovered (as each element lies in a set with probability 1/2). Hence in any feasible integral solution, for any $\ell > 0$, at least ℓ machines must be assigned at least $2^{-\ell}|U|/m$ jobs.

To exploit this, suppose we define ψ as the top-k norm with $k \approx \Omega(2^{-\ell}n/m)$, so that any machine with $\geq k$ jobs incurs the same load, say T. Then in any integral solution, at least ℓ machines have load T, while fractionally at most 2 machines (in total) have load T. By creating $m/2$ disjoint copies $I^{(1)}, \ldots, I^{(m/2)}$ of these set-cover instances, one would then expect that fractionally each machine has load T, while integrally the average load becomes $\Omega(\ell)$.

Unfortunately, this does not quite work as stated above, because once there are several instances $I^{(1)}, \ldots, I^{(m/2)}$, as these jobs share the same machines, an algorithm can find a low makespan even if it cannot figure out the good underlying set cover solution. In fact, this is provably so, as [8] gave a 3-approximation when ψ is a Top_k norm.

However, our key observation is that this idea can still be made to work by choosing the instances $I^{(1)}, \ldots, I^{(m/2)}$ at different scales (of the number of jobs and processing times) and defining the norm ψ as a suitable mixture of the top-k norms at these different scales. Roughly, this norm ψ still behaves as a top-k norm at each individual scale, but when jobs from different scales are combined, it takes on a large value, which be used to create a large gap in the reduction above.

Implementing this idea requires separating every two scales by $\Omega(2^\ell)$. So the ℓ scales leads to instances with size about 2^{ℓ^2} leading to the choice of $\ell = \Theta(\sqrt{\log n})$ to produce the $\Omega(\ell)$ gap. We now give the details.

The Instance. The instance will have n jobs and $m = \sqrt{\log n}$ machines. We first create $h = m/8$ disjoint set-cover instances $I^{(1)}, \ldots, I^{(h)}$, where $I^{(s)} = (U(s); A_1(s), A_2(s), \ldots, A_m(s))$ forms an (n_s, m, ℓ, β) set-system with $\ell = m/10, \beta = \exp(-m)$ and $n_s = (\sqrt{n}/2) \cdot \exp(4ms) = (\sqrt{n}/2)\beta^{-4s}$. Note that n_s increases as β^{-4s} with s, and $n_1 \geq \sqrt{n}$ and $n_h = n/2$, and it is easily checked that parameters for each $I^{(s)}$ satisfy the condition in Lemma 4.

For each $s \in [h]$, the elements in $U(s)$ correspond to jobs with size $p^{(s)} = \beta^{s-1}$ (or infinite if the job cannot be assigned to a machine). Each job j in $U(s)$ has size $p_{ij} = p^{(s)}$ on machine i iff $j \in A_i(s)$ and $p_{ij} = \infty$ otherwise. We abuse the notation slightly to refer to the resulting scheduling instance also as $I^{(s)}$. As the instances $I^{(1)}, \dots, I^{(h)}$ are at different scales in terms of the number of jobs and processing times, we refer to jobs in $I^{(s)}$ as being in the s-th *size class* [4].

We define the inner norm $\psi = \sum_{s\in[h]} \psi^{(s)}$, where each $\psi^{(s)}$ is a scaled top-k norm given by

$$\psi^{(s)}(\mathbf{v}) = \frac{\mathsf{Top}_{\beta^2 n_s}(\mathbf{v})}{(\beta^2 n_s p^{(s)})}. \tag{2}$$

Each machine i will have the same inner norm $\psi_i = \psi$. Informally, ψ has the following key property: for any subset C of $A_i(s)$ with at least $\beta^2 n_s$ jobs, $\psi_i^{(s)}(C) = 1$ and $\psi_i^{(s')}(C) \approx 0$ for $s' \neq s$. This in particular implies that $\psi_i(A_i(s)) \approx 1$ for any class s. Moreover, if C is an arbitrary subset of jobs with at least a β^2 fraction of jobs from r distinct sets among $A_i(1), A_i(2), \cdots, A_i(h)$, then $\psi_i(C) \approx r$ (roughly, each such size-class s affects a different term $\psi^{(s)}$ of the norm). This property motivates the following definition.

▶ **Definition 7** (Heavy Size Class)**.** *Given an assignment of jobs $\rho : J \to M$, we say that size class $s \in [h]$ is heavy on machine $i \in M$, if at least $\beta^2 n_s$ jobs from $A_i(s)$ are assigned to machine i.*

We now formally state and prove the property of the norm described above.

▶ **Lemma 8.** *For any $s \in [h]$ and $i \in M$, $\psi_i(A_i(s)) = 1 + o(1)$. Furthermore, for an assignment $\rho : J \to M$, if C is the set of jobs assigned to machine i, then $\psi_i(C)$ is at least the number of heavy size classes $s \in [h]$ on machine i.*

Proof. We first show that $\psi_i(A_i(s)) = 1 + o(1)$ by computing the value of $\psi^{(s')}$ for different s'. By the definition of $\psi^{(s')}$,

$$\psi_i^{(s')}(A_i(s)) = \frac{\mathsf{Top}_{\beta^2 n_{s'}}(p_i[A_i(s)])}{(\beta^2 n_{s'} p^{(s')})} = \frac{\min\{\beta^2 n_{s'}, |A_i(s)|\} \cdot p^{(s)}}{(\beta^2 n_{s'} p^{(s')})} = \left(\min\left\{1, \frac{|A_i(s)|}{\beta^2 n_{s'}}\right\}\right) \frac{p^{(s)}}{p^{(s')}}$$

For $s = s'$, this exactly equals 1 (as $\beta \ll 1$ and $|A_i(s)| \approx \frac{n_s}{2}$).
For $s' < s$, we have $\psi_i^{(s')}(A_i(s)) \leq \frac{p^{(s)}}{p^{(s')}} = \beta^{(s-s')} \leq \beta$.
Finally, for $s' > s$, we have

$$\psi_i^{(s')}(A_i(s)) \leq \frac{|A_i(s)|}{\beta^2 n_{s'}} \cdot \frac{p^{(s)}}{p^{(s')}} \leq \frac{n_s}{\beta^2 n_{s'}} \cdot \frac{p^{(s)}}{p^{(s')}} = \beta^{3(s'-s)-2} \leq \beta.$$

Let us now consider an arbitrary set of jobs C assigned to machine i. By the monotonicity of the norm,

$$\psi_i^{(s)}(C) \geq \psi_i^{(s)}(C \cap A_i(s)) = \min\left\{1, \frac{|C \cap A_i(s)|}{\beta^2 n_s}\right\} \geq \mathbb{1}[s \text{ is heavy on } i].$$

As $\psi_i(C) = \sum\limits_{s\in[h]} \psi_i^{(s)}(C)$, it follows that $\psi_i(C)$ is at least the number of heavy size classes on i. ◀

[4] The total number of jobs in the h instances is $\sum_{i=1}^{h} n_s \approx n/2$. To make the total number of jobs n, we add dummy jobs that have processing time 0 on all machines.

Fractional Solution. We claim that the following solution is feasible for the configuration LP Equation (1) with $T = 2$: for each machine $i \in M$, set $x_{i,A_i(s)} = 2/m$ for each $s \in [h]$.

Clearly, Equation (1a) is satisfied for each $i \in M$ as there are $h < m/2$ sets of jobs $A_i(1), A_i(2), \ldots, A_i(h)$, each with value $2/m$. Equation (1b) is satisfied as each job $j \in U(s)$, for each $s \in [h]$, lies in $m/2$ sets in $A_1(s), A_2(s), \ldots, A_m(s)$. Equation (1c) is also satisfied as $\psi_i(A_i(s)) = 1 + o(1) < 2 = T$ by Lemma 8.

Integral Solution. We now show that any integral solution has generalized makespan $\Omega(\sqrt{\log n})$.

▶ **Lemma 9.** *For any assignment $\rho : J \to M$, there is some machine with load $\Omega(\sqrt{\log n})$.*

Proof. We first show that each size class is heavy on at least $\ell = \sqrt{\log n}/10$ machines. Suppose this is not true for some size class $s \in [h]$. Let $H \subseteq M$ be the set of machines on which s is heavy, and so $|H| < \ell$. As $I^{(s)}$ forms an (n_s, m, ℓ, β) set-system, by Definition 3, $| \cup_{i \in H} A_i(s)| \le (1 - \beta)n_s$, and hence at least βn_s jobs from $U(s)$ are assigned to machines $i \notin H$. However, as any machine $i \notin H$ can have at most $\beta^2 n_s$ jobs from $A_i(s)$, the machines in $i \notin H$ can have is at most $m \cdot \beta^2 n_s < \beta n_s$, contradicting that each job in $U(s)$ was assigned to some machine.

By averaging over machines, there exists a machine i on which at least $(h\ell/m) = \Omega(\sqrt{\log n})$ size classes are heavy. By Lemma 8, this implies that $\psi_i = \Omega(\sqrt{\log n})$. ◀

This concludes the proof of Theorem 1.

4 Reduction from Label-Cover

We now prove Theorem 2 by reducing Label Cover to a GMP instance.

Overview. Our construction builds on the ideas used by Lund and Yannakakis [15] to show the $\Omega(\log n)$ hardness of set cover, using a gadget based on a natural $\Omega(\log n)$ integrality gap instance. We first give a rough sketch of their idea (see e.g. [2] for an excellent exposition), and then explain the additional steps needed in our setting and why we only get a $\log^{\delta} n$ hardness for some small $\delta > 0$, despite the $\Omega(\log^{1/2} n)$ integrality gap instance above.

Consider a label cover instance $\mathcal{L} = ((U, V, E), L, \Pi)$, as defined in Definition 5, with label set $[L]$ and vertex sets U and V. The main idea in [15] is the following. For each edge $e = (u, v) \in E$, one associates a (n, m, ℓ, β)-system $I^e = (U^e; A_1^e \ldots, A_m^e)$ (with disjoint universes for each edge). The sets will be associated with labels for vertices (so that $L = m$) and we associate the set $A^e_{\pi_e(a)}$ with label a for u and $\overline{A_b^e}$ with label b for vertex v. The point is that in the completeness case, where the labels a' and b' for u and v satisfy e (i.e., $\pi_e(a') = b'$), U^e can be covered by just the two corresponding sets $A^e_{\pi_e(a')}$ and $\overline{A^e_{b'}}$. Conversely, in the soundness case, if U^e is covered using less than ℓ sets, there must be a pair of sets that are complements of each other, which can be used to produce a good labeling for $\mathcal{L}$.

To adapt this to our setting, suppose we create a job for each element in U^e, and m machines per vertex, one for each label in $[L]$. Also suppose that for labels a, b, we set the processing times of jobs in the set $A^e_{\pi_e(a)}$ to be finite on the a-th machine of vertex u and the processing times of jobs in set $\overline{A_b^e}$ to be finite on the b-th machine of vertex v. In the completeness case, any perfect labeling gives an assignment where jobs are assigned to exactly one out of the m machines per vertex (similar to the value of the LP solution in the gap example in Section 3). However, the soundness argument fails, as a low makespan assignment can spread jobs from U^e on multiple machines, and not give any information to recover a good labeling (this is similar to the reason we needed multiple size classes in the gap example in Section 3).

To get around this, for each e, we will use h different set systems $I^{e,1}, \ldots, I^{e,h}$ (of geometrically increasing sizes) where the s-th set system is the following $I^{e,s} = (U^e(s); A_1^e(s), \ldots, A_m^e(s))$. Each vertex will have m machines, one for each label. The intended solution is that if vertex u is assigned label a, then we pick the sets $A^e_{\pi_e(a)}(1), \ldots, A^e_{\pi_e(a)}(h)$, and assign the corresponding jobs to the m machines for u with small makespan (this requires some care so that for any label a, jobs from different classes s can be assigned to different machines, but we ignore this issue here).

Using the properties of the norm ψ and the arguments in Section 3 for the integrality gap, one can show that if $h = \Omega(m)$, then given any schedule with low makespan, one can construct a good labeling thereby proving soundness. A key new idea here beyond [15] is to show that there is some fixed size class s^*, such that the assignment of jobs in class s^* gives a small set of good candidate labels for a large fraction of edges. However, as the hardness of Label-Cover is only $\Omega(L^c)$ for some small constant c as a function of the number of labels, our resulting hardness is only $\Omega(\log^{c'} n)$ for some small $c' > 0$, instead of $\Omega(\log^{1/2} n)$.

4.1 The Reduction

Suppose that we are given a label cover instance, $\mathcal{L} = ((U, V, E), L, \Pi)$ satisfying the properties of Lemma 6, i.e., the number of vertices $|U| = |V| = N$, the degree of every vertex is $d = O((\log N)^{c_1})$ and the number of labels $L = \sqrt{\log N}$. We now describe a polynomial-time (randomized) reduction from $\mathcal{L}$ to a GMP instance $\mathcal{I}$ with machines M, jobs J and assign processing times for each machine $i \in M$ and job $j \in J$.

Machines. For each vertex $w \in U \cup V$, we create $m = L = \sqrt{\log N}$ machines. We denote the set of machines corresponding to vertex w by $M_w = \{w_1, \ldots, w_m\}$. We denote the set of all machines by $M = \bigcup_{w \in U \cup V} M_w$. In total, we have $2N\sqrt{\log N}$ machines.

Jobs. For each edge $e \in E$, we create $O(N)$ jobs and partition them into $h = m/8$ size-classes $U^e(1), U^e(2), \ldots, U^e(h)$ of geometrically increasing size. More precisely, we pick the number of jobs in the s-th set to be $|U^e(s)| = \sqrt{N} \cdot \exp(4s\sqrt{\log N})$ which is always $O(N)$ since $s \leq \sqrt{\log N}/8$. Therefore, the total number n of jobs created is poly(N). We also remark that these sets can be constructed efficiently by the randomized procedure described in Lemma 4 and that this is the only randomized step of the reduction.

Processing times. To assign processing times, for each edge $e \in E$ and $s \in [h]$, we construct a $(|U^e(s)|, m, \ell, \beta)$ set system $I^{e,s} = (U^e(s); A_1^e(s), \ldots, A_m^e(s))$ with $\ell = m/10$ and $\beta = \exp(-m)$. For every vertex $u \in U$, label $a \in [L]$ and $s \in [h]$, define the set

$$S_{u,a,s} = \bigcup_{e \in \delta(u)} A^e_{\pi_e(a)}(s). \tag{3}$$

Similarly for every vertex $v \in V$, label $b \in [L]$ and $s \in [h]$, define the set

$$S_{v,b,s} = \bigcup_{e \in \delta(v)} \overline{A^e_b}(s). \tag{4}$$

For every vertex $w \in U \cup V$, label $a \in [L]$ and size class $s \in [h]$, we choose the processing times of all the jobs in $S_{w,a,s}$ to be $p^{(s)} = \beta^{s-1}$ on the machine w_i if the index i satisfies $i \equiv (a+s) \pmod m$. This way of assigning processing times ensures that for a fixed machine w_i and a label a, there is at most one set of jobs among $S_{w,a,1}, S_{w,a,2}, \ldots, S_{w,a,h}$ that have finite processing time on w_i. This is a useful property to have when we prove the completeness of our reduction.

Norm. Define the norm $\psi = \sum_{s\in[h]} \psi^{(s)}$, where each $\psi^{(s)}$ is a scaled top-k norm given by

$$\psi^{(s)}(\mathbf{v}) = \frac{\mathsf{Top}_{\beta^2 n_s}(\mathbf{v})}{(\beta^2 n_s p^{(s)})} \tag{5}$$

where $n_s = \left|\bigcup_{e\in\delta(w)} U^e(s)\right|$ is the number of jobs of size class s contained in edges incident to any vertex. Note that since the graph is d-regular this number is the same for each vertex.

The above reduction takes poly(N) time because there are only polynomially many jobs, machines and the (n, m, ℓ, β) set systems required can be constructed efficiently by Lemma 4.

4.2 Analysis

The set of jobs $S_{w,a,s}$ for any vertex w, label a and size class s, only contribute to the s-th top-k term of ψ on machine w_i. Similar to Definition 7, we define heavy and light size classes and state a lemma whose proof is analogous to that of Lemma 8.

▶ **Definition 10** (Heavy Size Class). *For a vertex w, label a and size class s, consider the set $S_{w,a,s}$ of jobs and the machine w_i satisfying $i \equiv (a+s) \pmod m$. For an assignment $\rho : J \to M$, we say that size class s is heavy on machine w_i, if ρ assigns at least $\beta^2 n_s$ jobs from $S_{w,a,s}$ to machine w_i; otherwise, we say the size class s is light on machine w_i.*

▶ **Lemma 11.** *For any vertex w, label a, size class s, and a machine w_i satisfying $i \equiv a + s \pmod m$, the norm $\psi_{w_i}(S_{w,a,s}) = 1 + o(1)$. Furthermore, for an assignment $\rho : J \to M$, let S be the set of jobs assigned to machine w_i. Then $\psi_{w_i}(S)$ is at least the number of heavy size classes heavy on w_i.*

4.2.1 Completeness

Given a labeling σ for the label cover instance $\mathcal{L}$ which satisfies all the edges, we use it to construct an assignment of jobs ρ with a low makespan.

▶ **Lemma 12.** *If the label cover instance $\mathcal{L}$ satisfies $OPT(\mathcal{L}) = 1$, the instance $\mathcal{I}$ has an assignment $\rho : J \to M$ with makespan less than 2.*

Proof. Let σ be the labeling of vertices that satisfies all edges in the label cover instance $\mathcal{L}$. Consider the assignment ρ of jobs to machines constructed using σ in the following way: for every vertex $w \in U \cup V$, and size class $s \in [h]$, assign the jobs in $S_{w,\sigma(w),s}$ to machine w_i where the index i satisfies $i \equiv \sigma(w) + s \pmod m$.

For an edge $e = (u, v) \in E$ we first show that each job in the sets $U^e(1), U^e(2), \ldots, U^e(h)$ is assigned to some machine. For a size class $s \in [h]$, the jobs in $S_{u,\sigma(u),s}$ are assigned to a machine in M_u, and similarly, the jobs in $S_{v,\sigma(v),s}$ are assigned to a machine in M_v. Since $\pi_e(\sigma(u)) = \sigma(v)$, we infer that $A^e_{\sigma(v)}(s) = A^e_{\pi_e(\sigma(u))}(s) \subseteq S_{u,\sigma(u),s}$ and $\overline{A^e_{\sigma(v)}}(s) \subseteq S_{v,\sigma(v),s}$. It follows that each job in $U^e(s)$ is assigned to some machine.

We now bound the makespan of the assignment by showing that each machine is assigned jobs from at most one size class. For a vertex w and $i \in [m]$ consider the machine w_i. Since $h < m$, there is at most one $s \in [h]$ for which $\sigma(w) + s \equiv i \pmod m$. Therefore, w_i is assigned jobs from at most one of the sets $S_{w,\sigma(w),1}, S_{w,\sigma(w),2}, \cdots, S_{w,\sigma(w),h}$. Therefore, by Lemma 11 its norm is at most $1 + o(1) < 2$. ◀

4.2.2 Soundness

Next, we show that if the instance $\mathcal{I}$ has a makespan much less than ℓ, then $OPT(\mathcal{L})$ is large. Towards this end, we first prove some useful lemmas. Consider an assignment ρ with makespan $T \leq \ell/100$. Call a size class s to be *good* for a vertex w if it is heavy on at most $32T$ of the machines $w_1, \ldots, w_m$; if not define it to be bad. We first show that a large fraction of size classes are good for any vertex.

▶ **Lemma 13.** *There are at least $3h/4$ good size classes for each vertex.*

Proof. Let b be the number of bad size classes for some vertex w. By averaging over the m machines on vertex w, there is a machine w_i for which at least $(32Tb)/m$ size classes are heavy. By Lemma 11, w_i has norm at least $(32Tb)/m$. As any machine has norm at most T, we get $b \leq m/32 = h/4$ and hence the claim follows. ◀

Call a size class *good* for an edge (u, v) if it is good for both u and v. By Lemma 13 each edge e has at least $h/2$ good size classes. By averaging over the edges $e \in E$, there must exist a size class $s^* \in [h]$ which is good for at least $|E|/2$ edges.

We will fix the class s^* henceforth, and use it to construct a good label cover solution by assigning a suitable label to each vertex. These labels will only depend on the class s^*.

Constructing a good labeling. For each vertex $w \in U \cup V$, we define $L(w)$ to be the set of all labels a such that size-class s^* is heavy on w_i where $i \equiv (a + s) \pmod m$. If no such label exists, add an arbitrary label to $L(w)$.

▶ **Lemma 14.** *Let $e = (u, v)$ be an edge for which s^* is good. There exists $a \in L(u)$ and $b \in L(v)$ such that $\pi_e(a) = b$.*

Proof. Assume there are no such labels $a \in L(u)$ and $b \in L(v)$ for which $\pi_e(a) = b$. Since $|L(u)| + |L(v)| \leq 64T < \ell$, then the union of all the sets $A^e_{\pi_e(a)}(s^*)$ and $\overline{A^e_b}(s^*)$ such that $a \in L(u)$ and $b \in L(v)$ covers at most $(1 - \beta)|U^e(s^*)|$ from Definition 3.

From the definition of a light size class, for all labels $a \notin L(u)$ (resp. $b \notin L(v)$), at most $\beta^2 n_{s^*} = \beta^2(|U^e(s^*)| \cdot d)$ jobs from the sets $A^e_{\pi_e(a)}(s^*)$ (resp. $\overline{A^e_b}(s^*)$) are assigned to some machines on vertex u (resp. v) . Notice that the degree of the graph (U, V, E) is $d = O((\log N)^{c_1})$. So, the union of all the jobs assigned from the sets $A^e_{\pi_e(a)}(s^*)$ (resp. $\overline{A^e_b}(s^*)$) such that $a \notin L(u)$ (resp. $b \notin L(v)$) has at most $(2m \cdot \beta^2 \cdot |U^e(s^*)| \cdot d) < \beta|U^e(s^*)|$ jobs which is a contradiction. ◀

For each vertex w, set $\sigma(w)$ to be a label selected uniformly at random from $L(w)$. We show that σ satisfies a large fraction of edges of $\mathcal{L}$ completing the proof of soundness.

▶ **Lemma 15.** *If the instance $\mathcal{I}$ has an assignment $\rho : J \rightarrow M$ with makespan $T \leq \ell/100$, then $OPT(\mathcal{L}) \geq 1/(2048T^2)$.*

Proof. Consider an edge $e = (u, v)$ for which size class s^* is good. In this case, we have $|L(u)| \leq 32T$ and $|L(v)| \leq 32T$. Also, by Lemma 14 there exists $a \in L(u)$ and $b \in L(w)$ for which $\pi_e(a) = b$. Therefore, $\pi_e(\sigma(u)) = \sigma(v)$ with probability at least $1/(32T)^2$. We have by the analysis above that the number of edges for which s^* is good is at least $|E|/2$. Therefore, the expected number of edges satisfied by σ is at least $|E|/(2(32)^2T^2) = |E|/(2048T^2)$ and the lemma follows. ◀

▶ **Theorem 2.** *There is a universal constant $\delta > 0$, such that any polynomial-time approximation algorithm for GMP has approximation ratio $\Omega(\log^\delta n)$ provided that* $\mathsf{NP} \not\subseteq \mathsf{ZTIME}(n^{O(\log \log n)})$.

Proof. Suppose that we have an algorithm that in polynomial time can decide if the *GMP* instance constructed has a makespan of at least T or at most 2. By the reduction above, from Lemmas 12 and 15 this algorithm can also distinguish between label cover instances that have value 1 and those that have value at most $1/2048T^2$. Due to the hardness of label cover (Lemma 6), this is not possible if $1/2048T^2 \geq 1/(\log N)^c$, i.e., if $T \leq O((\log N)^{c/2}) = O((\log n)^{c/2})$ since the number of jobs $n = \text{poly}(N)$. This, in particular, implies that any approximation algorithm for GMP has an approximation ratio of at least $\Omega((\log n)^{c/2})$ provided that $\mathsf{NP} \not\subseteq \mathsf{ZTIME}(n^{O(\log \log n)})$. ◀

5 Concluding Remarks

We conjecture that GMP admits an $O(\sqrt{\log n})$ approximation, based on suitably rounding the configuration LP. However, we are unable to prove any $o(\log n)$ approximation even in the restricted assignment case (note that the integrality gap and hardness instances in this paper only use restricted assignment). Finding a $o(\log n)$ approximation algorithm for any of these variants would be extremely interesting.

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 Sanjeev Arora and Carsten Lund. Hardness of approximations. In *Approximation algorithms for NP-hard problems*, pages 399–446. PWS Publishing Company, Boston, 1997.

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, May 1998. doi:10.1145/278298.278306.

4 Yossi Azar and Amir Epstein. Convex programming for scheduling unrelated parallel machines. In *Symposium on Theory of Computing*, pages 331–337, 2005. doi:10.1145/1060590.1060639.

5 Deeparnab Chakrabarty and Chaitanya Swamy. Approximation algorithms for minimum norm and ordered optimization problems. In *Symposium on Theory of Computing*, pages 126–137, 2019. doi:10.1145/3313276.3316322.

6 Deeparnab Chakrabarty and Chaitanya Swamy. Simpler and better algorithms for minimum-norm load balancing. In *27th Annual European Symposium on Algorithms*, volume 144 of *LIPIcs*, pages 27:1–27:12, 2019. doi:10.4230/LIPIcs.ESA.2019.27.

7 Siu On Chan. Approximation resistance from pairwise-independent subgroups. *J. ACM*, 63(3):27:1–27:32, 2016. doi:10.1145/2873054.

8 Shichuan Deng, Jian Li, and Yuval Rabani. Generalized unrelated machine scheduling problem. In *Symposium on Discrete Algorithms (SODA)*, pages 2898–2916. SIAM, 2023.

9 Uriel Feige. A threshold of $\ln n$ for approximating set cover (preliminary version). In *Symposium on the Theory of Computing*, pages 314–318, 1996. doi:10.1145/237814.237977.

10 Thomas Holenstein. Parallel repetition: simplifications and the no-signaling case. In *Symposium on Theory of Computing*, pages 411–419, 2007.

11 Sharat Ibrahimpur and Chaitanya Swamy. Minimum-norm load balancing is (almost) as easy as minimizing makespan. In *48th International Colloquium on Automata, Languages, and Programming*, volume 198 of *LIPIcs*, pages 81:1–81:20, 2021. doi:10.4230/LIPIcs.ICALP.2021.81.

12 Subhash Khot, Dor Minzer, and Muli Safra. Pseudorandom sets in grassmann graph have near-perfect expansion. In *Symposium on Foundations of Computer Science, FOCS*, pages 592–601, 2018. doi:10.1109/FOCS.2018.00062.

13 V. S. Anil Kumar, Madhav V. Marathe, Srinivasan Parthasarathy, and Aravind Srinivasan. A unified approach to scheduling on unrelated parallel machines. *J. ACM*, 56(5):28:1–28:31, 2009. doi:10.1145/1552285.1552289.

14 Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.*, 46:259–271, 1990. doi:10.1007/BF01585745.

15 Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM (JACM)*, 41(5):960–981, 1994.

16 Konstantin Makarychev and Maxim Sviridenko. Solving optimization problems with diseconomies of scale via decoupling. *J. ACM*, 65(6):42:1–42:27, 2018. doi:10.1145/3266140.

17 Anup Rao. Parallel repetition in projection games and a concentration bound. In *Symposium on Theory of Computing*, pages 1–10, 2008.

18 Ran Raz. A parallel repetition theorem. In *Symposium on Theory of computing*, pages 447–456, 1995.

An AFPTAS for Bin Packing with Partition Matroid via a New Method for LP Rounding

Ilan Doron-Arad ✉
Computer Science Department, Technion, Haifa, Israel

Ariel Kulik ✉
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Hadas Shachnai ✉
Computer Science Department, Technion, Haifa, Israel

Abstract

We consider the Bin Packing problem with a partition matroid constraint. The input is a set of items of sizes in $[0,1]$, and a partition matroid over the items. The goal is to pack the items in a minimum number of unit-size bins, such that each bin forms an independent set in the matroid. This variant of classic Bin Packing has natural applications in secure storage on the Cloud, as well as in equitable scheduling and clustering with fairness constraints.

Our main result is an *asymptotic* fully *polynomial-time approximation scheme (AFPTAS)* for Bin Packing with a partition matroid constraint. This scheme generalizes the known AFPTAS for Bin Packing with Cardinality Constraints and improves the existing *asymptotic polynomial-time approximation scheme (APTAS)* for Group Bin Packing, which are both special cases of Bin Packing with partition matroid. We derive the scheme via a new method for rounding a (fractional) solution for a configuration-LP. Our method uses this solution to obtain *prototypes*, in which items are interpreted as placeholders for other items, and applies *fractional grouping* to modify a fractional solution (prototype) into one having desired integrality properties.

2012 ACM Subject Classification Theory of computation

Keywords and phrases bin packing, partition-matroid, AFPTAS, LP-rounding

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.22

Category APPROX

Related Version *Full Version*: https://arxiv.org/abs/2212.01025 [9]

Funding *Ariel Kulik*: This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 852780-ERC (SUBMODULAR).

1 Introduction

The *bin packing (BP)* problem involves packing a set of items in a minimum number of containers (bins) of the same (unit) size. Bin Packing is one of the most studied problems in combinatorial optimization. Indeed, in many real-life scenarios, a solution for BP is essential for optimizing the allocation of resources. In this paper, we consider the Bin Packing problem with a partition matroid constraint. The input is a set of items of sizes in $[0,1]$, and a partition matroid over the items. The goal is to pack all the items in a minimum number of unit-size bins, such that each bin forms an independent set in the matroid.

Formally, a *bin packing with partition matroid (BPP)* instance is a tuple $\mathcal{I} = (I, \mathcal{G}, s, k)$, where I is a set of items, $\mathcal{G}$ is a partition of I into groups, $s : I \to [0,1]$ gives the item sizes, and $k : \mathcal{G} \to \mathbb{N}_{>0}$ sets a cardinality constraint for each group. The *instance matroid* of $\mathcal{I}$ is the partition matroid $\mathcal{M} = (I, \mathcal{S})$ where $\mathcal{S} = \{S \subseteq I \mid \forall G \in \mathcal{G} : \; |S \cap G| \leq k(G)\}$.

A *configuration* of the instance $\mathcal{I}$ is a subset of items $C \subseteq I$ such that $\sum_{\ell \in C} s(\ell) \leq 1$ and $C \in \mathcal{S}$. That is, the total size of items in C is at most one, and for each *group* $G \in \mathcal{G}$ the configuration C contains at most $k(G)$ items from G. A *packing* of $\mathcal{I}$ is a partition of I into m subsets called *bins* $(A_1, \ldots, A_m)$ such that A_b is a configuration for all $b \in [m]$.[1] The objective is to find a packing of all items in a minimal number of bins. Indeed, the special case where each group consists of a *single* item and $k(G) = 1$ for all $G \in \mathcal{G}$ is the classic Bin Packing problem.

Bin Packing with Partition Matroid has natural application in secure storage of project data in the Cloud. Computational projects of large data scale often rely on cloud computing. Commonly, the project data is also stored in the cloud. In this setting, a main concern is that a malicious entity might gain access to confidential data [13]. To strengthen security, data is dispersed among multiple cloud storage devices [24]. Projects are fragmented into critical tasks, so that access to several tasks cannot reveal substantial information about the entire project. To this end, at most $n(P)$ tasks of each project P can be stored on a single storage device. Viewing a cloud storage device as a *bin* and each project as a *group* containing a collection of critical tasks (*items*), the problem of storing a set of projects on a minimal number of (identical) storage devices yields an instance of BPP.

BPP arises also in machine scheduling. Consider the following variant of *equitable scheduling* on a single machine [25]. The input is a set of n clients, each having a collection of jobs with arbitrary processing times, and a single machine that is available at any day for a limited amount of time. To ensure fairness, at most k jobs of the same client can be processed in a single day, for some $k \geq 1$. The objective is to complete processing all the jobs in a minimum number of days, subject to the equitability constraints. An instance of the problem can be cast as an instance of BPP, where days are the bins, jobs are the items, and the set of jobs of each client forms a group.

Another application of BPP comes from clustering problems with fairness constraints, where we have a set of items, each belongs to certain *category*, and we seek a partition of the items into clusters (or, groups) with restriction on the number of items from each category selected to each cluster, to ensure fairness. One real-life example is grouping students for educational activities, in which there is some cost associated with each participating student, and an overall budget for each group. The goal is to partition the students into a minimal number of working groups subject to the budget constraints, such that each group is balanced in terms of protected attributes like gender or race (since studies indicate that students might learn better in a diverse group). This variant of the *multi-fair capacitated grouping problem* [39] yields an instance of BPP.

Let $\text{OPT} = \text{OPT}(\mathcal{I})$ be the value of an optimal solution for an instance $\mathcal{I}$ of a minimization problem $\mathcal{P}$. As in the Bin Packing problem, we distinguish between *absolute* and *asymptotic* approximation. For $\alpha \geq 1$, we say that $\mathcal{A}$ is an absolute α-approximation algorithm for $\mathcal{P}$ if for any instance $\mathcal{I}$ of $\mathcal{P}$ it holds that $\mathcal{A}(\mathcal{I})/\text{OPT}(\mathcal{I}) \leq \alpha$, where $\mathcal{A}(\mathcal{I})$ is the value of the solution returned by $\mathcal{A}$. Algorithm $\mathcal{A}$ is an *asymptotic* α-approximation algorithm if $\mathcal{A}(\mathcal{I}) \leq \alpha\text{OPT}(\mathcal{I}) + o(\text{OPT}(\mathcal{I}))$ for any instance $\mathcal{I}$ of $\mathcal{P}$. An *asymptotic polynomial-time approximation scheme (APTAS)* is a family of algorithms $(\mathcal{A}_\varepsilon)_{\varepsilon>0}$ such that, for every $\varepsilon > 0$, $\mathcal{A}_\varepsilon$ is a polynomial-time asymptotic $(1+\varepsilon)$-approximation algorithm for $\mathcal{P}$. An *asymptotic* fully *polynomial-time approximation scheme (AFPTAS)* is an APTAS $(\mathcal{A}_\varepsilon)_{\varepsilon>0}$ such that $\mathcal{A}_\varepsilon(\mathcal{I})$ runs in time $\text{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$, where $|\mathcal{I}|$ is the encoding length of the instance $\mathcal{I}$; that is, there is a bivariate polynomial p such that $\mathcal{A}_\varepsilon(\mathcal{I})$ runs in time $p(|\mathcal{I}|, \frac{1}{\varepsilon})$ or less.

[1] For any $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, 2, \ldots, n\}$.

By known results for the special case of Bin Packing [20], BP cannot be approximated within (absolute) ratio better than $\frac{3}{2}$, unless P=NP. Thus, we focus in this paper on deriving better *asymptotic* approximation ratio for BPP. Given that classic BP admits an AFPTAS [32, 26], it is natural to ask whether the same holds for the problem with the added partition matroid constraint. We answer this question affirmatively.

▶ **Theorem 1.** *There is an* AFPTAS *for* BPP.

As a special case, our scheme improves upon the recent APTAS of Doron-Arad et al. [8] for *group bin packing (GBP)*, where the cardinality constraint of all groups is equal to one (i.e., $k(G) = 1 \; \forall G \in \mathcal{G}$). This problem has been studied since the mid-1990's [37, 31].[2] Theorem 1 also generalizes the AFTPASs of [15, 30] for *bin packing with cardinality constraints (BPCC)*, the special case of BPP where all items belong to a single group (i.e., $\mathcal{G} = \{I\}$).[3]

1.1 Our Technique

We derive our scheme via a new technique for rounding a (fractional) *configuration* LP (c-LP) solution. Our technique interprets the standard c-LP solution as a prototype for a solution which is then modified via a sequence of rounding steps; a polytope associated with the prototype is used to ensure the prototype represents a valid solution following each of the rounding steps.

High Level Approach

To obtain a packing of the input BPP instance $\mathcal{I}$, our scheme partitions the set of items I into subsets $I_1, \ldots, I_r$; for each subset of items I_j it generates a packing $A'_1, \ldots, A'_m$ of the *large* items $L_j \subseteq I_j$ (of sizes at least δ), and extends the packing (in the *packing phase*) to include the *small* items $S_j = I_j \setminus L_j$ using a greedy algorithm (see Section 2). For this procedure to work, the following crucial properties must be satisfied.

1. The size $s(\ell) \leq \delta$ of a small item $\ell \in S_j$ is much smaller than the free space $\delta \ll 1 - s(A'_b)$ in a bin A'_b.
2. There is an upper bound $k_{j,G}$ on the number of large items from each group $G \in \mathcal{G}$ in any bin A'_b.
3. For each group $G \in \mathcal{G}$, there cannot be "too many" small items from G in S_j.

Figure 1 illustrates the packing of items in I_j. By the above properties, a main part of our scheme deals with generation of a partition of I into $I_1, \ldots, I_r$ and the corresponding partial packings.[4]

Let $\mathcal{C}$ denote the set of configurations of a BPP instance $\mathcal{I} = (I, \mathcal{G}, s, k)$. Our rounding technique interprets the vectors $\bar{x} \in \mathbb{R}^{\mathcal{C}}_{\geq 0}$ (i.e., vectors having a non-negative entry for each configuration $C \in \mathcal{C}$) as *prototypes*. The prototype $\bar{x} \in \mathbb{R}^{\mathcal{C}}_{\geq 0}$ serves as a blueprint for a (fractional) packing. Within the context of prototypes, each item $\ell \in C$ of a configuration C is interpreted as a placeholder (or, "slot") for items which can replace it. Also, some items may be added, thus utilizing the available capacity of the configuration C (given by $1 - s(C)$), while the items replacing the slots utilize the capacity $s(C)$ of the configuration. The value $\bar{x}_C$ represents a solution in which $\bar{x}_C$ configurations match the blueprint corresponding to C.

[2] See Table 1 for a summary of known results for GBP.
[3] An outline of the organization of this paper is given in Section 1.3.
[4] See the details in Section 3.

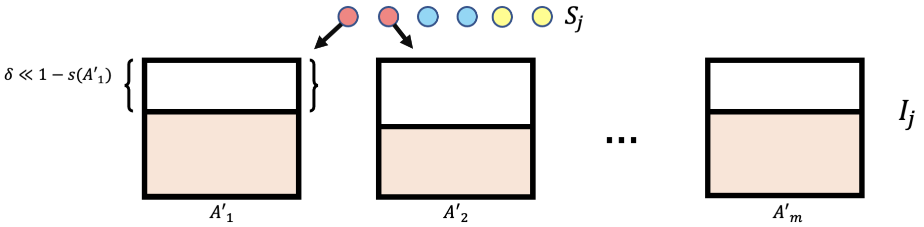


Figure 1 The set I_j in the partition $I_1, \dots, I_r$, along with a partial packing $A'_1, \dots, A'_m$ of the large items, L_j; the set S_j of remaining small items is represented by circles, whose colors indicate the groups. The items in S_j are added using a greedy algorithm.

A main building block in our scheme is an association of a polytope with each prototype. A non-empty polytope indicates the feasibility of a prototype. The partition of I into $I_1, \dots, I_r$ and S, and the generation of the partial packings $(A'_1, \dots, A'_m)$ relies on integrality properties of vertices in the polytope (associated with a prototype). However, for the integrality properties to be useful, we must obtain a *good prototype* $\bar{z}$, having a small number of configurations on the support (i.e, $\{C \in \mathcal{C} \mid \bar{z}_C > 0\}$), while the number of items in each configuration on the support must be small. We give below a high level description of how we construct a good prototype.

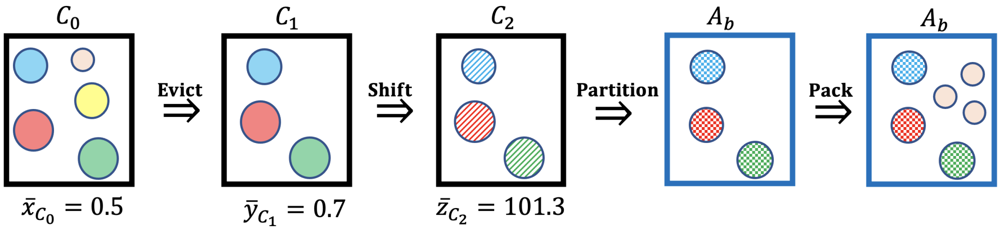


Figure 2 An illustration of the rounding process. Starting from a prototype $\bar{x}$, we follow a configuration C_0. The eviction process generates a prototype $\bar{y}$, where the value $\bar{x}_{C_0}$ is added to $\bar{y}_{C_1}$. In the shifting phase, $\bar{y}$ is transformed into a *good* prototype $\bar{z}$ in which items are replaced by a small number of representatives, and the entry $\bar{y}_{C_1}$ is added to $\bar{z}_{C_2}$. The partition phase uses the prototype $\bar{z}$ to define a partition $I_1, \dots, I_r$ of the items, and an initial packing $A_1, \dots, A_m$ for each I_j (containing the bin A_b in our example). Finally, in the packing phase, the remaining small items are added to the existing bins.

An initial prototype is obtained by solving a standard configuration-LP formulation of the problem. In this prototype, each item serves as a placeholder for itself. The algorithm modifies the prototype sequentially using two steps: *eviction* and *shifting*. The eviction step reduces the number of items per configuration on the support of the prototype. The shifting step reduces the overall number of distinct items used by configurations on the support. Thus, after the shifting, an item (or, a *slot*) may be used as a placeholder for many other items in the same group. Our construction in this step is non-trivial. Specifically, we use the *fractional grouping* technique introduced in [16] *constructively* (see the full version of the paper [9]). By the above, the number of configurations on the support of the constructed prototype is small, and our scheme can easily find a packing of the instance. We illustrate the main components of our multi-step rounding process in Figure 2.

Our technique may be useful in solving other packing problems, including, e.g., *bin packing with matching constraints* (see Section 4).

1.2 Prior Work

The special case of Group Bin Packing (GBP) (in which $k(G) = 1$ for all $G \in \mathcal{G}$) was first studied by Oh and Son [37]. This problem is a special case of *bin packing with conflicts (BPC)*, where the conflict graph is a *cluster graph* (see, e.g., [11] for a survey on recent results on BPC). An approximation ratio of 2.5 follows from the results of Jansen and Öhring [31] (for a generalization of GBP); this ratio was later improved by Epstein and Levin [14]. Better constants were given in several papers (e.g., [36, 1]). The best known asymptotic approximation for GBP prior to our work is an APTAS due to Doron-Arad et al. [8]. We summarize the known results for GBP in Table 1.

Table 1 Known Results for the special case of Group Bin Packing; $V(\mathcal{I}) = \max_{G \in \mathcal{G}} |G|$. The results of [31, 14, 1] for GBP follow as a special case from results for more general problems.

Authors	Year	Approximation
Oh and Son [37]	1995	$1.7 \cdot \text{OPT}(\mathcal{I}) + 2.19 \cdot V(\mathcal{I})$
Jansen and Öhring [31]	1997	$2.5 \cdot \text{OPT}(\mathcal{I})$
McCloskey and Shankar [36]	2005	$2 \cdot \text{OPT}(\mathcal{I}) + V(\mathcal{I})$
Epstein and Levin [14]	2008	$\frac{7}{3} \cdot \text{OPT}(\mathcal{I})$
Adany et al. [1]	2013	$2 \cdot \text{OPT}(\mathcal{I})$
Doron-Arad et al. [8]	2021	APTAS
This paper	2023	**AFPTAS**

The APTAS of [8] for GBP is based on extensive guessing of properties of an optimal solution which are then used as guidance for the assignment of items to bins; such enumeration is commonly used in studies of BP (e.g., [3]). We note that the algorithm of [8] does not round a solution for a configuration-LP, and cannot be viewed as a rounding algorithm in general. The extensive guessing leads to running time that is exponential in $\frac{1}{\varepsilon}$. For a special case of GBP, where the maximum cardinality of a group is some constant, an AFPTAS follows from a result of [27].

GBP was studied also in the context of scheduling on identical machines. Das and Wiese [6] introduced the problem of *makespan minimization with bag constraints*. In this generalization of the classic makespan minimization problem, each job belongs to a *bag*. The goal is to schedule the jobs on a set of m identical machines, for some $m \geq 1$, such that no two jobs in the same bag are assigned to the same machine, and the makespan is minimized. Das and Wiese [6] developed a PTAS for the problem with bag constraints. Later, Grage et al. [21] obtained an EPTAS.

Another special case of BPP is Bin Packing with Cardinality Constraints (BPCC), in which $|\mathcal{G}| = 1$, i.e., we have a single group G with $k(G) = k$, for some integer $k \geq 1$. BPCC has been studied since the 1970's [34, 35, 33, 4]. Epstein and Levin [15] presented an AFPTAS which relies on rounding a non-standard configuration-LP formulation of the problem. Later, Jansen et al. [30] gave an AFPTAS with improved additive term. The AFPTASs of [15, 30] rely on the property that if $k < \frac{1}{\varepsilon}$ then only $\frac{1}{\varepsilon}$ items fit into a bin; thus, linear shifting can be applied to the whole instance. We note that in a BPP instance (which consists of multiple groups) the cardinality bound for each group may be small (or even equal to 1), yet the number of items that can be packed in a single bin may be arbitrarily large. This is one of several hurdles encountered when attempting to adapt the techniques of [15, 30] to our setting of BP with a partition matroid constraint.

In the *matroid partitioning* problem, we are given a ground set U and a matroid $\mathcal{M} = (U, \mathcal{S})$, where $\mathcal{S}$ is a family of subsets of U, known as the *independent sets* of the matroid. We seek a partition of U into as few independent sets as possible. The problem is polynomially solvable for any matroid $\mathcal{M}$ over U using a combinatorial algorithm (see, e.g., [12, 19]). When $\mathcal{M}$ is a partition matroid, the matroid partitioning problem can be viewed as a variant of BPP with unbounded bin capacities.

The use of configuration-LP (c-LP) in approximation algorithms started in the seminal paper of Karmakar and Karp on BP [32]. Their approach is to round the item sizes prior to solving a c-LP. Similar approaches, in which item sizes are rounded or the instance is restructured prior to solving the c-LP, can be found also in later works (e.g., [15]). Other works obtain an integral solution by applying randomized rounding to the solution of a standard c-LP. This includes the *Round & Approx* technique of Bansal et al. [2] and the tight approximation for the *separable assignment problem (SAP)* due to Fleischer et al. [18]. In [28] and [29], Jansen combines techniques for *2D strip packing* to round the solution of c-LP for multiple knapsack. Our deterministic rounding technique deviates significantly from these known approaches.

To the best of our knowledge, Bin Packing with Partition Matroid is studied here for the first time.

1.3 Organization

In Section 2 we include some definitions and notation. Section 3 gives an overview of our scheme, that is applied to a *structured* instance. Due to space constraints, the technical sections along with most of the proofs are relegated to the full version of the paper [9]. In Section 4 we give a summary and directions for future work.

2 Preliminaries

Let $\mathcal{A}$ be an algorithm that accepts as input $\varepsilon > 0$. We say the running time of $\mathcal{A}$ is poly$(|\mathcal{I}|, \frac{1}{\varepsilon})$ if there is a two-variable polynomial $p(x, y)$ such that $p(|\mathcal{I}|, \frac{1}{\varepsilon})$ is an upper bound on the running time of $\mathcal{A}(\mathcal{I}, \varepsilon)$. To allow a simpler presentation of the results, we assume throughout the paper that the set of items I is $\{1, 2, \ldots, n\}$, and the items are sorted in non-increasing order by sizes, i.e., $s(1) \geq s(2) \geq \ldots s(n)$.

2.1 Tackling the Small Items

The classic asymptotic approximation schemes for Bin Packing (see, e.g., [17, 32]) rely on the key property that *small* items can be added to a partial packing of the instance with little overhead, using simple algorithms such as First-Fit (see, e.g., [40]). We note that packing small items in the presence of a partition matroid constraint is more involved. Even for the special case of BP with a cardinality constraint, the small items cannot be packed using simple classic BP heuristics (see, e.g., [15]).

We show below that an efficient packing of the small items of a BPP instance can still be found using a relatively simple algorithm; however, the setting in which the algorithm can be applied is more restrictive, and items cannot be easily added to a partial packing of the instance (i.e., a set of configurations). Furthermore, the quality of such a packing depends on the *cardinality bound* of the BPP instance $\mathcal{I} = (I, \mathcal{G}, s, k)$, defined by $V(\mathcal{I}) = \max_{G \in \mathcal{G}} \left\lceil \frac{|G|}{|k(G)|} \right\rceil$. Formally,

▶ **Lemma 2.** *Given a* BPP *instance $\mathcal{I} = (I, \mathcal{G}, s, k)$ and $\delta \in (0, 0.5)$, such that $s(\ell) \leq \delta$ for all $\ell \in I$, there is an algorithm* Greedy *that returns in polynomial time a packing of $\mathcal{I}$ in at most $(1 + 2\delta) \cdot \max\{s(I),\ V(\mathcal{I})\} + 2$ bins.*

The proof of Lemma 2 is given in [9].

2.2 Structuring the Instance

Our scheme initially transforms a given BPP instance into one having a structure which depends on the parameter $\varepsilon > 0$. In this new instance, only a small number of groups may contain relatively large items. Let $K : (0, 0.1) \to \mathbb{R}$, where $K(\varepsilon) = \varepsilon^{-\varepsilon^{-2}}$ for all $\varepsilon \in (0, 0.1)$. We use K for defining a structured instance.

▶ **Definition 3.** *Given a BPP instance $\mathcal{I} = (I, \mathcal{G}, s, k)$ and $\varepsilon \in (0, 0.1)$, we say that $\mathcal{I}$ is* ε-structured *if there is $\mathcal{B} \subseteq \mathcal{G}$ such that $|\mathcal{B}| \leq K(\varepsilon)$ and for all $G \in \mathcal{G} \setminus \mathcal{B}$ and $\ell \in G$ it holds that $s(\ell) < \varepsilon^2$.*

Following the structuring step, our scheme proceeds to solve BPP on the structured instance. As a final step, the packing found for the structured instance is transformed into a packing of the original instance. This is formalized in the next result.

▶ **Lemma 4.** *There is a pair of algorithms,* Reduce *and* Reconstruct, *which satisfy the following.*

1. *Given a* BPP *instance $\mathcal{J}$ and $\varepsilon > 0$ such that $\varepsilon^{-1} \in \mathbb{N}$, algorithm* Reduce *returns in time* $\text{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ *an ε-structured BPP instance $\mathcal{I}$, where* $\text{OPT}(\mathcal{I}) \leq \text{OPT}(\mathcal{J})$.
2. *Given a* BPP *instance $\mathcal{J}$, $\varepsilon > 0$ such that $\varepsilon^{-1} \in \mathbb{N}$, and a packing A' for $\mathcal{I} =$* Reduce$(\mathcal{J}, \varepsilon)$ *of size m', algorithm* Reconstruct *returns in time* $\text{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ *a packing A for the instance $\mathcal{J}$ of size m, where $m \leq m' + 13\varepsilon \cdot \text{OPT}(\mathcal{J}) + 1$.*

The structured instance $\mathcal{I}$ is obtained from $\mathcal{J}$ by reassigning items of size at least ε^2 from all but a few groups to a new group. The reconstruction algorithm modifies the packing of $\mathcal{I}$ such that each bin in the solution is a configuration of $\mathcal{J}$. The proof of Lemma 4 (given in [9]) is inspired by ideas of [6, 21, 8]. By Lemma 4, an AFTPAS for ε-structured BPP instances implies an AFTPAS for general BPP instances.

3 Approximation Algorithm for ε-Structured Instances

In this section we give an overview of our scheme. For $\varepsilon \in (0, 0.1)$ such that $\varepsilon^{-1} \in \mathbb{N}$, let $\mathcal{I} = (I, \mathcal{G}, s, k)$ be an ε-structured BPP instance. Recall that a *configuration* of $\mathcal{I}$ is a subset of items $C \subseteq I$ such that $\sum_{\ell \in C} s(\ell) \leq 1$, and $|C \cap G| \leq k(G)$ for all $G \in \mathcal{G}$. Let $\mathcal{C}(\mathcal{I})$ be the set of all configurations of $\mathcal{I}$; we use $\mathcal{C}$ when the instance $\mathcal{I}$ is clear from the context. Also, for every item $\ell \in I$ let $\mathcal{C}[\ell] = \{C \in \mathcal{C} \mid \ell \in C\}$ be the set of configurations of $\mathcal{I}$ that contain ℓ. A key component in our scheme is the construction of a *prototype* of a packing; a prototype gives a non-negative value to each configuration, which (informally) indicates the selection of the configuration. Specifically,

▶ **Definition 5.** *Given a* BPP *instance $\mathcal{I}$, a* prototype *is a vector $\bar{x} \in \mathbb{R}^{\mathcal{C}}_{\geq 0}$.*

In the context of prototypes, each configuration $C \in \mathcal{C}$ is interpreted as a set of placeholders called *types*. Each item $j \in C$ is a *slot-type*, which is a placeholder for a smaller or equally sized item of the same group. Also, the unused capacity of the configuration C (i.e., $1 - s(C)$) serves as a placeholder for additional items; we refer to this placeholder as the *configuration-type* C. Thus, C is interpreted as the set of slot-types of j for each $j \in C$ and the configuration-type of C itself.

Intuitively, a slot-type (configuration-type) can be replaced by an item (items) which *fit* into it. For any $j \in I$ define $\mathsf{group}(j) = G$, where $G \in \mathcal{G}$ is the unique group such that $j \in G$. The subset of items that fit in place of the slot-type $j \in I$ is

$$\mathsf{fit}(j) = \{\ell \in \mathsf{group}(j) \mid s(\ell) \leq s(j)\} \qquad \forall j \in I. \tag{1}$$

For our algorithm to work, the items which fit into the configuration-type $C \in \mathcal{C}$ must be *small* relative to the unused capacity of C. We define the subset of items that fit into the configuration-type $C \in \mathcal{C}$ by

$$\mathsf{fit}(C) = \{\ell \in I \mid s(\ell) \leq \min\{\varepsilon^2, \varepsilon \cdot (1 - s(C))\}\} \qquad \forall C \in \mathcal{C}. \tag{2}$$

In words, $\mathsf{fit}(C)$ contains items of sizes smaller than ε^2 and also at most ε-fraction of the unused capacity of C. Figure 3 illustrates the above definitions.

For any prototype $\bar{x}$ we define the *$\bar{x}$-polytope* as a set of fractional packings, in which items are fractionally assigned to slot-types and configuration-types. The set of *types* of the instance $\mathcal{I}$ is $I \cup \mathcal{C}$. That is, the slot-types I and configuration-types $\mathcal{C}$. A point in the $\bar{x}$-polytope has an entry for each pair of an item $\ell \in I$ and a type $t \in I \cup \mathcal{C}$ which represents the fractional *assignment* of the item to the type. Formally,

▶ **Definition 6.** *Given a* BPP *instance $\mathcal{I}$, the set $\mathcal{C}$ of configurations for $\mathcal{I}$, and a prototype $\bar{x}$ of $\mathcal{I}$, the $\bar{x}$-polytope is the set containing all points $\bar{\gamma} \in [0,1]^{I \times (I \cup \mathcal{C})}$ which satisfy the following constraints.*

$$\bar{\gamma}_{\ell,t} = 0 \qquad \forall \ell \in I, t \in I \cup \mathcal{C} \text{ s.t. } \ell \notin \mathsf{fit}\,(t) \tag{3}$$

$$\sum_{\ell \in I} \bar{\gamma}_{\ell,C} \cdot s(\ell) \leq (1 - s(C)) \cdot \bar{x}_C \qquad \forall C \in \mathcal{C} \tag{4}$$

$$\sum_{\ell \in G} \bar{\gamma}_{\ell,C} \leq \bar{x}_C \cdot (k(G) - |C \cap G|) \qquad \forall G \in \mathcal{G}, C \in \mathcal{C} \tag{5}$$

$$\sum_{\ell \in I} \bar{\gamma}_{\ell,j} \leq \sum_{C \in \mathcal{C}[j]} \bar{x}_C \qquad \forall j \in I \tag{6}$$

$$\sum_{t \in I \cup \mathcal{C}} \bar{\gamma}_{\ell,t} \geq 1 \qquad \forall \ell \in I \tag{7}$$

Constraints (3) indicate that an item $\ell \in I$ cannot be assigned to type t if ℓ does not fit in t. Constraints (4) set an upper bound on the total (fractional) size of items assigned to each configuration-type $C \in \mathcal{C}$. This bound is equal to the residual capacity of C times the fractional number of bins packed with C, given by $\bar{x}_C$. Constraints (5) bound the number of items in each group G assigned to configuration-type C; at most $k(G) - |C \cap G|$ items in G can be added to C without violating the cardinality constraint of G. Constraints (6) bound the number of items assigned to slot-type $j \in I$ by the total selection of configurations containing j in $\bar{x}$. Finally, constraints (7) guarantee that each item is fully assigned to the types.

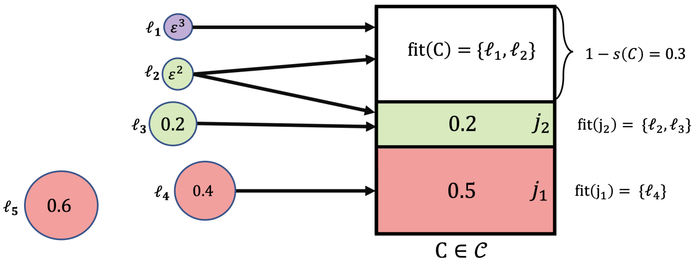


Figure 3 An example of items $\ell_1, \ldots, \ell_5$ of sizes $\varepsilon^3, \varepsilon^2, 0.2, 0.4, 0.6$, respectively, along with a configuration $C = \{j_1, j_2\} \in \mathcal{C}$ interpreted as the slot-types j_1, j_2 of sizes 0.5, 0.2 and the configuration-type C. The colors of the items and slot-types indicate their corresponding groups. An arrow indicates that the item on the left fits with a slot-type or with the configuration-type C.

Let $\mathsf{supp}(\bar{x}) = \{C \in \mathcal{C} \mid \bar{x}_C > 0\}$ be the *support* of $\bar{x}$. Throughout this paper, we use prototypes $\bar{x}$ for which $\mathsf{supp}(\bar{x})$ is polynomial in the input size; thus, these prototypes have sparse representations. Our scheme first converts an initial prototype $\bar{x}$ (defined by a solution for the configuration-LP) into a *good* prototype $\bar{z}$ as defined below, and then constructs a packing based on $\bar{z}$. Let $Q : (0, 0.1) \rightarrow \mathbb{R}$ where $Q(\varepsilon) = \exp(\varepsilon^{-17})$ for all $\varepsilon \in (0, 0.1)$. Then,

▶ **Definition 7.** *Given $\varepsilon \in (0, 0.1)$ and an ε-structured* BPP *instance $\mathcal{I}$, a prototype $\bar{z}$ of $\mathcal{I}$ is a* good prototype *if the $\bar{z}$-polytope is non-empty, $|\mathsf{supp}(\bar{z})| \leq Q(\varepsilon)$, and $|C| \leq \varepsilon^{-10}$ for all $C \in \mathsf{supp}(\bar{z})$.*

To construct a good prototype, our algorithm first finds an initial prototype $\bar{x}$ using a Configuration-LP of the problem.[5] However, $\bar{x}$ is not necessarily a good prototype, since it may have a support of large size. Therefore, we apply a non-trivial rounding process, starting from the initial prototype $\bar{x}$, and eventually generate a good prototype $\bar{z}$. We consider this rounding process, along with the notions of prototype and $\bar{x}$-polytope, the core technical contribution of this paper. The next result summarizes the main properties of our algorithm for finding a good prototype, presented in Section 3.1. We use $\|\bar{x}\| = \sum_{C \in \mathcal{C}} \bar{x}_C$ to denote the ℓ_1-norm of a prototype $\bar{x}$.

▶ **Lemma 8.** *There is an algorithm* Prototype *that given $\varepsilon \in (0, 0.1)$ and an ε-structured* BPP *instance $\mathcal{I}$, returns in time* $\text{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ *a good prototype $\bar{z}$ of $\mathcal{I}$ such that $\|\bar{z}\| \leq (1 + 19\varepsilon) \cdot \text{OPT}(\mathcal{I}) + Q(\varepsilon)$.*

Given a good prototype $\bar{z}$, our scheme finds an efficient packing of the instance. This phase relies on integrality properties of the $\bar{z}$-polytope, combined with a matching-based algorithm and a greedy assignment of relatively small items using Algorithm Greedy (Lemma 2).

▶ **Lemma 9.** *There is an algorithm* Solution *that given $\varepsilon \in (0, 0.1)$, an ε-structured* BPP *instance $\mathcal{I}$, and a good prototype $\bar{z}$ of $\mathcal{I}$, returns in time* $\text{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ *a packing of $\mathcal{I}$ in at most $(1 + 2\varepsilon) \cdot \|\bar{z}\| + 5\varepsilon^{-22} \cdot Q^2(\varepsilon)$ bins.*

The proof of Lemma 9 is given in Section 3.2. Using the above components, we obtain an AFPTAS for ε-structured instances. The pseudocode of the scheme is given in Algorithm 1.

▶ **Lemma 10.** *Given $\varepsilon \in (0, 0.1)$, $\varepsilon^{-1} \in \mathbb{N}$, and an ε-structured* BPP *instance $\mathcal{I}$, Algorithm 1 returns in time* $\text{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ *a packing of $\mathcal{I}$ of size at most $(1 + 60\varepsilon) \cdot \text{OPT}(\mathcal{I}) + (Q(\varepsilon))^3$.*

[5] See Section 3.1.

Algorithm 1 AFPTAS$(\mathcal{I}, \varepsilon)$.

```
Input   : An $\varepsilon$-structured instance $\mathcal{I}$ and $\varepsilon \in (0, 0.1)$ such that $\varepsilon^{-1} \in \mathbb{N}$
Output  : A packing of $\mathcal{I}$
1 Find a good prototype $\bar{z} = \mathsf{Prototype}(\mathcal{I}, \varepsilon)$.
2 Return a packing $\Phi = \mathsf{Solution}(\mathcal{I}, \varepsilon, \bar{z})$.
```

Lemma 10 follows from Lemmas 8 and 9. Theorem 1 can be easily derived using Lemmas 10 and 4. We give the full proofs in [9].

3.1 Algorithm Prototype

In this section we present Algorithm Prototype which finds a good prototype for a given ε-structured instance. The algorithm uses an LP relaxation of the given BPP instance. For $\varepsilon \in (0, 0.1)$ such that $\varepsilon^{-1} \in \mathbb{N}$, let $\mathcal{I} = (I, \mathcal{G}, s, k)$ be an ε-structured BPP instance. Define the *configuration-LP* of $\mathcal{I}$ as:

$$\begin{array}{lll} \min & \sum_{C \in \mathcal{C}} \bar{x}_C & \\ \text{s.t.} & \sum_{C \in \mathcal{C}[\ell]} \bar{x}_C = 1 & \forall \ell \in I \\ & \bar{x}_C \geq 0 & \forall C \in \mathcal{C} \end{array} \tag{8}$$

A solution for the LP (8) assigns to each configuration $C \in \mathcal{C}$ a real number $\bar{x}_C \in [0, 1]$ which indicates the fractional selection of C for the solution such that each item is fully *covered*. Observe that a solution for (8) is in particular a prototype of $\mathcal{I}$.

Note that the configuration-LP (8) has an exponential number of variables; thus, it cannot be solved in polynomial time by applying standard techniques. A common approach for solving such linear programs is to use a *separation oracle* for the dual program.

Consider the *configuration maximization problem (CMP)* in which we are given a BPP instance $\mathcal{I} = (I, \mathcal{G}, s, k)$ and a weight function $w : I \to \mathbb{R}_{\geq 0}$; the objective is to find a configuration $C \in \mathcal{C}$ such that $\sum_{\ell \in C} w(\ell)$ is maximized. By a well known connection between separation and optimization, an FPTAS for CMP implies an FPTAS for the configuration-LP of $\mathcal{I}$ [22, 18, 23, 38]. CMP can be solved via an easy reduction to *knapsack with partition matroid*, that is known to admit an FPTAS [10]. Thus, we have

▶ **Lemma 11.** *There is an algorithm* SolveLP *that given a* BPP *instance* $\mathcal{I}$ *and* $\varepsilon > 0$*, returns in time* poly$(|\mathcal{I}|, \frac{1}{\varepsilon})$ *a solution for the configuration*-LP *of* $\mathcal{I}$ *of value at most* $(1 + \varepsilon)\text{OPT}$*, where* OPT *is the value of an optimal solution for the configuration*-LP *of* $\mathcal{I}$*.*

We give the proof of Lemma 11 in [9]. A solution $\bar{x}$ for the Configuration-LP (8) is a prototype of the instance such that the $\bar{x}$-polytope is non-empty; in particular, it contains the point $\bar{\gamma}$ where $\bar{\gamma}_{\ell,j} = 1 \; \forall \ell, j \in I$ such that $\ell = j$, and $\bar{\gamma}_{\ell,t} = 0$ otherwise (note that this property does not hold for the good prototype $\bar{z}$ returned by Algorithm Prototype). Our intermediate goal is an *evicted prototype* $\bar{y}$, having configurations of bounded cardinality on its support, but with properties similar to those of solutions for (8). We say that an item $\ell \in I$ is ε-large if $s(\ell) \geq \varepsilon^2$. We use $L(\varepsilon, \mathcal{I})$ to denote the set of ε-large items of an instance $\mathcal{I}$. If $\mathcal{I}$ and ε are known by context we simply use L (instead of $L(\varepsilon, \mathcal{I})$). We now formalize the above.

▶ **Definition 12.** *Let $\varepsilon \in (0, 0.1)$ such that $\varepsilon^{-1} \in \mathbb{N}$ and $\mathcal{I}$ an ε-structured BPP instance. A prototype $\bar{y}$ of $\mathcal{I}$ is called an* evicted prototype *if the following holds.*

1. *For all $C \in \mathsf{supp}(\bar{y})$ it holds that $|C| \leq \varepsilon^{-10}$ and $s(C \setminus L) \leq \varepsilon$.*
2. *There exists $\bar{\gamma}$ in the $\bar{y}$-polytope such that $\bar{\gamma}_{\ell,j} = 0$ for all $\ell, j \in I$ where $\ell \neq j$.*
3. *$\sum_{C \in \mathcal{C}[\ell]} \bar{y}_C \leq 2$ for every $\ell \in I$.*

Given a solution $\bar{x}$ for the configuration-LP (8), we construct an evicted prototype $\bar{y}$ with $\|\bar{y}\| \approx \|\bar{x}\|$. Our technique fractionally maps each configuration $C \in \mathsf{supp}(\bar{x})$ to other configurations, where relatively small items are discarded in the mapping. To show that the $\bar{y}$-polytope is non-empty, we generate a point in the $\bar{y}$-polytope by assigning (fractionally) discarded items to configuration-types (see Definition 6). The result of this process is outlined in the next lemma.

▶ **Lemma 13.** *There is an algorithm* Evict *which given $\varepsilon \in (0, 0.1)$ such that $\varepsilon^{-1} \in \mathbb{N}$, an ε-structured* BPP *instance $\mathcal{I}$, and a solution $\bar{x}$ for the configuration-LP* (8)*, returns in time* $\text{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ *an evicted prototype $\bar{y}$ such that $\|\bar{y}\| \leq (1 + \varepsilon)\|\bar{x}\|$.*

A complete presentation of algorithm Evict and the proof of Lemma 13 are given in [9]. Observe that property 2 of Definition 12 allows $\bar{\gamma}_{\ell,C} > 0$ for item $\ell \in I$ and a configuration-type $C \in \mathcal{C}$; the property that $\bar{\gamma}_{\ell,j} > 0$ for $\ell \neq j$ is obtained only in the next step. Moreover, note that Evict does not return the vector $\bar{\gamma}$ but only guarantees its existence.

Given an evicted prototype $\bar{y}$, our scheme uses algorithm Shift to generate a good prototype $\bar{z}$ with $\|\bar{z}\| \approx \|\bar{y}\|$. As in algorithm Evict, we rely on a fractional mapping between configurations to construct $\bar{z}$; here, our goal is to significantly decrease $|\mathsf{supp}(\bar{z})|$ w.r.t. $|\mathsf{supp}(\bar{y})|$ while keeping the $\bar{z}$-polytope non-empty. One key observation utilizes combinatorial properties of the instance to show that items from most groups can be discarded in the mapping (the items may be assigned to configuration-types in the $\bar{z}$-polytope). Items of the remaining groups are mapped to a small number of *representatives*, using a non-trivial application of fractional grouping.

▶ **Lemma 14.** *There is an Algorithm* Shift *that given $\varepsilon \in (0, 0.1)$ such that $\varepsilon^{-1} \in \mathbb{N}$, an ε-structured* BPP *instance $\mathcal{I}$ and an evicted prototype $\bar{y}$, returns in time* $\text{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ *a good prototype $\bar{z}$ such that $\|\bar{z}\| \leq (1 + 5\varepsilon)\|\bar{y}\| + Q(\varepsilon)$.*

An elaborate presentation of Algorithm Shift and the proof of Lemma 14 are given in [9]. Finally, Algorithm Prototype finds a good prototype by computing Algorithms SolveLP, Evict, and Shift sequentially. We give the pseudocode in Algorithm 2. The proof of Lemma 8 easily follows from Lemmas 11, 13, and 14; we give the proof in [9].

Algorithm 2 Prototype$(\mathcal{I}, \varepsilon)$.

```
Input  : An $\varepsilon$-structured instance $\mathcal{I}$ and $\varepsilon \in (0, 0.1)$ such that $\varepsilon^{-1} \in \mathbb{N}$
Output : A good prototype
Find a solution for the configuration-LP of $\mathcal{I}$; that is, $\bar{x} = \mathsf{SolveLP}(\mathcal{I}, \varepsilon)$.
Find an evicted prototype $\bar{y} = \mathsf{Evict}(\mathcal{I}, \bar{x}, \varepsilon)$.
Return a good prototype $\bar{z} = \mathsf{Shift}(\mathcal{I}, \bar{y}, \varepsilon)$.
```

3.2 Algorithm Solution

In this section we show how integrality properties of a good prototype yield an efficient packing of the instance. For $\varepsilon \in (0, 0.1)$ such that $\varepsilon^{-1} \in \mathbb{N}$, let $\mathcal{I} = (I, \mathcal{G}, s, k)$ be an ε-structured BPP instance. The next lemma shows that if a prototype $\bar{x}$ has a small support, and each configuration in the support contains a few items (as for good prototypes), then the vertices of the $\bar{x}$-polytope are almost integral. Thus, given a vertex $\bar{\lambda}$ of such $\bar{x}$-polytope, the items assigned fractionally by $\bar{\lambda}$ can be packed using only a small number of extra bins.

▶ **Lemma 15.** *Let $\mathcal{I}$ be a* BPP *instance, $k \in \mathbb{N}_{\geq 1}$, and a prototype $\bar{x}$ of $\mathcal{I}$ such that for all $C \in \mathsf{supp}(\bar{x})$ it holds that $|C| \leq k$ and $\bar{x}_C \in \mathbb{N}$. Then, for any vertex $\bar{\lambda}$ in the $\bar{x}$-polytope for which constraints* (7) *hold with equality,*

$$\left|\left\{\ell \in I \mid \exists t \in I \cup \mathcal{C} \text{ s.t. } \bar{\lambda}_{\ell,t} \in (0,1)\right\}\right| \leq 8k^2 \cdot |\mathsf{supp}(\bar{x})|^2.$$

The proof of Lemma 15 bears some similarity to a proof of [7], which shows the integrality properties of a somewhat different polytope. We give the detailed proof in [9].

Given a good prototype $\bar{z}$, our scheme finds a packing of the instance using a partition of the items into slot-types and configuration-types; intuitively, items assigned to slot-types are already packed (using the integrality property of the $\bar{z}$-polytope) and items assigned to configuration-types will be added using Greedy. This relies on the following construction.

For configurations $S, C \in \mathcal{C}$, we say that S is *allowed* in C if each item $\ell \in S$ can be mapped to a distinct slot $j \in C$, such that $\ell \in \mathsf{fit}(j)$. We consider packings of a subset of items to which we call a *category.* Each category is associated with (i) a configuration $C \in \mathcal{C}$ such that all bins in the category are allowed in C, and (ii) a *completion*: a subset of (unpacked) items bounded by total size and number of items per group, where each item fits with C. This is formalized in the next definition.

▶ **Definition 16.** *Let $\varepsilon \in (0, 0.1)$, an ε-structured BPP instance $\mathcal{I} = (I, \mathcal{G}, s, k)$, a configuration $C \in \mathcal{C}$, a packing $B = (B_1, \ldots, B_m)$ of a subset of I (category), and $D \subseteq I$ (completion). We say that B is a category of C and D if the following holds.*

- *For any $i \in [m]$, B_i is allowed in C.*
- *$D \subseteq \mathsf{fit}(C)$.*
- *$s(D) \leq (1 - s(C)) \cdot m$.*
- *For any $G \in \mathcal{G}$ it holds that $|D \cap G| \leq m \cdot (k(G) - |C \cap G|)$.*

The motivation behind this construction, is that Algorithm Greedy (Lemma 2) can be used to assign the completion to the existing bins of the category using only a small number of extra bins. Thus, our end-goal from the good prototype $\bar{z}$ is to obtain an *ε-nice partition*: a packing of a subset of I such that the bins in the packing are partitioned into a bounded number of categories; also, we require that each item $\ell \in I$ is either in this packing or in a completion of a category. The above constraints are analogous to constraints (3)-(7) of the $\bar{z}$-polytope, that is used for finding an assignment of the items to slots and configurations. This is formalized in Definition 17. An example is given in Figure 4.

▶ **Definition 17.** *Given $\varepsilon \in (0, 0.1)$, an ε-structured BPP instance $\mathcal{I} = (I, \mathcal{G}, s, k)$, an ε-*nice partition *$\mathcal{B}$ of $\mathcal{I}$ is a packing $(A_1, ..., A_m)$ of a subset of I, configurations $\mathcal{H} \subseteq \mathcal{C}$, categories $(B_C)_{C \in \mathcal{H}}$, and completions $(D_C)_{C \in \mathcal{H}}$ such that the following holds.*

- *$|\mathcal{H}| \leq \varepsilon^{-22} Q^2(\varepsilon)$.*
- *$\{B_C\}_{C \in \mathcal{H}}$ is a partition of $\{A_i \mid i \in [m]\}$.*
- *$\{D_C\}_{C \in \mathcal{H}}$ is a partition of $I \setminus \bigcup_{i \in [m]} A_i$.*
- *For all $C \in \mathcal{H}$ it holds that B_C is a category of C and D_C.*

The size *of $\mathcal{B}$ is m.*

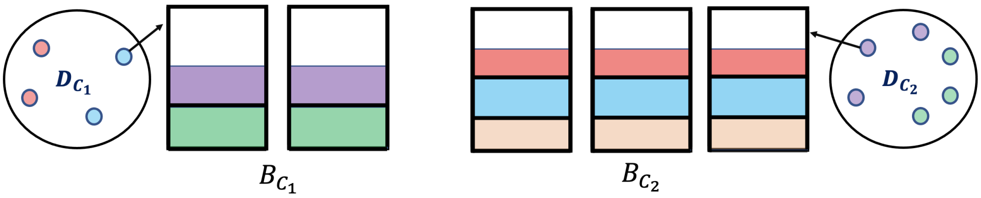


Figure 4 An example of an ε-nice partition, which consists of a packing in five bins partitioned into two categories $\mathcal{H} = \{B_{C_1}, B_{C_2}\}$. In addition, D_{C_1} and D_{C_2} are the completions of B_{C_1} and B_{C_2}, respectively. Colors indicate groups: if the cardinality bound of the blue group is 1, then D_{C_1} contains at most 2 blue items and D_{C_2} cannot contain blue items.

To obtain an ε-nice partition, Algorithm Partition initially rounds up the entries of $\bar{z}$ to obtain the prototype $\bar{z}^*$. It then finds a vertex $\bar{\lambda}$ of the $\bar{z}^*$-polytope, which is almost integral by Lemma 15. Thus, with the exception of a small number of items, each item is fully assigned either to a slot or to a configuration. Algorithm Partition uses $\bar{\lambda}$ to construct an ε-nice partition. We generate a category for each $C \in \mathsf{supp}(\bar{z}^*)$ and let $D_C = \{\ell \in I \mid \bar{\lambda}_{\ell,C} = 1\}$ be the set of all items assigned to C. We also generate $\bar{z}^*_C$ copies (bins) of each configuration and replace its slots by items via matching.

▶ **Lemma 18.** *There is an algorithm* Partition *that given $\varepsilon \in (0, 0.1)$ such that $\varepsilon^{-1} \in \mathbb{N}$, an ε-structured BPP instance $\mathcal{I}$, and a good prototype $\bar{z}$ of $\mathcal{I}$, returns in time* $\text{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ *an ε-nice partition of $\mathcal{I}$ of size at most $\|\bar{z}\| + \varepsilon^{-22}Q^2(\varepsilon)$.*

Algorithm Partition is presented in [9]. Given an ε-nice partition of size m, a packing of the instance in roughly m bins is obtained using the next lemma.

▶ **Lemma 19.** *There is a polynomial-time algorithm* Pack *which given $\varepsilon \in (0, 0.1)$ such that $\varepsilon^{-1} \in \mathbb{N}$, an ε-structured BPP instance $\mathcal{I}$, and ε-nice partition of $\mathcal{I}$ of size m, returns in time* $\text{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ *a packing of $\mathcal{I}$ in at most $(1 + 2\varepsilon)m + 2\varepsilon^{-22}Q^2(\varepsilon)$ bins.*

Algorithm Pack utilizes Algorithm Greedy to add the items in a completion of a category to the bins of this category, possibly using a few extra bins. Algorithm Pack and Algorithm Greedy are presented in [9]. Finally, we construct Algorithm Solution, which first finds an ε-nice partition and then use it to find a packing for the instance. We give the pseudocode in Algorithm 3.

Proof of Lemma 9. By Lemma 18, $\mathcal{B}$ is an ε-nice partition with size at most $\|\bar{z}\| + \varepsilon^{-22} \cdot Q^2(\varepsilon)$. Then, by Lemma 19, Φ is a full packing of $\mathcal{I}$ using at most $(1 + 2\varepsilon) \cdot \|\bar{z}\| + 5\varepsilon^{-22} \cdot Q^2(\varepsilon)$ bins. The running time is $\text{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ by Lemmas 18, 19. ◀

Algorithm 3 Solution($\mathcal{I}, \bar{z}, \varepsilon$).

```
Input  : $\varepsilon \in (0, 0.1)$, $\varepsilon^{-1} \in \mathbb{N}$, an $\varepsilon$-structured instance $\mathcal{I}$, a good prototype $\bar{z}$ of $\mathcal{I}$
Output : A packing of $\mathcal{I}$
Find an $\varepsilon$-nice partition $\mathcal{B}$ of $\mathcal{I}$ by Partition($\mathcal{I}, \bar{z}, \varepsilon$).
Return a packing $\Phi$ = Pack($\mathcal{I}, \mathcal{B}, \varepsilon$).
```

4 Discussion

In this paper we present an AFPTAS for Bin Packing with Partition Matroid. While BPP is a natural generalization of Bin Packing variants that have been studied in the past, to the best of our knowledge it is studied here for the first time. Our result improves upon the APTAS of [8] for the well studied special case of Group Bin Packing, and generalizes the AFPTASs of [15, 30] for the special case of Bin Packing with Cardinality Constraints. Our scheme applies a novel rounding method to solutions of the configuration-LP formulation of the problem. The rounding process relies on the key notion of a *prototype*, in which items are placeholders for other items, and sophisticated use of fractional grouping [16]. Our scheme demonstrates the power of this fractional version of linear grouping in solving constrained packing problems; it also shows how fractional grouping can be used *constructively*.

The rounding method introduced in this paper seems useful also for other settings. A preliminary study shows we can apply our method to obtain a polynomial time approximation scheme for *Multiple Knapsack with Partition Matroid*, a generalization of the Multiple Knapsack problem (see, e.g., [5, 28]) in which the items assigned to each bin form an independent set of a partition matroid. Furthermore, we can derive approximation algorithms for Machine Scheduling with Partition Matroid, a generalization of the classic Machine Scheduling problem in which the jobs assigned to a machine must be an independent set of a partition matroid. We note that this problem is a generalization of Machine Scheduling with Bag Constraints studied in [6, 21].

Another intriguing direction for future work is to apply our framework to Bin Packing with other types of constraints. We give two potential examples. The problem of Bin Packing with Partition Matroid is a special case of Bin Packing with Matroid, for which the input is a set of items I, a size function $s : I \to [0, 1]$ and a matroid $\mathcal{M}$. The objective is to partition I into a minimal number of bins $A_1, \ldots, A_m$ such that A_b is an independent set of the matroid $\mathcal{M}$, and $s(A_b) \leq 1$ for all $b \in [m]$. While this problem is a natural generalization of both Bin Packing and Matroid Partitioning, we were unable to find any published results. It would be interesting to obtain an APTAS for Bin Packing with Matroid using our framework. To this end, the reduction of the instance to a structured instance and the fractional shifting (see the full version of the paper [9]) need to be modified to tackle general matroids.

Finally, consider the Bin Packing with Matching Constraints problem (or, equivalently: Bin Packing with Line Graph Conflicts). The input is a graph $G = (V, E)$ and a size function on the edges $s : E \to [0, 1]$. The objective is to partition E into a minimal number of bins $A_1, \ldots, A_m$ such that $A_b \subseteq E$ is a matching in G, and $s(A_b) \leq 1$ for all $b \in [m]$. Our preliminary results suggest it may be possible to adapt the main components of our scheme for solving this problem.

References

1 Ron Adany, Moran Feldman, Elad Haramaty, Rohit Khandekar, Baruch Schieber, Roy Schwartz, Hadas Shachnai, and Tami Tamir. All-or-nothing generalized assignment with application to scheduling advertising campaigns. In *Integer Programming and Combinatorial Optimization - 16th International Conference, IPCO*, pages 13–24, 2013.

2 Nikhil Bansal, Alberto Caprara, and Maxim Sviridenko. A new approximation method for set covering problems, with applications to multidimensional bin packing. *SIAM Journal on Computing*, 39(4):1256–1278, 2010.

3 Nikhil Bansal, Marek Eliáš, and Arindam Khan. Improved approximation for vector bin packing. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on discrete algorithms*, pages 1561–1579. SIAM, 2016.

4 Alberto Caprara, Hans Kellerer, and Ulrich Pferschy. Approximation schemes for ordered vector packing problems. *Naval Research Logistics (NRL)*, 50(1):58–69, 2003.

5 Chandra Chekuri and Sanjeev Khanna. A polynomial time approximation scheme for the multiple knapsack problem. *SIAM Journal on Computing*, 35(3):713–728, 2005.

6 Syamantak Das and Andreas Wiese. On minimizing the makespan when some jobs cannot be assigned on the same machine. In *25th Annual European Symposium on Algorithms, ESA*, pages 31:1–31:14, 2017.

7 Ilan Doron-Arad, Ariel Kulik, and Hadas Shachnai. An APTAS for bin packing with clique-graph conflicts. *arXiv preprint*, 2020. `arXiv:2011.04273`.

8 Ilan Doron-Arad, Ariel Kulik, and Hadas Shachnai. An APTAS for bin packing with clique-graph conflicts. In *17th International Symposium on Algorithms and Data Structures, WADS*, pages 286–299, 2021.

9 Ilan Doron-Arad, Ariel Kulik, and Hadas Shachnai. Bin packing with partition matroid can be approximated within $o(opt)$ bins. *arXiv preprint*, 2022. `arXiv:2212.01025`.

10 Ilan Doron-Arad, Ariel Kulik, and Hadas Shachnai. An FPTAS for Budgeted Laminar Matroid Independent Set. *arXiv preprint*, 2023. `arXiv:2304.13984`.

11 Ilan Doron-Arad and Hadas Shachnai. Approximating bin packing with conflict graphs via maximization techniques. *Proc. WG*, 2023.

12 Jack Edmonds. Minimum partition of a matroid into independent subsets. *J. Res. Nat. Bur. Standards Sect. B*, 69:67–72, 1965.

13 Ibtissam Ennajjar, Youness Tabii, and Abdelhamid Benkaddour. Securing data in cloud computing by classification. In *Proceedings of the 2nd international Conference on Big Data, Cloud and Applications*, pages 1–5, 2017.

14 Leah Epstein and Asaf Levin. On bin packing with conflicts. *SIAM Journal on Optimization*, 19(3):1270–1298, 2008.

15 Leah Epstein and Asaf Levin. AFPTAS results for common variants of bin packing: A new method for handling the small items. *SIAM Journal on Optimization*, 20(6):3121–3145, 2010.

16 Yaron Fairstein, Ariel Kulik, and Hadas Shachnai. Modular and submodular optimization with multiple knapsack constraints via fractional grouping. In *29th Annual European Symposium on Algorithms, ESA*, pages 41:1–41:16, 2021.

17 W Fernandez de La Vega and George S. Lueker. Bin packing can be solved within 1+ ε in linear time. *Combinatorica*, 1(4):349–355, 1981.

18 Lisa Fleischer, Michel X Goemans, Vahab S Mirrokni, and Maxim Sviridenko. Tight approximation algorithms for maximum separable assignment problems. *Mathematics of Operations Research*, 36(3):416–431, 2011.

19 Harold N Gabow and Herbert H Westermann. Forests, frames, and games: algorithms for matroid sums and applications. *Algorithmica*, 7(1):465–497, 1992.

20 Michael R Garey and David S Johnson. Computers and intractability. *A Guide to the*, 1979.

21 Kilian Grage, Klaus Jansen, and Kim-Manuel Klein. An EPTAS for machine scheduling with bag-constraints. In *The 31st ACM Symposium on Parallelism in Algorithms and Architectures*, pages 135–144, 2019.

22 Michael D Grigoriadis, Leonid G Khachiyan, Lorant Porkolab, and Jorge Villavicencio. Approximate max-min resource sharing for structured concave optimization. *SIAM Journal on Optimization*, 11(4):1081–1091, 2001.

23 Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012.

24 Marcos Guerine, Murilo B Stockinger, Isabel Rosseti, Luidi G Simonetti, Kary ACS Ocaña, Alexandre Plastino, and Daniel de Oliveira. A provenance-based heuristic for preserving results confidentiality in cloud-based scientific workflows. *Future Generation Computer Systems*, 97:697–713, 2019.

25 Klaus Heeger, Danny Hermelin, George B Mertzios, Hendrik Molter, Rolf Niedermeier, and Dvir Shabtay. Equitable scheduling on a single machine. *Journal of Scheduling*, pages 1–17, 2022.

26 Rebecca Hoberg and Thomas Rothvoß. A logarithmic additive integrality gap for bin packing. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2616–2625. SIAM, 2017.

27 Klaus Jansen. An approximation scheme for bin packing with conflicts. *Journal of combinatorial optimization*, 3(4):363–377, 1999.

28 Klaus Jansen. Parameterized approximation scheme for the multiple knapsack problem. *SIAM Journal on Computing*, 39(4):1392–1412, 2010.

29 Klaus Jansen. A fast approximation scheme for the multiple knapsack problem. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 313–324. Springer, 2012.

30 Klaus Jansen, Marten Maack, and Malin Rau. Approximation schemes for machine scheduling with resource (in-) dependent processing times. *ACM Transactions on Algorithms (TALG)*, 15(3):1–28, 2019.

31 Klaus Jansen and Sabine R. Öhring. Approximation algorithms for time constrained scheduling. *Inf. Comput.*, 132(2):85–108, 1997.

32 Narendra Karmarkar and Richard M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *23rd Annual Symposium on Foundations of Computer Science*, pages 312–320. IEEE, 1982.

33 Hans Kellerer and Ulrich Pferschy. Cardinality constrained bin-packing problems. *Annals of Operations Research*, 92:335–348, 1999.

34 Kenneth L Krause, Vincent Y Shen, and Herbert D Schwetman. Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems. *Journal of the ACM (JACM)*, 22(4):522–550, 1975.

35 KL Krause, Vincent Y Shen, and Herbert D Schwetman. Errata:"analysis of several task-scheduling algorithms for a model of multiprogramming computer systems". *Journal of the ACM (JACM)*, 24(3):527, 1977.

36 Bill McCloskey and AJ. Shankar. *Approaches to bin packing with clique-graph conflicts*. Computer Science Division, University of California, 2005.

37 Y. Oh and S.H. Son. On a constrained bin-packing problem. *Technical Report CS-95-14*, 1995.

38 Serge A Plotkin, David B Shmoys, and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20(2):257–301, 1995.

39 Tai Le Quy, Gunnar Friege, and Eirini Ntoutsi. Multiple fairness and cardinality constraints for students-topics grouping problem. *arXiv preprint*, 2022. `arXiv:2206.09895`.

40 David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

Submodular Norms with Applications To Online Facility Location and Stochastic Probing

Kalen Patton ✉
School of Mathematics, Georgia Tech, Atlanta, GA, USA

Matteo Russo ✉
DIAG, Sapienza Università di Roma, Italy

Sahil Singla ✉
School of Computer Science, Georgia Tech, Atlanta, GA, USA

Abstract

Optimization problems often involve vector norms, which has led to extensive research on developing algorithms that can handle objectives beyond ℓ_p norms. Our work introduces the concept of *submodular norms*, which are a versatile type of norms that possess marginal properties similar to submodular set functions. We show that submodular norms can either accurately represent or approximate well-known classes of norms, such as ℓ_p norms, ordered norms, and symmetric norms. Furthermore, we establish that submodular norms can be applied to optimization problems such as online facility location and stochastic probing. This allows us to develop a logarithmic-competitive algorithm for online facility location with symmetric norms, and to prove logarithmic adaptivity gap for stochastic probing with symmetric norms.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Submodularity, Monotone Norms, Online Facility Location, Stochastic Probing

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.23

Category APPROX

Funding *Matteo Russo*: Supported by the ERC Advanced Grant 788893 AMDROMA.

Acknowledgements We thank the anonymous reviewers of APPROX/RANDOM 2023 for their valuable feedback.

1 Introduction

In the field of combinatorial optimization, norm objectives are frequently encountered. Canonical problems, such as the min-weight spanning tree and the k-median, involve searching for a feasible solution that minimizes the sum of costs, which is equivalent to the ℓ_1 norm of the edge cost vector. On the other hand, canonical problems like bottleneck spanning tree and k-center aim to minimize the maximum of costs, which is equivalent to the ℓ_∞ norm of the edge cost vector. However, because ℓ_1 and ℓ_∞ norms only capture the extreme Utilitarian and Egalitarian objectives respectively, significant research has been devoted to developing combinatorial optimization algorithms for more general norms (see references in Section 1.3). Among the commonly studied norms are ℓ_p norms, ordered norms, Orlicz norms, symmetric norms, and arbitrary monotone norms.

Over the past decade, there is also a lot of effort towards designing online and stochastic algorithms for more general norms. For instance, remarkable progress has been made in developing algorithms beyond ℓ_p norms for various problems, such as load balancing [16, 17, 29, 30, 31, 32], set cover [7, 36], spanning trees [28], and bandits with knapsacks [33, 32]. Notably, most of the recent progress is for the class of symmetric norms, i.e., monotone norms that remain unchanged upon permutation of coordinates. This progress is partly due to Ky Fan's Dominance Theorem (refer to [11]), which reduces the problem of designing algorithms for symmetric norms to ordered norms (see Section 1.1 for a formal definition). Ordered norms are comparatively more manageable due to their explicit form. Given this progress on some combinatorial problems for symmetric norms, a natural question arises:

> *What general norms and what combinatorial problems admit algorithms with good performance guarantees?*

A challenge in making progress beyond symmetric norms is that such norms are not explicit, e.g., they may not be well approximated by ordered norms. In this work we introduce the class of *submodular norms*, which is a broad class of norms with marginals properties mimicking submodular set functions. We show that submodular norms either capture or approximate popular classes of norms like ℓ_p norms, ordered norms, and symmetric norms. Moreover, submodular norms are amenable to some of the optimization problems that were previously intractable like online facility location and stochastic probing.

1.1 Norms and Submodularity

We start with the definitions of monotone, symmetric, and ordered norms. We will be only interested in norms defined in the positive orthant.

▶ **Definition 1** (Monotone Norm). *A* monotone norm *is a function* $\|\cdot\| : \mathbb{R}^n_+ \to \mathbb{R}_+$ *and is defined as*

$$\|x\| := \sup_{\boldsymbol{a} \in \mathcal{A}} \sum_i a_i x_i,$$

i.e, by a max of non-negative linear functions over set $\mathcal{A}$. This is equivalent to saying that $\|x\| \geq \|y\|$ whenever $x \geq y \geq 0$ coordinatewise (hence, the name monotone*).*

▶ **Definition 2** (Symmetric Norm). *A monotone norm $\|\cdot\|$ is a* symmetric norm *if, for any vector $x \in \mathbb{R}^n_+$ and for all of its coordinate permutations $\pi : [n] \to [n]$, $\|x\| = \|(x_{\pi(i)})_{i\in[n]}\|$.*

We remark that any symmetric norm can be written as $\sup_{\boldsymbol{a}\in\mathcal{A}}\langle \boldsymbol{a}, x^{\downarrow}\rangle$, where $x^{\downarrow}$ represents the sorted (in descending order) vector x, and $\mathcal{A}$ is a set on non-negative descending vectors. This follows from the fact that $\max_\pi \sum_i a_i x_{\pi(i)} = \langle \boldsymbol{a}^{\downarrow}, x^{\downarrow}\rangle$ for non-negative vectors a, x. In the special case where $\mathcal{A}$ is a singleton, we have ordered norms.

▶ **Definition 3** (Ordered Norm). *A monotone norm is an* ordered norm *if it can be written as $\|x\| = \sum_i a_i x_i^{\downarrow}$, where $a_1 \geq \ldots \geq a_n \geq 0$.*

Submodular Norms

Submodular set functions and their applications to optimization have been extensively studied; see books [38, 22]. Intuitively, they capture the notion of decreasing marginal gains. Although submodular functions were originally defined for discrete settings, the notion has been generalized to arbitrary lattices, in particular to real vectors [8]. This leads to the following notion of continuous submodularity, which has found several applications in machine learning [12, 10] and will be crucial in our definition of submodular norms. We discuss standard properties of continuous submodularity in Appendix A.1.

▶ **Definition 4** (Continuous Submodularity). *A real-valued function $f : \mathbb{R}_+^n \to \mathbb{R}_+$ is* continuously submodular *if for all $x, y \in \mathbb{R}_+^n$, we have $f(x \vee y) + f(x \wedge y) \le f(x) + f(y)$, where $x \vee y$ and $x \wedge y$ are the coordinate-wise max and min of x and y respectively.*

Our first contribution is to define the following natural class of submodular norms.

▶ **Definition 5** (Submodular Norm). *A monotone norm $\|\cdot\|$ is* submodular *if it is continuously submodular.*

Examples of submodular norms include all ℓ_p norms and Ordered norms (see Observation 12). Moreover, the following theorem proved in Section 2.2 shows that any symmetric norm can be approximated by a submodular norm.

▶ **Theorem 6.** *Any symmetric norm can be $O(\log \rho)$ approximated by a submodular norm, where $\rho := \frac{\|(1,1,\ldots,1)\|}{\|(1,0,0,\ldots,0)\|} \le n$. This approximation factor is tight up to $O(\operatorname{loglog} \rho)$ terms.*

There is an intimate connection between submodular norms and submodular set functions. Given a submodular norm $\|\cdot\|$, the set function $f : 2^{[n]} \to \mathbb{R}^+$ by $f(S) := \|\mathbf{1}_S\|$ is submodular, so every submodular norm is an extension of a submodular function. Moreover, if f is a monotone submodular function with $f(\emptyset) = 0$, then f can be extended to a submodular norm $\|\cdot\|$ by the *Lovász extension*:

$$\|x\| := \int_0^\infty f(\{j : t \le x_j\})dt.$$

This observation that every submodular set function induces a continuously submodular norm via its Lovász extension has appeared several times before [9, 8]. However, our definition of submodular norms can capture many more natural norms. E.g., all ℓ_p norms are submodular but for $1 < p < \infty$ they cannot be written as a Lovász extension of a submodular set function since the dual-norm unit ball has an infinite number of vertices.

▶ Remark 7. A commonly studied variant of continuous submodularity is DR-submodularity [13, 20, 37]: a function $f : \mathbb{R}_+^d \to \mathbb{R}_+$ is *DR-submodular* if it satisfies *diminishing returns* meaning $f(w + ae_i) - f(w) \le f(x + ae_i) - f(x)$ for all $x, w \in \mathbb{R}_+^d$ with $x \le w$, $i \in [d]$, and $a \ge 0$. It is known that continuous submodularity is equivalent to having this diminishing returns inequality only when $x_i = w_i$; hence continuous submodularity is a weaker property. The class of DR submodular functions turns out to be uninteresting when looking at norms since the only DR-submodular norm is the ℓ_1-norm (up to rescaling coordinate-wise). See Appendix A.1 for proofs.

1.2 Applications

In addition to being a natural class of norms, submodular norms find multiple applications. We will explore two specific applications-one in the domain of online algorithms and another in the field of stochastic optimization.

Online Facility Location

In this problem we are given a metric space $(\mathcal{M}, d)$ equipped with metric $d : \mathcal{M} \times \mathcal{M} \to \mathbb{R}_{\ge 0}$, along with a cost function $f : \mathcal{M} \to \mathbb{R}_+$ and a norm $\|\cdot\| : \mathbb{R}_+^n \to \mathbb{R}_+$. At each time step $i \in [n]$, an adversary produces a new request $x_i \in \mathcal{M}$ and the algorithm decides to either assign x_i to the closest already-open facility in the set F_{i-1}, thereby incurring a connection cost $d(x_i, F_{i-1})$, or to open a new facility q and assign request x_i to facility q, thereby incurring a connection cost $d(x_i, q)$ and an opening cost $f(q)$. Let F be the final set of opened facilities, let F_i be the set of facilities opened until (and including) the i-th request, and let $\boldsymbol{d} = (d_1, \ldots, d_n) \in \mathbb{R}_+^n$ be the vector of connection costs $d_i := d(x_i, F_i)$. Our goal is to minimize the total cost $\sum_{q \in F} f(q) + \|\boldsymbol{d}\|$.

Online facility location was introduced by Meyerson for ℓ_1 norm [35], where he showed an $O(\log n)$ competitive algorithm. A tight competitive ratio of $\Theta({}^{\log n}/_{\log\log n})$ was later obtained by Fotakis [21]. When all requests are given up front (offline setting), it is a classical NP-hard problem where we can design $O(1)$ approximation algorithm, even for general norms [27]. In the online setting, however, no non-trivial algorithm was previously known beyond ℓ_1 norms.

▶ **Theorem 8.** *For online facility location problem with a submodular norm $\|\cdot\|$, there exists a randomized online algorithm that obtains cost at most $O(\log \rho) \cdot \sum_{z \in F^*} f(z) + O(1) \cdot \|\boldsymbol{d}^*\|$, where F^* and $\boldsymbol{d}^*$ are the set of facilities and vector of assignment distances respectively given by the offline optimum algorithm and $\rho := \frac{\|(1,1,\ldots,1)\|}{\min_i \|e_i\|} \leq n \cdot \frac{\max_i \|e_i\|}{\min_i \|e_i\|}$.*

Since any symmetric norm can be $O(\log \rho)$ approximated by a submodular norm by Theorem 6, we get the following corollary.

▶ **Corollary 9.** *For online facility location problem with a symmetric norm, there exists an $O(\log \rho)$-competitive randomized algorithm.*

For concreteness, this corollary implies an $O(\log n)$-competitive algorithm for ℓ_1 norm, which matches Meyerson's bound [35], an $O(1)$-competitive algorithm for ℓ_∞ norm, and an $O(\log k)$-competitive algorithm for $\mathsf{Top}\text{-}k$ norm. This is tight up to an $O(\log \log \rho)$ factor for any symmetric norm by the lower bound construction given in Theorem 31.

The proof of Theorem 8 relies on generalizing Meyerson's algorithm beyond ℓ_1 norms. Meyerson's algorithm constructs a new facility at each demand point x_i with probability $d(x_i, F_{i-1})/f$, thereby balancing the cost of assigning the demand against the cost of constructing a new facility. A natural generalization of this algorithm to general norms is to construct a new facility with probability δ_i/f, where marginal cost $\delta_i = \|(d_1, \ldots, d_{i-1}, d(x_i, F_{i-1}), 0, \ldots, 0)\| - \|\boldsymbol{d}_{\leq i-1}\|$. Unfortunately, we will show that such an algorithm is $\Omega(n)$-competitive even for the ℓ_∞ norm. Our crucial change to Meyerson's algorithm is to carefully define *auxiliary assignment costs* $\hat{d}_i$ which upper bound the true costs d_i. Now we use $\hat{d}_i$ instead of d_i to calculate the marginal cost δ_i. Due to norm submodularity, this underestimates the marginal costs, making the algorithm more inclined to assign demand points instead of constructing new facilities.

Next, we discuss a stochastic optimization application of submodular norms.

Stochastic Probing

This problem is a natural stochastic generalization of constrained submodular maximization. Here, we are given probability distributions of n independent random variables $X = (X_1, \ldots, X_n) \in \mathbb{R}_+$, a downward-closed set family $\mathcal{F} \subseteq 2^{[n]}$, and a monotone objective $f : \mathbb{R}_+^n \to \mathbb{R}_+$. The goal is to select a feasible set $S \in \mathcal{F}$ of variables in order to maximize $f(X_S)$. The optimal strategy for this problem is generally *adaptive*, i.e., it selects elements of S one at a time and may change its decisions based on observations of the selected variables.

Since adaptive strategies are complicated (could be an exponential-sized decision tree) and hard to implement for many applications of stochastic probing, we are interested in finding non-adaptive algorithms that maximize $\max_{S \in \mathcal{F}} \mathbb{E}[f(X_S)]$. The main question, which has been studied in several papers [5, 24, 25, 26, 14, 19], is how much do we lose when we move from adaptive to non-adaptive algorithms, i.e., if $\mathsf{Adap}(X, \mathcal{F}, f)$ denotes the optimal adaptive strategy and $\mathsf{NA}(X, \mathcal{F}, f)$ denotes the optimal non-adaptive algorithm, then what is the maximum possible *adaptivity gap* $\frac{\mathsf{Adap}(X,\mathcal{F},f)}{\mathsf{NA}(X,\mathcal{F},f)}$.

For submodular set functions, the worst-case adaptivity gap is known to be 2 [26, 14]. An interesting conjecture posed in [26] is whether the adaptivity gap for *XOS* set functions is poly-logarithmic in n, where an XOS set function $f : 2^{[n]} \to \mathbb{R}_+$ is a max over linear set functions. Since a monotone norm is nothing but a max over linear functions (given by the dual-norm unit ball), they form an extension of XOS set functions from the hypercube to all non-negative real vectors. Thus, we can generalize the conjecture of [26] to the following:

▶ **Conjecture 10.** *The adaptivity gap for stochastic probing with monotone norms is poly-$\log(n)$.*

Although we are not able to resolve this general conjecture, we make progress by resolving it for all symmetric norms.

▶ **Theorem 11.** *The adaptivity gap for stochastic probing with symmetric norms is $O(\log n)$.*

The proof of this result relies on first approximating the symmetric norm by a submodular norm as given in Theorem 6. Next, we generalize the technique of bounding adaptivity gaps for submodular set functions in [14] to submodular norms.

1.3 Further Related Work

In recent years, there has been a surge of interest in the study of general norms. Some of the combinatorial problems that have been studied beyond ℓ_p norms are load balancing [16, 17, 29, 30], k-clustering [15, 16], vector scheduling [32, 18, 31], set cover [7, 36], spanning trees [28], and generalized assignment with convex costs [23, 32]. Beyond combinatorial optimization, general norms have been recently studied for problems such as mean estimation with statistical queries [34], nearest-neighbor search [4, 3], regression [2, 39], and communication complexity [1].

Continuous submodular functions have been extensively studied in the machine learning literature. We refer to the beautiful article of Bach [8] for their properties. Some of their applications to combinatorial optimization are discussed in [6, 37] and to learning are discussed in [40, 20]. The fact that submodular set function induces a norm via its Lovász extension has found several applications for regression since they induce sparsity [9, 10].

Paper Outline

Our work revolves around submodular norms and combinatorial optimization problems where the objective function is a submodular norm. In Section 2, we illustrate the key properties of submodular norms and the extent to which they serve as a good proxy for other classes of norms. In this respect, we identify a crucial parameter ρ that structurally characterizes a given submodular norm and may be of independent interest. In Section 3, we leverage these properties to derive a competitive algorithm for online facility location. Finally, in Section 4, we provide an application of submodular norms to adaptivity gaps for stochastic probing.

2 Submodular Norms

We study properties of submodular norms and how they relate to other commonly studied norms.

2.1 Properties and Important Special Cases

We first discuss some common examples of submodular norms.

▶ **Observation 12.** *The following norms are submodular:*

1. *All ℓ_p norms are submodular.*
2. *All* Top-k *and ordered norms are submodular.*
3. *For a matroid $\mathcal{M} = ([n], \mathcal{I})$, the* matroid rank norm $\|x\| := \max_{S\in\mathcal{I}}(\sum_{i\in S} x_i)$ *is submodular.*

Proof. To see that ℓ_p norms are submodular, it suffices to show that for any monotone concave $g : \mathbb{R}_+ \to \mathbb{R}_+$, and submodular $f : \mathbb{R}^n_+ \to \mathbb{R}_+$, the function $g \circ f$ is submodular. We can then apply this when $f = \|x\|_p^p$ and $g(y) = y^{1/p}$.

To prove the claim, notice that for $x, y \in \mathbb{R}^n_+$,

$$g(f(x \vee y)) - g(f(x)) \leq g(f(x \vee y) - f(x) + f(x \wedge y)) - g(f(x \wedge y)) \leq g(f(y)) - g(f(x \wedge y)).$$

On the other hand, Top-k norms and matroid rank norms are special cases of Lovász extensions. The matroid rank norm is the Lovász extension of the rank function, and a Top-k norm is a matroid rank norm for the k-uniform matroid. ◀

Submodular norms are also closed under several natural operations.

▶ **Lemma 13.** *The following operations return a submodular norm:*

1. *Any rescaling of the coordinates of a submodular norm.*
2. *Sums of partial*[1] *submodular norms.*
3. *Any conical combination of submodular norms is submodular.*[2]

Proof. The first property follows since coordinate-wise rescaling of vectors commutes with coordinate-wise max and min.

The second property follows from the fact that a partial submodular norm is a submodular semi-norm (i.e., a norm without the requirement to be positive definite). A sum of semi-norms remains a semi-norm, and from [8], a sum of continuously submodular functions is continuously submodular.

Finally, it is folklore that conical combinations of norms are norms, and it is also easy to show that such combinations also preserve continuous submodularity (see [8]). ◀

Besides their strict containment of many common norms, submodular norms are also powerful because they can be used to approximate other norms. In Section 2.2, we will discuss how symmetric norms can be approximated by submodular norms up to logarithmic factors. In addition, we note in Section 2.3 that submodular norms may approximate a much larger class of norms than just symmetric, although they are still far from the most general class of monotone norms. These approximation relations are summarized in Figure 1.

[1] Partial means norms defined on a subset of coordinates with every other coordinate treated as 0.

[2] Let $x_1, \ldots, x_m \in \mathbb{R}^n$ be real-valued vectors. We say that $y = \sum_{i\in[m]} \alpha_i x_i$ is a *conical* combination of the vectors if $\alpha_i \geq 0$ for all $i \in [m]$.

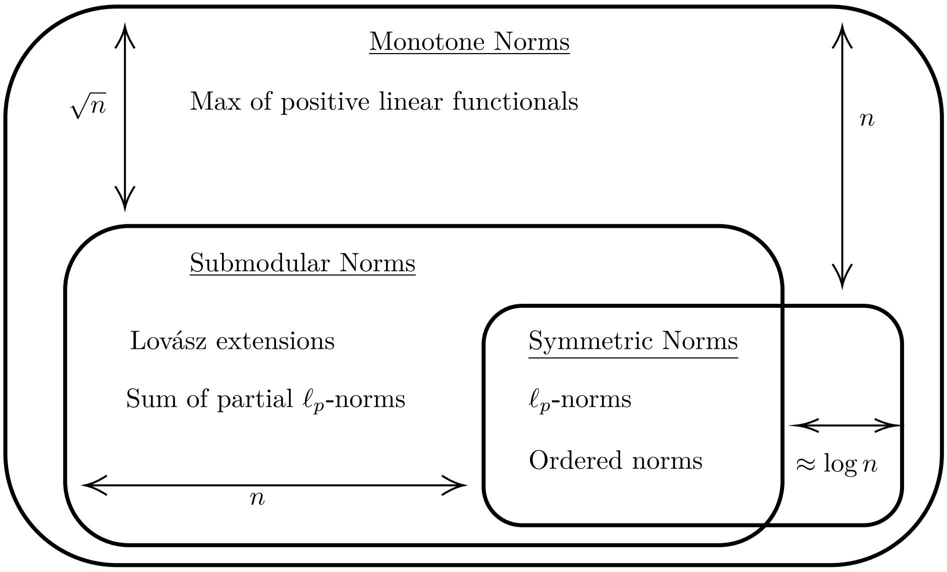


Figure 1 The containment relationships between monotone norms, submodular norms, and symmetric norms, along with some examples. The "distances" shown indicate the worst-case approximation factor (up to constants) for a norm in each outer class by a norm in the corresponding inner class.

2.2 Approximation of Symmetric Norms

A major benefit of studying submodular norms is that they can approximate any symmetric norm. Indeed, previous works have noted that symmetric norms can be approximated by an ordered norm up to a factor of $O(\log n)$ [16, 32]. For our purposes, it will be useful to make this approximation more precise by replacing $\log n$ with $\log \rho$, where parameter ρ is defined as follows:

▶ **Definition 14.** *If $e_1, \ldots, e_n$ denote the standard basis vector and let $\mathbf{1}_{\leq i} := \sum_{1\leq j\leq i} e_j$ denote the vector with 1s at the first i coordinates and 0 otherwise. Then for any monotone norm $\|\cdot\|$ we define the parameter*

$$\rho_{\|\cdot\|} := \frac{\|\mathbf{1}_{\leq n}\|}{\min_{i\in[n]} \|e_i\|}.$$

When the norm is clear from context, we simply write $\rho = \rho_{\|\cdot\|}$.

Notice that for symmetric norms, we have $\rho = \frac{\|\mathbf{1}_{\leq n}\|}{\|e_1\|} \leq n$. One can think of ρ for symmetric norms as a measure of how closely a norm behaves like $\|\cdot\|_1$ (large ρ) versus $\|\cdot\|_\infty$ (small ρ).

▶ **Observation 15.** *For ℓ_p norms, we have $\rho_{\|\cdot\|_p} = n^{1/p}$. For Top-$k$ norms, we have $\rho_{\|\cdot\|_{\text{Top-}k}} = k$.*

As we will see, the parameter ρ appears again in both the upper and lower bound analysis in Section 3 and Appendix B.3, making the improvement from $\log n$ to $\log \rho$ in Lemma 16 necessary for tight bounds in our applications.

▶ **Lemma 16.** *For any symmetric norm $\|\cdot\|$ with $\rho_{\|\cdot\|} = \rho$, there is an ordered norm $\|\cdot\|'$ such that $\|x\| \leq \|x\|' \leq 2(\log \rho + 1) \cdot \|x\|$.*

Proof. Let $\|x\| = \max_{a\in\mathcal{A}}\langle a, x^{\downarrow}\rangle$. Without loss of generality, assume $\|e_1\| = 1$, so $\|\mathbf{1}_{\le n}\| = \|(1,\dots,1)\| = \rho$.

Let $1 = m_0 \le m_1 \le \dots \le m_{\lfloor\log\rho\rfloor}$ be such that m_j is the least integer with $\|\mathbf{1}_{\le m_j}\| \ge 2^j$. Let $a_0,\dots,a_{\lfloor\log\rho\rfloor} \in \mathcal{A}$ be such that $\|\mathbf{1}_{\le m_j}\| = \langle a_j, \mathbf{1}_{\le m_j}\rangle$.

Now consider the ordered norm $\|x\|' := 2\langle a^*, x^{\downarrow}\rangle$, where $a^* = \sum_j a_j$. Clearly, we have

$$\frac{1}{2(\lfloor\log\rho\rfloor+1)}\|x\|' \le \max_j\langle a_j, x^{\downarrow}\rangle \le \max_{a\in\mathcal{A}}\langle a, x^{\downarrow}\rangle = \|x\|.$$

Additionally, notice that for any $x \in \mathbb{R}^n_+$, we can write $x^{\downarrow} = \sum_{k\in[n]}\lambda_k\mathbf{1}_{\le k}$ for some $\lambda_k \ge 0$. We have

$$\|x\|' = 2\sum_j\sum_k\lambda_k\langle a_j, \mathbf{1}_{\le k}\rangle \ge 2\sum_k\lambda_k\max_j\langle a_j, \mathbf{1}_{\le k}\rangle \ge^{\dagger} \sum_k\lambda_k\|\mathbf{1}_{\le k}\| \ge \|x\|.$$

Here, † follows from rounding each k down to the nearest m_j and using

$$\langle a_j, \mathbf{1}_{\le k}\rangle \ge \langle a_j, \mathbf{1}_{\le m_j}\rangle = \|\mathbf{1}_{\le m_j}\| \ge \frac{1}{2}\|\mathbf{1}_{\le k}\|. \qquad ◀$$

Tightness of approximation

In the worst case where $\rho = \Omega(n)$, the $\log n$ factor turns out to be nearly the best possible factor for approximation of a symmetric norm by a submodular norm. The following lemma shows that the construction in Lemma 16 is tight up to $O(\operatorname{loglog} n)$ factors.

▶ **Lemma 17.** *For any $\varepsilon \in (0, 1/2)$, define*

$$\|x\| := \max_{k\in[n]} k^{-\varepsilon}\cdot\langle\mathbf{1}_{\le k}, x^{\downarrow}\rangle.$$

For any submodular norm $\|\cdot\|'$ such that $\|x\|' \ge \|x\|$ for all $x \in \mathbb{R}^n_+$, there exists $y \in \mathbb{R}^n_+$ such that $\|y\|' \ge C\frac{\varepsilon}{1-\varepsilon}(\log n)^{1-\varepsilon}\|y\|$. Taking $\varepsilon = \frac{1}{\log\log n}$ gives $\|y\|' \ge \Omega(\frac{\log n}{\log\log n})\|y\|$.

Proof. Let y be defined by $y_k := \frac{k^{\varepsilon}-(k-1)^{\varepsilon}}{\varepsilon}$. A simple calculation yields that

$$k^{-(1-\varepsilon)} \le y_k \le (k-1)^{-(1-\varepsilon)}.$$

Additionally, we have

$$\|y\| = \max_{k\in[n]} k^{-\varepsilon}\cdot\sum_{i=1}^{k} y_i = \max_{k\in[n]} k^{-\varepsilon}\cdot\frac{k^{\varepsilon}}{\varepsilon} = \frac{1}{\varepsilon}.$$

Now to estimate $\|y\|'$, first note that we may assume $\|\cdot\|'$ to be symmetric, otherwise replace $\|\cdot\|'$ with its average over all permutations of inputs. We will inductively show that for $j \in [n/\log n]$, we have $\|y_{\le j}\|' \ge b_j := \frac{(\log(j+1))^{1-\varepsilon}}{4(1-\varepsilon)}$.

For $j = 1$, we check

$$b_1 \le \frac{1}{4(1-\varepsilon)} \le \frac{1}{2} \le 1 = \|y_{\le 1}\| \le \|y_{\le 1}\|'.$$

Now assume the claim holds for a given $j \ge 1$. Consider $z \in \mathbb{R}^d_+$ defined by

$$z_i = \begin{cases} y_i & 1 \le i \le j, \\ y_{j+1} & j < i \le j+\ell, \\ 0 & j+\ell < i, \end{cases}$$

where $\ell := \lceil (j+1)\log(j+1) \rceil$. Notice that by submodularity and symmetry, we have

$$\|y_{\leq j+1}\|' \geq \|y_{\leq j}\|' + \frac{\|z\|' - \|y_{\leq j}\|'}{\ell} \geq b_j + \frac{\|z\|' - b_j}{\ell}.$$

We see that

$$\|z\|' \geq \|z\| \geq (j+\ell)^{-\varepsilon} \cdot (j+\ell) y_{j+1} \geq \left(\frac{j+\ell}{j+1}\right)^{1-\varepsilon} \geq \log(j+1)^{1-\varepsilon}.$$

Thus, we have $\|z\|' - b_j \geq \frac{1}{2}\log(j+1)^{1-\varepsilon}$. Finally, we have

$$\|y_{\leq j+1}\|' \geq b_j + \frac{\log(j+1)^{1-\varepsilon}}{2\ell} \geq b_j + \frac{\log(j+1)^{-\varepsilon}}{4(j+1)} \geq b_j + \int_{j+1}^{j+2} \frac{(\log x)^{-\varepsilon}}{4x} dx = b_{j+1}. \blacktriangleleft$$

2.3 Beyond Symmetric Norms

Given that submodular norms allow us to approximate symmetric norms up to $\log n$ factors, we may ask if other classes of norms can be similarly approximated. We note that there exist submodular norms that are an $\Omega(n)$ factor away from any symmetric norm, which suggests that symmetric norms are not the largest class of norms which are approximated by submodular norms. Indeed, sums of partial ℓ_p or Top-k norms, such as those considered in [36], are submodular but can be highly asymmetric. Thus, although we focus on approximating symmetric norms by submodular norms, it is likely that many asymmetric norms admit submodular approximations. However, we leave the problem of characterizing these norms for future work.

In the following lemmas, we adopt the notation x_S for $x \in \mathbb{R}^n$ and $S \subseteq [n]$ to denote x after zeroing out all entries except those at indices in S, as well as $\mathbf{1}_S$ to denote the indicator vector of S.

▶ **Lemma 18.** *There exists a submodular norm $\|\cdot\|'$ for which any symmetric norm $\|\cdot\|$ satisfying $\|x\| \leq \|x\|'$ for all $x \in \mathbb{R}^n_+$, also has $\|y\|' \geq \Omega(n) \cdot \|y\|$ for some $y \in \mathbb{R}^n_+$.*

Proof. Let $A := \{1, \ldots, n/2\}$ and $B := \{n/2+1, \ldots, n\}$. Define the norm $\|x\|' := \|x_A\|_\infty + \|x_B\|_1$. Notice that $\|\cdot\|'$ is a sum of partial ℓ_p norms, so it is submodular by Lemma 13. Now suppose $\|\cdot\|$ is a symmetric norm with $\|x\| \leq \|x\|'$ for all $x \in \mathbb{R}^n_+$. Then $\|\mathbf{1}_A\| \leq \|\mathbf{1}_A\|' = 1$. However, we also have $\|\mathbf{1}_B\| = \|\mathbf{1}_A\|$ by symmetry, and $\|\mathbf{1}_B\|' = n/2$. Thus, taking $y = \mathbf{1}_B$, we have our lemma. ◀

In the most general setting of monotone norms, however, submodular norms cannot give better than an $\Omega(\sqrt{n})$ approximation. The proof of this fact is similar to the canonical proof of the $\Omega(\sqrt{n})$ factor gap between submodular set functions and XOS set functions.

▶ **Lemma 19.** *There exists a monotone norm $\|\cdot\|$ for which any submodular norm $\|\cdot\|'$ satisfying $\|x\| \leq \|x\|'$ for all $x \in \mathbb{R}^n_+$, also has $\|y\|' \geq \Omega(\sqrt{n}) \cdot \|y\|$ for some $y \in \mathbb{R}^n_+$.*

Proof. Partition $[n]$ into $\sqrt{n}$ blocks $B_1, \ldots, B_{\sqrt{n}}$, each of size $\sqrt{n}$. Define the norm $\|x\| := \max_{k \in [\sqrt{n}]} \left(\sum_{i \in B_k} x_i\right)$. Now suppose $\|\cdot\|'$ is a submodular norm satisfying $\|x\| \leq \|x\|'$ for all $x \in \mathbb{R}^n_+$. We will construct y by starting with the zero vector and iteratively choosing one element i_k of each B_k to activate (set $y_{i_k} = 1$). Clearly $\|y\| = 1$, so we just need to show $\|y\|' \geq \Omega(\sqrt{n})$.

Formally, let $y^{(0)} = 0$, and for each $k = 1, \dots, \frac{\sqrt{n}}{2}$, do the following. If $\|y^{(k-1)}\|' \geq \frac{\sqrt{n}}{2}$, we are done and simply choose $y = y^{(k-1)}$. Otherwise, notice that $\|y^{(k-1)} + \mathbf{1}_{B_k}\|' \geq \|y^{(k-1)} + \mathbf{1}_{B_k}\| = \sqrt{n}$, so $\|y^{(k-1)} + \mathbf{1}_{B_k}\|' - \|y^{(k-1)}\| \geq \frac{\sqrt{n}}{2}$. By submodularity, there exists some $i_k \in B_k$ such that $\|y^{(k-1)} + e_{i_k}\|' \geq \|y^{(k-1)}\|' + \frac{1}{2\sqrt{n}}$, for which we set $y^{(k)} := y^{(k-1)} + e_{i_k}$ By induction, if we do not terminate early, we have $\|y^{(\sqrt{n})}\|' \geq \frac{n}{2\sqrt{n}} = \frac{\sqrt{n}}{2}$. ◀

3 Online Facility Location with Submodular Norms

In this section, we illustrate how submodular norms can be applied to Online Facility Location. Recall from Section 1.2, in this problem we are given a metric space $(\mathcal{M}, d)$ along with a cost function $f : \mathcal{M} \to \mathbb{R}_+$. At each time step $i \in [n]$, an adversary produces a new request $x_i \in \mathcal{M}$, and the algorithm decides whether to assign x_i to the closest open facility F_{i-1} or to open a new facility q and assign request x_i to q. The goal is to minimize the total facility opening costs plus a given norm $\|\cdot\|$ of the connection costs, i.e., $\min \sum_{q \in F} f(q) + \|\boldsymbol{d}\|$, where $\boldsymbol{d} = (d_1, \dots, d_n) \in \mathbb{R}^n_+$ is the vector of connection costs $d_i := d(x_i, F_i)$.

In the case of uniform costs, $f(q) = f$ for all q, so the total facility opening cost becomes $f \cdot |F|$. We use $\boldsymbol{d}_{\leq i} = (d_1, \dots, d_i, 0, \dots, 0)$ to denote the first i coordinates of vector $\boldsymbol{d}$.

3.1 Uniform Costs

For now, we will focus on the case when facility costs are uniformly f.

▶ **Theorem 20.** *Let $\|\cdot\|$ be a submodular norm, and let $\rho := \rho_{\|\cdot\|}$. For the $\|\cdot\|$ norm online facility location problem with uniform facility costs f, there exists a randomized online algorithm that obtains cost at most $O(\log \rho) \cdot |F^*| f + O(1) \cdot \|\boldsymbol{d}^*\|$, where F^* and $\boldsymbol{d}^*$ are the set of facilities and vector of assignment distances, respectively, given by the optimal offline algorithm.*

Notice that because our algorithm obtains a constant factor approximation for the assignment costs, we have the following corollary.

▶ **Corollary 21.** *There is an $O(\log \rho)$-competitive algorithm for uniform costs online facility location with symmetric norms.*

Proof. Given the uniform cost facility location problem with a monotone symmetric norm $\|\cdot\|$, let F^* and $\boldsymbol{d}^*$ be the set of facilities and assignments distances given by the optimal offline algorithm. For our online algorithm, we will approximate $\|\cdot\|$ by a submodular norm $\|\cdot\|'$ using Lemma 16, and then run the algorithm in Theorem 20 on norm $\|\cdot\|'$. Since $\log \rho = \log \rho_{\|\cdot\|} = \Theta(\log \rho_{\|\cdot\|'})$, this algorithm will incur cost at most

$$|F|f + \|\boldsymbol{d}\| \;\leq\; |F|f + \|\boldsymbol{d}\|' \;\leq\; O(\log \rho)|F^*|f + O(1)\|\boldsymbol{d}^*\|' \;\leq\; O(\log \rho)|F^*|f + O(\log \rho)\|\boldsymbol{d}^*\|. \blacktriangleleft$$

Proof outline for Theorem 20

We want to generalize Meyerson's algorithm beyond ℓ_1 norms and use submodularity to complete the analysis. Meyerson's algorithm constructs a new facility at each demand point x_i with probability $d(x_i, F_{i-1})/f$, thereby balancing the cost of assigning the demand against the cost of constructing a new facility. To adapt this algorithm to more general norms, it is natural to construct a new facility at x_i with probability δ_i/f, where $\delta_i = \|(d_1, \dots, d_{i-1}, d(x_i, F_{i-1}), 0, \dots, 0)\| - \|\boldsymbol{d}_{\leq i-1}\|$ is the marginal cost of assigning x_i.

Unfortunately, the above natural generalization of Meyerson's algorithm can have an $\Omega(n)$ competitive ratio. For instance, consider the star graph $K_{1,n}$ equipped with the standard unweighted graph distance metric. Suppose that our construction costs are $f = 1$, our submodular norm is the ℓ_∞ norm, and the demand points are all the leaves of the star graph. The optimal solution constructs a single facility at the center, yielding a total cost of $1 + \|(1, \dots, 1)\|_\infty = 2$. On the other hand, the suggested algorithm constructs a facility for every demand point, as $\delta_i = 2$ for each $i \geq 2$, incurring a total cost of n.

To get around this issue, we will define *auxiliary assignment costs* $\hat{d}_i$ that upper bound the true costs d_i. The key modification to the algorithm is that we will use $\hat{d}_i$ instead of d_i for calculating the marginals δ_i. By overestimating the assignment costs that we have incurred, the algorithm underestimates potential marginal costs due to submodularity, making it more inclined to assign demand points instead of constructing new facilities. Moreover, the flexibility of the analysis allows us to show that the increased costs of $\hat{d}_i$ still obtain an $O(\log \rho)$ competitive ratio.

We now present the formal proof.

Proof of Theorem 20. To formalize the outline above, we will first inductively define our auxiliary cost vector $\hat{\boldsymbol{d}}$ and the marginals δ_i by

$$\hat{d}_i := \min\left\{d(x_i, F_{i-1}),\;\; \max\{z \geq 0 : f \geq \|(\hat{d}_1, \dots, \hat{d}_{i-1}, z, 0, \dots, 0)\| - \|\hat{\boldsymbol{d}}_{\leq i-1}\|\}\right\} \quad \text{and}$$
$$\delta_i := \|\hat{\boldsymbol{d}}_{\leq i}\| - \|\hat{\boldsymbol{d}}_{\leq i-1}\|.$$

Thus, $\hat{d}_i$ is the assignment distance $d(x_i, F_{i-1})$ capped such that $\delta_i \leq f$. For our algorithm, we construct a facility at x_i with probability δ_i/f, and assign x_i to the nearest facility otherwise. These are well-defined probabilities since $\delta_i \leq f$. To see the upper-bound $d_i \leq \hat{d}_i$, notice that if $\delta_i < f$, then $\hat{d}_i = d(x_i, F_{i-1}) \geq d(x_i, F_i) = d_i$. If $\delta_i = f$, then $d_i = 0$ since a facility is constructed at x_i with probability 1, so $d_i = 0 \leq \hat{d}_i$.

Let $\mathsf{cost}(i) := f \cdot \mathbb{1}_{|F_i| > |F_{i-1}|} + \delta_i$ be the marginal increase in auxiliary cost at step i, so $\sum_{i \in [n]} \mathsf{cost}(i) = f \cdot |F| + \|\hat{\boldsymbol{d}}\|$. We will bound separately the cost of demand points that arrive before and after the first nearby facility is constructed. Similar to Meyerson's proof, we have the following bound on costs incurred before a facility is constructed in a given set.

▷ Claim 22. Let $A \subseteq [n]$ be a fixed set of indices, and let $S \subseteq A$ be the subset of indices that arrive before the first facility is constructed at any step in A. Then $\mathbb{E}[\sum_{i \in S} \mathsf{cost}(i)] \leq 2f$.

The proof of this claim is essentially identical to Meyerson's, so we will defer it to Appendix B.1.

Now, let us enumerate our offline algorithm's facility set as $F^* = \{c_1^*, \dots, c_K^*\}$, where $K = |F^*|$. Let $C_1^*, \dots, C_K^*$ be the offline clusters, i.e., C_k^* is the set of $i \in [n]$ for which x_i is assigned to c_k^*.

Let $r := \frac{\|\boldsymbol{d}^*\|}{\|(1,\dots,1)\|}$. We partition each cluster into rings as $C_k^* = \bigcup_{\ell=0}^{L} C_k^\ell$, where $L = \lceil \log \rho \rceil$,

$$C_k^0 := \{i \in C_k^* : d(x_i, c_k^*) \leq r\}, \quad \text{and}$$
$$C_k^\ell := \{i \in C_k^* : 2^{\ell-1} r \leq d(x_i, c_k^*) \leq 2^\ell r\} \quad \text{for } \ell \in \{1, \dots, L\}.$$

Notice that this is a partition since

$$\max_{i \in C_k^*} d(x_i, C_k^*) \leq \|\boldsymbol{d}^*\|_\infty \leq \frac{\|\boldsymbol{d}^*\|}{\min_i \|e_i\|} = r\rho \leq 2^L r.$$

We will analyze the costs incurred by our algorithm on demand points in each ring. Within each ring, we will consider two types of demands separately: *long-distance* demands LD_k^ℓ and *short-distance* demands SD_k^ℓ, defined as

$$\mathsf{LD}_k^\ell := \{i \in C_k^\ell : d(c_k^*, F_{i-1}) > 2^\ell r\} \quad \text{and} \quad \mathsf{SD}_k^\ell := \{i \in C_k^\ell : d(c_k^*, F_{i-1}) \le 2^\ell r\}.$$

In other words, long (respectively, short) distance demands arrive before (respectively, after) a facility has been constructed within the outer perimeter of its corresponding ring. We now make the following claims.

▷ Claim 23. We have

$$\mathbb{E}\Big[\sum_{\ell=0}^{L}\sum_{k=1}^{K}\sum_{i\in\mathsf{LD}_k^\ell} \mathsf{cost}(i)\Big] \le 2(L+1)Kf.$$

▷ Claim 24. We have

$$\mathbb{E}\Big[\sum_{\ell=0}^{L}\sum_{k=1}^{K}\sum_{i\in\mathsf{SD}_k^\ell} \mathsf{cost}(i)\Big] \le 8\|\boldsymbol{d}^*\|.$$

These claims together give Theorem 20, since $L = O(\log \rho)$.

Notice that Claim 23 follows immediately from Claim 22 since long-distance demands must arrive before a facility is constructed in $C_\ell^{(j)}$. This implies $\mathbb{E}\left[\sum_{i\in\mathsf{LD}_k^\ell} \mathsf{cost}(i)\right] \le 2f$ for each ℓ and k.

To show Claim 24, let $\mathsf{SD} := \bigcup_{\ell=0}^{L}\bigcup_{k=1}^{K} \mathsf{SD}_k^\ell$. We seek to show that $\sum_{i\in\mathsf{SD}} \mathsf{cost}(i) \le 8\|\boldsymbol{d}^*\|$. Notice that if $i \in \mathsf{SD}_k^0$ for some k, then we have $\hat{d}_i \le d(x_i, F_{i-1}) \le d(x_i, c_k^*) + d(c_k^*, F_{i-1}) \le d_i^* + r$. Similarly, if $i \in \mathsf{SD}_k^\ell$ for some k and $\ell \ne 0$, we have $\hat{d}_i \le 3d_i^*$. Thus, we can say that $\hat{d}_i \le 3d_i^* + r$ for all $i \in \mathsf{SD}$. This gives

$$\mathbb{E}\Big[\sum_{i\in\mathsf{SD}} \mathsf{cost}(i)\Big] \;\le\; \mathbb{E}\Big[\sum_{i\in\mathsf{SD}} 2\delta_i\Big] \;\le\; 2\mathbb{E}\Big[\|\hat{\boldsymbol{d}}_{\mathsf{SD}}\|\Big] \;\le\; 2\|r\mathbf{1}_{\le n} + 3\boldsymbol{d}^*\| \;=\; 2r\|\mathbf{1}_{\le n}\| + 6\|\boldsymbol{d}^*\| \;\le\; 8\|\boldsymbol{d}^*\|,$$

where the second inequality comes from the submodular property, the third comes from norm monotonicity, and the last inequality is by choice of r. In particular, for the second inequality, we crucially use the submodularity of the norm to say

$$\sum_{i\in\mathsf{SD}} \delta_i \;=\; \sum_{i\in\mathsf{SD}} \Big(\|\hat{\boldsymbol{d}}_{\le i}\| - \|\hat{\boldsymbol{d}}_{\le i-1}\|\Big) \;\le\; \sum_{i\in\mathsf{SD}} \Big(\|\hat{\boldsymbol{d}}_{\mathsf{SD}\cap[i]}\| - \|\hat{\boldsymbol{d}}_{\mathsf{SD}\cap[i-1]}\|\Big) \;=\; \|\hat{\boldsymbol{d}}_{\mathsf{SD}}\|. \qquad ◀$$

3.2 Non-Uniform Costs

In this section, we argue how one could develop the ideas of the previous section further to extend the algorithm to Online Facility Location with different opening costs across facilities. We will show how to modify Meyerson's algorithm for non-uniform costs in a similar manner to the uniform cost setting, but in a way that also handles new challenges that arise. We motivate and describe the new algorithm, but to avoid clutter, we defer most of the proof details to Appendix B.2.

▶ **Theorem 8.** *For online facility location problem with a submodular norm $\|\cdot\|$, there exists a randomized online algorithm that obtains cost at most $O(\log\rho)\cdot\sum_{z\in F^*} f(z) + O(1)\cdot\|\boldsymbol{d}^*\|$, where F^* and $\boldsymbol{d}^*$ are the set of facilities and vector of assignment distances respectively given by the offline optimum algorithm and $\rho := \frac{\|(1,1,\ldots,1)\|}{\min_i \|e_i\|} \le n\cdot\frac{\max_i\|e_i\|}{\min_i\|e_i\|}$.*

Recall that previously, the algorithm's only choice at step i was whether to construct a facility at x_i or not. In the non-uniform setting, it might not be feasible to only ever construct at x_i, since the cost of opening there might be prohibitively high. Meanwhile a nearby location might have a much lower cost. Instead, the algorithm must consider all possible cost levels at which it could construct and how far facilities at different cost levels are from x_i.

First, let us recap how Meyerson's algorithm handles this in the ℓ_1 norm setting. By losing at most a factor of 2, we can assume that all opening costs are in some set $\{f_1, \dots, f_m\}$, where each f_j is a power of 2. For each $j \in [m]$, we define the $W_i^{(j)}$ to be the set of facilities which at step i are open or have opening cost at most f_j:

$$W_i^{(j)} := F_{i-1} \cup \{x \in \mathcal{M} : f(x) \leq f_j\}.$$

Additionally, $W_i^{(0)} := F_{i-1}$. Now Meyerson's algorithm [35] will, for each $j \in [m]$, open a facility at the nearest location in W_j with probability $\frac{d(x_i, W_i^{(j-1)}) - d(x_i, W_i^{(j)})}{f_j}$, capped at 1, then assign x_i to the nearest open facility. As in the uniform case, we can see that the expected facility opening cost incurred is $d(x_i, W_i^{(0)}) = d(x_i, F_{i-1})$. This allows us to again consider a "long-distance" and "short-distance" phase within each ring of each optimal cluster.

To adapt this algorithm to the submodular norm setting, the idea will again be to consider construction probabilities, not based on the distances $d(x_i, W_i^{(j)})$, but instead based on the marginal increase in an auxiliary assignment cost $\|\hat{\boldsymbol{d}}\|$. However, the definition of $\hat{\boldsymbol{d}}$ is not as straightforward as in the uniform cost setting. In particular, we need to satisfy $d_i \leq \hat{d}_i$ for each i deterministically. However, without carefully controlling dependencies between facility construction (for instance, by constructing independently for each j), it may be possible that $d_i = d(x_i, F_{i-1})$ if no facilities are constructed. This would force $\hat{d}_i$ to be too large to serve as a useful upper bound.

To avoid this, we instead construct facilities by sampling a single cost level f_j at which to build a facility. The probability of sampling f_j is given by a version of the probabilities used above. Suppose these probabilities sum to a value greater than 1. In that case, we crucially limit the probabilities for smaller f_j (corresponding to a facility at a greater distance) until the total probability is capped at 1: We name these probabilities $p_i^{(j)}$'s (Definition 30). This ensures that the true assignment distance d_i is never too large, which allows us to pick a conveniently small upper bound $\hat{d}_i$.

Algorithm 1 Non-Uniform Submodular Online Facility Location.

```
Data: Metric space $(\mathcal{M}, d)$ and online requests $x_1, \dots, x_n$
Result: Opened facilities set $F$ and assignment of requests to facilities (at the time
        of arrival)
for request $x_i$, $i = 1, \dots, n$ do
    Sample $j \in \{0, \dots, m\}$ according to the distribution given by $p_i^{(j)}$;
    Assign $x_i$ to the nearest location in $W_i^{(j)}$, constructing a facility there if necessary;
end
```

Our analysis proceeds by considering the partition of arrival indices into optimal clusters $[n] = \bigcup_{k \in [K]} C_k$, each with center c_k^*. Similarly to the uniform cost setting, we partition each cluster into rings as $C_k^* = \bigcup_{\ell=0}^{L} C_k^\ell$, where $L = \lceil \log \rho \rceil$ and

$$C_k^0 := \{i \in C_k^* : d(x_i, c_k^*) \leq r\},$$
$$C_k^\ell := \{i \in C_k^* : 2^{\ell-1} r \leq d(x_i, c_k^*) \leq 2^\ell r\}, \quad \text{for } \ell \in [L].$$

For each ring C_k^ℓ, we divide the analysis into two main stages: the *short-distance* and *long-distance* stages. We leave the formal definition of these stages for the proof details, but intuitively, each demand is considered a long-distance demand until a facility is constructed at a distance which is a constant multiple of the radius of C_k^ℓ. After this happens, subsequent demands in the ring are considered short-distance demands. In each of these stages, the algorithm obtains the following.

▶ **Lemma 25** (Short-Distance Stage). *The expected cost incurred by the algorithm in the short-distance stage is*

$$\mathbb{E}\left[\mathit{ALG}_{\mathit{SD}}\right] = \mathbb{E}\Big[\sum_{i \in \mathit{SD}} \mathit{cost}(i)\Big] \leq 36 \|\boldsymbol{d}^*\|.$$

▶ **Lemma 26** (Long-Distance Stage). *The expected cost incurred in the long-distance stage is*

$$\mathbb{E}\Big[\mathit{ALG}_{\mathit{LD}}\Big] \leq 48(\log \rho + 1) \cdot \sum_{k \in [K]} f(c_k^*).$$

It is now easy to see that Theorem 8 directly follows by combining Lemma 25 and Lemma 26. The proofs of these lemmas, along with a more formal definition of the probabilities $p_i^{(j)}$, are given in Appendix B.2.

4 Adaptivity Gaps for Stochastic Probing

In this section, we use submodular norms to prove small adaptivity gaps for the stochastic probing problem. Recall the stochastic probing problem from Section 1.2: Given n independent random variables $X = (X_1, \ldots, X_n) \in \mathbb{R}_+$, a downward closed set family $\mathcal{F} \subseteq 2^{[n]}$, and a monotone objective $f : \mathbb{R}_+^n \to \mathbb{R}_+$, the stochastic probing problem $(X, \mathcal{F}, f)$ is to open a feasible set $S \in \mathcal{F}$ of variables to maximize $f(X_S)$.

Denote by $\mathsf{Adap}(X, \mathcal{F}, f)$ the maximum expected objective achievable by an adaptive algorithm, i.e., one which selects elements of S one at a time, and may change its strategy based on its observations of the selected variables. We denote by $\mathsf{NA}(X, \mathcal{F}, f)$ the maximum expected objective by a non-adaptive algorithm, i.e., $\mathsf{NA}(X, \mathcal{F}, f) := \max_{S \in \mathcal{F}} \mathbb{E}[f(X_S)]$.

▶ **Theorem 27.** *If f is a submodular norm, then $\mathsf{Adap}(X, \mathcal{F}, f) \leq 2 \cdot \mathsf{NA}(X, \mathcal{F}, f)$.*

The following result for symmetric norms is an immediate corollary due to Theorem 6.

▶ **Theorem 11.** *The adaptivity gap for stochastic probing with symmetric norms is $O(\log n)$.*

Proof of Theorem 27. We follow the same proof approach as in [14]. Consider an adaptive algorithm Adap, and a non-adaptive algorithm Alg which selects each $S \in \mathcal{F}$ with the same probabilities as Adap, only non-adaptively. We will show by induction on n that Adap achieves an expected objective at most twice that of Alg. This is trivially true for $n = 1$, so we only need to show the inductive step.

We can compare the performance of these two algorithms by coupling their actions. Let's say Adap runs on random variables $X = (X_1, \ldots, X_n)$, and let $S \in \mathcal{F}$ be the (random) set of adaptively chosen variables. We can say Alg runs on variables $Y = (Y_1, \ldots, Y_n)$, i.i.d. copies of X, by choosing the same set S as Adap. Without loss of generality, say that Adap starts by selecting X_1. Since $1 \in S$ deterministically, we have that Adap achieves reward

$$\begin{aligned}
\mathbb{E}[f(X_S)] &= \mathbb{E}[f(X_1)] + \mathbb{E}\big[f_{X_1}(X_{S \setminus \{1\}}) \mid X_1\big] \\
&\leq \mathbb{E}[f(X_1 \vee Y_1)] + \mathbb{E}\big[f_{X_1 \vee Y_1}(X_{S \setminus \{1\}}) \mid X_1\big] \\
&\leq 2\mathbb{E}[f(Y_1)] + \mathbb{E}\big[f_{X_1 \vee Y_1}(X_{S \setminus \{1\}}) \mid X_1, Y_1\big],
\end{aligned}$$

where $f_x(Z) := f(x, Z) - f(x, 0)$ for $Z = (Z_2, \ldots, Z_n) \in \mathbb{R}_+^{n-1}$. Notice that, since f is submodular, we have f_x is submodular and decreasing in x for all $x \in \mathbb{R}_+$.

Now, notice that Alg achieves reward

$$\begin{aligned}\mathbb{E}[f(Y_S)] &= \mathbb{E}[f(Y_1)] + \mathbb{E}\big[f_{Y_1}(Y_{S\setminus\{1\}}) \mid X_1\big] \\ &\geq \mathbb{E}[f(Y_1)] + \mathbb{E}\big[f_{X_1 \vee Y_1}(Y_{S\setminus\{1\}}) \mid X_1, Y_1\big].\end{aligned}$$

Notice that given X_1, the set $S \setminus \{1\} \in \mathcal{F}|_{-1}$ is adaptively chosen among the variables $X_2, \ldots, X_n$ by Adap. Thus, by induction we can say $\mathbb{E}\big[f_{X_1 \vee Y_1}(X_{S\setminus\{1\}}) \mid X_1, Y_1\big] \leq 2 \cdot \mathbb{E}\big[f_{X_1 \vee Y_1}(Y_{S\setminus\{1\}}) \mid X_1, Y_1\big]$. Combining this with the above inequalities gives

$$\mathbb{E}[f(X_S)] \leq 2\mathbb{E}[f(Y_1)] + 2\mathbb{E}\big[f_{X_1 \vee Y_1}(Y_{S\setminus\{1\}}) \mid X_1, Y_1\big] \leq 2\mathbb{E}[f(Y_S)]. \qquad \blacktriangleleft$$

5 Conclusion

This paper introduces the concept of submodular norms and demonstrates their application in proving the efficiency of optimization problems beyond traditional ℓ_p objectives. We provide examples showcasing the utility of submodular norms in various scenarios. Specifically, we establish bounds on the competitive ratio of online facility location problems and the adaptivity gap of stochastic probing techniques when using symmetric norm objectives. These bounds crucially depend on the norm parameter ρ, and are approximately tight in the case of facility location. There are several natural directions for future work:

(i) *General Monotone Norms:* We have shown a logarithmic competitive ratio and adaptivity gap for online facility location and stochastic probing, respectively, when the objective is a symmetric norm or approximately a submodular norm. However, it remains open whether poly-logarithm bounds exist for either problem when the norm can be an arbitrary monotone norm.

(ii) *Symmetric Norm Stochastic Probing.* The logarithmic factor we get in our adaptivity gap bound for symmetric norm stochastic probing comes from the loss in approximating a symmetric norm by a submodular norm. However, it is not clear if such a loss is necessary. It would be interesting to determine if the true adaptivity gap is sub-logarithmic or even a constant.

(iii) *Parameter ρ.* Similar to online facility location, there are other optimization problems (e.g., online fractional set cover) which are known to have differing performance guarantees for ℓ_1 and ℓ_∞ objectives. We hypothesize that for such problems with symmetric norm objectives, the parameter ρ could provide a way of interpolating between ℓ_1 and ℓ_∞.

References

1 Alexandr Andoni, Jaroslaw Blasiok, and Arnold Filtser. Communication complexity of inner product in symmetric normed spaces. In *ITCS*, volume 251, pages 4:1–4:22, 2023.

2 Alexandr Andoni, Chengyu Lin, Ying Sheng, Peilin Zhong, and Ruiqi Zhong. Subspace embedding and linear regression with orlicz norm. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 224–233. PMLR, 2018.

3 Alexandr Andoni, Assaf Naor, Aleksandar Nikolov, Ilya P. Razenshteyn, and Erik Waingarten. Hölder homeomorphisms and approximate nearest neighbors. In *FOCS*, pages 159–169. IEEE Computer Society, 2018.

4 Alexandr Andoni, Huy L. Nguyen, Aleksandar Nikolov, Ilya P. Razenshteyn, and Erik Waingarten. Approximate near neighbors for general symmetric norms. In *STOC*, pages 902–913. ACM, 2017.

5 Arash Asadpour and Hamid Nazerzadeh. Maximizing stochastic monotone submodular functions. *Manag. Sci.*, 62(8):2374–2391, 2016.

6 Brian Axelrod, Yang P. Liu, and Aaron Sidford. Near-optimal approximate discrete and continuous submodular function minimization. In *SODA*, pages 837–853. SIAM, 2020.

7 Yossi Azar, Niv Buchbinder, T.-H. Hubert Chan, Shahar Chen, Ilan Reuven Cohen, Anupam Gupta, Zhiyi Huang, Ning Kang, Viswanath Nagarajan, Joseph Naor, and Debmalya Panigrahi. Online algorithms for covering and packing problems with convex objectives. In *FOCS*, pages 148–157. IEEE Computer Society, 2016.

8 Francis Bach. Submodular Functions: from Discrete to Continous Domains. *Mathematical Programming, Series A*, 2018. doi:10.1007/s10107-018-1248-6.

9 Francis R. Bach. Structured sparsity-inducing norms through submodular functions. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems*, pages 118–126, 2010.

10 Francis R. Bach. Learning with submodular functions: A convex optimization perspective. *Found. Trends Mach. Learn.*, 6(2-3):145–373, 2013.

11 Rajendra Bhatia. *Matrix Analysis*, volume 169. Springer, 1997.

12 Yatao Bian, Joachim M. Buhmann, and Andreas Krause. Continuous submodular function maximization. *CoRR*, abs/2006.13474, 2020. arXiv:2006.13474.

13 Yatao An Bian, Joachim M. Buhmann, and Andreas Krause. Optimal continuous dr-submodular maximization and applications to provable mean field inference. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 644–653. PMLR, 2019.

14 Domagoj Bradac, Sahil Singla, and Goran Zuzic. (Near) optimal adaptivity gaps for stochastic multi-value probing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 49:1–49:21, 2019.

15 Jaroslaw Byrka, Krzysztof Sornat, and Joachim Spoerhase. Constant-factor approximation for ordered k-median. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 620–631. ACM, 2018.

16 Deeparnab Chakrabarty and Chaitanya Swamy. Approximation algorithms for minimum norm and ordered optimization problems. In *STOC*, pages 126–137. ACM, 2019.

17 Deeparnab Chakrabarty and Chaitanya Swamy. Simpler and better algorithms for minimum-norm load balancing. In *ESA*, volume 144 of *LIPIcs*, pages 27:1–27:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

18 Shichuan Deng, Jian Li, and Yuval Rabani. Generalized unrelated machine scheduling problem. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023*, pages 2898–2916. SIAM, 2023.

19 Hossein Esfandiari, Amin Karbasi, and Vahab S. Mirrokni. Adaptivity in adaptive submodularity. In Mikhail Belkin and Samory Kpotufe, editors, *Conference on Learning Theory, COLT*, volume 134, pages 1823–1846. PMLR, 2021.

20 Moran Feldman and Amin Karbasi. Continuous submodular maximization: Beyond dr-submodularity. In *NeurIPS*, 2020.

21 Dimitris Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, 2008.

22 Satoru Fujishige. *Submodular functions and optimization.* Elsevier, 2005.

23 Anupam Gupta, Ravishankar Krishnaswamy, and Kirk Pruhs. Online primal-dual for non-linear optimization with applications to speed scaling. In *Approximation and Online Algorithms – 10th International Workshop, WAOA*, volume 7846, pages 173–186, 2012.

24 Anupam Gupta and Viswanath Nagarajan. A stochastic probing problem with applications. In *Integer Programming and Combinatorial Optimization – 16th International Conference, IPCO*, pages 205–216, 2013.

25 Anupam Gupta, Viswanath Nagarajan, and Sahil Singla. Algorithms and adaptivity gaps for stochastic probing. In *Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1731–1747, 2016.

26 Anupam Gupta, Viswanath Nagarajan, and Sahil Singla. Adaptivity gaps for stochastic probing: Submodular and XOS functions. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1688–1702. SIAM, 2017.

27 Swati Gupta, Jai Moondra, and Mohit Singh. Socially fair and hierarchical facility location problems. In *Proceedings of Economics and Computation (EC)*, 2023.

28 Sharat Ibrahimpur. Stochastic minimum norm combinatorial optimization. *Ph.D. Thesis, University of Waterloo, http://hdl.handle.net/10012/18471*, 2022.

29 Sharat Ibrahimpur and Chaitanya Swamy. Approximation algorithms for stochastic minimum-norm combinatorial optimization. In *FOCS*, pages 966–977. IEEE, 2020.

30 Sharat Ibrahimpur and Chaitanya Swamy. Minimum-norm load balancing is (almost) as easy as minimizing makespan. In *ICALP*, volume 198, pages 81:1–81:20, 2021.

31 Sharat Ibrahimpur and Chaitanya Swamy. A simple approximation algorithm for vector scheduling and applications to stochastic min-norm load balancing. In *SOSA*, pages 247–256. SIAM, 2022.

32 Thomas Kesselheim, Marco Molinaro, and Sahil Singla. Online and bandit algorithms beyond ℓ_p norms. In *SODA*, pages 1566–1593. SIAM, 2023.

33 Thomas Kesselheim and Sahil Singla. Online learning with vector costs and bandits with knapsacks. In *Proceedings of COLT*, pages 2286–2305, 2020.

34 Jerry Li, Aleksandar Nikolov, Ilya P. Razenshteyn, and Erik Waingarten. On mean estimation for general norms with statistical queries. In *COLT*, volume 99 of *Proceedings of Machine Learning Research*, pages 2158–2172. PMLR, 2019.

35 Adam Meyerson. Online facility location. In *FOCS*, pages 426–431. IEEE Computer Society, 2001.

36 Viswanath Nagarajan and Xiangkun Shen. Online covering with sum of ℓ_q- norm objectives. In *ICALP*, volume 80 of *LIPIcs*, pages 12:1–12:12, 2017.

37 Rad Niazadeh, Tim Roughgarden, and Joshua R. Wang. Optimal algorithms for continuous non-monotone submodular and dr-submodular maximization. In *NeurIPS*, pages 9617–9627, 2018.

38 Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.

39 Zhao Song, Ruosong Wang, Lin F. Yang, Hongyang Zhang, and Peilin Zhong. Efficient symmetric norm regression via linear sketching. In *NeurIPS*, pages 828–838, 2019.

40 Qixin Zhang, Zengde Deng, Zaiyi Chen, Haoyuan Hu, and Yu Yang. Stochastic continuous submodular maximization: Boosting via non-oblivious function. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 26116–26134. PMLR, 2022.

A Omitted Proofs from Section 2

A.1 Properties of Continuous Submodularity

As with submodular set functions, there are many equivalent definitions for continuous submodularity which may be helpful in different settings. These are folklore properties, but we prove them for completeness.

▶ **Lemma 28.** *Let $f : \mathbb{R}^d_+ \to \mathbb{R}_+$. The following are equivalent.*

1. *f is continuously submodular.*
2. *For all $x, y, z \in \mathbb{R}^d_+$ with $\mathsf{Supp}(y) \cap \mathsf{Supp}(z) = \emptyset$, we have*

$$f(x) + f(x + y + z) \leq f(x + y) + f(x + z).$$

3. *For all $x, y \in \mathbb{R}^d_+$ with $x \leq w$, and $i \in [d]$ such that $x_i = w_i$, and $a \geq 0$, we have*

$$f(w + ae_i) - f(w) \leq f(x + ae_i) - f(x).$$

4. *For all $x \in \mathbb{R}^d_+$ and $a, b \geq 0$ and distinct $i, j \in [d]$, we have*

$$f(x) + f(x + ae_i + be_j) \leq f(x + ae_i) + f(x + be_j).$$

Proof of Lemma 28. ($1 \iff 2$) Let $x, y, z \in \mathbb{R}^d_+$ with $y \perp z$. Notice that for non-negative vectors y, z, orthogonality implies that they have disjoint support. Hence, $(x+y) \vee (x+z) = x$ and $(x+y) \wedge (x+z) = x+y+z$. Then 2 follows from the definition of continuous submodularity. Likewise, if f satisfies condition 2, then applying condition 2 with $x' := x \wedge y$, $y' := y_{\{i : y_i > x_i\}}$, and $z' := x_{\{i : x_i > y_i\}}$ gives continuous submodularity.

($2 \implies 3$) Simply take $y = w - x$ and $z = ae_i$.

($3 \implies 4$) Simply take $w = a + be_j$.

($4 \implies 2$) Let $x^{(i,j)} := x + y_{<i} + z_{<j}$. Consider the sum

$$\begin{aligned} &f(x+y) + f(x+z) - f(x) - f(x+y+z) \\ &= \sum_{i,j \in [d]} \left[f(x^{(i,j)} + y_i e_i) + f(z_i^{(i,j)} + be_j) - f(x^{(i,j)}) - f(x^{(i,j)} + y_i e_i + z_j e_j) \right]. \end{aligned}$$

By condition 4, every term in the RHS sum is non-negative, so the LHS is non-negative as well. ◀

A commonly studied variant of continuous submodularity is DR-submodularity [13, 20, 37]: a function $f : \mathbb{R}^d_+ \to \mathbb{R}_+$ is *DR-submodular* if it satisfies the stronger condition that for all $x, w \in \mathbb{R}^d_+$ with $x \leq w$, $i \in [d]$, and $a \geq 0$, we have $f(w + ae_i) - f(w) \leq f(x + ae_i) - f(x)$. In other words, f satisfies condition 3 of Lemma 28 even where $w_i \neq x_i$. However, the only DR-submodular norm is the ℓ_1-norm.

▶ **Lemma 29.** *Any DR-submodular norm is equivalent to ℓ_1 up to rescaling the coordinates.*

Proof. Suppose $\|\cdot\| : \mathbb{R}^d_+ \to \mathbb{R}$ is a DR-submodular norm with $\|e_i\| = 1$ for each $i \in [d]$. Clearly, $\|x\| \leq \sum x_i = \|x\|_1$ by triangle inequality. Suppose that $\|x\| < \|x\|_1$ for some $x \in \mathbb{R}^d_+$. Then for some $i \in [d]$, we have $\|x_{\leq i}\| - \|x_{\leq i-1}\| \leq x_i - \varepsilon$, where $\varepsilon > 0$. By DR submodularity, this means $\|x_{\leq i-1} + kx_i e_i\| \leq k(x_i - \varepsilon)$ for all $k \in \mathbb{N}$. However, with the continuity of norms, this gives

$$x_i - \varepsilon \geq \lim_{k \to \infty} \frac{\|x_{\leq i-1} + kx_i e_i\|}{k} = \lim_{k \to \infty} \left\| \frac{1}{k} \cdot x_{\leq i-1} + x_i e_i \right\| = \|x_i e_i\| = x_i,$$

which is a contradiction, so we have $\|x\| = \|x\|_1$. ◀

B Omitted Proofs from Section 3

B.1 Uniform Costs

▷ Claim 22. Let $A \subseteq [n]$ be a fixed set of indices, and let $S \subseteq A$ be the subset of indices that arrive before the first facility is constructed at any step in A. Then $\mathbb{E}[\sum_{i \in S} \mathsf{cost}(i)] \leq 2f$.

Proof. Consider the following game: For $i \in A$, a player is shown δ_i, and has to option to pay a cost of δ_i to play a lottery, which has a $\frac{\delta_i}{f}$ chance of giving reward f. Since the expected reward of playing the lottery is exactly the cost, the player is indifferent to playing at each step. This means any strategy for the player has zero expected reward. In particular, the strategy of playing the lottery only until the first win has an expected reward of 0.

Let R be the total lottery winnings of this strategy and C be the total cost of playing. We have that $\mathbb{E}[R - C] = 0$, and since at most one lottery is won, $\mathbb{E}[R] \leq f$. Thus, $\mathbb{E}[R + C] = 2\mathbb{E}[R] \leq 2f$. But $R + C$ has exactly the distribution of $\sum_{i \in S} \mathsf{cost}(i)$, which gives the desired result. ◁

B.2 Non-uniform Costs

We now introduce the notation needed to prove that Algorithm 1 shows Theorem 8. Let us recall that $f : X \to \mathbb{R}_+$ is the cost function of opening a facility. First, we assume without loss of generality that $f(x)$ is a power of 2 for each $x \in X$.[3] Let $f^{(1)} \leq \cdots \leq f^{(m)}$ be the distinct cost levels, so $2f^{(i)} \leq f^{(i+1)}$. Additionally, let $f^{(0)} := 0$ for completeness. As before, let F_i denote the set of facilities that have been opened after the arrival of x_i, before the arrival of x_{i+1}.

▶ **Definition 30.** *For each step $i \in [n]$ and cost level $j \in \{0, \ldots, m\}$, let us define*

1. $W_i^{(j)} := F_{i-1} \cup \{x \in X : f(x) \leq f^{(j)}\}$ *to be the set of locations which are open or have an opening cost at most $f^{(j)}$;*
2. $\hat{d}_i^{(j)} := \min\{d(x_i, W_i^{(j)}), \tau_i\}$ *to be the capped value of $d_i^{(j)}$ (where cap τ_i is defined in Item 6);*
3. $\hat{d}_i := \hat{d}_i^{(0)} = \min\{d(x_i, F_{i-1}), \tau_i\}$ *for simplicity.*
4. $\delta_i^{(j)} := \frac{\hat{d}_i^{(j)}}{\hat{d}_i^{(0)}} \left(\|\hat{\boldsymbol{d}}_{\leq i}^{(0)}\| - \|\hat{\boldsymbol{d}}_{\leq i-1}^{(0)}\|\right)$ *to be the fraction of marginal increase in assignment cost we attribute to cost levels $\leq j$;*
5. $p_i^{(j)} := \frac{\delta_i^{(j-1)} - \delta_i^{(j)}}{f^{(j)}}$ *for $j \geq 1$ to be the assigned probability of opening a facility in $W_i^{(j)}$, and $p_i^{(0)} := 1 - \sum_{j=1}^m p_i^{(j)}$;*
6. $\tau_i := \arg\max\{\tau \in \mathbb{R}_{\geq 0} \cup \{+\infty\} \mid \sum_{j=1}^m p_i^{(j)} \leq 1\}$ *to be the cap value, i.e., the largest nonnegative cap such that $\sum_{j=1}^m p_i^{(j)} \leq 1$. This exists as each $p_i^{(j)}$ is monotone decreasing in τ_i.*

Given the above definitions, we notice that since $d_i^{(j)}$ is decreasing in j, this means for some j we have

$$0 = \hat{d}_i^{(m)} \leq \hat{d}_i^{(m-1)} \leq \cdots \leq \hat{d}_i^{(j+1)} \leq \tau_i = \hat{d}_i^{(j)} = \cdots = \hat{d}_i^{(0)}.$$

We shall prove the following theorem as discussed in Section 3.2.

▶ **Theorem 8.** *For online facility location problem with a submodular norm $\|\cdot\|$, there exists a randomized online algorithm that obtains cost at most $O(\log \rho) \cdot \sum_{z \in F^*} f(z) + O(1) \cdot \|\boldsymbol{d}^*\|$, where F^* and $\boldsymbol{d}^*$ are the set of facilities and vector of assignment distances respectively given by the offline optimum algorithm and $\rho := \frac{\|(1,1,\ldots,1)\|}{\min_i \|e_i\|} \leq n \cdot \frac{\max_i \|e_i\|}{\min_i \|e_i\|}$.*

To prove this theorem, we will separately the so-called short distance demands SD_k^ℓ and long distance demands LD_k^ℓ in each ring C_k^ℓ. Formally, we define

$$\mathsf{LD}_k^\ell := \{i \in C_k^\ell : \hat{d}_i^{(0)} > (\lambda+1)2^\ell r\}, \qquad \mathsf{LD} := \bigcup_{k=1}^{K} \bigcup_{\ell=0}^{L} \mathsf{LD}_k^\ell,$$

$$\mathsf{SD}_k^\ell := \{i \in C_k^\ell : \hat{d}_i^{(0)} \leq (\lambda+1)2^\ell r\}, \qquad \mathsf{SD} := \bigcup_{k=1}^{K} \bigcup_{\ell=0}^{L} \mathsf{SD}_k^\ell.$$

[3] That is, by rounding costs down to powers of 2, we lose only a factor of 2.

B.2.1 Short-distance stage

▶ **Lemma 25** (Short-Distance Stage). *The expected cost incurred by the algorithm in the short-distance stage is*

$$\mathbb{E}\left[ALG_{SD}\right] = \mathbb{E}\Big[\sum_{i \in SD} cost(i)\Big] \leq 36\|\boldsymbol{d}^*\|.$$

Proof. Let us fix a set C_k^ℓ. If $\ell = 0$, then for all $i \in \mathsf{SD}_k^\ell$, we have that $\hat{d}_i \leq (\lambda+1)r$. If $\ell > 0$, we still have that $\hat{d}_i \leq (\lambda+1)2^\ell r \leq 2(\lambda+1)d_i^*$. Summing up overall demands arriving in the short distance stage we have,

$$\begin{aligned}\mathbb{E}\left[\mathsf{ALG}_{\mathsf{SD}}\right] &\leq \sum_{i \in \mathsf{SD}} \mathbb{E}[\mathsf{cost}(i)] \leq \sum_{i \in \mathsf{SD}} 2\delta_i^{(0)} = \sum_{i \in \mathsf{SD}: d_i^* \leq r} 2\delta_i^{(0)} + \sum_{i \in \mathsf{SD}: d_i^* > r} 2\delta_i^{(0)} \\ &\leq 2 \cdot \|(\hat{d}_i)_{i \in \mathsf{SD}: d_i^* \leq r}\| + 2 \cdot \|(\hat{d}_i)_{i \in \mathsf{SD}: d_i^* > r}\| \\ &\leq 2(\lambda+1) \cdot r \cdot \|(1 \dots 1)\| + 4(\lambda+1) \cdot \|\boldsymbol{d}^*\| \\ &\leq 6(\lambda+1) \cdot \|\boldsymbol{d}^*\|.\end{aligned}$$

Here, the second inequality comes from the fact that we need to account for the facility opening cost as well as the connection cost. Moreover, the third inequality holds by norm submodularity, the fourth by what was argued earlier on distances, and the last by definition of r. The lemma then follows from choosing $\lambda = 5$, which is needed for the proof of Lemma 26. ◀

B.2.2 Long-distance stage

▶ **Lemma 26** (Long-Distance Stage). *The expected cost incurred in the long-distance stage is*

$$\mathbb{E}\Big[ALG_{LD}\Big] \leq 48(\log\rho + 1) \cdot \sum_{k \in [K]} f(c_k^*).$$

Proof. Let us fix a cluster ring C_k^ℓ, and let j_k^* be defined such that $f(c_k^*) = f^{(j_k^*)}$. Denote by $\gamma_i^{(j)} := d(c_k^*, W_i^{(j)})$, the distance at step i between the cluster center and a facility whose opening cost is at most $f^{(j)}$. We denote by $\mathcal{E}_\ell^{(j)}$ the event that a facility is opened within a $\gamma_0^{(j)} + 2^{\ell+1}r$ distance from optimal center c_k^*. It is easy to see that such an event occurs whenever the algorithm constructs a facility of cost $f^{(j)}$ or higher for a demand in C_k^ℓ. We now analyze the expected cost accumulated by the algorithm before, and after $\mathcal{E}_\ell^{(j)}$ has occurred. We denote by $t_\ell^{(j)}$ the time of event $\mathcal{E}_\ell^{(j)}$ occurrence.

Before $\mathcal{E}_\ell^{(j)}$ has occurred, we have $\sum_{i \leq t_\ell^{(j)}} \mathbb{E}[\mathsf{cost}^{(j)}(i)] \leq 2f^{(j)}$, by the same reasoning as Claim 22. Hence, we have that the total cost all levels $j \leq j_k^*$ before event $\mathcal{E}_\ell^{(j)}$ is

$$\sum_{j \leq j_k^*} \sum_{\substack{i \in \mathsf{LD}_k^\ell \\ i \leq t_\ell^{(j)}}} \mathbb{E}[\mathsf{cost}^{(j)}(i)] \leq 2 \sum_{j \leq j_k^*} f^{(j)} \leq 4 f^{(j_k^*)}.$$

We seek to demonstrate that these costs make up a constant fraction of all costs during the long-distance stage. Notice that by re-indexing, we can write

$$\sum_{j \leq j_k^*} \sum_{\substack{i \in \mathsf{LD}_k^\ell \\ i < t_\ell^{(j)}}} \mathbb{E}[\mathsf{cost}^{(j)}(i)] = \sum_{j=0}^{j_k^*-1} \sum_{\substack{i \in \mathsf{LD}_k^\ell \\ t_\ell^{(j)} < i \leq t_\ell^{(j+1)}}} \sum_{j'=j+1}^{j_k^*} \mathbb{E}[\mathsf{cost}^{(j')}(i)] = \mathbb{E} \sum_{j=0}^{j_k^*-1} \sum_{\substack{i \in \mathsf{LD}_k^\ell \\ t_\ell^{(j)} < i \leq t_\ell^{(j+1)}}} 2(\delta_i^{(j)} - \delta_i^{(j_k^*)}),$$

i.e., these are also the costs that occur in the range $\{j+1,\dots,j_k^*\}$, during each period between event $\mathcal{E}_\ell^{(j)}$ and $\mathcal{E}_\ell^{(j+1)}$. In particular, we will show for each term in the sum, ${(\delta_i^{(j)}-\delta_i^{(j_k^*)})}/{\delta_i^{(0)}} \geq {\lambda-4}/{\lambda+1}$, so these costs comprise a constant fraction of the total expected cost $2\delta_i^{(0)}$ at each step i.

We start with the simple observation that

$$\frac{\delta_i^{(j)}-\delta_i^{(j_k^*)}}{\delta_i^{(0)}} = \frac{\hat{d}_i^{(j)}-\hat{d}_i^{(j_k^*)}}{\hat{d}_i^{(0)}} \geq \frac{d(x_i, W_i^{(j)})}{d(x_i, F_{i-1})} - \frac{d(x_i, c_k^*)}{\hat{d}_i^{(0)}},$$

by definition and subadditivity. We now proceed with bounding each term. Since $\mathcal{E}_\ell^{(j)}$ has occurred, but we are still in the long distance stage, we have

$$(\lambda+1)2^\ell r < \hat{d}_i^{(0)} \leq d(x_i, F_{i-1}) \leq d(x_i, c_k^*) + d(c_k^*, F_{i-1}) \leq \gamma_i^{(j)} + 3\cdot 2^\ell r.$$

This implies both $d(x_i, F_{i-1}) \leq \gamma_i^{(j)} + 3\cdot 2^\ell r$ and $\gamma_i^{(j)} \geq (\lambda-2)2^\ell r$. Additionally, we have $d(x_i, c_k^*) \leq 2^\ell r$, and by triangle inequality, we have that $\gamma_i^{(j)} \leq d(x_i, W_i^{(j)}) + 2^\ell r$.

Altogether, we get

$$\frac{d(x_i, W_i^{(j)})}{d(x_i, F_{i-1})} - \frac{d(x_i, c_k^*)}{\hat{d}_i^{(0)}} \geq \frac{\gamma_i^{(j)} - 2^\ell r}{\gamma_i^{(j)} + 3\cdot 2^{\ell+1} r} - \frac{2^\ell r}{(\lambda+1)2^\ell r} \geq \frac{\lambda-4}{\lambda+1},$$

as desired. Thus, the total cost of points in LD_k^ℓ is bounded as follows:

$$\begin{aligned}\sum_{i\in\mathsf{LD}_k^\ell} \mathbb{E}[\mathsf{cost}(i)] &= \sum_{j=0}^{j_k^*-1} \sum_{\substack{i\in\mathsf{LD}_k^\ell \\ t_\ell^{(j)}<i\leq t_\ell^{(j+1)}}} \mathbb{E}[\mathsf{cost}(i)] = \mathbb{E}\sum_{j=0}^{j_k^*-1} \sum_{\substack{i\in\mathsf{LD}_k^\ell \\ t_\ell^{(j)}<i\leq t_\ell^{(j+1)}}} 2\delta_i^{(0)} \\ &\leq 2\frac{\lambda+1}{\lambda-4}\cdot\mathbb{E}\sum_{j=0}^{j_k^*-1} \sum_{\substack{i\in\mathsf{LD}_k^\ell \\ t_\ell^{(j)}<i\leq t_\ell^{(j+1)}}} (\delta_i^{(j)}-\delta_i^{(j_k^*)}) \\ &\leq 8\frac{\lambda+1}{\lambda-4} f^{(j_k^*)}.\end{aligned}$$

We now sum across all concentric rings across all K optimal clusters, to obtain that

$$\sum_{i\in\mathsf{LD}} \mathbb{E}[\mathsf{cost}(i)] = \sum_{k=1}^{K}\sum_{\ell=0}^{L}\sum_{i\in\mathsf{LD}_k^\ell} \mathbb{E}[\mathsf{cost}(i)] \leq 8L\left(\frac{\lambda+1}{\lambda-4}\right)\cdot\sum_{k\in[K]} f^{(j_k^*)},$$

and the claim follows from choosing $\lambda = 5$. ◀

B.3 Lower Bound

▶ **Theorem 31.** *For any monotone norm $\|\cdot\|$, there exists a uniform-cost OFL problem with norm $\|\cdot\|$ such that any online algorithm only achieves $\Omega\left(\frac{\log\sigma}{\log\log\sigma}\right)$ competitive ratio, where $\sigma = \frac{\|\mathbf{1}_{\leq n}\|}{\max_i \|e_i\|}$. Notice that for symmetric norms, $\sigma = \rho$.*

Proof. We may assume $\max_i \|e_i\| = 1$ (otherwise we rescale costs), so $\|\mathbf{1}_{\leq n}\| = \sigma$. Let k be the largest integer such that $k^k \leq \sigma$, so we have $k = \Theta\left(\frac{\log \sigma}{\log \log \sigma}\right)$. Assume $k \geq 2$.

Now, let $G = (V, E)$ be a complete N-ary tree with height k, where N is sufficiently large (intuitively, think of N as infinite). For $j = 0, \dots k-1$, each downwards edge from a node at depth level j will have length k^{-j}. We also define the facility opening cost to be k.

Let $0 \leq m_0 \leq m_1 \leq \cdots \leq m_k$ be defined such that m_j is the least positive integer with $\|\mathbf{1}_{\leq m_j}\| \geq k^j$. Notice that this implies $\|\mathbf{1}_{\leq m_j}\| \leq \|\mathbf{1}_{\leq m_j - 1}\| + \|e_{m_j}\| < k^j + 1$.

Our adversary will supply the demand locations as follows. First, they will choose a random path $v_0 v_1 \dots v_k$ from the root v_0 to a leaf v_k. Then, for $j = 0, \dots, k$, the adversary supply v_j as a demand repeated $m_j - m_{j-1}$ times (m_0 times for $j = 0$).

In the offline setting, one may simply place a single facility at v_k and assign all demands to it. This gives

$$\mathsf{OPT} = k + \|\boldsymbol{d}^*\| \leq k + \sum_{j=0}^{k-1} k^{-j} \|\mathbf{1}_{\leq m_j}\| \leq k + \sum_{j=0}^{k-1} k^{-j}(k^j + 1) = O(k).$$

In the online setting, we will show that no algorithm can achieve an expected cost of less than $\Omega(k^2)$.

Notice that any online algorithm, upon receiving a demand at v_j, should only consider the options of allocating the demand or constructing a facility at v_j. Constructing a facility anywhere else is strictly disadvantageous, as there is a negligible probability (by choice of N) that the chosen location is in the subtree rooted at v_{j+1}. Thus, after the algorithm is complete, it will have constructed a set of facilities $F \subseteq \{v_0, \dots, v_k\}$, and each demand will be allocated to the most recently constructed facility above it. Let $\boldsymbol{d}$ be the vector of allocation distances.

If there is some $j \geq 2$ such that $v_j, v_{j-1} \notin F$, then notice that every demand at v_j will have allocation distance at least k^{-j+2}. Thus, we have

$$\|\boldsymbol{d}\| \geq k^{-j+2} \|\mathbf{1}_{\leq m_j} - \mathbf{1}_{\leq m_{j-1}}\| \geq k^{-j+2}(\|\mathbf{1}_{\leq m_j}\| - \|\mathbf{1}_{\leq m_{j-1}}\|) \geq k^2 - k - k^{-j+2} = \Omega(k^2)$$

However, if no such j exists, then $|F| \geq k/2$, so construction costs are at least $k^2/2 = \Omega(k^2)$. ◀

▶ **Corollary 32.** *In the case of a symmetric norm $\|\cdot\|$, our lower bound becomes $\Omega\left(\frac{\log \rho}{\log \log \rho}\right)$ as $\rho = \sigma = \frac{\|\mathbf{1}_{\leq n}\|}{\|e_1\|}$.*

Independent Sets in Elimination Graphs with a Submodular Objective

Chandra Chekuri
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA

Kent Quanrud
Department of Computer Science, Purdue University, West Lafayette, IN, USA

Abstract

Maximum weight independent set (MWIS) admits a $\frac{1}{k}$-approximation in inductively k-independent graphs [2, 40] and a $\frac{1}{2k}$-approximation in k-perfectly orientable graphs [34]. These are a parameterized class of graphs that generalize k-degenerate graphs, chordal graphs, and intersection graphs of various geometric shapes such as intervals, pseudo-disks, and several others [40, 34]. We consider a generalization of MWIS to a submodular objective. Given a graph $G = (V, E)$ and a non-negative submodular function $f : 2^V \to \mathbb{R}_+$, the goal is to approximately solve $\max_{S \in \mathcal{I}_G} f(S)$ where $\mathcal{I}_G$ is the set of independent sets of G. We obtain an $\Omega(\frac{1}{k})$-approximation for this problem in the two mentioned graph classes. The first approach is via the multilinear relaxation framework and a simple contention resolution scheme, and this results in a randomized algorithm with approximation ratio at least $\frac{1}{e(k+1)}$. This approach also yields parallel (or low-adaptivity) approximations.

Motivated by the goal of designing efficient and deterministic algorithms, we describe two other algorithms for inductively k-independent graphs that are inspired by work on streaming algorithms: a preemptive greedy algorithm and a primal-dual algorithm. In addition to being simpler and faster, these algorithms, in the monotone submodular case, yield the first deterministic constant factor approximations for various special cases that have been previously considered such as intersection graphs of intervals, disks and pseudo-disks.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Submodular optimization and polymatroids

Keywords and phrases elimination graphs, independent set, submodular maximization, primal-dual

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.24

Category APPROX

Related Version *Full Version*: https://arxiv.org/abs/2307.02022

Funding *Chandra Chekuri*: Supported in part by NSF grants CCF-1910149 and CCF-1907937.
Kent Quanrud: Supported in part by NSF grant CCF-2129816.

1 Introduction

Given a graph $G = (V, E)$ a set $S \subseteq V$ of vertices is an independent set (also referred to as a stable set) if there is no edge between any two vertices in S. Let $\alpha(G)$ denote the cardinality of a maximum independent set in G. Finding $\alpha(G)$ is a classical problem with many applications; we refer to the search problem of finding a maximum cardinality independent set as MIS. We also consider the weighted version where the input consists of G and a vertex weight function $w : V \to \mathbb{Z}_+$ and the goal is to find a maximum weight independent set; we refer to the weighted problem as MWIS. MIS is NP-Hard, and moreover it is also NP-Hard to approximate $\alpha(G)$ to within a $\frac{1}{n^{1-\epsilon}}$-factor for any fixed $\epsilon > 0$ [32, 41]. For this reason, MIS and MWIS are studied in various special classes of graphs that capture interesting problems while also being tractable. It is easy to see that graphs with maximum degree k admit a $\frac{1}{k}$-approximation. In fact, the same approximation ratio holds for k-degenerate graphs – a

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 24; pp. 24:1–24:22
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

graph $G = (V, E)$ is a k-degenerate if there is an ordering of the vertices $\mathcal{V} = \{v_1, \ldots, v_n\}$ such that for each v_i, $|N(v_i) \cap \{v_i, \ldots, v_n\}| \leq k$. A canonical example is the class of planar graphs which are 5-degenerate.

In this paper we are interested in two parameterized classes of graphs called inductively k-independent graphs [40] and k-perfectly orientable graphs [34]. These graphs are motivated by the well-known class of chordal graphs, and capture several other interesting classes such as intersection graphs of intervals, disks (and hence planar graphs), low-treewidth graphs, t-interval graphs, and many others. A more recent example is the intersection graph of a collection of pseudo-disks which were shown to be inductively 156-independent [38]. Graphs in these classes can be dense and have large cliques. We formally define the classes.

Given a graph $G = (V, E)$ and a vertex v we let $N(v)$ denote the set of neighbors of v (excluding v). A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with n vertices has a *perfect elimination ordering* if there is an ordering of vertices $\mathcal{V} = \{v_1, \ldots, v_n\}$ such that for each v_i, $\alpha(G[N(v_i) \cap \{v_i, \ldots, v_n\}]) = 1$; in other words $N(v_i) \cap \{v_i, \ldots, v_n\}$ is a clique. It is well-known that these graphs are the same as chordal graphs.[1] For example, the intersection graph of a given set of intervals is chordal. One can generalize the perfect elimination property ordering of chordal graphs.

▶ **Definition 1** ([34]). *For a fixed integer $k \geq 1$, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is* k-simplicial *if there is an ordering of vertices $\mathcal{V} = \{v_1, \ldots, v_n\}$ such that for each v_i, $G[N(v_i) \cap \{v_i, \ldots, v_n\}]$ can be covered by k cliques.*

Note that if $G[N(v_i) \cap \{v_i, \ldots, v_n\}]$ is covered by k cliques then $\alpha(G[N(v_i) \cap \{v_i, \ldots, v_n\}]) \leq k$. Hence one can define a class based on this weaker property.

▶ **Definition 2** ([2, 40]). *For a fixed integer $k \geq 1$, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is* inductively k-independent *if there is an ordering of vertices $\mathcal{V} = \{v_1, \ldots, v_n\}$ such that for each v_i, $\alpha(G[N(v_i) \cap \{v_i, \ldots, v_n\}]) \leq k$. The* inductive independence number *of $\mathcal{G}$ is the minimum k for which $\mathcal{G}$ is inductively k-independent.*

Although inductively k-independent graphs generalize k-simplicial graphs there is no known natural class of graphs that differentiates the two; typically one establishes inductive k-independence via k-simpliciality. The ordering-based definition can be further relaxed based on orientations of G.

▶ **Definition 3** ([34]). *For a fixed integer $k \geq 1$, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is* k-perfectly orientable *if there is an orientation $H = (V, A)$ of $\mathcal{G}$ such that for each vertex $v \in \mathcal{V}$, $G[S_v]$ can be covered by k cliques, where $S_v = N_H^+(v)$ is the out-neighborhood of v in H.*

▶ Remark 4. In this paper we will use the term k-perfectly orientable for the following class of graphs: there is an orientation $H = (V, A)$ of $\mathcal{G}$ such that for each vertex $v \in \mathcal{V}$, $\alpha(G[S_v]) \leq k$ where $S_v = N_H^+(v)$ is the out-neighborhood of v in H. This is more general than the preceding definition. We observe that the algorithm in [34] for MWIS works also for this larger class, although there are no known natural examples that differentiate the two.

We observe that if G is inductively k-independent then it is also k-perfectly orientable according to our relaxed definition. Indeed, if $v_1, v_2, \ldots, v_n$ is an ordering that certifies inductive k-independence we simply orient the edges of G according to this ordering which yields a DAG. The advantage of the k-perfect orientability is that it allows arbitrary orientations. Note that a cycle is 1-perfectly orientable while it is 2-inductively independent. This factor

[1] A graph is chordal iff there is no induced cycle of length more than 3.

of 2 gap shows up in the known approximation bounds for MWIS in these two classes of graphs. It is known that for arbitrarily large n there are 2-perfectly orientable graphs on n vertices such that the graphs are not inductively $\sqrt{n}$-independent [5]. These come from the intersection graphs of so-called 2-interval graphs. Thus, k-perfect orientability can add substantial modeling power.

Akcoglu et al. [2] described a $\frac{1}{k}$-approximation for the MWIS problem in graphs that are inductively k-independent. They used the local-ratio technique, and subsequently [40] derived it using a stack-based algorithm. Both algorithms require as input an ordering of the vertices that certifies the inductive k-independent property. For k-perfectly orientable graphs [34] described a $\frac{1}{2k}$-approximation for the MWIS problem following the ideas in [5] for a special case. Given a graph $G = (V, E)$ and integer k there is an $n^{O(k)}$-time algorithm to check if G is inductively k-independent [40]. Typically, the proof that a specific class of graphs is inductively k-independent for some fixed value of k, yields an efficient algorithm that also computes a corresponding ordering. This is also true for k-perfect orientability. We refer the reader to [30] for additional discussion on computational aspects of computing orderings. In this paper we will assume that we are given both G and the ordering that certifies inductive k-independence, or an orientation that certifies k-perfect orientability.

1.1 Independent sets with a submodular objective

We consider an extension of MWIS to submodular objectives. A real-valued set function $f : 2^V \to \mathbb{R}$ is modular iff $f(A)+f(B) = f(A\cup B)+f(A\cap B)$ for all $A, B \subseteq V$. It is easy to show that f is modular iff there a weight function $w : V \to \mathbb{R}$ where $f(A) = w(A) = \sum_{v\in A} w(v)$. A real-valued set function $f : 2^V \to \mathbb{R}$ is *submodular* if $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$ for all $A, B \subseteq V$. An equivalent definition is via decreasing marginal value property: for any $A \subset B \subset V$ and $v \in V - B$, $f(A + v) - f(A) \geq f(B + v) - f(B)$. Here $A + v$ is a convenient notation for $A \cup \{v\}$. f is monotone if $f(A) \leq f(B)$ for all $A \subseteq B$. We will confine our attention in this paper to non-negative submodular functions and we will also assume that $f(\emptyset) = 0$. Given a graph $G = (V, E)$ and a non-negative submodular function $f : 2^V \to \mathbb{R}_+$, we consider the problem $\max_{S\subseteq \mathcal{I}_G} f(S)$ where $\mathcal{I}_G$ is the collection of independent sets in G. This problem generalizes MWIS since a modular function is also submodular. We assume throughout that f is available through a value oracle that returns $f(S)$ on query S. Our focus is on developing approximation algorithms for this problem in the preceding graph classes, since even very simple special cases are NP-Hard.

Motivation and related work. Submodular function maximization subject to various "independence" constraints has been a very active area of research in the last two decades. There have been several important theoretical developments, and a variety of applications ranging from algorithmic game theory, machine learning and artificial intelligence, data analysis, and network analysis; see [10, 6, 22] for some pointers. We are motivated to consider this objective in inductive k-independent graphs and k-perfectly orientable graphs for several reasons. First, it is a natural generalization of MWIS. Second, various special cases of this problem have been previously studied: Feldman [24] considered the case of interval graphs, and Chan and Har-Peled considered the case of intersection graphs of disks and pseudo-disks [16]. Third, previous algorithms have relied on the multilinear relaxation based approach combined with contention resolution schemes for rounding. This is a computationally expensive approach and also requires randomization. The known approximation algorithms for MWIS in inductive k-independent graphs are based on simple combinatorial methods such as local-ratio, and this raises the question of developing similar combinatorial algorithms for submodular objectives. In particular, we are inspired by the connection to *preemptive* greedy algorithms for submodular function maximization that have been developed in the context of streaming algorithms [15, 3, 17]. Although a natural greedy algorithm has been extensively studied for submodular function maximization [36, 28], the utility of the preemptive version for offline approximation has not been explored as far as we are aware of. This is partly due to the fact that the standard greedy algorithm works well for matroid like constraints. More recently [35] developed a primal-dual based algorithm for submodular streaming under b-matching constraints which is inspired by the stack based algorithm of [37] for the modular setting; the latter has close connections to stack based algorithms for inductive k-independent graphs [40]. The algorithm in [35] was generalized to matroid intersection in [29]. Finally, at a meta-level, we are also interested in understanding the relationship in approximability between optimizing with modular objectives and submodular objectives. For many "independence" constraints the approximability of the problem with a submodular objective is often within a constant factor of the approximability with a modular objective, but there are also settings in which the submodular objective is provably harder (see [7]). A substantial amount of research on submodular function optimization is for constraints defined by *exchange* systems such as (intersections of) matroids and their generalizations such as k-exchange systems [27] and k-systems [33, 14]. Independent sets in the graph classes we consider provide a different parameterized family of constraints.

1.2 Results

We obtain an $\Omega(\frac{1}{k})$-approximation for $\max_{S\subseteq\mathcal{I}} f(S)$ in inductively k-independent graphs and in k-perfectly orientable graphs. We explore different techniques to achieve these results since they have different algorithmic benefits.

First, we obtain a randomized algorithm via the multilinear relaxation framework [21] by considering a natural polyhedral relaxation and developing simple contention resolution schemes (CRS). The CRS schemes are useful since one can combine the rounding with other side constraints in various applications.

► **Theorem 5.** *There is a randomized algorithm that given a k-perfectly orientable graph G (along with its orientation) and a monotone submodular function f, outputs an independent set S' such that with high probability $f(S') \geq (\frac{1}{k+1} \cdot \frac{1}{(1+1/k)^k}) \max_{A\in\mathcal{I}_G} f(A)$. For non-negative functions there is an algorithm that outputs an independent set S' such that with high probability $f(S') \geq \frac{1}{e(k+1)} \max_{A\in\mathcal{I}_G} f(A)$.*

The multilinear relaxation based approach yields parallel (or low-adaptivity) algorithms with essentially similar approximation ratios, following ideas in [20, 23]. Although the multilinear approach is general and powerful, there are two drawbacks; algorithmic complexity and randomization which are inherent to the approach. An interesting question in the submodular maximization literature is whether one can obtain deterministic algorithms via alternate methods, or by derandomizing the multilinear relaxation approach. There have been several results along these lines [9, 11, 31], and several open problems.

Motivated by these considerations we develop simple and efficient approximation algorithms for inductively k-independent graphs. We show that a preemptive greedy algorithm, inspired by the streaming algorithm in [17], yields a deterministic $\Omega(\frac{1}{k})$-approximation when f is monotone. This can be combined with a simple randomized approach when f is non-monotone. Inspired by [35], we describe a primal-dual algorithm that also yields a $\Omega(\frac{1}{k})$-approximation; the primal-dual approach yields better constants and we state the result below.

▶ **Theorem 6.** *There is a deterministic combinatorial algorithm that given an inductively k-independent graph G (along with its orientation) and a monotone submodular function f, outputs an independent set S' such that $f(S') \geq \frac{1}{k+1+2\sqrt{k}} \max_{A \in \mathcal{I}_G} f(A)$. For non-negative functions there is a randomized algorithm that outputs an independent set S' such that $\mathbf{E}[f(S')] \geq \frac{1}{2k+1+\sqrt{8k}} \max_{A \in \mathcal{I}_G} f(A)$. Both algorithms use $O(|V(G)|)$ value oracle calls to f and in addition take linear time in the size of G.*

▶ Remark 7. We obtain deterministic $1/4$-approximation for monotone submodular function maximization for independent sets in chordal graphs, and hence also for interval graphs. This matches the best ratio known via the multilinear relaxation approach [24], and is the first deterministic algorithm as far as we know. Similarly, this is the first deterministic algorithm for disks and pseudo-disks that were previously handled via the multilinear relaxation approach [16]. Are there deterministic algorithms for k-perfectly orientable graphs? See Section 5.

▶ Remark 8. Matchings in a graph G, when viewed as independent sets in the line graph H of G, form an inductively 2-independent graph. In fact *any* ordering of the edges of G forms a valid 2-inductive ordering of H. Thus our algorithm is also a semi-streaming algorithm. Our approximation bound for monotone functions matches the approximation achieved in [35] for matchings although we use a different LP relaxation and view the problem from a more general viewpoint. However, for non-monotone functions, our ratio is slightly weaker, and highlights some differences.

The primal-dual algorithm is a two-phase algorithm. The preemptive greedy algorithm is a single phase algorithm. It gives slightly weaker approximation bounds when compared to the primal-dual algorithm, but has the advantage that it can be viewed as an *online preemptive* algorithm. Algorithms in such a model for submodular maximization were developed in [12, 25]. Streaming algorithms for submodular function maximization in [15, 17] can be viewed as online preemptive algorithms. Our work shows that there is an online preemptive algorithm for independent sets of inductive k-independent graphs if the vertices arrive in the proper order. There are interesting examples where any ordering of the vertices is a valid k-inductive ordering.

Our main contribution in this paper is conceptual. We study the problem to unify and generalize existing results, understand the limits of existing techniques, and raise some directions for future research (see Section 5). As we mentioned, our techniques are inspired by past and recent work on submodular function maximization [21, 24, 17, 35].

Organization

Section 2 sets up the relevant technical background on submodular functions. Section 3 describes the multilinear relaxation approach and proves Theorem 5. Section 4 describes the primal-dual approach and proves Theorem 6. Section 5 concludes with a discussion of some open problems. The description and analysis of the preemptive greedy algorithm can be found in Appendix A.

2 Preliminaries

Let $f : 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ be a real-valued nonnegative set function defined over a finite ground set $\mathcal{N}$. The function f is *monotone* if $f(S) \leq f(T)$ for any nested sets $S \subseteq T \subseteq \mathcal{N}$, and *submodular* if it has decreasing marginal returns: if $S \subseteq T \subseteq \mathcal{N}$ are two nested sets and $e \in \mathcal{N} \setminus T$ is an element, then $f(S+e) - f(S) \geq f(T+e) - f(T)$. For two sets $A, B \subseteq \mathcal{N}$, we denote the marginal value of adding B to A by $f_A(B) \overset{\text{def}}{=} f(A \cup B) - f(A)$.

Incremental values

In this paper, there is always an implicit ordering $<$ over the ground set $\mathcal{N}$. For a set $S \subseteq \mathcal{N}$ and an element $e \in \mathcal{N}$, the *incremental value* of e in S, denoted $\nu(f, S, e)$, is defined as

$$\nu(f, S, e) = f_{S'}(e), \text{ where } S' = \{s \in S : s < e\}.$$

Incremental value has some simple but very useful properties, proved in [17, Lemmas 1–3] and summarized in the following.

▶ **Lemma 9.** *Let $\mathcal{N}$ be an ordered set and $f : 2^{\mathcal{N}} \to \mathbb{R}$ a set function.*

(a) *For any set $S \subseteq \mathcal{N}$, we have $f(S) = \sum\limits_{e \in S} \nu(f, S, e)$.*

(b) *Let $S \subseteq T \subseteq \mathcal{N}$ be two nested subsets of $\mathcal{N}$ and $e \in \mathcal{N}$ an element. If f is submodular, then $\nu(f, T, e) \leq \nu(f, S, e)$.*

(c) *Let $S, Z \subseteq \mathcal{N}$ be two sets, and $e \in S$. If f is submodular, then $\nu(f_Z, S, e) \leq \nu(f, Z \cup S, e)$.*

Multilinear Extension and Relaxation

▶ **Definition 10.** *Given a set function $f : 2^{\mathcal{N}} \to \mathbb{R}$, the* multilinear extension *of f, denoted F, extends f to the product space $[0,1]^{\mathcal{N}}$ by interpreting each point $x \in [0,1]^{\mathcal{N}}$ as an independent sample $S \subseteq \mathcal{N}$ with sampling probabilities given by x, and taking the expectation of $f(S)$. Equivalently,*

$$F(x) = \sum_{S \subseteq \mathcal{N}} \left(\prod_{i \in S} x_i \prod_{i \notin S} (1 - x_i) \right).$$

An *independence family* $\mathcal{I}$ over a ground set $\mathcal{N}$ is a subset of $2^{\mathcal{N}}$ that is downward closed, that is, if $A \in \mathcal{I}$ and $B \subset A$ then $B \in \mathcal{I}$. A polyhedral/convex relaxation P for a given independence family $\mathcal{I}$ over $\mathcal{N}$ is a polyhedra/convex subset of $[0,1]^{\mathcal{N}}$ such that for each $A \in \mathcal{I}$, $\chi_A \in P$ where χ_A is the characteristic vector of A (a vector in $\{0,1\}^{\mathcal{N}}$ with a 1 in coordinate i iff $i \in A$). We say that P is a *solvable* relaxation for $\mathcal{I}$ if there is a polynomial time algorithm to optimize a linear objective over P. Given a ground set $\mathcal{N}$, and a non-negative submodular function f over $\mathcal{N}$, and an independence family $\mathcal{I} \subseteq 2^{\mathcal{N}}$,[2] we are interested in the problem $\max_{S \in \mathcal{I}} f(S)$. For this general problem the multilinear relaxation approach is to approximately solve the multilinear relaxation $\max_{x \in P} F(x)$ followed by rounding – see [14, 21, 10]. For monotone f there is a randomized $(1 - 1/e)$-approximation to the multilinear relaxation when P is solvable [14]. For general non-negative functions there is a 0.385-approximation [11].

Concave closure and relaxation

▶ **Definition 11.** *Given a set function $f : 2^{\mathcal{N}} \to \mathbb{R}$, the* concave closure *of f, denoted f^+, extends f to the product space $[0,1]^{\mathcal{N}}$ as follows. For $x \in [0,1]^{\mathcal{N}}$ we let*

$$f^+(x) = \max \left\{ \sum_{S \subseteq \mathcal{N}} \alpha_S f(S) : \sum_{S \ni i} \alpha_S = x_i \textit{ for all } i \in \mathcal{N}, \sum_{S} \alpha_S = 1, \alpha_S \geq 0 \textit{ for all } S \subseteq N \right\}.$$

[2] We assume that an independence family is specified implicitly via an independence oracle that returns whether a given $A \subseteq \mathcal{N}$ belongs to $\mathcal{I}$.

As the name suggests, f^+ is a concave function over $[0,1]^{\mathcal{N}}$ for any set function f. The definition of $f^+(x)$ involves the solution of an exponential sized linear program. The concave closure of a submodular set function is in general NP-Hard to evaluate. Nevertheless, the concave closure is useful indirectly in several ways. One can relate the concave closure to the multilinear extension via the notion of correlation gap [1, 13, 39, 19]. We can consider a relaxation based on the concave closure for the problem of $\max_{S \in \mathcal{I}} f(S)$, namely, $\max_{x \in P} f^+(x)$ where P is a polyhedral or convex relaxation for the constraint set $\mathcal{I}$. Although we may not be able to solve this relaxation directly, it provides an upper bound on the optimum solution and moreover, unlike the multilinear relaxation, the relaxation can be rewritten as a large linear program when P is polyhedral.

Contention Resolution Schemes

Contention resolution schemes are a way to round fractional solutions for relaxations to packing problems and they are a powerful and useful tool in submodular function maximization [21]. For a polyhedral relaxation P for $\mathcal{I}$ and a real $b \in [0,1]$, bP refers to the polyhedron $\{bx \mid x \in P\}$.

▶ **Definition 12.** *Let $b, c \in [0,1]$. A (b,c)-balanced* CR *scheme π for a polyhedral relaxation P for $\mathcal{I}$ is a procedure that for every $bx \in bP$ and $A \subseteq N$, returns a* random *set $\pi_x(A) \subseteq A \cap \mathit{support}(x)$ and satisfies the following properties:*
(a) *$\pi_x(A) \in \mathcal{I}$ with probability 1 $\quad \forall A \subseteq N, x \in bP$, and*
(b) *for all $i \in \mathit{support}(x)$, $\mathbf{P}[i \in \pi_x(R(x)) \mid i \in R(x)] \geq c \quad \forall x \in bP$.*
The scheme is said to be monotone *if $\mathbf{P}[i \in \pi_x(A_1)] \geq \mathbf{P}[i \in \pi_x(A_2)]$ whenever $i \in A_1 \subseteq A_2$. A $(1,c)$-balanced* CR *scheme is also called a* c-balanced CR scheme. *The scheme is* deterministic *if π is a deterministic algorithm (hence $\pi_x(A)$ is a single set instead of a distribution). It is* oblivious *if π is deterministic and $\pi_x(A) = \pi_y(A)$ for all x, y and A, that is, the output is independent of x and only depends on A. The scheme is* efficiently implementable *if π is a polynomial-time algorithm that given x, A outputs $\pi_x(A)$.*

3 Approximating via Contention Resolution Schemes

Let $G = (V,E)$ be an inductively k-independent graph and let $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ be the corresponding order. Let $\mathcal{I}$ denote the set of independent sets of G. We consider the following simple polyhedral relaxation for $\mathcal{I}$ where there is a variable x_i for each vertex v_i. For notational simplicity we let A_i denote the set $N(v_i) \cap \{v_{i+1}, \ldots, v_n\}$ which is the set of neighbors of v_i that come after v_i in the ordering.

$$x_i + \sum_{v_j \in A_i} x_j \leq k \quad \text{for all } i \in [n]$$

$$x_i \in [0,1] \quad \text{for all } i \in [n]$$

This is a valid polyhedral relaxation for $\mathcal{I}$. Indeed, consider an independent set $S \subseteq V$, and let x be the indicator vector of S. Fix a vertex v_i and consider the first inequality. If $v_i \in S$, then since $A_i \subseteq N(v_i)$, we have $A_i \cap S = \emptyset$, and the left hand side (LHS) is 1. Otherwise $\sum_{v_j \in A_i} x_j = |A_i \cap S| \leq \alpha(A_i) \leq k$, so the LHS is at most k.

In fact, the $\frac{1}{k}$-approximation for MWIS in [2, 40] are implicitly based on this relaxation. Moreover, the relaxation has a polynomial number of constraints and hence is solvable. We refer to this relaxation as Q_G and omit G when clear from the context. The multilinear relaxation is to solve $\max_{x \in Q_G} F(x)$.

Now we consider the case when $G = (V, E)$ is a k-perfectly orientable graph. Let $H = (V, A)$ be an orientation of G. For a given $v \in V$ we let $N^+_H(v) = \{u \in V \mid (v, u) \in A\}$ denote the out-neighbors of v in H. We can write a simple polyhedral relaxation for independent sets in G where we have a variable x_v for each $v \in V$ as follows:

$$x_v + \sum_{u \in N^+_H(v)} x_u \leq k \quad \text{for all } v \in V$$

$$x_v \in [0,1] \quad \text{for all } v \in V$$

To avoid notational overhead we will use Q_G to refer to the preceding relaxation for a k-perfectly orientable graph G. In [34] a stronger relaxation than the preceding relaxation is used to obtain a $\frac{1}{2k}$-approximation for MWIS. It is not hard to see, however, that the proof in [34] can be applied to the simpler relaxation above.

We will only consider k-perfectly orientable graphs in the rest of this section since the CR scheme applies for this more general class and we do not have a better scheme for inductively k-independent graphs. We consider two simple CR schemes for Q. The first is an oblivious deterministic one. Given a set R it outputs S where $S = \{v \in R \mid N^+_H(v) \cap R = \emptyset\}$. In other words it discards from R any vertex v which has an out-neighbor in R. We claim that S is an independent set. To see this suppose $uv \in E(G)$. In H, uv is oriented as (u, v) or (v, u). Thus, both u and v cannot be in S even if they are both are picked in R. It is also easy to see that the scheme is monotone.

We now describe a randomized non-oblivious scheme which yields slightly better constants and is essentially the same as the one from [24] where interval graphs were considered (a special case of $k = 1$). This scheme works as follows. Given R and x it creates a subsample $R' \subseteq R$ by sampling each $v \in R$ independently with probability $(1 - e^{-x_v})/x_v$ (Note that $1 - e^{-y} \leq y$ for all $y \in [0, 1]$.). Equivalently R' is obtained from x by sampling each v with probability $1 - e^{-x_v}$. It then applies the preceding deterministic scheme to R'. Note that this scheme is randomized and non-oblivious since it uses x in the sub-sampling step. It is also easy to see that it is monotone.

We analyze the two schemes.

▶ **Theorem 13.** *For each $b \in [0, 1]$ there is a deterministic, oblivious, monotone $(b/k, 1 - b)$ CR scheme for Q. There is a randomized monotone $(b/k, e^{-b})$ CR scheme for Q.*

Proof. Let $x \in \frac{b}{k} Q$ and Let R be a random set obtained by picking each $v \in V$ independently with probability x_v. We first analyze the deterministic CR scheme. Fix a vertex $v \in \text{support}(x)$ and condition on $v \in R$. The vertex v is included in the final output iff $N^+_H(v) \cap R = \emptyset$. Since $x \in \frac{b}{k} Q$ we have $\sum_{u \in N^+(v)} x_u \leq b - x_v \leq b$.

$$\mathbf{P}[v \in S \mid v \in R] = \mathbf{P}[N^+(v) \cap R = \emptyset] = \prod_{u \in N^+(v)} (1 - x_u) \geq 1 - \sum_{u \in N^+(v)} x_u \geq 1 - b.$$

This shows that the scheme is a $(b/k, 1 - b)$ CR scheme.

Now we analyze the randomized scheme which follows [24]. Consider $v \in R(x)$. We see that $v \in S$ conditioned on $v \in R$, if $v \in R'$ and $R' \cap N^+(v) = \emptyset$. Since the vertices are picked independently,

$$\mathbf{P}[v \in S \mid v \in R] = \mathbf{P}[v \in R' \mid v \in R] \cdot \mathbf{P}[N^+(v) \cap R' = \emptyset] = \frac{(1 - e^{-x_v})}{x_v} \prod_{u \in N^+(v)} e^{-x_u}$$

$$\geq \frac{(1 - e^{-x_v})}{x_v} e^{-(b - x_v)} \geq \frac{(e^{x_v} - 1)}{x_v} e^{-b} \geq e^{-b}.$$

This finishes the proof. ◀

One can apply the preceding CR schemes for Q_G along with the known framework via the multilinear relaxation to approximate $\max_{S\in\mathcal{I}} f(S)$. Let OPT be the value of an optimum solution. For monotone functions the Continuous Greedy algorithm [14] can be used to find a point $x \in \frac{b}{k}Q$ such that $F(x) \geq (1-e^{-b/k})\,\mathrm{OPT}$. When combined with the $(b/k, 1-b)$ CR scheme this yields a $(1-e^{-b/k})(1-b)$-approximation. The randomized CR scheme yields a $(1-e^{-b/k})e^{-b}$-approximation; this bound is maximized when $b = k\ln(1+1/k)$ and the ratio is $\frac{1}{k+1}\cdot\frac{1}{(1+1/k)^k} \geq \frac{1}{e(k+1)}$. For non-negative functions one can use Measured Continuous Greedy [26, 24] to obtain $x \in \frac{b}{k}Q$ such that $F(x) \geq \frac{b}{k}e^{-b/k}\,\mathrm{OPT}$. Combined with the CR scheme this yields a $(\frac{b}{k}e^{-b(1+/k)})$-approximation. Setting $b = k/(k+1)$ yields a $\frac{1}{e(k+1)}$-approximation.

▶ **Theorem 14.** *There is a randomized algorithm that given a k-perfectly orientable graph G (along with its orientation) and a monotone submodular function f, outputs an independent set S' such that with high probability $f(S') \geq (\frac{1}{k+1}\cdot\frac{1}{(1+1/k)^k})\max_{A\in\mathcal{I}} f(A)$. For non-negative functions there is an algorithm that outputs an independent set S' such that with high probability $f(S') \geq \frac{1}{e(k+1)}\max_{A\in\mathcal{I}} f(A)$.*

Efficiency and Parallelism

Approximately solving the multilinear relaxation is typically a bottleneck. [18] develops faster algorithms via the multiplicative-weight update (MWU) based method. We refer the reader to [18] for concrete running times that one can obtain in terms of the number of oracle calls to f or F. Once the relaxation is solved, rounding via the CR scheme above is simple and efficient. Another aspect is the design of parallel algorithms, or algorithms with low adaptivity – we refer the reader to [4] for the motivation and set up. Via results in [20, 23], and the CR scheme above, we can obtain algorithms with adaptivity $O(\frac{\log^2 n}{\epsilon^2})$ while only losing a $(1-\epsilon)$-factor in the approximation compared to the sequential approximation ratios. We defer details.

4 Primal-Dual Approach for Inductively k-Independent Graphs

We now consider a primal-dual algorithm. This is inspired by previous algorithms for MWIS in inductively k-independent graphs, and the work of Levin and Wajc [35] who considered a primal-dual based semi-streaming algorithm for submodular function maximization under matching constraints.

The stack based algorithm in [40] for MWIS is essentially a primal-dual algorithm. It is instructive to explicitly consider the LP relaxation and the analysis for MWIS before seeing the algorithm and analysis for the submodular setting. An interested reader can find this exposition in the full version.

Following [35] we consider an LP relaxation based on the concave closure of f. For independent sets in an inductively k-independent graph, we consider the relaxation $\max_{x\in Q_G} f^+(x)$. We write this as an explicit LP and describe its dual. See Fig 1. The primal has a variable x_i for each $v_i \in \mathcal{V}$ as we saw in the relaxation for MWIS. In addition to these variables, we have variables $\alpha_L, L \subseteq \mathcal{V}$ to model the objective $f^+(x)$. The dual has three types of variables. μ is for the equality constraint $\sum_L \alpha_L = 1$, y_i is corresponds to the primal packing constraint for x_i coming from the independence constraint, and z_i is for the equality constraint coming from modeling $f^+(x)$.

$$
\begin{array}{rcll}
\max \sum_{S\subseteq V} \alpha_L f(L) & & & \\
\sum_{L\subseteq V} \alpha_L & = & 1 & \\
\sum_{L\ni v_i} \alpha_L & = & x_i & i\in[n] \\
x_i + \sum_{v_j\in A_i} x_j & \leq & k & i\in[n] \\
x_i & \geq & 0 & i\in[n]
\end{array}
$$

$$
\begin{array}{rcll}
\min \mu + k\sum_{i=1}^{n} y_i & & & \\
\mu + \sum_{v_i\in L} z_i & \geq & f(L) & L\subseteq V \\
y_i + \sum_{v_j\in B_i} y_j & \geq & z_i & i\in[n] \\
y_i & \geq & 0 & i\in[n]
\end{array}
$$

Figure 1 Primal and Dual LPs via the concave closure relaxation for an inductively k-independent graph $\mathcal{G}=(\mathcal{V},\mathcal{E})$ with a given ordering $\{v_1,v_2,\ldots,v_n\}$.

4.1 Algorithm for monotone submodular functions

We describe a deterministic primal-dual algorithm for the monotone case. The algorithm and analysis are inspired by [35] and we note that the algorithm has some similarities to the preemptive greedy algorithm. The primal-dual algorithm takes a two phase approach similar to algorithm for the modular case. In the first phase it processes the vertices in the given order and creates a set $S\subseteq\mathcal{V}$. In the second phase it process the vertices in the reverse order of insertion and creates a maximal independent set. Unlike the modular case, the decision to add a vertex v_i to S in the first phase is based on an inflation factor $(1+\beta)$. The formal algorithm is described in Fig 2. The algorithm creates a feasible dual as it goes along – the variables y,z,μ are from the dual LP. It also maintains and uses auxiliary weight variables w_i, $1\leq i\leq n$ that will be useful in the analysis.

```
primal-dual-monotone-submod($f:2^{\mathcal{V}}\to\mathbb{R}_{\geq 0}$,$k\in\mathbb{N}$,$\beta\in\mathbb{R}_{>0}$).
1. Initialize an empty stack $S$. Let $\mathcal{V}=\{v_1,\ldots,v_n\}$ be a $k$-independence ordering of $\mathcal{V}$.
   Set $w,z,y\leftarrow\mathbb{0}_n$.
2. For $i=1,\ldots,n$:
   A. Let $C_i=N(v_i)\cap S=\{u\in S: uv_i\in\mathcal{E}\}$
   B. If $(f_S(v_i)>(1+\beta)\sum_{v_j\in C_i} w_j)$ then
      1. Call $S$.push($v_i$) and set $x_i\leftarrow 1$.
      2. Set $w_i\leftarrow f_S(v_i)-\sum_{v_j\in C_i} w_j$ and $y_i\leftarrow(1+\beta)w_i$.
   C. Otherwise set $z_i\leftarrow f_S(v_i)$
3. Let $\mu\leftarrow f(S)$ and $\hat{S}\leftarrow\emptyset$
4. While $S$ is not empty:
   A. $v\leftarrow S$.pop()
   B. If $\hat{S}+v_i$ is independent in $\mathcal{G}$ then set $\hat{S}\leftarrow\hat{S}+v_i$.
5. Return $\hat{S}$
```

Figure 2 Primal-dual algorithm for monotone submodular maximization. The algorithm creates a feasible dual solution in the first phase along with a set S_{end}. In the second phase it processes S_{end} in reverse order of insertion and creates a maximal independent set.

Let S_{end} be the set of vertices in the stack S at the end of the first phase. S is a monotonically increasing set during the algorithm. Note that $\mu = f(S_{\text{end}})$ at the end of the algorithm. We observe that for each i, the algorithm sets the variables w_i, y_i, z_i exactly once when v_i is processed, and does not alter the values after they are set.

▶ **Lemma 15.** *The algorithm* `primal-dual-monotone-submod` *creates a feasible dual solution* $\mu, \bar{y}, \bar{z}$ *when* f *is monotone.*

Proof. We observe that $z_i = 0$ if $v_i \in S_{\text{end}}$ and $z_i = \nu\big(f, S^-_{v_i}, v_i\big)$ otherwise. By submodularity it follows that if $v_i \notin S_{\text{end}}$, $z_i \geq f_{S_{\text{end}}}(v_i)$ since $S^-_{v_i} \subseteq S_{\text{end}}$.

Consider the first set of constraints in the dual of the form $\mu + \sum_{v_i \in L} z_i \geq f(L)$ for $L \subseteq V$. We have

$$\mu + \sum_{v_i \in L} z_i \geq f(S_{\text{end}}) + \sum_{v_i \in L \setminus S_{\text{end}}} f_{S_{\text{end}}}(v_i) \geq f(S_{\text{end}} \cup L) \geq f(L).$$

We used submodularity in the second inequality and monotonicity of f in the last inequality.

Now consider the second set of constraints in the dual of the form $y_i + \sum_{v_j \in B_i} y_j \geq z_i$ for each i. If $v_i \in S_{\text{end}}$ then $z_i = 0$ and the constraint is trivially satisfied since the y variables are non-negative. Assume $v_i \notin S_{\text{end}}$. The algorithm did not add v_i to S because

$$z_i = \nu\big(f, S^-_{v_i}, v_i\big) \leq (1+\beta) \sum_{v_j \in C_i} w_j = \sum_{v_j \in C_i} y_j$$

which implies that the constraint for v_i is satisfied. ◀

Feasibility of the dual solution implies an upper bound on the optimal value.

▶ **Corollary 16.** $\text{OPT} \leq f(S_{end}) + k(1+\beta) \sum_{i=1}^{n} w_i$.

We now lower bound the value of $f(\hat{S})$.

▶ **Lemma 17.** $f(\hat{S}) \geq \sum_{i=1}^{n} w_i$.

Proof. A vertex v_i is added to S_{end} since $\nu\big(f, S^-_{v_i}, v_i\big) > (1+\beta) \sum_{v_j \in C_i} w_j$. Moreover, we have $w_i + \sum_{v_j \in C_i} w_j = \nu\big(f, S^-_{v_i}, v_i\big)$ via the algorithm. Therefore,

$$f(\hat{S}) = \sum_{v_i \in \hat{S}} \nu\Big(f, \hat{S}, v_i\Big) \geq \sum_{v_i \in \hat{S}} \nu\big(f, S^-_{v_i}, v_i\big) = \sum_{v_i \in \hat{S}} (w_i + \sum_{j \in C_i} w_j).$$

We see that for every i' such that $v_{i'} \in S_{\text{end}}$ the term $w_{i'}$ appears at least once in $\sum_{v_i \in \hat{S}}(w_i + \sum_{j \in C_i} w_j)$; either $v_{i'} \in \hat{S}$ or if it is not then it was removed in the second phase since $v_{i'} \in C_i$ for some $v_i \in \hat{S}$. In the latter case $w_{i'}$ appears in the $\sum_{j \in C_i} w_j$. Thus $f(\hat{S}) \geq \sum_{i=1}^{n} w_i$ (recall that $w_i = 0$ if $v_i \notin S_{\text{end}}$). ◀

We now upper bound $f(S_{\text{end}})$ via the weights.

▶ **Lemma 18.** $f(S_{end}) \leq \frac{1+\beta}{\beta} \sum_{i=1}^{n} w_i$.

Proof. Let $v_i \in S_{\text{end}}$. Recall that $\nu\big(f, S^-_{v_i}, v_i\big) \geq (1+\beta) \sum_{j \in C_i} w_j$ and $w_i = \nu\big(f, S^-_{v_i}, v_i\big) - \sum_{j \in C_i} w_j$. This implies that $w_i \geq \frac{\beta}{1+\beta} \nu\big(f, S^-_{v_i}, v_i\big)$. Therefore

$$f(S_{\text{end}}) = \sum_{v_i \in S_{\text{end}}} \nu(f, S_{\text{end}}, v_i) = \sum_{v_i \in S_{\text{end}}} \nu\big(f, S^-_{v_i}, v_i\big) \leq \frac{1+\beta}{\beta} \sum_{v_i \in S_{\text{end}}} w_i,$$

as desired. ◀

▶ **Theorem 19.** $\mathrm{OPT} \leq (1+\beta)(1/\beta + k)f(\hat{S})$*. In particular, for* $\beta = \frac{1}{\sqrt{k}}$, $\mathrm{OPT} \leq (k+1+2\sqrt{k})f(\hat{S})$.

Proof. From Corollary 16 and Lemma 18 and Lemma 17,

$$\mathrm{OPT} \leq f(S_{\text{end}}) + k(1+\beta)\sum_{i=1}^{n} w_i \leq \frac{1+\beta}{\beta}\sum_{i=1}^{n} w_i + k(1+\beta)\sum_{i=1}^{n} w_i$$
$$\leq (1+\beta)(\frac{1}{\beta}+k)\sum_{i=1}^{n} w_i \leq (1+\beta)(\frac{1}{\beta}+k)f(\hat{S}),$$

as desired. ◀

▶ Remark 20. For $k = 1$ we obtain a 1/4-approximation which yields a deterministic 1/4-approximation for chordal graphs and interval graphs. For $k = 2$ we obtain a bound of $3 + 2\sqrt{2}$ which is the same as what [35] obtain for matchings. Note that matchings can be interpreted, via the line graph, as inductive 2-independent and in fact any ordering of the edges is an inductive 2-independent order. This explains why the ordering does not matter. [35] use a different LP relaxation for matchings, and hence it is a bit surprising that we obtain the same bound for all 2-independent graphs. For the non-monotone case we obtain a weaker bound for 2-independent graphs than what [35] obtain for matchings.

4.2 Non-monotone submodular maximization

We now consider the case of non-negative submodular function which may not be necessarily monotone. This class of functions requires some additional technical care and a key lemma that is useful in handling non-monotone function is the following.

▶ **Lemma 21** ([8]). *Let* $f : 2^V \to \mathbb{R}_+$ *be a non-negative submodular function. Fix a set* $T \subseteq V$*. Let* S *be a random subset of* V *such that for any* $v \in V$ *the probability of* $v \in S$ *is at most* p *for some* $p < 1$*. Then* $\mathbf{E}[f(S \cup T)] \geq (1-p)f(T)$*.*

We describe a randomized primal-dual algorithm which is adapted from the one form [35]. It differs from the monotone algorithm in one simple but crucial way; even when a vertex v has good value compared to its conflict set it adds it to the stack only with probability p which is a parameter that is chosen later.

As in the monotone case let S_{end} be the set of vertices in the stack at the end of the first phase (note that S_{end} is now a random set). The analysis of the randomized version of the algorithm is technically more involved. The sets $S_{\text{end}}, \hat{S}$ and the dual variables are now random variables. Since very high-value vertices can be discarded probabilistically, the dual values constructed by the algorithm may not satisfy the dual constraints for each run of the algorithm. Levin and Wajc [35] analyze their algorithm for matchings via an "expected" dual solution. We do a more direct analysis via weak duality.

The following two lemmas are essentially the same as in the monotone case and they relate the expected value of $\hat{S}$ and S_{end} to the dual weight values.

▶ **Lemma 22.** *For each run of the algorithm:* $f(\hat{S}) \geq \sum_{i=1}^{n} w_i$ *and hence* $\mathbf{E}\Big[f(\hat{S})\Big] \geq \sum_{i=1}^{n} \mathbf{E}[w_i]$.

▶ **Lemma 23.** *For each run of the algorithm,* $f(S_{end}) \leq \frac{1+\beta}{\beta}\sum_{i=1}^{n} w_i$ *and hence*

$$\mathbf{E}[f(S_{end})] \leq \frac{1+\beta}{\beta}\sum_{i=1}^{n} \mathbf{E}[w_i].$$

```
primal-dual-nonneg-submod($f : 2^{\mathcal{V}} \to \mathbb{R}_{\geq 0}$, $k \in \mathbb{N}$, $\beta \in \mathbb{R}_{>0}$).
1. Initialize an empty stack $S$. Let $\mathcal{V} = \{v_1, \dots, v_n\}$ be a $k$-independence ordering of $\mathcal{V}$.
   Let $w, y, z = \mathbb{0}_n$.
2. For $i = 1, \dots, n$:
   A. Let $C_i = N(v_i) \cap S = \{u \in S : uv_i \in \mathcal{E}\}$
   B. If $(f_S(v_i) > (1+\beta) \sum_{v_j \in C_i} w_j)$, then with probability $p$:
      1. Call $S$.push($v_i$) and $x_i \leftarrow 1$
      2. Set $w_i \leftarrow f_S(v_i) - \sum_{v_j \in C_i} w_j$ and $y_i \leftarrow (1+\beta) w_i$
   C. Otherwise set $z_i \leftarrow f_S(v_i)$.
3. Set $\mu \leftarrow f(S)$ and $\hat{S} \leftarrow \emptyset$
4. While $S$ is not empty:
   A. $v \leftarrow S$.pop().
   B. If $\hat{S} + v_i$ is independent in $\mathcal{G}$ then set $\hat{S} \leftarrow \hat{S} + v_i$
5. Return $\hat{S}$
```

Figure 3 Randomized primal-dual algorithm for non-negative submodular maximization.

The next two lemmas provide a way to upper bound the optimum value via the expected dual objective value.

▶ **Lemma 24.** *For each vertex v_i, let $1_{v_i \notin S_{end}}$ indicate if v_i is excluded from S_{end}. Let $B_i' = B_i + v_i$. Then*

$$\mathbf{E}[f(v_i \,|\, S_i) 1_{v_i \notin S_{end}}] \leq \max\left\{\frac{1-p}{p}, 1+\beta\right\} \mathbf{E}[w(B_i' \cap S_{end})].$$

Proof. Let E_i be the event that $f(v_i \,|\, S_i) > (1+\beta) w(B_i \cap S_i)$. Condition on $\bar{E}_i$, that is E_i *not* occurring, in which case v_i is not added to the stack. In this case we have

$$\begin{aligned}\mathbf{E}\big[w(B_i) \,\big|\, \bar{E}_i\big] &= \mathbf{E}\big[w(B_i' \cap S_{\text{end}}) \,\big|\, \bar{E}_i\big] \geq \frac{1}{1+\beta} \mathbf{E}\big[f(v_i \,|\, S_i) \,\big|\, \bar{E}_i\big] \\ &= \frac{1}{1+\beta} \mathbf{E}\big[f(v_i \,|\, S_i) 1_{v_i \notin S_{\text{end}}} \,\big|\, \bar{E}_i\big]\end{aligned}$$

On the other hand, condition on E_i, we have

$$\mathbf{E}[w(B_i' \cap S_{\text{end}}) \,|\, E_i] \overset{\text{(a)}}{\geq} p\, \mathbf{E}[f(v_i \,|\, S_i) \,|\, E_i] \overset{\text{(b)}}{=} \frac{p}{1-p} \mathbf{E}[f(v_i \,|\, S_i) 1_{v_i \notin S_{\text{end}}} \,|\, E_i].$$

(a) is because with probability p, we add v to the stack, in which case $w(B_i' \cap S_{\text{end}}) \geq f(v_i \,|\, S_i)$. (b) is because conditional on E_i and $f(v_i \,|\, S_i)$, $v_i \notin S_{\text{end}}$ with probability $1-p$. We combine the two bounds by taking conditional expectations, as follows:

$$\begin{aligned}\mathbf{E}\big[f(v_i \,|\, S_i) 1_{v_i \notin S_{\text{end}}}\big] &= \mathbf{E}\big[f(v_i \,|\, S_i) 1_{v_i \notin S_{\text{end}}} \,\big|\, E_i\big] \mathbf{P}[E_i] + \mathbf{E}\big[f(v_i \,|\, S_i) 1_{v_i \notin S_{\text{end}}} \,\big|\, \bar{E}_i\big] \mathbf{P}\big[\bar{E}_i\big] \\ &\leq \frac{1-p}{p} \mathbf{E}\big[w(B_i' \cap S_{\text{end}}) \,\big|\, E_i\big] \mathbf{P}[E_i] + (1+\beta) \mathbf{E}\big[w(B_i' \cap S_{\text{end}}) \,\big|\, \bar{E}_i\big] \mathbf{P}\big[\bar{E}_i\big] \\ &\leq \max\left\{\frac{1-p}{p}, 1+\beta\right\} \Big(\mathbf{E}\big[w(B_i' \cap S_{\text{end}})\big] \mathbf{P}[E_i] + \mathbf{E}\big[w(B_i' \cap S_{\text{end}})\big] \mathbf{P}\big[\bar{E}_i\big]\Big) \\ &= \max\left\{\frac{1-p}{p}, 1+\beta\right\} \mathbf{E}\big[w(B_i' \cap S_{\text{end}})\big],\end{aligned}$$

as desired. ◀

▶ **Lemma 25.** *For any set T, $\mathbf{E}[f(S_{end} \cup T)] \leq \mathbf{E}[f(S)] + k \max\left\{\frac{1-p}{p}, 1+\beta\right\} \mathbf{E}[w(S_{end})]$.*

Proof. We have

$$\begin{aligned}\mathbf{E}[f(T \cup S_{\text{end}}) - f(S_{\text{end}})] &\overset{\text{(c)}}{\leq} \mathbf{E}\left[\sum_{v_i \in T \setminus S_{\text{end}}} f(v_i \mid S_{\text{end}})\right] \overset{\text{(d)}}{\leq} \mathbf{E}\left[\sum_{v_i \in T \setminus S_{\text{end}}} f(v_i \mid S_i)\right] \\ &= \sum_{v_i \in T} \mathbf{E}[f(v_i \mid S_i) 1_{v_i \notin S_{\text{end}}}] \\ &\overset{\text{(e)}}{\leq} \max\left\{\frac{1-p}{p}, 1+\beta\right\} \sum_{v_i \in T} \mathbf{E}[w(B_i' \cap S_{\text{end}})] \\ &\overset{\text{(f)}}{\leq} \max\left\{\frac{1-p}{p}, 1+\beta\right\} k\, \mathbf{E}[w(S_{\text{end}})],\end{aligned}$$

as desired up to rearrangement of terms. Here (c,d) is by submodularity. (e) is by the Lemma 24. (f) is by k-inductive independence. ◀

We now put the lemmas together to relate $\mathbf{E}\left[f(\hat{S})\right]$ to the optimum.

▶ **Lemma 26.** *Let T^* be an optimum independent set with* OPT $= f(T^*)$. *Then*

$$\text{OPT} \leq \frac{k \max\left\{\frac{1-p}{p}, 1+\beta\right\} + \left(\frac{1+\beta}{\beta}\right)}{1-p} \mathbf{E}\left[f\left(\hat{S}\right)\right].$$

Proof. Let T be any independent set, in particular T^*. We observe that the algorithm ensures that for any vertex v, $\mathbf{P}[v \in S_{\text{end}}] \leq p$ and hence $\mathbf{P}\left[v \in \hat{S}\right] \leq p$.

$$\begin{aligned}(1-p) f(T) &\leq \mathbf{E}[f(T \cup S_{\text{end}})] \quad \text{(Lemma 21)} \\ &\leq \mathbf{E}[f(S)] + k \max\left\{\frac{1-p}{p}, 1+\beta\right\} \mathbf{E}[w(S_{\text{end}})] \quad \text{(Lemma 25)} \\ &\leq \left(k \max\left\{\frac{1-p}{p}, 1+\beta\right\} + \left(\frac{1+\beta}{\beta}\right)\right) \mathbf{E}[w(S_{\text{end}})] \quad \text{(Lemma 23)} \\ &\leq \left(k \max\left\{\frac{1-p}{p}, 1+\beta\right\} + \left(\frac{1+\beta}{\beta}\right)\right) \mathbf{E}\left[f\left(\hat{S}\right)\right] \quad \text{(Lemma 22)}.\end{aligned}$$ ◀

It remains to choose $p \in [0,1]$ and $\beta > 0$ to minimize the RHS. Consider the term $\max\{(1-p)/p, 1+\beta\}$. If $(1-p)/p \geq 1+\beta$, then $p \geq 1/2$ (to force $(1-p)/p \geq 1$), and the RHS is minimized by taking β as large as possible – that is, such that $1+\beta = (1-p)/p$. If $(1-p)/p \leq 1+\beta$, then the RHS is minimized by taking p as small as possible – that is, such that $(1-p)/p = 1/p - 1 = 1+\beta$. Thus $(1-p)/p = 1+\beta$ at the optimum. In terms of just p, then, we have

$$\text{OPT} \leq \left(\frac{1}{1-p}\right)\left(\frac{k(1-p)}{p} + \frac{1-p}{1-2p}\right) \mathbf{E}\left[f\left(\hat{S}\right)\right] = \left(\frac{k}{p} + \frac{1}{1-2p}\right) \mathbf{E}\left[f\left(\hat{S}\right)\right].$$

(Here we note that $\beta = (1-2p)/p$, hence $(1+\beta)/\beta = (1-p)/(1-2p)$.)

In the special case of $k = 2$, as in matching, the RHS is

$$\text{OPT} \leq \left(\frac{2}{p} + \frac{1}{1-2p}\right) \mathbf{E}\left[f\left(\hat{S}\right)\right]$$

The RHS is minimized by $p = 1/3$, giving an approximation factor of 9.

For general k, the minimum is $2k + \sqrt{8k} + 1$.

It is easy to see that the primal-dual algorithm makes $O(n)$ evaluation calls to f and the overall running time is linear in the size of the graph. The results for the monotone and non-negative functions, together yield Theorem 6.

5 Concluding Remarks and Open Problems

We described $\Omega(\frac{1}{k})$-approximation algorithms for independent sets in two parameterized families of graphs that capture several problems of interest. Although the multilinear relaxation based framework yields such algorithms, the resulting algorithms are computationally expensive and randomized. We utilized ideas from streaming and primal-dual based algorithms to give simple and fast algorithms for inductively k-independent graphs with the additional property that they are deterministic for monotone functions. Our work raises several interesting questions that we summarize below.

- The CR scheme that we described in Section 3 is unable to distinguish k-perfectly orientable graphs and inductive k-independent graphs. Is a better bound possible for inductively k-independent graphs?
- Our combinatorial algorithms only apply to inductively k-independent graphs. Can we obtain combinatorial algorithms for k-perfectly orientable graphs? Even for MIS the only approach appears to be via primal rounding of the LP solution [34].
- Can we obtain deterministic $\Omega(\frac{1}{k})$-approximation algorithms for these graph classes when f is non-negative? Interval graphs seem to be a natural first step to consider.
- Are better approximation ratios achievable? For instance, can we obtain better than 1/4-approximation for monotone submodular function maximization in interval graphs? Can we prove better lower bounds under complexity theory assumptions or in the oracle model for interval graphs or other concrete special cases of interest?
- For both classes of graphs our algorithms are based on having an ordering that certifies that they belong to the class. For MWIS in k-simplicial and k-perfectly orientable graphs, [30] describes algorithms based on the Lovász number of a graph and the Lovász θ-function of a graph, and these algorithms do not require an ordering. It may be feasible to extend their approach to the submodular setting via the multilinear relaxation. However, the resulting algorithms are computationally quite expensive. It would be interesting to obtain fast algorithms for these classes of graphs (or interesting special cases) when the ordering is not explicitly given.

References

1 Shipra Agrawal, Yichuan Ding, Amin Saberi, and Yinyu Ye. Correlation robust stochastic optimization. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1087–1096. SIAM, 2010.

2 Karhan Akcoglu, James Aspnes, Bhaskar DasGupta, and Ming-Yang Kao. Opportunity-cost algorithms for combinatorial auctions. In E. J. Kontoghiorghes, B. Rustem, and S. Siokos, editors, *Applied Optimization 74: Computational Methods in Decision-Making, Economics and Finance*, pages 455–479. Kluwer Academic Publishers, 2002.

3 A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause. Streaming submodular optimization: Massive data summarization on the fly. In *Proc. 20th ACM Conf. Knowl. Disc. and Data Mining* (KDD), pages 671–680, 2014.

4 Eric Balkanski and Yaron Singer. The adaptive complexity of maximizing a submodular function. In *Proceedings of the 50th annual ACM SIGACT symposium on theory of computing*, pages 1138–1151, 2018.

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 Jeff A. Bilmes. Submodularity in machine learning and artificial intelligence. *CoRR*, abs/2202.00132, 2022. arXiv:2202.00132.

7 Simon Bruggmann and Rico Zenklusen. Submodular maximization through the lens of linear programming. *Mathematics of Operations Research*, 44(4):1221–1244, 2019.

8 N. Buchbinder, M. Feldman, J. Naor, and R. Schwartz. Submodular maximization with cardinality constraints. In *Proc. 25th ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 1433–1452, 2014.

9 Niv Buchbinder and Moran Feldman. Deterministic algorithms for submodular maximization problems. *ACM Transactions on Algorithms (TALG)*, 14(3):1–20, 2018.

10 Niv Buchbinder and Moran Feldman. Submodular functions maximization problems. In *Handbook of Approximation Algorithms and Metaheuristics, Second Edition*, pages 753–788. Chapman and Hall/CRC, 2018.

11 Niv Buchbinder, Moran Feldman, and Mohit Garg. Deterministic (1/2+ ε)-approximation for submodular maximization over a matroid. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 241–254. SIAM, 2019.

12 Niv Buchbinder, Moran Feldman, and Roy Schwartz. Online submodular maximization with preemption. *ACM Transactions on Algorithms (TALG)*, 15(3):1–31, 2019.

13 G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint (extended abstract). In *Proc. 12th Int. Conf. Int. Prog. Comb. Opt.* (IPCO), pages 182–196, 2007.

14 Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

15 Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: matchings, matroids, and more. *Mathematical Programming*, 154:225–247, 2015.

16 Timothy M Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, 2012.

17 Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. Streaming algorithms for submodular function maximization. In *Automata, Languages, and Programming: 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I 42*, pages 318–330. Springer, 2015. Longer version: http://arxiv.org/abs/1504.08024.

18 Chandra Chekuri, T. S. Jayram, and Jan Vondrák. On multiplicative weight updates for concave and submodular function maximization. In Tim Roughgarden, editor, *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 201–210. ACM, 2015. doi:10.1145/2688073.2688086.

19 Chandra Chekuri and Vasilis Livanos. On submodular prophet inequalities and correlation gap. *arXiv preprint arXiv:2107.03662*, 2021.

20 Chandra Chekuri and Kent Quanrud. Submodular function maximization in parallel via the multilinear relaxation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 303–322. SIAM, 2019.

21 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.

22 Andrew Clark, Basel Alomair, Linda Bushnell, and Radha Poovendran. *Submodularity in dynamics and control of networked systems.* Springer, 2016.

23 Alina Ene, Huy L Nguyen, and Adrian Vladu. Submodular maximization with matroid and packing constraints in parallel. In *Proceedings of the 51st annual ACM SIGACT symposium on theory of computing*, pages 90–101, 2019.

24 Moran Feldman. *Maximization problems with submodular objective functions.* PhD thesis, Computer Science Department, Technion, 2013.

25 Moran Feldman, Amin Karbasi, and Ehsan Kazemi. Do less, get more: Streaming submodular maximization with subsampling. *Advances in Neural Information Processing Systems*, 31, 2018.

26 Moran Feldman, Joseph Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 570–579. IEEE, 2011.

27 Moran Feldman, Joseph Naor, Roy Schwartz, and Justin Ward. Improved approximations for k-exchange systems. In *Algorithms–ESA 2011: 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings 19*, pages 784–798. Springer, 2011.

28 M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions – II. *Math. Prog. Studies*, 8:73–87, 1978.

29 Paritosh Garg, Linus Jordan, and Ola Svensson. Semi-streaming algorithms for submodular matroid intersection. *Mathematical Programming*, pages 1–24, 2022.

30 Magnús M. Halldórsson and Tigran Tonoyan. Computing inductive vertex orderings. *Information Processing Letters*, 172:106159, 2021. doi:10.1016/j.ipl.2021.106159.

31 Kai Han, zongmai Cao, Shuang Cui, and Benwei Wu. Deterministic approximation for submodular maximization over a matroid in nearly linear time. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 430–441. Curran Associates, Inc., 2020. URL: https://proceedings.neurips.cc/paper/2020/file/05128e44e27c36bdba71221bfccf735d-Paper.pdf.

32 Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Math*, 182, 1999.

33 Th Jenkyns. The efficacy of the" greedy" algorithm. In *Proc. 7th Southeastern Conf. on Combinatorics, Graph Theory and Computing*, pages 341–350, 1976.

34 Frank Kammer and Torsten Tholey. Approximation algorithms for intersection graphs. *Algorithmica*, 68(2):312–336, 2014.

35 Roie Levin and David Wajc. Streaming submodular matching meets the primal-dual method. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1914–1933. SIAM, 2021.

36 G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions – I. *Math. Prog.*, 14(1):265–294, 1978.

37 Ami Paz and Gregory Schwartzman. A (2+ ε)-approximation for maximum weight matching in the semi-streaming model. *ACM Transactions on Algorithms (TALG)*, 15(2):1–15, 2018.

38 Rom Pinchasi. A finite family of pseudodiscs must include a “small” pseudodisc. *SIAM Journal on Discrete Mathematics*, 28(4):1930–1934, 2014.

39 Jan Vondrák. *Submodularity in combinatorial optimization.* PhD thesis, Univerzita Karlova, Matematicko-fyzikální fakulta, 2007.

40 Yuli Ye and Allan Borodin. Elimination graphs. *ACM Transactions on Algorithms (TALG)*, 8(2):1–23, 2012.

41 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 681–690, 2006.

A A Preemptive Greedy Algorithm

We now describe a preemptive greedy algorithm for maximizing a monotone submodular function $f : 2^V \to \mathbb{R}_+$ over independent sets of a inductively k-independent graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ assuming that we are also given the ordering. The algorithm is simple and intuitive, and is inspired by algorithms developed in the streaming model.

The pseudocode for the algorithm is given in Figure 4, and is designed as follows. Starting from an empty solution $S = \emptyset$, `preemptive-greedy` processes the vertices in the given ordering one by one. When considering v_i, the algorithm gathers the subset $C_i \subseteq S$ of all vertices in the current set S that are neighbors of v_i (those that conflict with v_i). The algorithm has to decide whether to reject v_i or to accept v_i in which case it has to remove C_i from S. It accepts v_i if the marginal gain $f_S(v_i) \overset{\text{def}}{=} f(S + v_i) - f(S)$ of adding v_i directly to S is at least $(1 + \beta)$ times the value $\sum_{u \in C_i} f_{S \setminus C_i}(u)$. Here $\beta > 0$ is a parameter that is fixed based on the analysis. After processing all vertices, we return the final set S.

```
preemptive-greedy($\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $f : 2^{\mathcal{V}} \to \mathbb{R}_{\geq 0}$, $k \in \mathbb{N}$, $\beta \in \mathbb{R}_{>0}$).
1. Let $S = \emptyset$. Let $\mathcal{V} = \{v_1, \ldots, v_n\}$ by a $k$-independence ordering of $\mathcal{V}$
2. For $i = 1, \ldots, n$:
   A. Let $C_i = N(v_i) \cap S = \{u \in S : uv_i \in \mathcal{E}\}$
   B. If $f_S(v_i) \geq (1 + \beta) \sum_{u \in C_i} \nu(f, S, u)$
      1. Set $S \leftarrow (S \setminus C_i) + v_i$
3. Return $S$
```

Figure 4 The algorithm `preemptive-greedy` for finding an independent set in a inductively k-independent graph to maximize a monotone submodular objective function.

`preemptive-greedy` for inductively k-independent graphs has the following approximation bound. The proof is deferred to the subsection following the theorem statements.

▶ **Theorem 27.** *Given an inductively k-independent graph with a k-inductive ordering, the algorithm* `preemptive-greedy` *returns an independent set $\hat{S}$ such that for any independent set T, $f(T) \leq (k(1 + \beta) + 1)(1 + \beta^{-1}) f(\hat{S})$.*

`preemptive-greedy` can be extended to nonnegative (and non-monotone) submodular functions with a constant factor loss in approximation by random sampling. As a preprocessing step, we let $\mathcal{V}'$ randomly sample each vertex in $\mathcal{V}$ independently with probability $1/2$. We then apply `preemptive-greedy` to the subgraph $\mathcal{G}' = \mathcal{G}[\mathcal{V}']$ induced by $\mathcal{V}'$. It is easy to see that any subgraph of an inductively k-independent graph is also inductively k-independent. The net effect of the random sampling is an approximation factor for nonnegative submodular functions that is a factor 4 worse than for the monotone case. The modified algorithm, called `randomized-preemptive-greedy`, is given in Figure 5.

▶ **Theorem 28.** *Given an inductively k-independent graph with a k-inductive ordering, the algorithm* `randomized-preemptive-greedy` *returns an independent set $\hat{S}$ such that for any independent set T, $f(T) \leq 4(k(1 + \beta) + 1)(1 + \beta^{-1}) f(\hat{S})$.*

```
randomized-preemptive-greedy($\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $f : 2^{\mathcal{V}} \to \mathbb{R}_{\geq 0}$, $k \in \mathbb{N}$, $\beta \in \mathbb{R}_{>0}$).
1. Let $\mathcal{V}' \subseteq \mathcal{V}$ sample each $v \in \mathcal{V}$ independently with probability $1/2$
2. Let $\mathcal{G}' = \mathcal{G}[\mathcal{V}']$ be the subgraph of $\mathcal{G}$ induced by $\mathcal{V}'$
3. Return preemptive-greedy($\mathcal{G}'$, $f : 2^{\mathcal{V}'} \to \mathbb{R}_{\geq 0}$, $k \in \mathbb{N}$, $\beta \in \mathbb{R}_{>0}$).
```

Figure 5 The algorithm `randomized-preemptive-greedy` for finding an independent set in an inductively k-independent graph to maximize a nonnegative submodular objective function.

▶ Remark 29. The randomized strategy we outline is simple and oblivious. It loses a factor of 4 over the monotone case. One could try to improve the approximation ratio by using randomization within the algorithm which would make the analysis more involved. However, we have not done this since the primal-dual algorithm yields better approximation bounds. This subsampling strategy is not new and has been used previously in [25], and is also implicit in [17].

The rest of the section is devoted to proving the claimed approximation guarantees.

A.1 Analysis of preemptive greedy

We follow the notation of [17]. Let $\hat{S}$ be the final set of vertices returned by `preemptive-greedy`. It is easy to see that the algorithm returns an independent set. For each $u \in \mathcal{V}$ let S_u^- denote the set of vertices in S just before u is processed, and let S_u^+ denote the set after u is processed. Thus a vertex u is added to S iff $S_u^+ \setminus S_u^- = \{u\}$. Let $U = \bigcup_{u \in \mathcal{V}} S_u^+$ be the set of all vertices that were ever (even momentarily) added to S. Alternatively, $\mathcal{V} \setminus U$ is the set of vertices that are discarded by the algorithm when it considers them. For each vertex u, let $\delta_u \overset{\text{def}}{=} f(S_u^+) - f(S_u^-)$ be the value added to S from processing u. We have $\delta_u = 0$ for all $u \notin U$, and $f(\hat{S}) = \sum_{u \in \mathcal{V}} \delta_u = \sum_{u \in U} \delta_u$.

Let $T \subseteq \mathcal{V}$ be an independent set in the given graph, in particular an optimum set. We would like to compare $f(\hat{S})$ with $f(T)$. Directly comparing T with $\hat{S}$ is difficult since $\hat{S}$ is obtained by deleting vertices in S along the way; thus a vertex $v \in T \setminus \hat{S}$ may have been discarded due to a vertex $u \in S$ when v was considered but u may not be in $\hat{S}$. Thus, the analysis is broken into two parts that detour through U. First, we relate the value of $f(\hat{S})$ to the value of $f(U)$. This part of the analysis bounds the amount of value lost by kicking out vertices from S during the exchanges. We then relate $f(U)$ and $f(T)$; this is easier because any vertex in T is always compared against *some* subset of vertices in U. Chaining the inequalities from $f(\hat{S})$ to $f(U)$ to $f(T)$ gives the final approximation ratio.

Relating $f(\hat{S})$ to $f(U)$

The analysis is similar to that in [17]. We provide proofs for the sake of completeness. The following claim is easy to see since elements before s can only be deleted from S as the algorithm proceeds.

▷ Claim 30. Over the course of the algorithm, the incremental value $\nu(f, S, s)$ of an element $s \in S$ is nondecreasing.

For a vertex $u \in U \setminus \hat{S}$ we let u' denote the vertex that caused u to be removed from S. And we let $\chi(u)$ denote its incremental value just before it is removed. Therefore, $\chi(u) = \nu\big(f, S_{u'}^-, u\big)$.

▶ **Lemma 31.** *Let $u \in U$ then $\delta_u \geq \beta \sum_{c \in C_u} \nu(f, S_u^-, c)$.*

Proof. Since the vertex u was added to S when it was considered, we have $\delta_u = f(S_u^+) - f(S_u^-)$ where $S_u^+ = S_u^- - C_u + u$. The vertex u was added by the algorithm since $f_S(u) \geq (1 + \beta) \sum_{c \in C_u} \nu(f, S, c)$ where $S = S_u^-$. Therefore $\beta \sum_{c \in C_u} \nu(f, S_u^-, c) \leq f_{S_u^-}(u) - \sum_{c \in C_u} \nu(f, S_u^-, c)$. It suffices to prove that $f(S_u^+) - f(S_u^-) \geq f_S(u) - \sum_{c \in C_u} \nu(f, S, c)$ which we do below. For notational convenience let $A = S_u^- - C_u$.

$$\begin{aligned} f(S_u^+) - f(S_u^-) &= f(A+u) - f(S_u^-) \\ &= f_A(u) + f(A) - f(S_u^-) \\ &\geq f_{S_u^-}(u) - (f(S_u^-) - f(A)) && \text{by submodularity since } A \subseteq S_u^- \\ &\geq f_{S_u^-}(u) - \sum_{c \in C_u} \nu\big(f, S_u^-, c\big) && \text{by submodularity and defn of } \nu. \end{aligned}$$ ◀

▶ **Lemma 32.** $\sum_{u \in U \setminus \hat{S}} \chi(u) \leq \beta^{-1} f(\hat{S})$.

Proof. Indeed,

$$\begin{aligned} \sum_{u \in U \setminus \hat{S}} \chi(u) &= \sum_{u \in U} \sum_{c \in C_u} \chi(c) && \text{since } \{C_u : u \in U\} \text{ partitions } U \setminus \hat{S} \\ &\leq \sum_{u \in U} \frac{1}{\beta} \sum_{u \in U} \delta_u && \text{from Lemma 31} \\ &= \frac{1}{\beta} f(\hat{S}). \end{aligned}$$ ◀

The next lemma shows that $f(U)$ is not much larger than $f(\hat{S})$.

▶ **Lemma 33.** $f(U) \leq \big(1 + \beta^{-1}\big) f(\hat{S})$.

Proof. Let $U' = U \setminus \hat{S}$ and let $U' = \{v_{i_1}, \ldots, v_{i_h}\}$ where $i_1 < i_2 \ldots < i_h$. We have $f(U) = f(\hat{S}) + f_{\hat{S}}(U')$. It suffices to upper bound $f_{\hat{S}}(U')$ by $f(\hat{S})/\beta$. For $1 \leq j \leq h$ let $U'_j = \{v_{i_1}, \ldots, v_{i_j}\}$. We have $f_{\hat{S}}(U') = \sum_{j=1}^{h} f_{\hat{S} \cup U'_{j-1}}(v_{i_j})$. We claim that $f_{\hat{S} \cup U'_{j-1}}(v_{i_j}) \leq \chi\big(v_{i_j}\big)$. This follows by submodularity and the fact that $\hat{S} \cup U'_{j-1}$ is a superset of the vertices that are in S when v_{i_j} is deleted. Putting things together,

$$f_{\hat{S}}(U') = \sum_{j=1}^{h} f_{\hat{S} \cup U'_{j-1}}(v_{i_j}) \leq \sum_{u \in U'} \chi(u) \leq \frac{1}{\beta} f(\hat{S})$$

where the last inequality follows from Lemma 32. ◀

Relating OPT to $f(U)$

It remains to bound $f(T)$ (for some competing set T) to $f(U)$ and hence to $f(\hat{S})$. The critical question, addressed in the following lemmas, is how to charge the value of elements in T off to elements in U.

▶ **Lemma 34.** *Let $T \subseteq \mathcal{V}$ be an independent set disjoint from U. Each element $u \in U$ appears in the conflict list C_t for at most k vertices $t \in T$.*

Proof. Fix $u \in U$. The set $T \cap N(u) \cap \{v : v > u\}$ consists of precisely the vertices $t \in T$ for which $u \in C_t$. As a subset of T, this set is certainly independent. By definition of k-inductive independence, the cardinality of this set is at most k. ◀

▶ **Lemma 35.** *Let $T \subseteq \mathcal{V}$ be an independent set. Then*

$$f_U(T) \leq k(1+\beta)(1+\beta^{-1}) f(\hat{S}).$$

Proof. Since $f_U(T) = f_U(T \setminus U)$, it suffices to assume that T is disjoint from U. For each vertex $t \in T$, since t is not in U, we have $f_{S_t^-}(t) \le (1+\beta)\sum_{c \in C_t} \nu\big(f, S_t^-, c\big)$. Fix a vertex $u \in C_t$. If $u \in \hat{S}$, then u is in the final output; then we have $\nu\big(f, S_t^-, u\big) \le \nu\Big(f, \hat{S}, u\Big)$ because the incremental value of an element in S is nondecreasing. If $u \notin \hat{S}$, and u was deleted to make room for some later element u', then we have $\nu\big(f, S_t^-, u\big) \le \chi(u)$ again because incremental values are nondecreasing.

By the preceding lemma, each element $u \in U$ appears in C_t for at most k choices of t. Therefore, in sum, we have

$$\begin{aligned} f_U(T) &\le \sum_{t \in T} f_{S_t^-}(t) && \text{by submodularity,} \\ &\le (1+\beta)\sum_{t \in T}\sum_{c \in C_t} \nu\big(f, S_t^-, c\big) && \text{since } t \notin U, \\ &\le k(1+\beta)\left(\sum_{u \in \hat{S}} \nu\Big(f, \hat{S}, u\Big) + \sum_{u \in U \setminus \hat{S}} \chi(u)\right) && \text{Lemma 34 and argument above,} \\ &\le k(1+\beta)\left(f(\hat{S}) + \sum_{u \in U \setminus \hat{S}} \chi(u)\right) \\ &\le k(1+\beta)(1+\beta^{-1})f(\hat{S}) && \text{by Lemma 32} \end{aligned}$$

as desired. ◀

From here, it is relatively straightforward to get a final approximation bound.

▶ **Theorem 36.** *Given an inductively k-independent graph with a k-inductive ordering, the algorithm* `preemptive-greedy` *returns an independent set $\hat{S}$ such that for any independent set T,*

$$f(T) \le (k(1+\beta)+1)(1+\beta^{-1})f(\hat{S}).$$

Proof. Let T be an optimal solution. We have

$$f(T) \le f_U(T) + f(U) \le (k(1+\beta)+1)(1+\beta^{-1})f(\hat{S}) \tag{1}$$

via Lemma 35 and Lemma 33. ◀

The bound is minimized by taking $\beta = \sqrt{1+k^{-1}}$, which at which point

$$f(T) \le (4k+2+o(1))f(\hat{S}),$$

where the $o(1)$ goes to 0 as k increases. For $k = 1$, the approximation ratio is $3 + 2\sqrt{2}$.

A.2 Randomized preemptive greedy for nonnegative functions

Here we analyze the `randomized-preemptive-greedy` for non-negative submodular functions that may not be monotone. A key observation is that the analysis of `preemptive-greedy` does not invoke the monotonicity of f until the very end, in equation (1). In particular, Lemma 35 and Lemma 33 hold for nonnegative submodular functions.

(1) invokes monotonicity when it takes the inequality $f(U \cup T) \geq f(T)$. Informally speaking, by injecting randomization, we will be able recover a similar inequality, except losing a factor of 4.

Fix a set T. Let $\mathcal{V}'$ sample each element in $\mathcal{V}$ with probability 1/2. Let $T' = T \cap \mathcal{V}'$. Conditional on $\mathcal{V}'$, we have

$$f(U) \leq \left(1 + \beta^{-1}\right) f\left(\hat{S}\right)$$

and

$$f_U(T') \leq k(1+\beta)(1+\beta)^{-1} f\left(\hat{S}\right)$$

via Lemma 33 and Lemma 35 respectively.

Now, conditional on T', $U \setminus T = U \setminus T'$ is a randomized set, where any vertex $v \in \mathcal{V}$ appears in $U \setminus T$ with probability at most 1/2. By Lemma 21,

$$\mathbf{E}[f(U \cup T') \mid T'] \geq \frac{1}{2} f(T').$$

We also have, via the concavity of F along any non-negative direction [39],

$$\mathbf{E}[f(T')] = F(\frac{1}{2}\mathbb{1}_T) \geq \frac{1}{2} F(\mathbb{1}_T) = \frac{1}{2} f(T)$$

where $\mathbb{1}_T$ is the indicator vector of T.

Altogether, we have

$$\begin{aligned} f(T) &\leq 2\,\mathbf{E}[f(T')] \leq 4\,\mathbf{E}[f(U \cup T')] \\ &= 4\,\mathbf{E}[f_U(T') + f(U)] \leq 4(k(1+\beta)+1)(1+\beta)^{-1}\,\mathbf{E}\left[f\left(\hat{S}\right)\right], \end{aligned}$$

as desired.

Improved Diversity Maximization Algorithms for Matching and Pseudoforest

Sepideh Mahabadi ✉
Microsoft Research, Redmond, WA, USA

Shyam Narayanan[1] ✉
Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

In this work we consider the diversity maximization problem, where given a data set X of n elements, and a parameter k, the goal is to pick a subset of X of size k maximizing a certain diversity measure. Chandra and Halldórsson [11] defined a variety of diversity measures based on pairwise distances between the points. A constant factor approximation algorithm was known for all those diversity measures except "remote-matching", where only an $O(\log k)$ approximation was known. In this work we present an $O(1)$ approximation for this remaining notion. Further, we consider these notions from the perpective of composable coresets. Indyk et al. [23] provided composable coresets with a constant factor approximation for all but "remote-pseudoforest" and "remote-matching", which again they only obtained a $O(\log k)$ approximation. Here we also close the gap up to constants and present a constant factor composable coreset algorithm for these two notions. For remote-matching, our coreset has size only $O(k)$, and for remote-pseudoforest, our coreset has size $O(k^{1+\varepsilon})$ for any $\varepsilon > 0$, for an $O(1/\varepsilon)$-approximate coreset.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Theory of computation → Computational geometry

Keywords and phrases diversity maximization, approximation algorithms, composable coresets

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.25

Category APPROX

Related Version *Full Version*: http://arxiv.org/abs/2307.04329

1 Introduction

Diverse Subset Selection is the task of searching for a subset of the data that preserves its diversity as much as possible. This task has a large number of applications in particular while dealing with large amounts of data, including data summarization (e.g. [26, 18, 25]), search and information retrieval (e.g. [5, 2, 24, 13, 31]), and recommender systems (e.g. [33, 1, 34, 32]), among many others (e.g. [17, 29]). Here, given a ground set of n vectors X in a metric space $(\mathcal{X}, \rho)$, representing a data set of objects (e.g. using their feature vectors), and a parameter k, the goal is to choose a subset $S \subseteq X$ of this data set of size k, that maximizes a pre-specified optimization function measuring the diversity.

Many diversity measures have been introduced and used in the literature that fit different tasks. A large number of these measures are defined based on pairwise distances between the vectors in X. In particular the influential work of [11] introduced a taxonomy of pairwise-distance based diversity measures which is shown in Table 1. For example, remote-edge measures the distance of the closest points picked in the subset S, and remote-clique measures the sum of pairwise distances between the points in the subset S. Remote-pseudoforest falls between these two where it wants to ensure that the average distance of a point to its nearest

[1] Work done as an intern at Microsoft Research.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 25; pp. 25:1–25:22
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Table 1 This table includes the notions of diversity considered by [11] ($S = S_1|...|S_t$ is used to denote that $S_1 \dots S_t$ is a partition of S into t sets). We also include the best previously-known approximation factors, both in the standard (offline) and coreset setting. If not explicitly stated, the approximation factor holds for both the offline setting and the coreset setting. We note that the previously known $O(1)$-approximate remote-pseudoforest offline algorithm is randomized, whereas the rest of the previously known algorithms are deterministic.

Problem	Diversity of the point set S	Apx factor
Remote-edge	$\min_{x,y \in S} dist(x, y)$	$O(1)$
Remote-clique	$\sum_{x,y \in S} dist(x, y)$	$O(1)$
Remote-tree	$wt(MST(S))$, weight of the minimum spanning tree of S	$O(1)$
Remote-cycle	$\min_C wt(C)$ where C is a TSP tour on S	$O(1)$
Remote t-trees	$\min_{S=S_1\|...\|S_t} \sum_{i=1}^{t} wt(MST(S_i))$	$O(1)$
Remote t-cycles	$\min_{S=S_1\|...\|S_t} \sum_{i=1}^{t} wt(TSP(S_i))$	$O(1)$
Remote-star	$\min_{x \in S} \sum_{y \in S \setminus \{x\}} dist(x, y)$	$O(1)$
Remote-bipartition	$\min_B wt(B)$, where B is a bipartition (i.e., bisection) of S	$O(1)$
Remote-pseudoforest	$\sum_{x \in S} \min_{y \in S \setminus \{x\}} dist(x, y)$	$O(1)$ (Offline) $O(\log k)$ (Coreset)
Remote-matching	$\min_M wt(M)$, where M is a perfect matching of S	$O(\log k)$

neighbor is large. Remote-matching measures the diversity as the cost of minimum-weight-matching. Various other measures have also been considered: Table 1 includes each of their definitions along with the best known approximation factor for these measures known up to date. In particular, by [11] it was known that all these measures except remote-pseudoforest and remote-matching admit a constant factor approximation. More recently, [7] showed a constant factor randomized LP-based algorithm for remote-pseudoforest. They also showed the effectiveness of the remote-pseudoforest measure on real data over the other two common measures (remote-edge and remote-clique). On the lower bound side, it was known by [20] that for remote-matching, one cannot achieve an approximation factor better than 2. However, despite the fact that there has been a large body of work on diversity maximization problems [8, 3, 9, 10, 14], the following question had remained unresolved for over two decades.

▶ **Question 1.** *Is it possible to get an $O(1)$ approximation algorithm for the remaining notion of remote-matching?*

Later following a line of work on diversity maximization in big data models of computations, [23] presented algorithms producing a composable coreset for the diversity maximization problem under all the aforementioned diversity measures. An α-approximate composable coreset for a diversity objective is a mapping that given a data set X, outputs a small subset $C \subset X$ with the following composability property: given multiple data sets $X^{(1)}, \cdots, X^{(m)}$, the maximum achievable diversity over the union of the composable coresets $\bigcup_i C^{(i)}$ is within an α factor of the maximum diversity over the union of those data sets $\bigcup_i X^{(i)}$. It is shown that composable coresets naturally lead to solutions in several massive data processing models including distributed and streaming models of computations, and this has lead to recent interest in composable coresets since their introduction [28, 6, 27, 22, 4, 15]. [23] showed α-approximate compsable coresets again for all measures of diversity introduced by [11]. They presented constant factor α-approximate composable coresets for all diversity measures except remote-pseudoforest and remote-matching, where they provided only $O(\log k)$-approximations for these measures. Again the following question remained open.

▶ **Question 2.** *Is it possible to get a constant factor composable coreset for the remote-pseudoforest and remote-matching objective functions?*

In this work we answer above two questions positively and close the gap up to constants for these problems.

Our Results

In this work, we resolve a longstanding open question of [11] by providing polynomial-time $O(1)$-approximation algorithms for the remote-matching problem. We also resolve a main open question of [23] by providing polynomial-time algorithms that generate $O(1)$-approximate composable coresets for both the remote-pseudoforest and remote-matching problems. Hence, our paper establishes $O(1)$-approximate offline algorithms *and* $O(1)$-approximate composable coresets for *all* remaining diversity measures proposed in [11]. Specifically, we have the following theorems.

▶ **Theorem 3** (Remote-Matching, Offline Algorithm). *Given a dataset $X = \{x_1, \ldots, x_n\}$ and an integer $k \le \frac{n}{3}$, w.h.p. Algorithm 4 outputs an $O(1)$-approximate set for remote-matching.*

While we assume n is at least a constant factor bigger than k, this is usually a standard assumption (for instance both the remote-pseudoforest and remote-matching algorithms in [11] assume $n \ge 2k$, and the $O(1)$-approximate remote-pseudoforest algorithm in [7] assumes $n \ge 3k$). We now state our theorems regarding composable coresets.

▶ **Theorem 4** (Pseudoforest, Composable Coreset). *Given a dataset $X = \{x_1, \ldots, x_n\}$ and an integer $k \le n$, Algorithm 2 outputs an $O(1/\varepsilon)$-approximate composable coreset C for remote-pseudoforest, of size at most $O(k^{1+\varepsilon})$.*

By this, we mean that if we partitioned the dataset X into $X^{(1)}, \ldots, X^{(m)}$, and ran the algorithm separately on each piece to obtain $C^{(1)}, \ldots, C^{(m)}$, each $C^{(i)}$ will have size at most $O(k^{1+\varepsilon})$ and $\max_{Z \subset C : |Z| = k} \mathrm{PF}(Z) \ge \Omega(\varepsilon) \cdot \max_{Z \subset X : |Z| = k} \mathrm{PF}(Z)$, where $C = \bigcup_{i=1}^{m} C^{(i)}$.

▶ **Theorem 5** (Minimum-Weight Matching, Composable Coreset). *Given a dataset $X = \{x_1, \ldots, x_n\}$ and an integer $k \le n$, Algorithm 3 outputs an $O(1)$-approximate composable coreset C for remote-matching, of size at most $3k$.*

In all of our results, we obtain $O(1)$-approximation algorithms, whereas the previous best algorithms for all 3 problems was an $O(\log k)$-approximation algorithm, meaning the diversity was at most $\Omega(\frac{1}{\log k})$ times the optimum. We remark that our composable coreset in Theorem 4 has size $k^{1+\varepsilon}$, for some arbitrarily small constant ε (to obtain a constant approximation) which is possibly suboptimal. We hence ask an open question as to whether an $O(1)$-approximate composable coreset for remote-pseudoforest, of size $O(k)$, exists.

Finally, as an additional result we also prove an alternative method of obtaining an $O(1)$-approximate offline algorithm for remote-pseudoforest. Unlike [7], which uses primal-dual relaxation techniques, our techniques are much simpler and are based on ε-nets and dynamic programming. In addition, our result works for all $k \le n$, whereas the work of [7] assumes $k \le \frac{n}{3}$ and that k is at least a sufficiently large constant. Also, our algorithm is deterministic, unlike [7]. We defer the statement and proof to the full version of the paper on arXiv.

2 Preliminaries

2.1 Definitions and Notation

We use $\rho(x, y)$ to represent the metric distance between two points x and y. For a point x and a set S, we define $\rho(x, S) = \min_{s \in S} \rho(x, s)$. Likewise, for two sets S, T, we define $\rho(S, T) = \min_{s \in S, t \in T} \rho(S, T)$. We define the diameter of a dataset X as $\text{diam}(X) = \max_{x,y \in X} \rho(x, y)$.

Costs and diversity measures

For a set of points $Y = \{y_1, \ldots, y_k\}$, we use $\text{div}(Y)$, as a generic term to denote its diversity which we measure by the following cost functions.

If $k = |Y|$ is even, we define the minimum-weight matching cost $\text{MWM}(Y)$ as the minimum total weight over all perfect matchings of Y. Equivalently,

$$\text{MWM}(Y) := \min_{\text{permutation } \pi:[k]\to[k]} \sum_{i=1}^{k/2} \rho(y_{\pi(2i)}, y_{\pi(2i-1)}).$$

Likewise, we define the pseudoforest cost $\text{PF}(Y)$, also known as the sum-of-nearest-neighbor cost, of Y as

$$\text{PF}(Y) := \sum_{y \in Y} \rho(y, Y \backslash y).$$

Finally, we define the minimum spanning tree cost $\text{MST}(Y)$ of Y as the minimum total weight over all spanning trees of Y. Equivalently,

$$\text{MST}(Y) = \min_{G\text{: spanning tree of } [k]} \sum_{e=(i,j)\in G} \rho(y_i, y_j).$$

The pseudoforest cost and minimum spanning tree cost do not require Y to be even.

▶ **Definition 6** (Diversity maximization). *Given a dataset X, and a parameter k, the goal of the diversity maximization problem is to choose a subset $Y \subset X$ of size k with maximum diversity, where in this work we focus on $\text{div}(Y) = \text{MWM}(Y)$ and $\text{div}(Y) = \text{PF}(Y)$.*

We also define $\text{div}_k(X)$ to be this maximum achievable diversity, i.e., $\text{div}_k(X) = \max_{Y \subset X, |Y|=k} \text{div}(Y)$. In particular we use $\text{MWM}_k(X)$, or the remote-matching *cost of X, as $\max_{Y \subset X, |Y|=k} \text{MWM}(Y)$, and define $\text{PF}_k(X) = \max_{Y \subset X, |Y|=k} \text{PF}(Y)$. These objectives are also known as k-*matching *and k-*pseudoforest*.*

For a specific diversity maximization objective div_k (such as MWM_k), an *α-approximation algorithm* ($\alpha \ge 1$) for div is an algorithm that, given any dataset $X = \{x_1, \ldots, x_n\}$, outputs some dataset $Z \subset X$ of size k such that

$$\text{div}(Z) \ge \frac{1}{\alpha} \cdot \text{div}_k(X) = \frac{1}{\alpha} \cdot \max_{Y \subset X : |Y|=k} \text{div}(Y).$$

▶ **Definition 7** (Composable coresets). *We say that an algorithm $\mathcal{A}$ that acts on a dataset X and outputs a subset $\mathcal{A}(X) \subset X$ forms an α-*approximate composable coreset *($\alpha \ge 1$) for* div *if, for any collection of datasets $X^{(1)}, \ldots, X^{(m)}$, we have*

$$\text{div}_k\left(\bigcup_{i=1}^m \mathcal{A}(X^{(i)})\right) \ge \frac{1}{\alpha} \cdot \text{div}_k\left(\bigcup_{i=1}^m X^{(i)}\right).$$

Algorithm 1 The GMM Algorithm.

```
Input: data $X = \{x_1, \dots, x_n\}$, integer $k$.
$y_1$ is an arbitrary point in $X$.
Initialize $Y \leftarrow \{y_1\}$.
for $p = 2$ to $k$ do
    $y_p \leftarrow \arg\max_{y \in X} \rho(y, Y)$, i.e., $y_p$ is the furthest point in $X$ from the current $Y = \{y_1, \dots, y_{p-1}\}$.
    $Y \leftarrow Y \cup \{y_p\}$.
end for
Return $Y$.
```

Throughout the paper, for our coreset construction algorithms, we use $X^{(1)}, \cdots, X^{(m)}$ to denote the collection of the data sets. Note that since $\mathcal{A}(X^{(i)}) \subset X^{(i)}$, the diversity of the combined coresets is always at most the diversity of the combined original datasets. We also say that the coreset is of size k' if $|\mathcal{A}(X^{(i)})| \le k'$ for each $X^{(i)}$. We desire for the size k' to only depend (polynomially) on k, and not on n.

2.2 The GMM Algorithm

The GMM algorithm [19] is an algorithm that was first developed for the k-center clustering, but has since been of great use in various diversity maximization algorithms and dispersion problems, starting with [30]. The algorithm is a simple greedy procedure that finds k points Y in a dataset X that are well spread out. It starts by picking an arbitrary point $y_1 \in X$. Given $y_1, \dots, y_p$ for $p < k$, it chooses y_{p+1} as the point that maximizes the distance $\rho(y, \{y_1, \dots, y_p\})$ over all choices of $y \in X$. We provide pseudocode in Algorithm 1.

The GMM algorithm serves as an important starting point in many of our algorithms, as well as in many of the previous state-of-the-art algorithms for diversity maximization. The GMM algorithm has the following crucial property.

▶ **Proposition 8.** *Suppose we run GMM for k steps to produce $Y = \{y_1, \dots, y_k\}$. Let $r = \max_{x \in X} \rho(x, Y)$. Then, every pair of points y_i, y_j has $\rho(y_i, y_j) \ge r$.*

3 Composable Coreset Constructions

In this section, we design algorithms for constructing $O(1)$-approximate composable coresets for remote-pseudoforest and remote-matching. In this section, we provide a technical overview and pseudocode for the algorithms, but defer the proof (and algorithm descriptions in words) to Appendix A (for remote-pseudoforest) and Section 5 (for remote-matching).

3.1 Coreset Constructions: Technical Overview

For both remote-pseudoforest and remote-matching, we start by considering the heuristic of GMM, where in each group we greedily select k points. For simplicity, suppose we only have one group for now. After picking the set $Y = \{y_1, \dots, y_k\}$ from GMM, define r to be the maximum distance $\rho(x, Y)$ over all remaining points x. Then, Proposition 8 implies that all distances $\rho(y_j, y_{j'}) \ge r$ for $j, j' \le k$. Hence, running GMM will ensure we have a set of k points with minimum-weight-matching or pseudoforest cost at least $\Omega(k \cdot r)$. Hence, we only fail to get an $O(1)$-approximation if the optimum remote-matching (or remote-pseudoforest) cost is much larger than $k \cdot r$.

However, in this case, note that every point $x \in X$ satisfies $\rho(x, Y) \leq r$, meaning every point x is within r of some y_i. This suggests that if the optimum cost is $\omega(k \cdot r)$, achieved by some points $z_1, \dots, z_k$, we could just map each z_i to its closest y_i, and this would change each distance by no more than $O(k \cdot r)$. Hence, we can ostensibly use the GMM points to obtain a cost within $O(k \cdot r)$ of the right answer, which is within an $\Omega(1)$ (in fact a $1 - o(1)$) multiplicative factor! Additionally, this procedure will compose nicely, because if we split the data into m components $X^{(1)}, \dots, X^{(m)}$, each with corresponding radius $r^{(1)}, \dots, r^{(m)}$, then each individual coreset has cost at least $r^{(j)}$ (so the combination has cost at least $\max r^{(i)}$), whereas we never move a point more than $r^{(i)} \leq \max r^{(i)}$.

The problem with this, however, is that we may be using each y_j multiple times: for instance, if both z_1 and z_2 are closest to y_1, we would use y_1 twice. Our goal is to find a subset of k points, meaning we cannot duplicate any point.

Note that in this duplication, it is never necessary to duplicate a point more than k times. So, if we could somehow pick k copies of each y_i, we would have a coreset. However, note that it is not crucial for each z_i to be mapped to its closest GMM point y_j: any point within distance r of y_j is also acceptable. Using this observation, it suffices to pick k points among those closest to y_j if possible - if there are fewer than k points, picking all of the points is sufficient. It is also important to choose all of Y, in the case where the optimum cost is only $O(k \cdot r)$. Together, this generates a composable coreset for both remote-matching and remote-pseudoforest, of size only k^2.

3.1.1 Improving the coreset for remote-matching

In the case of remote-matching, we can actually improve this to $O(k)$. The main observation here is to show that if a set Z of size k had two identical points, getting rid of both of them does not affect the minimum-weight-matching cost. (This observation does *not* hold for pseudoforest). One can similarly show that if the two points were close in distance, removing both of the points does not affect the matching cost significantly. This also implies we can, rather than removing both points, move them both to a new location as long as they are close together. At a high level, this means that there must exist a near-optimal k-matching that only has $O(1)$ points closest to each y_j: as a result we do not have to store k points for each y_j: only $O(1)$ points suffice.

3.1.2 Improving the coreset for remote-pseudoforest

In the case of remote-pseudoforest, the improvement is more involved. Consider a single group, and suppose GMM gives us the set $Y = \{y_1, \dots, y_k\}$. Let X_i represent the set of points in X closest to Y_i. The first observation we make is that if the optimal solution had multiple points in a single X_i, each such point can only contribute $O(r)$ cost. Assuming that the optimum cost is $\omega(k \cdot r)$, it may seem sufficient to simply pick 1 point in each Y_i for the coreset, as we can modify the optimum solution by removing points in X_i if there are two or more of them. While this will allow us to obtain a set with nearly optimum cost, the problem is the set has size less than k. So, we need to add additional points while preventing the pseudoforest cost from decreasing by too much.

To develop intuition for how this can be accomplished, we first suppose that $|X_1|, |X_2| \geq k$. In this case, we could choose the coreset as $X_1 \cup X_2 \cup Y$. We know there is a subset $Z \subset Y$ with pseudoforest cost close to optimum, though $|Z|$ may be much smaller than k. However, since y_1, y_2 are far away from each other (they were chosen first in the greedy procedure of GMM), so all points in X_1 and all points in X_2 are far from each other. That means

Algorithm 2 PFCoreset: $O(1)$-approximate remote-pseudoforest composable coreset algorithm.

```
Input: data $X = \{x_1, \dots, x_n\}$, integer $k$, parameter $\varepsilon \in (0, 1]$.
if $n < 2k^{1+\varepsilon} + k$ then
    Return $X$.
end if
$Y = \{y_1, \dots, y_k\} \leftarrow \mathrm{GMM}(x_1, \dots, x_n, k)$.
for $i = 1$ to $n$ do
    $\tilde{r}_i \leftarrow (k+1)^{\text{th}}$ largest value of $\rho(x_i, x_j)$ across all $j \le n$.
end for
$\tilde{r} \leftarrow \min_i \tilde{r}_i$, $x \leftarrow \arg\min_i \tilde{r}_i$
$U \leftarrow k$ furthest points in $X$ from $x$.
$P \leftarrow k$ arbitrary points within distance $\tilde{r}$ of $x$.
$S, T \leftarrow \mathrm{FindST}(X, k, \varepsilon, \tilde{r})$. {See Algorithm 5}
Return $C \leftarrow P \cup S \cup T \cup U \cup Y$.
```

that if we randomly choose either X_1 or X_2, and add enough points from the chosen set so that we have k points, each point $y \in Y$, with 50% probability, is not close to the new points added (because every point y must either be far from X_1 or far from X_2). Thus, the expected distance from y to the closest new point is large.

In general, it is not actually important that the points come from X_1 and X_2: we just need two sets S, T of k points such that $\rho(S, T)$, the minimum distance between $s \in S$ and $t \in T$, is large. Then, any point y cannot be close to points in both S and T. This also composes nicely, because to find the final set of k points, we only need there to be two sets S, T throughout the union of the coresets with large $\rho(S, T)$.

To find large S, T with large $\rho(S, T)$, we will require $|X| \ge k^{1+\varepsilon}$ for some small constant ε. For simplicity, we focus on the case when $|X| \ge k^{1.5}$. Suppose all but k points are in some ball B of radius r. If there exists x that is within distance $r/10$ of k points (we can make S these k points), then all points in S must be far away from the furthest k points from x (which we can set as T), or else we could have found a smaller ball B'. Otherwise, there are two options.

1. The majority of points $x \in X$ are within $r/100$ of at least $\sqrt{k}$ other points, but no $x \in X$ is within $r/10$ of at least k other points. Intuitively (we will make this intuition formal in Appendix A), a random set S_0 of size $O(\sqrt{k})$ should be within $r/100$ of at least k other points in total (we can make S these k points), but there are at least $|X| - k \cdot |S_0| \ge k$ points (we can make T these points) that are not within $r/10$ of S_0.
2. The majority of points aren't within $r/100$ of even $O(\sqrt{k})$ points. In this case, we can pick k of these points to form S, and they will not be within $r/100$ of at least $|X| - O(\sqrt{k}) \cdot |S| \ge k$ points. We make this intuition formal and prove the result for the more general $k^{1+\varepsilon}$.

3.2 Algorithm Pseudocode

We provide pseudocode for the remote-pseudoforest coreset in Algorithm 2 and for the remote-matching coreset in Algorithm 3. The proofs, as well as algorithm descriptions in words, are deferred to Section 5 (for remote-matching) and Appendix A (for remote-pseudoforest).

Algorithm 3 MWMCoreset: $O(1)$-approximate remote-matching composable coreset algorithm.

```
**Input:** data $X = \{x_1, \dots, x_n\}$, integer $k$.
**if** $n \leq 3k$ **then**
    **Return** $X$
**else**
    $Y = \{y_1, \dots, y_k\} \leftarrow \mathrm{GMM}(x_1, \dots, x_n, k)$.
    **Initialize** $S_1, \dots, S_k \leftarrow \emptyset$.
    **for** $i = 1$ to $n$ **do**
        Add $i$ to $S_j$ for $j = \arg\min \rho(x_i, y_j)$.
    **end for**
    **Initialize** $C \leftarrow Y$.
    **for** $i = 1$ to $k/2$ **do**
        **Find** $x, x' \in X \backslash C$ such that $x, x'$ are in the same $S_j$.
        $C \leftarrow C \cup \{x, x'\}$.
    **end for**
    **Return** $C$.
**end if**
```

4 Offline Remote-Matching Algorithm

In this section, we design $O(1)$-approximate offline algorithms for remote-matching. In this section, we first provide a technical overview, then the algorithm description and pseudocode, and finally we provide the full analysis.

4.1 Technical Overview

The remote-matching offline algorithm first utilizes some simple observations that we made in Section 3.1. Namely, we may assume the largest minimum-weight matching cost of any subset of k points is $\omega(k \cdot r)$, or else GMM provides an $O(1)$-approximation. Next, if the optimum solution was some $Z = \{z_1, \dots, z_k\}$, we can again consider mapping each z_i to its closest y_j, at the cost of having duplicates. However, as noted in Section 3.1, we may delete a point twice without affecting the matching cost: this means we can keep deleting a point twice until each y_j is only there 0 times (if the total number of z_i's closest to y_j was even) or 1 time (if the total number of z_i's closest to y_j was odd).

However, we have no idea what Z actually is, so we have no idea whether each y_j should be included or not. However, this motivates the following simpler problem: among the k points Y, choose a (even-sized) subset of Y maximizing the matching cost.

One attempt at solving this problem is to choose $\{y_1, \dots, y_p\}$ for some $p \leq k$: this will resemble an argument in [11]. The idea is that if we define r_p to be the maximum value $\rho(x, \{y_1, \dots, y_p\})$ over all $x \in X$, the same argument as Proposition 8 implies that all points among $y_1, \dots, y_p$ are separated by at least r_p. Hence, for the best p we can obtain matching cost $\Omega(\max_{1 \leq p \leq k} p \cdot r_p)$. Conversely, it is known that the minimum-weight-matching cost of any set of points Z is upper-bounded by the cost of the minimum spanning tree of Z. But the minimum spanning tree has cost at most $\sum_{p=1}^{k} r_p$, since we can create a tree by adding an edge from each y_{p+1} to its closest center among $y_1, \dots, y_p$, which has distance r_p. Since $\max(p \cdot r_p) \geq \Omega\left(\frac{1}{\log k}\right) \cdot \sum_{p=1}^{k} r_p$ (with equality for instance if $r_p = \frac{1}{p}$), we can obtain an $O(\log k)$-approximation.

For simplicity, we focus on the case where $r_p = \Theta(1/p)$ for all $1 \le p \le k$. We would hope that either the minimum spanning tree cost of Y, which we call $\mathrm{MST}(Y)$, is actually much smaller than $\log k$, or there is some alternative selection to obtain matching cost $\Omega(\log k)$ rather than $O(1)$. Suppose that $\mathrm{MST}(Y) = \Omega(\log k)$: furthermore, for simplicity suppose the pth largest edge of the tree has weight $\frac{1}{p}$. If we considered the graph on Y connecting two points if their distance is less than $\frac{1}{p}$, it is well-known that the graph must therefore split into p disconnected components.

Now, for some fixed p suppose that we chose a subset Z of Y such that each connected component in the graph above has an odd number of points in Z. Then, any matching must send at least one point in each $Z \cap CC_j$ (where CC_j is the jth connected component) to a point in a different connected component, forcing an edge of weight at least $\frac{1}{p}$. Since each of p connected components has such a point, together we obtain weight at least 1. In addition, if we can ensure this property for $p = 2, 4, 8, 16 \dots, k$, we can in fact get there must be at least 2^i edges of weight $1/2^i$, making the total cost $\Omega(\log k)$, as desired.

While such a result may not be possible exactly, it turns out that even a *random* subset of Y satisfies this property asymptotically! Namely, if we choose each point $y \in Y$ to be in Z with 50% probability, each CC_j is odd with 50% probability. So in expectation, for all p, the number of connected components of odd size is $p/2$. Even if we make sure Z has even size, this will still be true, replacing $p/2$ with $\Omega(p)$. Since this is true for all p in expectation, by adding over powers of 2 for p, we will find k points with $\Omega(\log k)$ matching cost in expectation.

4.2 Algorithm Desciption and Pseudocode

Given a dataset $X = \{x_1, \dots, x_n\}$, we recall that the goal of the remote-matching problem is to find a subset $Z = \{z_1, \dots, z_k\} \subset X$ of k points, such that the minimum-weight matching cost of Z, $\mathrm{MWM}(Z)$, is approximately maximized. In this subsection, we describe our $O(1)$-approximate remote-matching algorithm. We also provide pseudocode in Algorithm 4.

Algorithm Description

The algorithm proceeds as follows. First, run the GMM algorithm for k steps, to obtain k points $Y = \{y_1, \dots, y_k\} \subset X$. Define the subsets $S_1, \dots, S_k$ as a partitioning of X, where $x \in X$ is in S_i if y_i is the closest point to x in Y. (We break ties arbitrarily.) We use the better of the following two options, with the larger minimum-weight matching cost.

1. Simply use $Y = \{y_1, \dots, y_k\}$.
2. Let $\hat{Z} \subset Y$ be a uniformly random subset of Y. Initialize W to $\hat{Z}$ if $|\hat{Z}|$ is even, and otherwise initialize W to $\hat{Z} \backslash \hat{z}$ for some arbitrary $\hat{z} \in \hat{Z}$. Now, if there exist two points not in $W \cup Y$ but in the same subset S_i, add both of them to W. Repeat this procedure until $|W| = k$.

We will use whichever of Y or W has the larger minimum-weight matching cost. Since minimum-weight matching can be computed in polynomial time, we can choose the better of these two in polynomial time.

In Theorem 3, we assume $n \ge 3k$. Because of this assumption, if $|W| < k$, then $|W \cup Y| \le 2k - 1$, which means $|X \backslash (W \cup Y)| \ge k + 1$. Hence, by Pigeonhole Principle, two of these points must be in the same set S_i, which means that the procedure described above is indeed doable.

Algorithm 4 MWMOffline: $O(1)$-approximate remote-matching algorithm.

```
 1: Input: data $X = \{x_1, \ldots, x_n\}$, even integer $k$.
 2: $Y = \{y_1, \ldots, y_k\} \leftarrow \mathrm{GMM}(x_1, \ldots, x_n, k)$.
 3: Initialize $S_1, \ldots, S_k \leftarrow \emptyset$.
 4: for $i = 1$ to $n$ do
 5:     Add $i$ to $S_j$ if $j = \arg\min \rho(x_i, y_j)$.
 6: end for
 7: $Z \leftarrow$ random subset of $Y$.
 8: if $|Z|$ is odd then
 9:     Remove an arbitrary element from $Z$
10: end if
11: Initialize $W \leftarrow Z$.
12: while $|W| < k$ do
13:     Find some $x, x' \in X \backslash (W \cup Y)$, such that $x, x'$ are in the same subset $S_j$.
14:     Add $x, x'$ to $W$.
15: end while
16: Return whichever of $Y, W$ has larger minimum-weight matching cost.
```

4.3 Analysis

The first ingredient in proving Theorem 3 is the following lemma, which shows that assuming the random subset Z we chose in Line 7 of Algorithm 4 is sufficiently good, the algorithm produces an $O(1)$-approximation.

▶ **Lemma 9.** *For some constant $\frac{1}{2} \geq \alpha > 0$, suppose that*

$$\mathrm{MWM}(Z) \geq \alpha \cdot \max_{Z' \subset Y : |Z'| \text{ is even}} \mathrm{MWM}(Z').$$

Then, Algorithm 4 provides a $\frac{4}{\alpha}$-approximation for the remote-matching problem.

Proof. Let M be the optimal remote-matching cost. Let r be the maximum distance from any point in $X \backslash Y$ to its closest point in Y. Note that $\rho(y_i, y_j) \geq r$ for all $i, j \leq k$, by Proposition 8.

First, suppose that $M \leq 2\alpha^{-1} \cdot r \cdot k$. In this case, because every pair in Y has pairwise distance at least r, we have $\mathrm{MWM}(Y) \geq r \cdot \frac{k}{2}$. Hence, $\mathrm{MWM}(Y) \geq \frac{\alpha}{4} \cdot M$, which means we have a $\frac{4}{\alpha}$-approximation.

Alternatively, suppose $M \geq 2\alpha^{-1} \cdot r \cdot k$. Let $W_0 \subset X$ be the set of p points that achieves this, i.e., $\mathrm{MWM}(W_0) = M$. Consider the following multiset $\tilde{W}_0$ of size p in Y, where each point in W_0 is mapped to its closest center in Y (breaking ties in the same way as in the algorithm). Then, every pair of distances between W_0 and $\tilde{W}_0$ changes by at most $2r$. This means *every* matching has its cost change by at most $\frac{k}{2} \cdot 2r = rk$, so $\mathrm{MWM}(\tilde{W}_0) \geq M - rk$.

Now, let $Z_0 \subset Y$ be the set of points where $y_i \in Z_0$ if and only if y_i is in $\tilde{W}_0$ an odd number of times. Then, $\mathrm{MWM}(\tilde{W}_0) = \mathrm{MWM}(Z_0)$. To see why, first note that $\mathrm{MWM}(\tilde{W}_0) \leq \mathrm{MWM}(Z_0)$ since we can convert any matching of Z_0 to a matching of $\tilde{W}_0$ by simply matching duplicate points in $\tilde{W}_0$ until only Z_0 is left. To see why $\mathrm{MWM}(\tilde{W}_0) \geq \mathrm{MWM}(Z_0)$, note that if an optimal matching of $\tilde{W}_0$ connected some copy of y to a point $y' \neq y$ and another copy of y to a point $y'' \neq y$, we can always replace the edges (y, y') and (y, y'') with (y, y) and (y', y''), which by Triangle inequality will never increase the cost. We may keep doing this until a maximal number of duplicate points are matched together, and only one copy of each element in Z_0 will be left. Hence, we have

$$\mathrm{MWM}(Z_0) = \mathrm{MWM}(\tilde{W}_0) \geq M - rk. \tag{1}$$

Similarly, let Z, W be the sets found in the algorithm described above, and let $\tilde{W}$ be the multiset formed by mapping each point in W to its nearest center in Y. As in the case with Z_0 and $\tilde{W}_0$, we have that for Z and $\tilde{W}$, a point z is in Z if and only if z is in $\tilde{W}$ an odd number of times. Hence, $\text{MWM}(\tilde{W}) = \text{MWM}(Z)$. Likewise, each point in $\tilde{W}$ has distance at most r from its corresponding point in W, which means $\text{MWM}(W) \ge \text{MWM}(\tilde{W}) - rk$. Hence, we have

$$\text{MWM}(Z) = \text{MWM}(\tilde{W}) \le \text{MWM}(W) + rk. \tag{2}$$

Overall, $\text{MWM}(Z) \ge \alpha \cdot \max_{Z \subset Y: |Z| \text{ is even}} \text{MWM}(Z) \ge \alpha \cdot \text{MWM}(Z_0)$, so

$$\begin{aligned} \text{MWM}(W) &\ge \text{MWM}(Z) - rk \ge \alpha \cdot \text{MWM}(Z_0) - rk \\ &\ge \alpha \cdot M - (1+\alpha) rk. \end{aligned}$$

But note that $M \ge 2\alpha^{-1} rk$, which means that $(1+\alpha)rk \le \frac{\alpha(1+\alpha)}{2} \cdot M \le \frac{3}{4} \cdot \alpha \cdot M$. Hence, $\text{MWM}(W) \ge \frac{\alpha}{4} \cdot M$, which again means we have a $\frac{4}{\alpha}$-approximation. ◀

The main technical lemma that we will combine with Lemma 9 shows that Z has the desired property. We now state the lemma, but we defer the proof slightly, to Section 4.5. We remark that the proof roughly follows the intuition described at the end of Section 4.1.

▶ **Lemma 10.** *Let $\hat{Z}$ be a random subset of Y where each element is independently selected with probability $1/2$. If $|\hat{Z}|$ is even, we set $Z = \hat{Z}$, and if $|\hat{Z}|$ is odd, we arbitrarily remove 1 element from $\hat{Z}$ to generate Z. Then,*

$$\mathbb{E}[\text{MWM}(Z)] \ge \frac{1}{16} \cdot \max_{Z' \subset Y: |Z'| \text{ is even}} \text{MWM}(Z').$$

Given Lemmas 9 and 10, we explain how combine them to prove Theorem 3.

Proof of Theorem 3. Suppose we generate a random subset Z of Y (possibly removing an element), and suppose that $\text{MWM}(Z) = \alpha \cdot \max_{Z \subset Y: |Z| \text{ is even}} \text{MWM}(Z)$. Then, the output of the algorithm has matching cost at least $\frac{\alpha}{4}$ times the optimum k-matching cost $\text{MWM}_k(X)$, by Lemma 9. However, by Lemma 10, $\mathbb{E}[\alpha] \ge \frac{1}{16}$, which means that the expected matching cost of the output is $\frac{\mathbb{E}[\alpha]}{4} \cdot \text{MWM}_k(X) \ge \frac{1}{64} \cdot \text{MWM}_k(X)$.

If we want this to occur with high probability, note that the matching cost of the output can never be more than $\text{MWM}_k(X)$. Hence, by Markov's inequality, with at least $\frac{1}{64^2} = \frac{1}{4096}$ probability, the output has matching cost at least $\frac{1}{65} \cdot \text{MWM}_k(X)$. If we repeat this $O(1)$ times and return the set with best matching cost, we can find a set of size k with matching cost at least $\frac{1}{65} \cdot \text{MWM}_k(X)$, with probability at least 0.99. ◀

Before proving Lemma 10, we will need some additional preliminaries.

4.4 Preliminaries for Lemma 10

To prove Lemma 10, we will need several preliminary facts relating to the cost of a minimum weight matching, as well as the cost of a minimum spanning tree of a set of points.

First, we have the following fact, bounding the minimum weight matching in terms of the MST.

▶ **Proposition 11** (Classical, see Proof of Lemma 5.2 in [11])**.** *For any (finite, even sized) set of data points Z in a metric space,* $\text{MWM}(Z) \le \text{MST}(Z)$.

Next, given a subset Z of Y in a metric space, we can bound the minimum spanning tree cost of Z in terms of the minimum spanning cost of Y.

▶ **Proposition 12** (Classical, see [16])**.** *Let $Z \subset Y$ be (finite) sets of data points in some metric space. Then,* $\mathrm{MST}(Z) \le 2 \cdot \mathrm{MST}(Y)$.

Next, we equate the minimum spanning tree of a dataset Y with the number of connected components in a family of graphs on Y. The following proposition essentially follows from the same argument as in [12, Lemma 2.1]. We prove it here for completeness since the statement we desire is not explicitly proven in [12].

▶ **Proposition 13.** *Given a dataset Y in a metric space and a radius $r > 0$, define $G_r(Y)$ to be the graph on Y that connects two data points if and only if their distance is at most r. Define $P_r(Y)$ to be the number of connected components in $G_r(Y)$. Then,*

$$\mathrm{MST}(Y) \in \left[\frac{1}{2}, 1\right] \cdot \left(\sum_{i \in \mathbb{Z}} 2^i \cdot (P_{2^i}(Y) - 1)\right).$$

Proof. Note that if 2^i is at least $\mathrm{diam}(Y)$, the diameter of Y, $P_{2^i}(Y) = 1$, which means we may ignore the summation for i with $2^i > \mathrm{diam}(Y)$. Hence, by scaling by some power of 2, we may assume WLOG that $\mathrm{diam}(Y) < 1$, and that the summation is only over $i < 0$.

Now, for any $t \ge 0$, let $Q_t(Y)$ be the number of edges in the MST of Y with weight at most 2^{-t} and strictly more than $2^{-(t+1)}$ (assuming we run Kruskal's algorithm for MST). Note that $R_t(Y) := \sum_{t' \ge t} Q_{t'}(Y)$ is the number of edges with weight at most 2^{-t}. Note that $R_t(Y)$ is precisely $n - P_{2^{-t}}(Y)$. To see why, note that the $R_t(Y)$ edges form a forest with $n - R_t(Y)$ connected components. In addition, in the graph $G_{2^{-t}}(Y)$, none of the $n - R_t(Y)$ components can be connected to each other, or else there would have been another edge of weight at most 2^{-t} that Kruskal's algorithm would have had to add. Therefore, $\sum_{t' \ge t} Q_{t'}(Y) = n - P_{2^{-t}}(Y)$. By subtracting this equation from the same equation replacing t with $t + 1$, we obtain

$$Q_t(Y) = P_{2^{-(t+1)}}(Y) - P_{2^{-t}}(Y). \tag{3}$$

Now, note that by definition of $Q_t(Y)$, the cost of $\mathrm{MST}(Y)$ is between $\sum_{t \ge 0} 2^{-(t+1)} \cdot Q_t(Y)$ and $\sum_{t \ge 0} 2^{-t} \cdot Q_t(Y)$. Equivalently, it equals $\alpha \cdot \left(\sum_{t \ge 0} 2^{-t} \cdot Q_t(Y)\right)$, for some $\alpha \in [1/2, 1]$. Therefore,

$$\begin{aligned}
\mathrm{MST}(Y) &= \alpha \cdot \left(\sum_{t \ge 0} 2^{-t} \cdot Q_t(Y)\right) \\
&= \alpha \cdot \left(\sum_{t \ge 0} 2^{-t} \cdot (P_{2^{-(t+1)}}(Y) - P_{2^{-t}}(Y))\right) \\
&= \alpha \cdot \left(\sum_{t \ge 0} (2^{-t} - 2^{-(t+1)}) P_{2^{-(t+1)}}(Y) - P_1(Y)\right) \\
&= \alpha \cdot \left(\sum_{t \ge 1} 2^{-t} P_{2^{-t}}(Y) - 1\right) \\
&= \alpha \cdot \left(\sum_{t \ge 1} 2^{-t} (P_{2^{-t}}(Y) - 1)\right).
\end{aligned}$$

The second-to-last line follows since the diameter is at most 1 so $G_1(Y)$ has one connected component, and the last line follows because $\sum_{t \ge 1} 2^{-t} = 1$. ◀

Finally, we need to consider the minimum weight matching cost in a *hierarchically well-separated tree* (HST).

▶ **Definition 14.** *A* hierarchically-well seprated tree (HST) *is a depth-d tree (for some integer $d \geq 1$) with the root as depth 0, and every leaf has depth d. For any node u in the tree of depth $1 \leq t \leq d$, each edge from u to its parent has weight 2^{-t}. For two nodes v, w in the HST, the distance $d_{\text{HST}}(v, w)$ is simply the sum of the edge weights along the shortest path from v to w in the tree.*

Note that for any two leaf nodes v, w in an HST, if their least common ancestor has depth t, the distance between v and w is $2 \cdot \left(2^{-(t+1)} + 2^{-(t+2)} + \cdots + 2^{-d}\right) = 2 \cdot (2^{-t} - 2^{-d})$.

We will make use of the following result about points in an HST metric.

▶ **Proposition 15** ([21], Claim 3)**.** *Let Z be a (finite, even sized) set of points that are leaves in a depth-d HST. Let m_i be the number of nodes at level i with an* odd *number of descendants in Z. Then, with respect to the HST metric, the minimum weight matching cost equals*

$$\text{MWM} = \sum_{i=0}^{d} 2^{-i} \cdot m_i.$$

We remark that the corresponding statement in [21] has an additional additive factor of $n = |Z|$ in the right-hand side. This is because we include the bottom level in our sum (which consists of n nodes each with exactly one descendant), whereas [21] does not.

4.5 Proof of Lemma 10

We are now ready to prove Lemma 10

Proof of Lemma 10. Assume WLOG (by scaling) that the diameter of Y is at most 1. Let Z be a subset of Y with even size. By Proposition 11, $\text{MWM}(Z) \leq \text{MST}(Z)$. By Proposition 12, $\text{MST}(Z) \leq 2 \cdot \text{MST}(Y)$. Combining these together, we have

$$\max_{Z \subset Y : |Z| \text{ even}} \text{MWM}(Z) \leq 2 \cdot \text{MST}(Y). \tag{4}$$

Now, for our dataset Y and any positive real $r > 0$, recall that $G_r(Y)$ is defined as the graph on Y that connects two data points if their distance is at most r. In addition, define $\mathcal{P}_r(Y)$ to be the partitioning of Y into connected components based on $G_r(Y)$, and recall that $P_r(Y) = |\mathcal{P}_r(Y)|$ equals the number of connected components in $G_r(Y)$. Then, Proposition 13 tells us that

$$\text{MST}(Y) \leq \sum_{i \in \mathbb{Z}} 2^i \cdot (P_{2^i}(Y) - 1). \tag{5}$$

Now, consider the following "embedding" of Y into a depth-d hierarchically well-separated tree (where we will choose d later) as follows. By scaling, assume WLOG that the diameter of Y is 1. For each integer $0 \leq t \leq d$, the nodes at level t will be the connected components in $G_{2^{-t}}(Y)$, where the children of any node at depth t, represented by a subset Z of Y, are simply the connected components in $\mathcal{P}_{2^{-(t+1)}}(Y)$ contained in Z.

The distance $d_{\text{HST}}(y_i, y_j)$ between any two vertices y_i, y_j in the HST is precisely $2(2^{-t} - 2^{-d})$ if y_i, y_j have common ancestor at level t of the HST. Note that if $d_{\text{HST}}(y_i, y_j) = 2(2^{-t} - 2^{-d})$, then y_i, y_j are not in the same connected component of $G_{2^{-(t+1)}}$, which means that $\rho(y_i, y_j) > 2^{-(t+1)}$. Importantly, this means that $d_{\text{HST}}(y_i, y_j) \leq 4\rho(y_i, y_j)$ for all pairs i, j. Hence, for any subset $Z \subset Y$ of even size, the minimum weight matching cost $\mathrm{MWM}_{\mathrm{HST}}(Z)$ with respect to the HST metric is at most 4 times the true minimum weight matching cost, i.e.,

$$\mathrm{MWM}_{\mathrm{HST}}(Z) \le 4 \cdot \mathrm{MWM}(Z). \tag{6}$$

Finally, we consider selecting a random subset $\hat{Z} \subset Y$, and provide a lower bound for $\mathrm{MWM}_{\mathrm{HST}}(Z)$, where $Z = \hat{Z}$ if $|\hat{Z}|$ is even and otherwise Z equals $\hat{Z}$ after removing a single (arbitrary) element. Note that for each node v of depth t, corresponding to a connected component in $\mathcal{P}_{2^{-t}}(Y)$, the probability that it has an odd number of descendants in $\hat{Z}$ if $\hat{Z}$ is picked at random is precisely $1/2$. This implies that the expectation of $\sum_{i=0}^d 2^{-i} \cdot m_i$, where m_i is the number of nodes at level i with an odd number of descendants in $\hat{Z}$, is $\frac{1}{2} \cdot \sum_{i=0}^d 2^{-i} \cdot P_{2^{-i}}(Y)$.

Note, however, that $\hat{Z}$ has odd size with $1/2$ probability. In this event, we remove an arbitrary element of $\hat{Z}$, which may reduce each m_i by 1. This only happens with 50% probability, so after this potential removal of a point, the expectation of $\sum_{i=0}^d 2^{-i} \cdot m_i(Z)$, where $m_i(Z)$ is the number of nodes at level i with an odd number of descendants in Z, is at least $\frac{1}{2} \cdot \sum_{i=0}^d 2^{-i} \cdot (P_{2^{-i}}(Y) - 1)$. Since $\mathrm{diam}(Y)$ is at most 1, this implies $P_{2^i}(Y) - 1 = 0$ for all $i \ge 1$. Also, for $i > d$, $2^{-i} \cdot (P_{2^{-i}}(Y) - 1) \le 2^{-i} \cdot n$. If we sum this up over all $i > d$, this is still at most $2^{-d} \cdot n$. Hence, we have that

$$\mathbb{E}_Z\left[\sum_{i=0}^{d} 2^{-i} \cdot m_i(Z)\right] \ge \frac{1}{2} \cdot \left(\sum_{i \in \mathbb{Z}} 2^i (P_{2^i}(Y) - 1)\right) - n \cdot 2^{-d}. \tag{7}$$

In summary, we have that

$$\begin{aligned}
\max_{Z' \subset Y : |Z'| \text{ even}} \mathrm{MWM}(Z') &\le 2 \cdot \mathrm{MST}(Y) && \text{By Equation (4)} \\
&\le 2 \cdot \sum_{i \in \mathbb{Z}} 2^i \cdot (P_{2^i}(Y) - 1) && \text{By Equation (5)} \\
&\le 4 \cdot \left(\mathbb{E}_Z\left[\sum_{i=0}^{d} 2^{-i} \cdot m_i(Z)\right] + n \cdot 2^{-d}\right) && \text{By Equation (7)} \\
&= 4 \cdot \left(\mathbb{E}_Z\left[\mathrm{MWM}_{\mathrm{HST}}(Z)\right] + n \cdot 2^{-d}\right) && \text{By Proposition 15} \\
&\le 16 \cdot \left(\mathbb{E}_Z\left[\mathrm{MWM}(Z)\right] + n \cdot 2^{-d}\right). && \text{By Equation (6)}
\end{aligned}$$

We can choose the depth of the HST to be arbitrarily large, which therefore implies that

$$\mathbb{E}_Z[\mathrm{MWM}(Z)] \ge \frac{1}{16} \cdot \max_{Z' \subset Y : |Z'| \text{ even}} \mathrm{MWM}(Z'). \qquad \blacktriangleleft$$

5 Coreset for Remote-Matching

In this section, we prove why the algorithm given in Algorithm 3 creates an $O(1)$-approximate composable coreset. First, we describe the algorithm in words.

5.1 Algorithm Description

We start by running GMM on the dataset X for k steps, to return k points $Y = \{y_1, \dots, y_k\}$. Again, let the subsets $S_1, \dots, S_k$ be a partitioning of X, where $x \in X$ is in S_i if y_i is the closest point in Y to x (breaking ties arbitrarily). Note that $y_i \in S_i$ for all i.

To create our coreset C for X, if $|X| \le 3k$ we simply define $C = X$. Otherwise, we start by initializing C to be Y, so C currently has size k. Next, for $k/2$ steps, we find any two points in $X \backslash C$ that are in the same partition piece S_i, and add both of them to C. Hence, at the end $|C| = 2k$. Note that this procedure is always doable, since we are assuming $|X| \ge 3k+1$, which means if we have picked at most $2k$ total elements, there are $k+1$ remaining elements in X, of which at least 2 must be in the same S_i by the pigeonhole principle.

5.2 Analysis

In this subsection, we prove that the algorithm generates an $O(1)$-approximate composable coreset, by proving Theorem 5.

Proof of Theorem 5. Suppose we run this algorithm for each of m datasets, $X^{(1)}, \dots, X^{(m)}$, to generate coresets $C^{(1)}, \dots, C^{(m)}$. We wish to show that the optimum k-matching cost of $C = \bigcup_{j=1}^m C^{(j)}$ is at least $\Omega(1)$ times the optimum k-matching cost of $X = \bigcup_{j=1}^m X^{(j)}$.

Let $Y^{(j)} = \{y_1^{(j)}, \dots, y_k^{(j)}\}$ represent the k points we obtained by running GMM on $X^{(j)}$, and let $r^{(j)}$ be the maximum distance from any point in $X^{(j)} \backslash Y^{(j)}$ to its closest point in $Y^{(j)}$. Then, note that all points in $Y^{(j)}$ are pairwise separated by at least $r^{(j)}$. Let $r = \max_{1 \le j \le m} r^{(j)}$.

First, suppose that the optimum k-matching cost of X is $M \le 5r \cdot k$. In this case, for the $r^{(j)}$ that equals r, the GMM algorithm finds k points that are pairwise separated by at least $r^{(j)} = r$. Since $C^{(j)} \supset Y^{(j)}$, this means that the full coreset C contains k points that are pairwise separated by r, which has k-matching cost at least $r \cdot \frac{k}{2}$. Hence, we have a 10-approximate coreset.

Alternatively, the optimum k-matching cost of X is $M \ge 5r \cdot k$. Let $S_i^{(j)}$ represent the set S_i for $X^{(j)}$, and suppose W is an optimal set of k points in X with $\text{MWM}(W) = M$. Let $W^{(j)} = W \cap X^{(j)}$. Also, let $W_i^{(j)} = W \cap S_i^{(j)}$ and $b_i^{(j)}$ be the *parity* of $|W_i^{(j)}|$, i.e., $b_i^{(j)} = 1$ if $|W_i^{(j)}|$ is odd and $b_i^{(j)} = 0$ if $|W_i^{(j)}|$ is even. In addition, let $\tilde{W}$ be the multiset of k points formed by mapping each point in $W_i^{(j)}$ to $y_i^{(j)}$. In other words, $\tilde{W}$ consists of each $y_i^{(j)}$ repeated $|W_i^{(j)}|$ times. Note that since each $W_i^{(j)}$ has distance at most $r^{(j)} \le r$ from $y_i^{(j)}$, all pairwise distances change by at most $2r$, which means the matching cost difference $|\text{MWM}(\tilde{W}) - \text{MWM}(W)| \le \frac{1}{2} \cdot 2r \cdot k = rk$. Also, note that $\tilde{W}$ only consists of points of the form $y_i^{(j)}$, with the parity of the number of times $y_i^{(j)}$ appears in $\tilde{W}$ equaling $b_i^{(j)}$.

Next, we create a similar set $W' \subset C$. For each $j \le m$, define $k^{(j)} = |W^{(j)}|$. We will find a set $(W')^{(j)} \subset C^{(j)}$ of size $k^{(j)}$, such that the parity of $|(W')^{(j)} \cap S_i^{(j)}|$ equals $b_i^{(j)}$ for all $i \le k$. To do so, first note that if $|X^{(j)}| \le 3k$, then $C^{(j)} = X^{(j)}$, so we can just choose $(W')^{(j)} = W^{(j)}$. Otherwise, $|X^{(j)}| \ge 3k+1$, and $C^{(j)}$ consists of $2k$ points. In addition, $C^{(j)} \supset Z^{(j)}$. Now, we start by including in $(W')^{(j)}$ each point $y_i^{(j)}$ such that $b_i^{(j)} = 1$. Since $b_i^{(j)} = 1$ means that $|W_i^{(j)}|$ is odd, for any fixed j the number of $b_i^{(j)} = 1$ is at most $k^{(j)}$ and has the same parity as $k^{(j)}$. Now, as long as $|(W')^{(j)}| < k^{(j)}$, this means $|C^{(j)} \backslash (W')^{(j)}| \ge k+1$, which means there are two points in $C^{(j)} \backslash (W')^{(j)}$ that are in the same $S_i^{(j)}$, by pigeonhole principle. We can add both of them to $(W')^{(j)}$. We can keep repeating this procedure until $|(W')^{(j)}| = k^{(j)}$, and note that this never changes the parity of each $|(W')^{(j)} \cap S_i^{(j)}|$.

Our set $W' \subset C$ will just be $\bigcup_{j=1}^m (W')^{(j)}$. Note that $|W'| = \sum_{j=1}^m k^{(j)} = k$, and $|W' \cap S_i^{(j)}|$ has parity $b_i^{(j)}$, just like W. Hence, we can create the multiset $\tilde{W}'$ by mapping each point $w' \in W' \cap S_i^{(j)}$ to $y_i^{(j)}$. Again, each point moves by at most r, so all pairwise distances change by at most $2r$, which means that $|\text{MWM}(\tilde{W}') - \text{MWM}(W')| \le rk$.

Finally, we will see that $\text{MWM}(\tilde{W}') = \text{MWM}(\tilde{W})$. Note that both $\tilde{W}$ and $\tilde{W}'$ are multisets of $y_i^{(j)}$ points, each repeated an odd number of times if and only if $b_i^{(j)} = 1$. However, we saw in the proof of Lemma 9 that $\text{MWM}(\tilde{W})$ equals the minimum-weight matching cost of simply including each point $y_i^{(j)}$ exactly $b_i^{(j)}$ times. This is because there exists an optimal matching that keeps matching duplicate points together as long as it is possible. The same holds for $\text{MWM}(\tilde{W}')$, which means $\text{MWM}(\tilde{W}) = \text{MWM}(\tilde{W}')$.

Overall, this means that $|\text{MWM}(W') - \text{MWM}(W)| \le |\text{MWM}(W') - \text{MWM}(\tilde{W}')| + |\text{MWM}(\tilde{W}') - \text{MWM}(\tilde{W})| + |\text{MWM}(\tilde{W}) - \text{MWM}(W)| \le rk + 0 + rk = 2rk$. But since we assumed that $\text{MWM}(W) = M \ge 5rk$, this means the k-matching for C is at least $M - 2rk \ge \frac{M}{2}$. Hence, we get a 2-approximate coreset.

In either case, we obtain an $O(1)$-approximate coreset, as desired. ◀

References

1 Sofiane Abbar, Sihem Amer-Yahia, Piotr Indyk, and Sepideh Mahabadi. Real-time recommendation of diverse related articles. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1–12, 2013.

2 Sofiane Abbar, Sihem Amer-Yahia, Piotr Indyk, Sepideh Mahabadi, and Kasturi R Varadarajan. Diverse near neighbor problem. In *Proceedings of the twenty-ninth annual symposium on Computational geometry*, pages 207–214, 2013.

3 Zeinab Abbassi, Vahab S Mirrokni, and Mayur Thakur. Diversity Maximization Under Matroid Constraints. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 32–40, 2013.

4 Sepideh Aghamolaei, Majid Farhadi, and Hamid Zarrabi-Zadeh. Diversity maximization via composable coresets. In *CCCG*, pages 38–48, 2015.

5 Albert Angel and Nick Koudas. Efficient diversity-aware search. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 781–792, 2011.

6 Sepehr Assadi and Sanjeev Khanna. Randomized composable coresets for matching and vertex cover. *arXiv preprint*, 2017. arXiv:1705.08242.

7 Aditya Bhaskara, Mehrdad Ghadiri, Vahab S. Mirrokni, and Ola Svensson. Linear relaxations for finding diverse elements in metric spaces. In *Advances in Neural Information Processing Systems*, pages 4098–4106, 2016.

8 Allan Borodin, Hyun Chul Lee, and Yuli Ye. Max-sum diversification, monotone submodular functions and dynamic updates. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*, pages 155–166, 2012.

9 Matteo Ceccarello, Andrea Pietracaprina, Geppino Pucci, and Eli Upfal. Mapreduce and streaming algorithms for diversity maximization in metric spaces of bounded doubling dimension. *arXiv preprint*, 2016. arXiv:1605.05590.

10 Alfonso Cevallos, Friedrich Eisenbrand, and Sarah Morell. Diversity maximization in doubling metrics. *arXiv preprint*, 2018. arXiv:1809.09521.

11 Barun Chandra and Magnús M. Halldórsson. Approximation algorithms for dispersion problems. *J. Algorithms*, 38(2):438–465, 2001.

12 Artur Czumaj and Christian Sohler. Estimating the weight of metric minimum spanning trees in sublinear time. *SIAM J. Comput.*, 39(3):904–922, 2009.

13 Marina Drosou and Evaggelia Pitoura. Search result diversification. *ACM SIGMOD Record*, 39(1):41–47, 2010.

14 Alessandro Epasto, Mohammad Mahdian, Vahab Mirrokni, and Peilin Zhong. Improved sliding window algorithms for clustering and coverage via bucketing-based sketches. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3005–3042. SIAM, 2022.

15 Alessandro Epasto, Vahab Mirrokni, and Morteza Zadimoghaddam. Scalable diversity maximization via small-size composable core-sets (brief announcement). In *The 31st ACM symposium on parallelism in algorithms and architectures*, pages 41–42, 2019.

16 E. N. Gilbert and H. O. Pollak. Steiner minimal trees. *SIAM J. Appl. Math.*, 16:1–29, 1968.

17 Sreenivas Gollapudi and Aneesh Sharma. An axiomatic approach for result diversification. In *Proceedings of the 18th international conference on World wide web*, pages 381–390, 2009.

18 Boqing Gong, Wei-Lun Chao, Kristen Grauman, and Fei Sha. Diverse sequential subset selection for supervised video summarization. *Advances in neural information processing systems*, 27, 2014.

19 Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.

20 Magnús M Halldórsson, Kazuo Iwano, Naoki Katoh, and Takeshi Tokuyama. Finding subsets maximizing minimum structures. *SIAM Journal on Discrete Mathematics*, 12(3):342–359, 1999.

21 Piotr Indyk. Algorithms for dynamic geometric problems over data streams. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 373–380. ACM, 2004.

22 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.

23 Piotr Indyk, Sepideh Mahabadi, Mohammad Mahdian, and Vahab S. Mirrokni. Composable core-sets for diversity and coverage maximization. In Richard Hull and Martin Grohe, editors, *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*, pages 100–108. ACM, 2014.

24 Anoop Jain, Parag Sarda, and Jayant R Haritsa. Providing diversity in k-nearest neighbor query results. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 404–413. Springer, 2004.

25 Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 510–520, 2011.

26 Hui Lin, Jeff Bilmes, and Shasha Xie. Graph-based submodular selection for extractive summarization. In *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 381–386. IEEE, 2009.

27 Sepideh Mahabadi, Piotr Indyk, Shayan Oveis Gharan, and Alireza Rezaei. Composable core-sets for determinant maximization: A simple near-optimal algorithm. In *International Conference on Machine Learning*, pages 4254–4263. PMLR, 2019.

28 Vahab Mirrokni and Morteza Zadimoghaddam. Randomized composable core-sets for distributed submodular maximization. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 153–162. ACM, 2015.

29 Julien Pilourdault, Sihem Amer-Yahia, Dongwon Lee, and Senjuti Basu Roy. Motivation-aware task assignment in crowdsourcing. In *EDBT*, 2017.

30 S. S. Ravi, D. J. Rosenkrantz, and G. K. Tayi. Facility dispersion problems: Heuristics and special cases. *Algorithms and Data Structures*, 519:355–366, 1991.

31 Michael J Welch, Junghoo Cho, and Christopher Olston. Search result diversity for informational queries. In *Proceedings of the 20th international conference on World wide web*, pages 237–246, 2011.

32 Cong Yu, Laks VS Lakshmanan, and Sihem Amer-Yahia. Recommendation diversification using explanations. In *2009 IEEE 25th International Conference on Data Engineering*, pages 1299–1302. IEEE, 2009.

33 Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling, and Yi-Cheng Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences*, 107(10):4511–4515, 2010.

34 Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32, 2005.

A Composable Coreset for Remote-Pseudoforest

In this section, we describe and analyze the composable coreset algorithm for remote-pseudoforest.

A.1 Algorithm

In this subsection, we prove why the algorithm given in Algorithm 2 creates an $O(1)$-approximate composable coreset. First, we describe the algorithm in words. We recall that we have m datasets $X^{(1)}, \dots, X^{(m)}$: we wish to create a coreset $C^{(j)}$ of each $X^{(j)}$ so that $\bigcup_{j=1}^m C^{(j)}$ contains a set Z of k points such that $\text{PF}(Z) \ge \Omega(1) \cdot \text{PF}_k\left(\bigcup_{j=1}^m X^{(j)}\right)$.

A.1.1 Coreset Construction

Suppose that $|X^{(j)}| \ge 2k^{1+\varepsilon} + k$. Let $\tilde{r}^{(j)}$ represent the smallest value such that there exists a ball $\tilde{B}^{(j)}$ of radius $\tilde{r}^{(j)}$ around some $x^{(j)} \in X^{(j)}$ that contains all but at most k of the points in $X^{(j)}$. The point $\tilde{r}^{(j)}$ and a corresponding $x^{(j)}, \tilde{B}^{(j)}$ can be found in $O(|X^{(j)}|^2)$ time.

Choose $U^{(j)}$ to be the set of k points furthest from $x^{(j)}$. These will either be precisely the k points outside $\tilde{B}^{(j)}$, or all of the points outside $\tilde{B}^{(j)}$ plus some points on the boundary of $\tilde{B}^{(j)}$, to make a total of k points. In addition, we choose some arbitrary set $P^{(j)} \subset X^{(j)}$ of any k points in the ball $B^{(j)}$.

Next, we will choose sets $S^{(j)}, T^{(j)} \subset X^{(j)}$ of size k, such that $\rho(S^{(j)}, T^{(j)}) \ge \frac{\varepsilon}{2} \cdot \tilde{r}^{(j)}$. It is not even clear that such sets exist, but we will show how to algorithmically find such sets in $O(|X^{(j)}|^2)$ time (assuming $|X^{(j)}| \ge 2k^{1+\varepsilon} + k$).

Finally, we run the GMM algorithm on $X^{(j)}$ to obtain $Y^{(j)} = \{y_1^{(j)}, y_2^{(j)}, \dots, y_k^{(j)}\}$. The final coreset will be $C^{(j)} := P^{(j)} \cup S^{(j)} \cup T^{(j)} \cup U^{(j)} \cup Y^{(j)}$. Note that each of $P^{(j)}, S^{(j)}, T^{(j)}, U^{(j)}, Y^{(j)}$ has size at most k, so $|C^{(j)}| \le 5k$.

Alternatively, if $|X^{(j)}| < 2k^{1+\varepsilon} + k$, we choose the coreset to simply be $X^{(j)}$. For convenience, in this setting, we define $U^{(j)} := X^{(j)}$ and $P^{(j)}, S^{(j)}, T^{(j)}, Y^{(j)}$ to all be empty.

We have not yet described how to find $S^{(j)}, T^{(j)}$, let alone prove they even exist. We now describe an $O(|X^{(j)}|^2)$ time algorithm that finds $S^{(j)}, T^{(j)} \subset X^{(j)}$ of size k, such that $\rho(S^{(j)}, T^{(j)}) \ge \frac{\varepsilon}{2} \cdot \tilde{r}^{(j)}$.

A.1.2 Efficiently finding $S^{(j)}, T^{(j)}$

Here, we show that we can efficiently find $S^{(j)}, T^{(j)}$ from the coreset construction, as long as $|X^{(j)}| \ge 2k^{1+\varepsilon} + k$. We formalize this with the following lemma.

▶ **Lemma 16.** *Let $\varepsilon > 0$ be a fixed constant, and consider a dataset X of size at least $2k^{1+\varepsilon} + k$, and suppose that no ball of radius smaller than $\tilde{r}$ around any $x \in X$ contains all but at most k points in X. (In other words, for every $x \in X$, there are at least k points in X of distance at least $\tilde{r}$ from x.) Then, we in $O(|X|^2)$ time, we can find two disjoint sets $S, T \subset X$, each of size k, such that $\rho(S, T) \ge \frac{\varepsilon \cdot \tilde{r}}{2}$.*

Proof. The algorithm works as follows. First, define $r' = \frac{\varepsilon \cdot \tilde{r}}{2}$. For each point $x \in X$, and for every nonnegative integer i, define $N_i(x)$ as the set of points in X of distance at most $i \cdot r'$ from x. We can compute the set $N_i(x)$ for all $x \in X$ and $0 \le i \le 2/\varepsilon$ in $O(|X|^2)$ time, as ε is a constant. Suppose there exists $x \in X$ such that $|N_{1/\varepsilon}(x)| \ge k$. By our assumption on $\tilde{r}$, there are at least k points of distance at least $\tilde{r} = \frac{2}{\varepsilon} \cdot r'$ from x (or else we could have chosen $\tilde{r}$ to be smaller). Therefore, we can let S be a subset of size k from $N_{1/\varepsilon}(x)$ and T be a subset of size k of points of distance at least $\frac{2}{\varepsilon} \cdot r'$ from x. The minimum distance between any $s \in S$ and $t \in T$ is at least $r' \cdot \left(\frac{2}{\varepsilon} - \frac{1}{\varepsilon}\right) = \frac{\tilde{r}}{2}$.

Alternatively, every $x \in X$ satisfies $|N_{1/\varepsilon}(x)| < k$. Now, consider the following peeling procedure. Let $X_0 := X$: for each $h \ge 1$, we will inductively create $X_h \subsetneq X_{h-1}$ from X_{h-1}, as follows. First, we pick an arbitrary point $x_h \in X_{h-1}$. For any point $x \in X_{h-1}$ and any integer $i \ge 0$, define $N_i(x_h; X_{h-1}) = N_i(x_h) \cap X_{h-1}$ to be the set of points of distance at most $i \cdot r'$ from x_h in X_{h-1}. By our assumption, we have that $|N_{1/\varepsilon}(x_h; X_{h-1})| \le |N_{1/\varepsilon}(x_h)| < k$, so there exists some $i(h)$ with $0 \le i(h) \le \frac{1}{\varepsilon}$, such that $\frac{|N_{i(h)+1}(x_h, X_{h-1})|}{|N_{i(h)}(x_h, X_{h-1})|} \le k^\varepsilon$ and $|N_{i(h)}(x_h, X_{h-1})| \le k$. Choose such an $i = i(h)$, and let $X_h := X_{h-1} \backslash N_{i(h)}(x_h, X_{h-1})$.

We repeat this process until we have found the first X_ℓ with $|X \backslash X_\ell| \ge k$. Note that each removal process removes at least 1 and at most k elements, so $|X \backslash X_\ell| \le 2k$. Let $S_h = N_{i(h)}(x_h, X_{h-1}) = X_{h-1} \backslash X_h$ for each $1 \le h \le \ell$, so $X \backslash X_\ell = S_1 \cup \cdots \cup S_\ell$. Note, however, that any point within distance r' of some $x \in S_h$ was either in $S_1 \cup \cdots \cup S_{h-1}$, or was in $N_{i(h)+1}(x_h, X_{h-1})$. In other words, every point of distance r' of $x \in S_1 \cup \cdots \cup S_\ell$ is in $\bigcup_{h=1}^{\ell} N_{i(h)+1}(x_h, X_{h-1})$. But this has size at most

$$\sum_{i=1}^{\ell} k^\varepsilon \cdot |N_{i(h)}(x_h, X_{h-1})| = k^\varepsilon \cdot \sum_{i=1}^{\ell} |S_i| \le k^\varepsilon \cdot 2k = 2k^{1+\varepsilon}.$$

So, assuming that $|X| \ge 2k^{1+\varepsilon} + k$, defining $S := S_1 \cup \cdots \cup S_\ell$, we have that $|S| \ge k$ and there are at least k points in X that are *not* within distance $r' = \frac{\varepsilon \cdot \tilde{r}}{2}$ of S. ◀

We include pseudocode for the algorithm described in the proof of Lemma 16, in Algorithm 5.

A.2 Analysis

In this section, we prove that the algorithm indeed generates an $O(1/\varepsilon)$-approximate composable coreset of size at most $O(k^{1+\varepsilon})$. By making ε an arbitrarily small constant, this implies we can find a constant-approximate composable coreset of size $O(k^{1+\varepsilon})$ for any arbitrarily small constant ε.

Let OPT represent the optimal set of k points in all of $X = \bigcup_{j=1}^{m} X^{(j)}$, that maximizes remote-pseudoforest cost. Our goal is to show that there exists a set of k points in $\bigcup_j (P^{(j)} \cup S^{(j)} \cup T^{(j)} \cup U^{(j)} \cup Y^{(j)})$ with pseudoforest cost at least $\Omega(\text{PF}(\text{OPT}))$.

Let $r^{(j)}$ represent the maximum distance from any point in $X^{(j)}$ to its closest point in $Y^{(j)}$. Note that by Proposition 8, $\text{PF}(Y^{(j)}) \ge k \cdot r^{(j)}$. Hence, if $\text{PF}(\text{OPT}) < 10 \cdot k \cdot \max_j r^{(j)}$, there exists some choice of j with $\text{PF}(Y^{(j)}) \ge 0.1 \cdot \text{OPT}$ and $|Y^{(j)}| = k$. Hence, we get a constant-factor approximation in this case. Otherwise, we may assume that $\text{PF}(\text{OPT}) \ge 10 \cdot k \cdot \max_j r^{(j)}$.

Next, for a fixed $X^{(j)}$, let $X_i^{(j)}$ represent the set of points in $X^{(j)}$ closest to $y_i^{(j)}$ among all points in $Y^{(j)}$. Given the optimal solution OPT of k points, let $\text{OPT}_i^{(j)} = \text{OPT} \cap X_i^{(j)}$.

Algorithm 5 FINDST: Find two sets S, T of size k with large $\rho(S, T)$.

```
Input: data $X = \{x_1, \dots, x_n\}$, integer $k$, parameter $\varepsilon \in (0, 1]$, radius $\tilde{r}$.
$r' \leftarrow \frac{\varepsilon \cdot \tilde{r}}{2}$.
for $x$ in $X$ do
    for $i = 0$ to $2/\varepsilon$ do
        $N_i(x) = \{z \in X : \rho(x, z) \leq i \cdot r'\}$.
    end for
    if $|N_{1/\varepsilon}(x)| \geq k$ then
        $S \leftarrow$ arbitrary subset of size $k$ in $N_{1/\varepsilon}(x)$.
        $T \leftarrow$ arbitrary $k$ points of distance at least $\tilde{r}$ from $x$.
        Return $S, T$.
    end if
end for
$S = \emptyset, X_0 \leftarrow X, h \leftarrow 0$
while $|S| < k$ do
    $h \leftarrow h + 1$
    $x_h \in X_{h-1}$ chosen arbitrarily.
    Find $0 \leq i \leq \frac{1}{\varepsilon}$ such that $\frac{|N_{i+1}(x_h) \cap X_{h-1}|}{|N_i(x_h) \cap X_{h-1}|} \leq k^{\varepsilon}$.
    $S_h \leftarrow N_i(x_h) \cap X_{h-1}$.
    $X_h \leftarrow X_{h-1} \backslash S_h$, $S \leftarrow S \cup S_h$
end while
$S \leftarrow$ arbitrary subset of size $k$ in $S$
$T \leftarrow$ arbitrary subset of $k$ points of distance at least $r'$ from all points in $S$.
Return $S, T$.
```

Now, we will define sets $G_i^{(j)}, G_i'^{(j)}$ based on the following cases.

1. If $|X^{(j)}| < 2k^{1+\varepsilon} + k$, define $G_i^{(j)} = G_i'^{(j)} = \mathrm{OPT}_i^{(j)}$ for all $i \leq k$.
2. Else, if $y_i^{(j)} \in \mathrm{OPT}_i^{(j)}$, define $G_i^{(j)}$ as $y_i^{(j)} \cup (U^{(j)} \cap \mathrm{OPT}_i^{(j)})$ and $G_i'^{(j)} = \mathrm{OPT}_i^{(j)}$.
3. Else, if $\mathrm{OPT}_i^{(j)} \backslash U^{(j)} = \emptyset$ (i.e., all points in $\mathrm{OPT}_i^{(j)}$ happen to be in $U^{(j)}$), define $G_i^{(j)} = G_i'^{(j)} = \mathrm{OPT}_i^{(j)}$.
4. Else, define $G_i^{(j)} = y_i^{(j)} \cup (U^{(j)} \cap \mathrm{OPT}_i^{(j)})$, and define $G_i'^{(j)}$ as $\mathrm{OPT}_i^{(j)}$ with the slight modification of moving a single (arbitrary) point in $\mathrm{OPT}_i^{(j)} \backslash U^{(j)}$ to $y_i^{(j)}$.

We will define the sets $G = \bigcup_{i,j} G_i^{(j)}$ and $G' = \bigcup_{i,j} G_i'^{(j)}$.

Importantly, the following five properties always hold for all $i \leq m, j \leq k$. (They even hold in the setting when $|X^{(j)}| < 2k^{1+\varepsilon} + k$, because we defined $U^{(j)} = X^{(j)}$ and $G_i^{(j)} = G_i'^{(j)} = \mathrm{OPT}_i^{(j)}$.)

1. $|G_i'^{(j)}| = |\,\mathrm{OPT}_i^{(j)}\,|$. This means that $|G'| = k$.
2. $G_i^{(j)} \subset G_i'^{(j)} \subset X_i^{(j)}$. This means that $G \subset G'$.
3. $G_i^{(j)} \subset U^{(j)} \cup Y^{(j)}$. This means that $G \subset \bigcup_j (U^{(j)} \cup Y^{(j)})$.
4. Every point in $G_i'^{(j)} \backslash G_i^{(j)}$ is not in $U^{(j)}$. This means that $\bigcup U^{(j)}$ and $G' \backslash G$ are disjoint.
5. If $G_i'^{(j)} \backslash G_i^{(j)}$ is nonempty, then $y_i^{(j)} \in G_i^{(j)}$, so $G_i^{(j)}$ is also nonempty.

Now, note that from changing OPT to G', we never move a point by more than $\max_j r^{(j)}$, which means that $|\,\mathrm{PF}(G') - \mathrm{PF}(\mathrm{OPT})| \leq 2k \cdot \max_j r^{(j)}$. As we are assuming that $\mathrm{PF}(\mathrm{OPT}) \geq 10k \cdot \max_j r^{(j)}$, we have $\mathrm{PF}(G') \geq 0.8 \cdot \mathrm{OPT}$. Next, if a point x is in $G_i'^{(j)}$ but not in $G_i^{(j)}$, then

$x \in X^{(j)} \backslash U^{(j)}$ and $y_i^{(j)} \in G_i^{(j)}$, which means that the cost of x with respect to G' is at most $\max_j r^{(j)}$. So for any set A, if we define $\text{cost}_A(x)$ for $x \in A$ to denote $\min_{y \in A: y \neq x} \rho(x, y)$, then

$$\sum_{x \in G} \text{cost}_{G'}(x) \geq \text{PF}(G') - \sum_{x \in G' \backslash G} \text{cost}_{G'}(x) \geq \text{PF}(G') - k \cdot \max_j r^{(j)} \geq 0.7 \cdot \text{OPT}\,. \tag{8}$$

We now try to find a set $G'' \supset G$ of size k with large pseudoforest cost, but this time we must ensure that $G'' \subset \bigcup_j (P^{(j)} \cup S^{(j)} \cup T^{(j)} \cup U^{(j)} \cup Y^{(j)})$. In other words, to finish the analysis, it suffices to prove the following lemma.

▶ **Lemma 17.** *There exists $G'' \subset \bigcup_j (P^{(j)} \cup S^{(j)} \cup T^{(j)} \cup U^{(j)} \cup Y^{(j)})$ of size at least k, such that $G'' \supset G$ and $\sum_{x \in G} \text{cost}_{G''}(x) \geq \Omega(\varepsilon) \cdot \text{PF}(\text{OPT})$.*

To see why Lemma 17 is sufficient to prove Theorem 4, since $|G| \leq k$ and $|G''| \geq k$ we can choose a set $\hat{G}$ of size k such that $G \subseteq \hat{G} \subseteq G''$. Then, $\hat{G} \subset \bigcup_j (P^{(j)} \cup S^{(j)} \cup T^{(j)} \cup U^{(j)} \cup Y^{(j)})$ and

$$\text{PF}(\hat{G}) = \sum_{x \in \hat{G}} \text{cost}_{\hat{G}}(x) \geq \sum_{x \in G} \text{cost}_{\hat{G}}(x) \geq \sum_{x \in G} \text{cost}_{G''}(x),$$

where the last inequality holds because the cost of x never increases from $\hat{G}$ to a larger set G''. Finally, by Lemma 17, this means $\text{PF}(\hat{G}) \geq \Omega(\varepsilon) \cdot \text{PF}(\text{OPT})$, as desired.

We now prove Lemma 17. First, we show how to construct G''. Let $g = |G|$: note that $g \leq k$. If $g = k$, then in fact $G = G'$ and we can set $G'' = G$, which completes the proof by (8) and Property 3.

Hence, from now on, we may assume that $g < k$. Recall that $X^{(j)} \backslash U^{(j)}$ is contained in a ball of radius $\tilde{r}^{(j)}$. Next, let $\tilde{r}$ be the radius of $\bigcup_j (X^{(j)} \backslash U^{(j)})$. (Note that for $|X^{(j)}| < 2k^{1+\varepsilon} + k$, $X^{(j)} \backslash U^{(j)}$ is empty.) We claim the following proposition.

▶ **Proposition 18.** *There exist $j, j' \leq m$, possibly equal, such that $\rho(S^{(j)}, T^{(j')}) \geq \frac{\varepsilon}{10} \cdot \tilde{r}$.*

Proof. Let $A \subset [m]$ be the subset of indices j such that $|X^{(j)}| \geq 2k^{1+\varepsilon} + k$. Suppose that $\tilde{r} \leq 5 \cdot \max_{j \in A} \tilde{r}^{(j)}$. Then, by setting $j = j'$ to be $\arg\max_{j \subset A} \tilde{r}^{(j)}$, we have that $\rho(S^{(j)}, T^{(j')}) \geq \frac{\varepsilon}{2} \cdot \max_{j \in A} \tilde{r}^{(j)} \geq \frac{\varepsilon}{10} \cdot \tilde{r}$.

Otherwise, $\tilde{r} > 5 \cdot \max_{j \in A} \tilde{r}^{(j)}$. So, if we pick j arbitrarily, the distance between the center of the ball $\tilde{B}^j$ and the furthest center $\tilde{B}^{j'}$ must be at least $0.8 \cdot \tilde{r}$, or else the ball of radius $0.8 \cdot \tilde{r} + \max_{j \in A} \tilde{r}^{(j)} < \tilde{r}$ around the center of $\tilde{B}^j$ contains all of $\bigcup_j (X^{(j)} \backslash U^{(j)})$. Then, $d(S^{(j)}, T^{(j')}) \geq 0.8 \cdot \tilde{r} - \tilde{r}^{(j)} - \tilde{r}^{(j')} \geq 0.4 \cdot \tilde{r}$, which is at least $\frac{\varepsilon}{10} \cdot \tilde{r}$. ◀

We now prove Lemma 17.

Proof. Recall that we already proved the lemma in the case that $G = G'$. So, we may assume $|G| < k$ and $G' \backslash G$ is nonempty. We claim that we can set G'' to be one of $G \cup P^{(j)}$, $G \cup S^{(j)}$, or $G \cup T^{(j')}$, for j, j' chosen in Proposition 18.

First, note that $P^{(j)}, S^{(j)}$, and $T^{(j')}$ have size k, so all three choices of G'' have size at least k.

First, note that $G' \backslash G$ is assumed to be nonempty, which means it is contained in $\bigcup_j (X^{(j)} \backslash U^{(j)})$ by Property 4, which is contained in the radius $\tilde{r}$ ball. Therefore, $G' \backslash G$ has nonempty intersection with $X \backslash (\bigcup_j U^{(j)})$. Now, fix any point $x \in G$. If $\text{cost}_{G'}(x) \geq 3 \cdot \tilde{r}$, then because $G' \backslash G$ has a point in the radius $\tilde{r}$ ball containing $X \backslash (\bigcup_j U^{(j)})$ (and this point is not x since $x \in G$), x has distance at least $\tilde{r}$ from the radius $\tilde{r}$ ball. So, $\text{cost}_{G \cup P^{(j)}}(x) \geq \text{cost}_{G'}(x) - 2\tilde{r} \geq \frac{1}{3} \cdot \text{cost}_{G'}(x)$. Alternatively, if $\text{cost}_{G'}(x) < 3 \cdot \tilde{r}$, then $\text{cost}_{G \cup S^{(j)}}(x) \geq \min(\text{cost}_{G'}(x), \rho(x, S^{(j)}))$ and likewise, $\text{cost}_{G \cup T^{(j')}}(x) \geq \min(\text{cost}_{G'}(x), \rho(x, T^{(j')}))$. So,

$$\begin{aligned}\text{cost}_{G \cup S^{(j)}}(x) + \text{cost}_{G \cup T^{(j')}}(x) &\geq \min(\text{cost}_{G'}(x), \rho(S^{(j)}, T^{(j')})) \\ &\geq \min\left(\text{cost}_{G'}(x), \frac{\varepsilon}{10} \cdot \tilde{r}\right) \geq \frac{\varepsilon}{30} \cdot \text{cost}_{G'}(x).\end{aligned}$$

In all cases, we have that

$$\text{cost}_{G \cup P^{(j)}}(x) + \text{cost}_{G \cup S^{(j)}}(x) + \text{cost}_{G \cup T^{(j')}}(x) \geq \frac{\varepsilon}{30} \cdot \text{cost}_{G'}(x),$$

so adding over all $x \in G$ and choosing among the three choices randomly, we have that the total cost in expectation is at least

$$\frac{1}{3} \cdot \left(\sum_{x \in G} \frac{\varepsilon}{30} \cdot \text{cost}_{G'}(x)\right) = \frac{\varepsilon}{90} \cdot \sum_{x \in G} \text{cost}_{G'}(x) \geq \frac{\varepsilon}{90} \cdot 0.7 \cdot \text{PF}(\text{OPT}) \geq \frac{\varepsilon}{150} \cdot \text{PF}(\text{OPT}).$$

Hence, for one of the three choices of G'', the pseudoforest cost is at least $\frac{\varepsilon}{150} \cdot \text{PF}(\text{OPT})$. ◀

Approximating Pandora's Box with Correlations

Shuchi Chawla
University of Texas – Austin, TX, USA

Evangelia Gergatsouli
University of Wisconsin – Madison, WI, USA

Jeremy McMahan
University of Wisconsin – Madison, WI, USA

Christos Tzamos
University of Wisconsin – Madison, WI, USA
University of Athens, Greece

Abstract

We revisit the classic Pandora's Box (PB) problem under correlated distributions on the box values. Recent work of [13] obtained constant approximate algorithms for a restricted class of policies for the problem that visit boxes in a fixed order. In this work, we study the complexity of approximating the optimal policy which may adaptively choose which box to visit next based on the values seen so far.

Our main result establishes an approximation-preserving equivalence of PB to the well studied Uniform Decision Tree (UDT) problem from stochastic optimization and a variant of the Min-Sum Set Cover (MSSC_f) problem. For distributions of support m, UDT admits a $\log m$ approximation, and while a constant factor approximation in polynomial time is a long-standing open problem, constant factor approximations are achievable in subexponential time [43]. Our main result implies that the same properties hold for PB and MSSC_f.

We also study the case where the distribution over values is given more succinctly as a mixture of m product distributions. This problem is again related to a noisy variant of the Optimal Decision Tree which is significantly more challenging. We give a constant-factor approximation that runs in time $n^{\tilde{O}(m^2/\varepsilon^2)}$ when the mixture components on every box are either identical or separated in TV distance by ε.

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

Keywords and phrases Pandora's Box, Min Sum Set Cover, stochastic optimization, approximation preserving reduction

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.26

Category APPROX

Related Version *Full Version*: https://arxiv.org/pdf/2108.12976.pdf

Funding This work was funded in part by NSF awards CCF-2225259 and CCF-2217069.

1 Introduction

Many everyday tasks involve making decisions under uncertainty; for example driving to work using the fastest route or buying a house at the best price. Although we don't know how the future outcomes of our current decisions will turn out, we can often use some prior information to facilitate the decision making process. For example, having driven on the possible routes to work before, we know which is usually the busiest one. It is also common in such cases that we can remove part of the uncertainty by paying some additional cost. This type of online decision making in the presence of costly information can be modeled as the so-called Pandora's Box problem, first formalized by Weitzman in [52]. In this

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 26; pp. 26:1–26:24
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

problem, the algorithm is given n alternatives called *boxes*, each containing a value from a known distribution. The exact value is not known, but can be revealed at a known *opening* cost specific to the box. The goal of the algorithm is to decide which box to open next and whether to select a value and stop, such that the total *opening cost plus the minimum value revealed* is minimized. In the case of independent distributions on the boxes' values, this problem has a very elegant and simple optimal solution, as described by Weitzman [52]: calculate an index for each box[1], open the boxes in increasing order of index, and stop when the expected gain is worse than the value already obtained.

Weitzman's model makes the crucial assumption that the distributions on the values are independent across boxes. This, however, is not always the case in practice and as it turns out, the simple algorithm of the independent case fails to find the optimal solution under correlated distributions. Generally, the complexity of the Pandora's Box with correlations is not yet well understood. **In this work we develop the first approximately-optimal policies for the Pandora's Box problem with correlated values.**

We consider two standard models of correlation where the distribution over values can be specified explicitly in a succinct manner. In the first, the distribution over values has a small support of size m. In the second the distribution is a mixture of m product distributions, each of which can be specified succinctly. We present approximations for both settings.

A primary challenge in approximating Pandora's Box with correlations is that the optimal solution can be an adaptive policy that determines which box to open depending on the instantiations of values in all of the boxes opened previously. It is not clear that such a policy can even be described succinctly. Furthermore, the choice of which box to open is complicated by the need to balance two desiderata – finding a low value box quickly versus learning information about the values in unopened boxes (a.k.a. the state of the world or realized scenario) quickly. Indeed, the value contained in a box can provide the algorithm with crucial information about other boxes, and inform the choice of which box to open next; an aspect that is completely missing in the independent values setting studied by Weitzman.

Contribution 1: Connection to Decision Tree and a general purpose approximation

Some aspects of the Pandora's Box problem have been studied separately in other contexts. For example, in the Optimal Decision Tree problem (DT) [30, 43], the goal is to identify an unknown hypothesis, out of m possible ones, by performing a sequence of costly tests, whose outcomes depend on the realized hypothesis. This problem has an informational structure similar to that in Pandora's Box. In particular, we can think of every possible joint instantiation of values in boxes as a possible hypothesis, and every opening of a box as a test. The difference between the two problems is that while in Optimal Decision Tree we want to identify the realized hypothesis exactly, in Pandora's Box it suffices to terminate the process as soon as we have found a low value box.

Another closely related problem is the Min Sum Set Cover [21], where boxes only have two kinds of values – *acceptable* or *unacceptable* – and the goal is to find an *acceptable* value as quickly as possible. A primary difference relative to Pandora's Box is that unacceptable boxes provide no further information about the values in unopened boxes.

One of the main contributions of our work is to unearth connections between Pandora's Box and the two problems described above. We show that Pandora's Box is essentially equivalent to a special case of Optimal Decision Tree (called Uniform Decision Tree

[1] This is a special case of Gittins index [25].

or UDT) where the underlying distribution over hypotheses is uniform – the approximation ratios of these two problems are related within log-log factors. Surprisingly, in contrast, the non-uniform DT appears to be harder than non-uniform PANDORA'S BOX. We relate these two problems by showing that both are in turn related to a new version of MIN SUM SET COVER, that we call MIN SUM SET COVER WITH FEEDBACK (MSSC$_f$). These connections are summarized in Figure 1. We can thus build on the rich history and large collection of results on these problems to offer efficient algorithms for PANDORA'S BOX. We obtain a polynomial time $\tilde{O}(\log m)$ approximation for PANDORA'S BOX, where m is the number of distinct value vectors (a.k.a. scenarios) that may arise; as well as constant factor approximations in subexponential time.

$$\underbrace{\mathrm{PB} \xleftrightarrow{\text{Section 4}} \mathrm{UMSSC}_f}_{\text{Log-log factors}} \underbrace{\xleftrightarrow{\text{Section 5}} \mathrm{UDT}}_{\text{Constant factors}}$$

Figure 1 A summary of our approximation preserving reductions.

It is an important open question whether constant factor approximations exist for UNIFORM DECISION TREE: the best known lower-bound on the approximation ratio is 4 while it is known that it is not NP-hard to obtain super-constant approximations under the Exponential Time Hypothesis. The same properties transfer also to PANDORA'S BOX and MIN SUM SET COVER WITH FEEDBACK. Pinning down the tight approximation ratio for any of these problems will directly answer these questions for any other problem in the equivalence class we establish.

The key technical component in our reductions is to find an appropriate stopping rule for PANDORA'S BOX: after opening a few boxes, how should the algorithm determine whether a small enough value has been found or whether further exploration is necessary? We develop an iterative algorithm that in each phase finds an appropriate threshold, with the exploration terminating as soon as a value smaller than the threshold is found, such that there is a constant probability of stopping in each phase. Within each phase then the exploration problem can be solved via a reduction to UDT. The challenge is in defining the stopping thresholds in a manner that allows us to relate the algorithm's total cost to that of the optimal policy.

Contribution 2: Approximation for the mixture of distributions model

Having established the general purpose reductions between PANDORA'S BOX and DT, we turn to the mixture of product distributions model of correlation. This special case of PANDORA'S BOX interpolates between Weitzman's independent values setting and the fully general correlated values setting. In this setting, we use the term "scenario" to denote the different product distributions in the mixture. The information gathering component of the problem is now about determining which product distribution in the mixture the box values are realized from. Once the algorithm has determined the realized scenario (a.k.a. product distribution), the remaining problem amounts to implementing Weitzman's strategy for that scenario.

We observe that this model of correlation for PANDORA'S BOX is related to the noisy version of DT, where the results of some tests for a given realized hypothesis are not deterministic. One challenge for DT in this setting is that any individual test may give us very little information distinguishing different scenarios, and one needs to combine information across sequences of many tests in order to isolate scenarios. This challenge is inherited by Pandora's Box.

Previous work on noisy DT obtained algorithms whose approximations and runtimes depend on the amount of noise. In contrast, we consider settings where the level of noise is arbitrary, but where the mixtures satisfy a *separability assumption.* In particular, we assume that for any given box, if we consider the marginal distributions of the value in the box under different scenarios, these distributions are either identical or sufficiently different (e.g., at least ε in TV distance) across different scenarios. Under this assumption, we design a constant-factor approximation for Pandora's Box that runs in $n^{\tilde{O}(m^2/\varepsilon^2)}$ (Theorem 18), where n is the number of boxes. The formal result and the algorithm is presented in Section 6.

1.1 Related work

The Pandora's Box problem was first introduced by Weitzman in the Economics literature [52]. Since then, there has been a long line of research studying Pandora's Box and its many variants ; non-obligatory inspection [19, 8, 7, 22], with order constraints [37, 9], with correlation [13, 24], with combinatorial costs [6], competitive information design [18], delegated version [5], and finally in an online setting [20]. Multiple works also study the generalized setting where more information can be obtained for a price [12, 32, 15, 14] and in settings with more complex combinatorial constraints [50, 26, 33, 1, 35, 36, 31].

Chawla et al. [13] were the first to study Pandora's Box with correlated values, but they designed approximations relative to a simpler benchmark, namely the optimal performance achievable using a so-called *Partially Adaptive* strategy that cannot adapt the order in which it opens boxes to the values revealed. In general, optimal strategies can decide both the ordering of the boxes and the stopping time based on the values revealed. [13] designed an algorithm with performance no more than a constant factor worse than the optimal Partially Adaptive strategy.

In Min Sum Set Cover the line of work was initiated by [21], and continued with improvements and generalizations to more complex constraints by [3, 46, 4, 51].

Optimal decision tree is an old problem studied in a variety of settings ([49, 48, 30, 29]), while its most notable application is in active learning settings. It was proven to be NP-Hard by Hyafil and Rivest [38]. Since then the problem of finding the best approximation algorithm was an active one [23, 45, 42, 17, 10, 11, 30, 34, 16, 2], where finally a greedy $\log m$ for the general case was given by [30]. This approximation ratio is proven to be the best possible [10]. For the case of Uniform decision tree less is known, until recently the best algorithm was the same as the optimal decision tree, and the lower bound was 4 [10]. The recent work of Li et al. [43] showed that there is an algorithm strictly better than $\log m$ for the uniform decision tree.

The noisy version of optimal decision tree was first studied in [29][2], which gave an algorithm with runtime that depends exponentially on the number of noisy outcomes. Subsequently, Jia et al. in [40] gave an $(\min(r, h) + \log m)$-approximation algorithm, where r (resp. h) is the maximum number of different test results per test (resp. scenario) using a reduction to Adaptive Submodular Ranking problem [41]. In the case of large number of noisy outcome they obtain a $\log m$ approximation exploiting the connection to Stochastic Set Cover [44, 39].

[2] This result is based on a result from [27] which turned out to be wrong [47]. The correct results are presented in [28]

2 Preliminaries

In this paper we study the connections between three different sequential decision making problems – Optimal Decision Tree, Pandora's Box, and Min Sum Set Cover. We describe these problems formally below.

Optimal Decision Tree

In the Optimal Decision Tree problem (denoted DT) we are given a set $\mathcal{S}$ of m scenarios $s \in \mathcal{S}$, each occurring with (known) probability p_s; and n tests $\mathcal{T} = \{T_i\}_{i\in[n]}$, each with cost 1. Nature picks a scenario $s \in \mathcal{S}$ from the distribution p but this scenario is unknown to the algorithm. The goal of the algorithm is to determine which scenario is realized by running a subset of the tests $\mathcal{T}$. When test T_i is run and the realized scenario is s, the test returns a result $T_i(s) \in \mathbb{R}$.

Output. The output of the algorithm is a decision tree where at each node there is a test that is performed, and the branches are the outcomes of the test. In each of the leaves there is an individual scenario that is the only one consistent with the results of the test in the unique path from the root to this leaf. Observe that there is a single leaf corresponding to each scenario s. We can represent the tree as an *adaptive policy* defined as follows:

▶ **Definition 1** (Adaptive Policy π). *An adaptive policy $\pi : \cup_{X\subseteq\mathcal{T}}\mathbb{R}^X \to \mathcal{T}$ is a function that given a set of tests done so far and their results, returns the next test to be performed.*

Objective. For a given decision tree or policy π, let $\text{cost}_s(\pi)$ denote the total cost of all of the tests on the unique path in the tree from the root to the leaf labeled with scenario s. The objective of the algorithm is to find a policy π that minimizes the average cost $\sum_{s\in\mathcal{S}} p_s \text{cost}_s(\pi)$.
We use the term Uniform Decision Tree (UDT) to denote the special case of the problem where $p_s = 1/m$ for all scenarios s.

Pandora's Box

In the Pandora's Box problem we are given n boxes, each with cost $c_i \geq 0$ and value v_i. The values $\{v_i\}_{i\in[n]}$ are distributed according to known distribution $\mathcal{D}$. We assume that $\mathcal{D}$ is an arbitrary correlated distribution over vectors $\{v_i\}_{i\in[n]} \in \mathbb{R}^n$. We call vectors of values *scenarios* and use $s = \{v_i\}_{i\in[n]}$ to denote a possible realization of the scenario. As in DT, nature picks a scenario from the distribution D and this realization is a priori unknown to the algorithm. The goal of the algorithm is to pick a box of small value. The algorithm can observe the values realized in the boxes by opening any box i at its respective costs c_i.

Output. The output of the algorithm is an adaptive policy π for opening boxes and a stopping condition. The policy π takes as input a subset of the boxes and their associated values, and either returns the index of a box $i \in [n]$ to be opened next or stops and selects the minimum value seen so far. That is, $\pi : \cup_{X\subseteq[n]}\mathbb{R}^X \to [n] \cup \{\perp\}$ where $\perp$ denotes stopping.

Objective. For a given policy π, let $\pi(s)$ denote the set of boxes opened by the policy prior to stopping when the realized scenario is s. The objective of the algorithm is to minimize the expected cost of the boxes opened plus the minimum value discovered, where the expectation is taken over all possible realizations of the values in each box.[3] Formally the objective is given by

$$\mathbb{E}_{s\sim\mathcal{D}}\left[\min_{i\in\pi(s)} v_{is} + \sum_{i\in\pi(s)} c_i\right],$$

For simplicity of presentation, from now on we assume that $c_i = 1$ for all boxes, but we show in the Appendix of the full version how to adapt our results to handle non-unit costs, without any loss in the approximation factors.

We use UPB to denote the special case of the problem where the distribution $\mathcal{D}$ is uniform over m scenarios.

Min Sum Set Cover with Feedback

In Min Sum Set Cover, we are given n elements and a collection of m sets $\mathcal{S}$ over them, and a distribution $\mathcal{D}$ over the sets. The output of the algorithm is an ordering π over the elements. The cost of the ordering for a particular set $s \in \mathcal{S}$ is the index of the first element in the ordering that belongs to the set s, that is, $\text{cost}_s(\pi) = \min\{i : \pi(i) \in s\}$. The goal of the algorithm is to minimize the expected cost $\mathbb{E}_{s\sim\mathcal{D}}[\text{cost}_s(\pi)]$.

We define a variant of the Min Sum Set Cover problem, called Min Sum Set Cover with Feedback (MSSC$_f$). As in the original problem, we are given a set of n elements, a collection of m sets $\mathcal{S}$ and a distribution $\mathcal{D}$ over the sets. Nature instantiates a set $s \in \mathcal{S}$ from the distribution $\mathcal{D}$; the realization is unknown to the algorithm. Furthermore, in this variant, each element provides *feedback* to the algorithm when the algorithm "visits" this element; this feedback takes on the value $f_i(s) \in \mathbb{R}$ for element $i \in [n]$ if the realized set is $s \in \mathcal{S}$.

Output. The algorithm once again produces an ordering π over the elements. Observe that the feedback allows the algorithm to adapt its ordering to previously observed values. Accordingly, π is an adaptive policy that maps a subset of the elements and their associated feedback, to the index of another element $i \in [n]$. That is, $\pi : \cup_{X\subseteq[n]}\mathbb{R}^X \to [n]$.

Objective. As before, the cost of the ordering for a particular set $s \in \mathcal{S}$ is the index of the first element in the ordering that belongs to the set s, that is, $\text{cost}_s(\pi) = \min\{i : \pi(i) \in s\}$. The goal of the algorithm is to minimize the expected cost $\mathbb{E}_{s\sim\mathcal{D}}[\text{cost}_s(\pi)]$.

Commonalities and notation

As the reader has observed, we capture the commonalities between the different problems through the use of similar notation. Scenarios in DT correspond to value vectors in PB and to sets in MSSC$_f$; all are denoted by s, lie in the set $\mathcal{S}$, and are drawn by nature from a known joint distribution $\mathcal{D}$. Tests in DT correspond to boxes in PB and elements in MSSC$_f$;

[3] In the original version of the problem studied by Weitzman [52] the values are independent across boxes, and the goal is to maximize the value collected minus the costs paid, in contrast to the minimization version we study here.

we index each by $i \in [n]$. The algorithm for each problem produces an adaptive ordering π over these tests/boxes/elements. Test outcomes $T_i(s)$ in DT correspond to box values $v_i(s)$ in PB and feedback $f_i(s)$ in MSSC_f. We will use the terminology and notation across different problems interchangeably in the rest of the paper.

2.1 Modeling Correlation

In this work we study two general ways of modeling the correlation between the values in the boxes. **Explicit Distributions.** In this case, $\mathcal{D}$ is a distribution over m *scenarios* where the j'th scenario is realized with probability p_j, for $j \in [m]$. Every scenario corresponds to a fixed and known vector of values contained in each box. Specifically, box i has value $v_{ij} \in \mathbb{R}^+ \cup \{\infty\}$ for scenario j.

Mixture of Distributions. We also consider a more general setting, where $\mathcal{D}$ is a mixture of m product distributions. Specifically, each scenario j is a product distribution; instead of giving a deterministic value for every box i, the result is drawn from distribution $\mathcal{D}_{ij}$. This setting is a generalization of the explicit distributions setting described before.

3 Roadmap of the Reductions and Implications

In Figure 2, we give an overview of all the main technical reductions shown in Sections 4 and 5. An arrow $A \to B$ means that we gave an approximation preserving reduction from problem A to problem B. Therefore an algorithm for B that achieves approximation ratio α gives also an algorithm for A with approximation ratio $O(\alpha)$ (or $O(\alpha \log \alpha)$ in the case of black dashed lines). For the exact guarantees we refer to the formal statement of the respective theorem. The gray lines denote less important claims or trivial reductions (e.g. in the case of A being a subproblem of B).

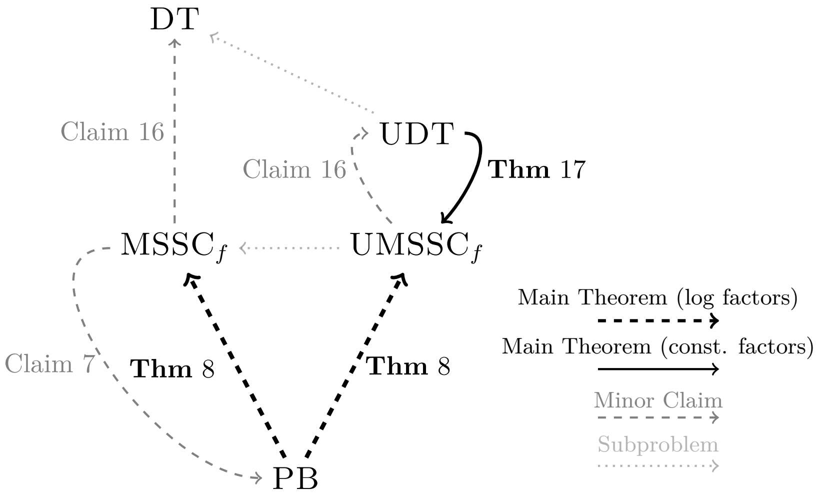


Figure 2 Summary of all our reductions. Bold black lines denote our main theorems, gray dashed are minor claims, and dotted lines are trivial reductions.

3.1 Approximating Pandora's Box

Given our reductions and using the best known results for Uniform Decision Tree from [43] we immediately obtain efficient approximation algorithms for Pandora's Box. We repeat the results of [43] below.

▶ **Theorem 2** (Theorems 3.1 and 3.2 from [43]).

- *There is a $O(\log m/\log \mathrm{OPT})$-approximation algorithm for UDT that runs in polynomial time, where OPT is the cost of the optimal solution of the UDT instance.*
- *There is a $\frac{9+\varepsilon}{\alpha}$-approximation algorithm for UDT that runs in time $n^{\tilde{O}(m^\alpha)}$ for any $\alpha \in (0,1)$.*

Using the results of Theorem 2 combined with Theorem 8 and Claim 16 we get the following corollary.

▶ **Corollary 3.** *From the best-known results for UDT, we have that*

- *There is a $\tilde{O}(\log m)$-approximation algorithm for PB that runs in polynomial time*[4].
- *There is a $\tilde{O}(1/\alpha)$-approximation algorithm for PB that runs in time $n^{\tilde{O}(m^\alpha)}$ for any $\alpha \in (0,1)$.*

An immediate implication of the above corollary is that it is not NP-hard to obtain a superconstant approximation for PB, formally stated below.

▶ **Corollary 4.** *It is not NP-hard to achieve any superconstant approximation for PB assuming the Exponential Time Hypothesis.*

Observe that the logarithmic approximation achieved in Corollary 3 loses a $\log\log m$ factor (hence the $\tilde{O}$) as it relies on the more complex reduction of Theorem 8. If we choose to use the more direct naive reduction (given in the full version of our paper) to the Optimal Decision Tree where the tests have non-unit costs (which also admits a $O(\log m)$-approximation [34, 41]), we get the following corollary.

▶ **Corollary 5.** *There exists an efficient algorithm that is $O(\log m)$-approximate for Pandora's Box and with or without unit-cost boxes.*

3.2 Constant approximation for Partially Adaptive PB

Moving on, we show how our reduction can be used to obtain and improve the results of [13]. Recall that in [13] the authors presented a constant factor approximation algorithm against a Partially Adaptive benchmark where the order of opening boxes must be fixed up front.

In such a case, the reduction of Section 4 can be used to reduce PB to the standard Min Sum Set Cover (i.e. without feedback), which admits a 4-approximation [21].

▶ **Corollary 6.** *There exists a polynomial time algorithm for PB that is $O(1)$-competitive against the partially adaptive benchmark.*

The same result applies even in the case of non-uniform opening costs. This is because a 4-approximate algorithm for Min Sum Set Cover is known even when elements have arbitrary costs [46]. The case of non-uniform opening costs has also been considered for Pandora's Box by [13] but only provide an algorithm to handle polynomially bounded opening costs.

[4] If additionally the possible number of outcomes is a constant K, this gives a $O(\log m)$ approximation without losing an extra logarithmic factor, since $\mathrm{OPT} \geq \log_K m$, as observed by [43].

4 Connecting Pandora's Box and MSSC$_f$

In this section we establish the connection between Pandora's Box and Min Sum Set Cover with Feedback. We show that the two problems are equivalent up to logarithmic factors in approximation ratio.

One direction of this equivalence is easy to see in fact: Min Sum Set Cover with Feedback is a special case of Pandora's Box. Note that in both problems we examine boxes/elements in an adaptive order. In PB we stop when we find a sufficiently small value; in MSSC$_f$ we stop when we find an element that belongs to the instantiated scenario. To establish a formal connection, given an instance of MSSC$_f$, we can define the "value" of each element i in scenario s as being 0 if the element belongs to the set s and as being $L + f_i(s)$ for some sufficiently large value L where $f_i(s)$ is the feedback of element i for set s. This places the instance within the framework of PB and a PB algorithm can be used to solve it. We formally describe this reduction in Section A of the Appendix.

▷ Claim 7. If there exists an $\alpha(n, m)$-approximation algorithm for PB then there exists a $\alpha(n, m)$-approximation for MSSC$_f$.

The more interesting direction is a reduction from PB to MSSC$_f$. In fact we show that a general instance of PB can be reduced to the simpler *uniform* version of Min Sum Set Cover with Feedback. We devote the rest of this section to proving the following theorem.

▶ **Theorem 8** (Pandora's Box to MSSC$_f$). *If there exists an $a(n, m)$ approximation algorithm for UMSSC$_f$ then there exists a $O(\alpha(n + m, m^2) \log \alpha(n + m, m^2))$-approximation for PB.*

Guessing a stopping rule and an intermediate problem

The feedback structure in PB and MSSC$_f$ is quite similar, and the main component in which the two problems differ is the stopping condition. In MSSC$_f$, an algorithm can stop examining elements as soon as it finds one that "covers" the realized set. In PB, when the algorithm observes a value in a box, it is not immediately apparent whether the value is small enough to stop or whether the algorithm should probe further, especially if the scenario is not fully identified. The key to relating the two problems is to "guess" an appropriate stopping condition for PB, namely an appropriate threshold T such that as soon as the algorithm observes a value smaller than this threshold, it stops. We say that the realized scenario is "covered".

To formalize this approach, we introduce an intermediate problem called *Pandora's Box with costly outside option T* (also called *threshold*), denoted by PB$_{\leq T}$. In this version the objective is to minimize the cost of finding a value $\leq T$, while we have the extra option to quit searching by opening an *outside option* box of cost T. We say that a scenario is *covered* in a given run of the algorithm if it does not choose the outside option box T.

We show that Pandora's Box can be reduced to PB$_{\leq T}$ with a logarithmic loss in approximation factor, and then PB$_{\leq T}$ can be reduced to Min Sum Set Cover with Feedback with a constant factor loss. The following two results capture the details of these reductions.

▷ Claim 9. If there exists an $\alpha(n, m)$ approximation algorithm for UMSSC$_f$ then there exists an $3\alpha(n + m, m^2)$-approximation for UPB$_{\leq T}$.

▶ **Main Lemma 10.** *Given a polynomial-time $\alpha(n,m)$-approximation algorithm for UPB$_{\leq T}$, there exists a polynomial-time $O(\alpha(n,m)\log\alpha(n,m))$-approximation for PB.*

The relationship between PB$_{\leq T}$ and Min Sum Set Cover with Feedback is relatively straightforward and requires explicitly relating the structure of feedback in the two problems. We describe the details in Section A of the Appendix.

Putting it all together. The proof of Theorem 8 follows by combining Claim 9 with Lemmas 11 and 10 presented in the following sections. Proofs of Claims 7, 9 deferred to Section A of the Appendix. The rest of this section is devoted to proving Lemmas 11 and 10.

4.1 Reducing Pandora's Box to PB$_{\leq T}$

Recall that a solution to Pandora's Box involves two components ; (1) the order in which to open boxes and (2) a stopping rule. The goal of the reduction to PB$_{\leq T}$ is to simplify the stopping rule of the problem, by making values either 0 or ∞, therefore allowing us to focus on the order in which boxes are opened, rather than which value to stop at. We start by presenting our main tool, a reduction to Min Sum Set Cover with Feedback in Section 4.1.1 and then improve upon that to reduce from the **uniform** version of MSSC$_f$ (Section 4.1.2).

4.1.1 Main Tool

The high level idea in this reduction is that we repeatedly run the algorithm for PB$_{\leq T}$ with increasingly larger value of T with the goal of covering some mass of scenarios at every step. The thresholds for every run have to be cleverly chosen to guarantee that enough mass is covered at every run. The distributions on the boxes remain the same, and this reduction does not increase the number of boxes, therefore avoiding the issues faced by the naive reduction given in the full version of the paper. Formally, we show the following lemma.

▶ **Main Lemma 11.** *Given a polynomial-time $\alpha(n,m)$-approximation algorithm for PB$_{\leq T}$, there exists a polynomial-time $O(\alpha(n,m)\log\alpha(n,m))$-approximation for PB.*

Algorithm 1 Reduction from PB to PB$_{\leq T}$.

```
Input: Oracle $\mathcal{A}(T)$ for PB$_{\leq T}$, set of all scenarios $\mathcal{S}$.
$i \leftarrow 0$ // Number of current Phase
while $\mathcal{S} \neq \emptyset$ do
    Use $\mathcal{A}$ to find smallest $T_i$ via Binary Search s.t.
      $\mathbf{Pr}$ [accepting the outside option $T_i$] $\leq 0.2$
    Call the oracle $\mathcal{A}(T_i)$ on set $\mathcal{S}$ to obtain policy $\pi_i$
    $\mathcal{S} \leftarrow \mathcal{S} \setminus$ {scenarios with total cost $\leq T_i$}
end
for $i \leftarrow 0$ to $\infty$ do
    Run policy $\pi_i$ until it terminates and selects a box, or accumulates probing cost
      $T_i$.
end
```

We will now analyze the policy produced by this algorithm.

Proof of Main Lemma 11. We start with some notation. Given an instance $\mathcal{I}$ of PB, we repeatedly run $\text{PB}_{\le T}$ in *phases*. Phase i consists of running $\text{PB}_{\le T}$ with threshold T_i on a sub instance of the original problem where we are left with a smaller set of scenarios, with their probabilities reweighted to sum to 1. Call this set of scenarios $\mathcal{S}_i$ for phase i and the corresponding instance $\mathcal{I}_i$. After every phase i, we remove the probability mass that was covered[5], and run $\text{PB}_{\le T}$ on this new instance with a new threshold T_{i+1}. In each phase, the boxes, costs and values remain the same, but the stopping condition changes: thresholds T_i increase in every subsequent phase. The thresholds are chosen such that at the end of each phase, 0.8 of the remaining probability mass is covered. The reduction process is formally shown in Algorithm 1.

Accounting for the cost of the policy. We first note that the total cost of the policy in phase i conditioned on reaching that phase is at most $2T_i$: if the policy terminates in that phase, it selects a box with value at most T_i. Furthermore, the policy incurs probing cost at most T_i in the phase. We can therefore bound the total cost of the policy as $\le 2\sum_{i=0}^{\infty}(0.2)^i T_i$. We will now relate the thresholds T_i to the cost of the optimal PB policy for $\mathcal{I}$. To this end, we define corresponding thresholds for the optimal policy that we call *p-thresholds*. Let $\pi^*_{\mathcal{I}}$ denote the optimal PB policy for $\mathcal{I}$ and let c_s denote the cost incurred by $\pi^*_{\mathcal{I}}$ when scenario i is realized. A p-threshold is the minimum possible threshold T such that at most p mass of the scenarios has cost more than T in PB, formally defined below.

▶ **Definition 12** (p-Threshold). *Let $\mathcal{I}$ be an instance of PB and c_s be the cost of scenario $s \in \mathcal{S}$ in $\pi^*_{\mathcal{I}}$, we define the p-threshold as*

$$t_p = \min\{T : \boldsymbol{Pr}\left[c_s > T\right] \le p\}.$$

The following two lemmas relate the cost of the optimal policy to the p-thresholds, and the p-thresholds to the thresholds T_i our algorithm finds. The proofs of both lemmas are deferred to Section A.1 of the Appendix. We first formally define a *sub-instance* of the given Pandora's Box instance.

▶ **Definition 13** (Sub-instance). *Let $\mathcal{I}$ be an instance of $\{PB_{\le T}, PB\}$ with set of scenarios $\mathcal{S}_{\mathcal{I}}$ each with probability $p^{\mathcal{I}}_s$. For any $q \in [0,1]$ we call $\mathcal{I}'$ a q-sub instance of $\mathcal{I}$ if $\mathcal{S}_{\mathcal{I}'} \subseteq \mathcal{S}_{\mathcal{I}}$ and $\sum_{s\in\mathcal{S}_{\mathcal{I}'}} p^{\mathcal{I}}_s = q$.*

▶ **Lemma 14** (Optimal Lower Bound). *Let $\mathcal{I}$ be the instance of PB. For any $q < 1$, any $\alpha > 1$, and $\beta \ge 2$, for the optimal policy $\pi^*_{\mathcal{I}}$ for PB it that*

$$\text{cost}(\pi^*_{\mathcal{I}}) \ge \sum_{i=0}^{\infty} \frac{1}{\beta\alpha} \cdot (q)^i\, t_{q^i/\beta\alpha}.$$

▶ **Lemma 15.** *Given an instance $\mathcal{I}$ of PB; an α-approximation algorithm $\mathcal{A}_T$ to $PB_{\le T}$; and any $q < 1$ and $\beta \ge 2$, suppose that the threshold T satisfies*

$$T \ge t_{q/(\beta\alpha)} + \beta\alpha \sum_{\substack{c_s\in[t_q, t_{q/(\beta\alpha)}]\\ s\in\mathcal{S}}} c_s \frac{p_s}{q}.$$

Then if $\mathcal{A}_T$ is run on a q-sub instance of $\mathcal{I}$ with threshold T, at most a total mass of $(2/\beta)q$ of the scenarios pick the outside option box T.

[5] Recall, a scenario is *covered* if it does not choose the outside option box.

Calculating the thresholds. For every phase i we choose a threshold T_i such that $T_i = \min\{T : \mathbf{Pr}\left[c_s > T\right] \leq 0.2\}$ i.e. at most 0.2 of the probability mass of the scenarios are not covered. In order to select this threshold, we do binary search starting from $T = 1$, running every time the α-approximation algorithm for $\mathrm{PB}_{\leq T}$ with outside option box T and checking how many scenarios select it. We denote by $\mathrm{Int}_i = [t_{(0.2)^i}, t_{(0.2)^i/(10\alpha)}]$ the relevant interval of costs at every run of the algorithm, then by Lemma 15 for $\beta = 10$, we know that for remaining total probability mass $(0.2)^i$, any threshold which satisfies

$$T_i \geq t_{(0.2)^{i-1}/10a} + 10\alpha \sum_{\substack{s\in\mathcal{S}\\ c_s\in\mathrm{Int}_i}} c_s \frac{p_s}{(0.2)^i}$$

also satisfies the desired covering property, i.e. at least 0.8 mass of the current scenarios is covered. Therefore the threshold T_i found by our binary search satisfies the following

$$T_i = t_{(0.2)^{i-1}/10a} + 10\alpha \sum_{\substack{s\in\mathcal{S}\\ c_s\in\mathrm{Int}_i}} c_s \frac{p_s}{(0.2)^i}. \tag{1}$$

Bounding the final cost. To bound the final cost, we recall that at the end of every phase we cover 0.8 of the remaining scenarios. Furthermore, we observe that each threshold T_i is charged in the above Equation (1) to optimal costs of scenarios corresponding to intervals of the form $\mathrm{Int}_i = [t_{(0.2)^i}, t_{(0.2)^i/(10\alpha)}]$. Note that these intervals are overlapping. We therefore get

$$\begin{aligned}
\mathrm{cost}(\pi_{\mathcal{I}}) &\leq 2\sum_{i=0}^{\infty}(0.2)^i T_i \\
&= 2\sum_{i=0}^{\infty}\left((0.2)^i t_{(0.2)^{i-1}/10a} + 10\alpha \sum_{\substack{s\in\mathcal{S}\\ c_s\in\mathrm{Int}_i}} c_s p_s\right) && \text{From equation (1)} \\
&\leq 4\cdot 10\alpha\pi^*_{\mathcal{I}} + 20\alpha\sum_{i=0}^{\infty}\sum_{\substack{s\in\mathcal{S}\\ c_s\in\mathrm{Int}_i}} c_s p_s && \text{Using Lemma 14 for } \beta = 10, q = 0.2 \\
&\leq 40\alpha\log\alpha\cdot\pi^*_{\mathcal{I}}.
\end{aligned}$$

Where the last inequality follows since each scenario with cost c_s can belong to at most $\log\alpha$ intervals, therefore we get the theorem. ◀

Notice the generality of this reduction; the distributions on the values are preserved, and we did not make any more assumptions on the scenarios or values throughout the proof. Therefore we can apply this tool regardless of the type of correlation or the way it is given to us, e.g. we could be given a parametric distribution, or an explicitly given distribution, as we see in the next section.

4.1.2 An Even Stronger Tool

Moving one step further, we show that if we instead of $\mathrm{PB}_{\leq T}$ we had an α-approximation algorithm for $\mathrm{UPB}_{\leq T}$ we can obtain the same guarantees as the ones described in Lemma 11. Observe that we cannot directly use Algorithm 1 since the oracle now requires that all scenarios have the same probability, while this might not be the case in the initial PB instance. The theorem stated formally follows.

▶ **Main Lemma 10.** *Given a polynomial-time $\alpha(n,m)$-approximation algorithm for $UPB_{\leq T}$, there exists a polynomial-time $O(\alpha(n,m)\log\alpha(n,m))$-approximation for PB.*

We are going to highlight the differences with the proof of Main Lemma 11, and show how to change Algorithm 1 to work with the new oracle, that requires the scenarios to have uniform probability. The function **Expand** shown in Algorithm 2 is used to transform the instance of scenarios to a uniform one where every scenario has the same probability by creating multiple copies of the more likely scenarios. The function is formally described in Algorithm 3 in Section A.2 of the Appendix, alongside the proof of Main Lemma 10.

Algorithm 2 Reduction from PB to $\text{UPB}_{\leq T}$.

```
Input: Oracle $\mathcal{A}(T)$ for $\text{UPB}_{\leq T}$, set of all scenarios $\mathcal{S}$, $c = 1/10, \delta = 0.1$.
$i \leftarrow 0$ // Number of current Phase
while $\mathcal{S} \neq \emptyset$ do
    Let $\mathcal{L} = \left\{ s \in \mathcal{S} : p_s \leq c \cdot \frac{1}{|\mathcal{S}|} \right\}$ // Remove low probability scenarios
    $\mathcal{S}' = \mathcal{S} \setminus \mathcal{L}$
    $\mathcal{UI} = \text{Expand}(\mathcal{S}')$
    In instance $\mathcal{UI}$ use $\mathcal{A}$ to find smallest $T_i$ via Binary Search s.t.
      $\mathbf{Pr}[\text{accepting } T_i] \leq \delta$
    Call the oracle $\mathcal{A}(T_i)$
    $\mathcal{S} \leftarrow \left( \mathcal{S}' \setminus \{ s \in \mathcal{S}' : c_s \leq T_i \} \right) \cup \mathcal{L}$
end
```

5 Connecting MSSC_f and Optimal Decision Tree

In this section we establish the connection between Min Sum Set Cover with Feedback and Optimal Decision Tree. We show that the uniform versions of these problems are equivalent up to constant factors in approximation ratio. The proofs of this section are deferred to the full version of the paper in ArXiv.

▷ Claim 16. If there exists an $\alpha(n,m)$-approximation algorithm for DT (UDT) then there exists a $(1+\alpha(n,m))$-approximation algorithm for MSSC_f (resp. UMSSC_f).

▶ **Theorem 17** (Uniform Decision Tree to UMSSC_f)**.** *Given an $\alpha(m,n)$-approximation algorithm for $UMSSC_f$ then there exists an $O(\alpha(n+m,m))$-approximation algorithm for UDT.*

The formal proofs of these statements can be found in the full version, here we sketch the main ideas.

One direction of this equivalence is again easy to see. The main difference between Optimal Decision Tree and MSSC_f is that the former requires scenarios to be exactly identified whereas in the latter it suffices to simply find an element that covers the scenario. In particular, in MSSC_f an algorithm could cover a scenario without identifying it by, for example, covering it with an element that covers multiple scenarios. To reduce MSSC_f to DT we simply introduce extra feedback into all of the elements of the MSSC_f instance such that the elements isolate any scenarios they cover. (That is, if the algorithm picks an element that covers some subset of scenarios, this element provides feedback about which of the covered scenarios materialized.) This allows us to relate the cost of isolation and the cost of covering to within the cost of a single additional test, implying Claim 16.

Proof Sketch of Theorem 17. The other direction is more complicated, as we want to ensure that covering implies isolation. Given an instance of UDT, we create a special element for each scenario which is the unique element covering the scenario and also isolates the scenario from all other scenarios. The intention is that an algorithm for MSSC_f on this new instance only chooses the special isolating element in a scenario after it has identified the scenario. If that happens, then the algorithm's policy is a feasible solution to the UDT instance and incurs no extra cost. The problem is that an algorithm for MSSC_f over the modified instance may use the special covering element before isolating a scenario. We argue that this choice can be "postponed" in the policy to a point at which isolation is nearly achieved without incurring too much extra cost. This involves careful analysis of the policy's decision tree and we present details in the appendix.

Why our reduction does not work for DT. Our analysis above heavily uses the fact that the probabilities of all scenarios in the UDT instance are equal. This is because the "postponement" of elements charges increased costs of some scenarios to costs of other scenarios. In fact, our reduction above fails in the case of non-uniform distributions over scenarios – it can generate an MSSC_f instance with optimal cost much smaller than that of the original DT instance.

To see this, consider an example with m scenarios where scenarios 1 through $m-1$ happen with probability $\varepsilon/(m-1)$ and scenario m happens with probability $1-\varepsilon$. There are $m-1$ tests of cost 1 each. Test i for $i \in [m-1]$ isolates scenario i from all others. Observe that the optimal cost of this DT instance is at least $(1-\varepsilon)(m-1)$ as all $m-1$ tests need to be run to isolate scenario m. Our construction of the MSSC_f instance adds another isolating test for scenario m. A solution to this instance can use this new test at the beginning to identify scenario m and then run other tests with the remaining ε probability. As a result, it incurs cost at most $(1-\varepsilon)+\varepsilon(m-1)$, which is a factor of $1/\varepsilon$ cheaper than that of the original DT instance.

6 Mixture of Product Distributions

In this section we switch gears and consider the case where we are given a mixture of m product distributions. Observe that using the tool described in Section 4.1.1, we can reduce this problem to $\mathrm{PB}_{\leq T}$. This now is equivalent to the noisy version of DT [28, 40] where for a specific scenario, the result of each test is not deterministic and can get different values with different probabilities.

Comparison with previous work. previous work on noisy decision tree, considers limited noise models or the runtime and approximation ratio depends on the type of noise. For example in the main result of [40], the noise outcomes are binary with equal probability. The authors mention that it is possible to extend the following ways:

- to probabilities within $[\delta, 1-\delta]$, incurring an extra $1/\delta$ factor in the approximation
- to non-binary noise outcomes, incurring an extra at most m factor in the approximation

Additionally, their algorithm works by expanding the scenarios for every possible noise outcome (e.g. to 2^m for binary noise). In our work the number of noisy outcomes does not affect the number of scenarios whatsoever.

In our work, we obtain a **constant approximation** factor, that does not depend in any way on the type of the noise. Additionally, the outcomes of the noisy tests can be arbitrary, and do not affect either the approximation factor or the runtime. We only require a *separability* condition to hold ; the distributions either differ *enough* or are exactly the same. Formally, we require that for any two scenarios $s_1, s_2 \in \mathcal{S}$ and for every box i, the distributions $\mathcal{D}_{is_1}$ and $\mathcal{D}_{is_2}$ satisfy $|\mathcal{D}_{is_1} - \mathcal{D}_{is_2}| \in \mathbb{R}_{\geq \varepsilon} \cup \{0\}$, where $|\mathcal{A} - \mathcal{B}|$ is the total variation distance of distributions $\mathcal{A}$ and $\mathcal{B}$.

6.1 A DP Algorithm for noisy PB$_{\leq T}$

We move on to designing a dynamic programming algorithm to solve the PB$_{\leq T}$ problem, in the case of a mixtures of product distributions. The guarantees of our dynamic programming algorithm are given in the following theorem.

▶ **Theorem 18.** *For any $\beta > 0$, let π_{DP} and π^* be the policies produced by Algorithm* DP(β) *described by Equation* (2) *and the optimal policy respectively and* $\text{UB} = \frac{m^2}{\varepsilon^2} \log \frac{m^2 T}{c_{\min} \beta}$. *Then it holds that*

$$c(\pi_{\text{DP}}) \leq (1 + \beta) c(\pi^*).$$

and the DP *runs in time n^{UB}, where n is the number of boxes and $c_{\min}$ is the minimum cost box.*

Using the reduction described in Section 4.1.1 and the previous theorem we can get a constant-approximation algorithm for the initial PB problem given a mixture of product distributions. Observe that in the reduction, for every instance of PB$_{\leq T}$ it runs, the chosen threshold T satisfies that $T \leq (\beta + 1)c(\pi^*_T)/0.2$ where π^*_T is the optimal policy for the threshold T. The inequality holds since the algorithm for the threshold T is a $(\beta + 1)$ approximation and it covers 80% of the scenarios left (i.e. pays $0.2T$ for the rest). This is formalized in the following corollary.

▶ **Corollary 19.** *Given an instance of PB on m scenarios, and the DP algorithm described in Equation* (2)*, then using Algorithm 1 we obtain an $O(1)$-approximation algorithm for PB that runs in $n^{\tilde{O}(m^2/\varepsilon^2)}$.*

Observe that the naive DP, that keeps track of all the boxes and possible outcomes, has space exponential in the number of boxes, which can be very large. In our DP, we exploit the separability property of the distributions by distinguishing the boxes in two different types based on a given set of scenarios. Informally, the *informative* boxes help us distinguish between two scenarios, by giving us enough TV distance, while the *non-informative* always have zero TV distance. The formal definition follows.

▶ **Definition 20** (Informative and non-informative boxes)**.** *Let $S \subseteq \mathcal{S}$ be a set of scenarios. Then we call a box k* informative *if there exist $s_i, s_j \in \mathcal{S}$ such that*

$$|\mathcal{D}_{ks_i} - \mathcal{D}_{ks_j}| \geq \varepsilon.$$

We denote the set of all informative *boxes by* IB(S)*. Similarly, the boxes for which the above does not hold are called* non-informative *and the set of these boxes is denoted by* NIB(S)*.*

Recursive calls of the DP. Our dynamic program chooses at every step one of the following options:

1. open an **informative** box: this step contributes towards *eliminating* improbable scenarios. From the definition of informative boxes, every time such a box is opened, it gives TV distance at least ε between at least two scenarios, making one of them more probable

than the other. We show (Lemma 21) that it takes a finite amount of these boxes to decide, with high probability, which scenario is the one realized (i.e. eliminating all but one scenarios).

2. open a **non-informative** box: this is a greedy step; the best non-informative box to open next is the one that maximizes the probability of finding a value smaller than T. Given a set S of scenarios that are not yet eliminated, there is a unique next non-informative box which is best. We denote by $\text{NIB}^*(S)$ the function that returns this next best non-informative box. Observe that the non-informative boxes do not affect the greedy ordering of which is the next best, since they do not affect which scenarios are eliminated.

State space of the DP. the DP keeps track of the following three quantities:

1. **a list $\boldsymbol{M}$** which consists of sets of informative boxes opened and numbers of non-informative ones opened in between the sets of informative ones. Specifically, M has the following form: $M = S_1|x_1|S_2|x_2|\dots|S_L|x_L$[6] where S_i is a set of informative boxes, and $x_i \in \mathbb{N}$ is the number of non-informative boxes opened exactly after the boxes in set S_i. We also denote by $\text{IB}(M)$ the informative boxes in the list M.
   In order to update M at every recursive call, we either append a new informative box b_i opened (denoted by $M|b_i$) or, when a non-informative box is opened, we add 1 at the end, denoted by $M + 1$.
2. **a list $\boldsymbol{E}$** of m^2 tuples of integers (z_{ij}, t_{ij}), one for each pair of distinct scenarios (s_i, s_j) with $i, j \in [m]$. The number z_{ij} keeps track of the number of informative boxes between s_i and s_j that the value discovered had higher probability for scenario s_i, and the number t_{ij} is the total number of informative for scenarios s_i and s_j opened. Every time an informative box is opened, we increase the t_{ij} variables for the scenarios the box was informative and add 1 to the z_{ij} if the value discovered had higher probability in s_i. When a non-informative box is opened, the list remains the same.We denote this update by E^{++}.
3. **a list $\boldsymbol{S}$** of the scenarios not yet eliminated. Every time an informative test is performed, and the list E updated, if for some scenario s_i there exists another scenario s_j such that $t_{ij} > 1/\varepsilon^2 \log(1/\delta)$ and $|z_{ij} - \mathbb{E}[z_{ij}|s_i]| \leq \varepsilon/2$ then s_j is removed from S, otherwise s_i is removed[7]. This update is denoted by S^{++}.

Base cases. if a value below T is found, the algorithm stops. The other base case is when $|S| = 1$, which means that the scenario realized is identified, we either take the outside option T or search the boxes for a value below T, whichever is cheapest. If the scenario is identified correctly, the DP finds the expected optimal for this scenario. We later show that we make a mistake only with low probability, thus increasing the cost only by a constant factor. We denote by $\text{Nat}(\cdot,\cdot,\cdot)$ the "nature's" move, where the value in the box we chose is realized, and $\text{Sol}(\cdot,\cdot,\cdot)$ is the minimum value obtained by opening boxes. The recursive formula is shown below.

$$\text{Sol}(M,E,S) = \begin{cases} \min(T, c_{\text{NIB}^*(S)} + \text{Nat}(M{+}1, E, S)) & \text{if } |S| = 1 \\ \min\Big(T, \min\limits_{i\in \text{IB}(M)} (c_i{+}\,\text{Nat}(M|i, E, S)) & \\ \qquad\qquad , c_{\text{NIB}^*(S)} + \text{Nat}(M{+}1, E, S)\Big) & \text{else} \end{cases}$$

[6] If b_i for $i \in [n]$ are boxes, the list M looks like this: $b_3b_6b_{13}|5|b_{42}b_1|6|b_2$

[7] This is the process of elimination in the proof of Lemma 21

$$\text{Nat}(M, E, S) = \begin{cases} 0 & \text{if } v_{\text{last box opened}} \leq T \\ \text{Sol}(M, E^{++}, S^{++}) & \text{else} \end{cases} \tag{2}$$

The final solution is $\text{DP}(\beta) = \text{Sol}(\emptyset, E^0, \mathcal{S})$, where E^0 is a list of tuples of the form $(0, 0)$, and in order to update S we set $\delta = \beta c_{\min}/(m^2 T)$.

▶ **Lemma 21.** *Let $s_1, s_2 \in \mathcal{S}$ be any two scenarios. Then after opening $\frac{\log(1/\delta)}{\varepsilon^2}$ informative boxes, we can eliminate one scenario with probability at least $1 - \delta$.*

References

1 Marek Adamczyk, Maxim Sviridenko, and Justin Ward. Submodular stochastic probing on matroids. *Math. Oper. Res.*, 41(3):1022–1038, 2016. doi:10.1287/moor.2015.0766.

2 Micah Adler and Brent Heeringa. Approximating optimal binary decision trees. *Algorithmica*, 62(3-4):1112–1121, 2012. doi:10.1007/s00453-011-9510-9.

3 Yossi Azar, Iftah Gamzu, and Xiaoxin Yin. Multiple intents re-ranking. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 – June 2, 2009*, pages 669–678. ACM, 2009. doi:10.1145/1536414.1536505.

4 Nikhil Bansal, Anupam Gupta, and Ravishankar Krishnaswamy. A constant factor approximation algorithm for generalized min-sum set cover. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1539–1545, 2010. doi:10.1137/1.9781611973075.125.

5 Curtis Bechtel, Shaddin Dughmi, and Neel Patel. Delegated pandora's box. In David M. Pennock, Ilya Segal, and Sven Seuken, editors, *EC '22: The 23rd ACM Conference on Economics and Computation, Boulder, CO, USA, July 11 – 15, 2022*, pages 666–693. ACM, 2022. doi:10.1145/3490486.3538267.

6 Ben Berger, Tomer Ezra, Michal Feldman, and Federico Fusco. Pandora's problem with combinatorial cost. *CoRR*, abs/2303.01078, 2023. doi:10.48550/arXiv.2303.01078.

7 Hedyeh Beyhaghi and Linda Cai. Pandora's problem with nonobligatory inspection: Optimal structure and a PTAS. *CoRR*, abs/2212.01524, 2022. doi:10.48550/arXiv.2212.01524.

8 Hedyeh Beyhaghi and Robert Kleinberg. Pandora's problem with nonobligatory inspection. In Anna Karlin, Nicole Immorlica, and Ramesh Johari, editors, *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019*, pages 131–132. ACM, 2019. doi:10.1145/3328526.3329626.

9 Shant Boodaghians, Federico Fusco, Philip Lazos, and Stefano Leonardi. Pandora's box problem with order constraints. In Péter Biró, Jason D. Hartline, Michael Ostrovsky, and Ariel D. Procaccia, editors, *EC '20: The 21st ACM Conference on Economics and Computation, Virtual Event, Hungary, July 13-17, 2020*, pages 439–458. ACM, 2020. doi:10.1145/3391403.3399501.

10 Venkatesan T. Chakaravarthy, Vinayaka Pandit, Sambuddha Roy, Pranjal Awasthi, and Mukesh K. Mohania. Decision trees for entity identification: Approximation algorithms and hardness results. *ACM Trans. Algorithms*, 7(2):15:1–15:22, 2011. doi:10.1145/1921659.1921661.

11 Venkatesan T. Chakaravarthy, Vinayaka Pandit, Sambuddha Roy, and Yogish Sabharwal. Approximating decision trees with multiway branches. 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 210–221. Springer, 2009. doi:10.1007/978-3-642-02927-1_19.

12 Moses Charikar, Ronald Fagin, Venkatesan Guruswami, Jon M. Kleinberg, Prabhakar Raghavan, and Amit Sahai. Query strategies for priced information (extended abstract). In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 582–591, 2000. doi:10.1145/335305.335382.

13 Shuchi Chawla, Evangelia Gergatsouli, Yifeng Teng, Christos Tzamos, and Ruimin Zhang. Pandora's box with correlations: Learning and approximation. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1214–1225. IEEE, 2020. doi:10.1109/FOCS46700.2020.00116.

14 Yuxin Chen, S. Hamed Hassani, Amin Karbasi, and Andreas Krause. Sequential information maximization: When is greedy near-optimal? In *Proceedings of The 28th Conference on Learning Theory, COLT 2015, Paris, France, July 3-6, 2015*, pages 338–363, 2015. URL: http://proceedings.mlr.press/v40/Chen15b.html.

15 Yuxin Chen, Shervin Javdani, Amin Karbasi, J. Andrew Bagnell, Siddhartha S. Srinivasa, and Andreas Krause. Submodular surrogates for value of information. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 3511–3518, 2015. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9841.

16 Ferdinando Cicalese, Tobias Jacobs, Eduardo Sany Laber, and Marco Molinaro. On greedy algorithms for decision trees. In Otfried Cheong, Kyung-Yong Chwa, and Kunsoo Park, editors, *Algorithms and Computation – 21st International Symposium, ISAAC 2010, Jeju Island, Korea, December 15-17, 2010, Proceedings, Part II*, volume 6507 of *Lecture Notes in Computer Science*, pages 206–217. Springer, 2010. doi:10.1007/978-3-642-17514-5_18.

17 Sanjoy Dasgupta. Analysis of a greedy active learning strategy. In *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*, pages 337–344, 2004. URL: https://proceedings.neurips.cc/paper/2004/hash/c61fbef63df5ff317aecdc3670094472-Abstract.html.

18 Bolin Ding, Yiding Feng, Chien-Ju Ho, Wei Tang, and Haifeng Xu. Competitive information design for pandora's box. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 353–381. SIAM, 2023. doi:10.1137/1.9781611977554.ch15.

19 Laura Doval. Whether or not to open pandora's box. *J. Econ. Theory*, 175:127–158, 2018. doi:10.1016/j.jet.2018.01.005.

20 Hossein Esfandiari, Mohammad Taghi Hajiaghayi, Brendan Lucier, and Michael Mitzenmacher. Online pandora's boxes and bandits. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 – February 1, 2019*, pages 1885–1892. AAAI Press, 2019. doi:10.1609/aaai.v33i01.33011885.

21 Uriel Feige, László Lovász, and Prasad Tetali. Approximating min sum set cover. *Algorithmica*, 40(4):219–234, 2004. doi:10.1007/s00453-004-1110-5.

22 Hu Fu, Jiawei Li, and Daogao Liu. Pandora box problem with nonobligatory inspection: Hardness and improved approximation algorithms. *CoRR*, abs/2207.09545, 2022. doi:10.48550/arXiv.2207.09545.

23 M. R. Garey and Ronald L. Graham. Performance bounds on the splitting algorithm for binary testing. *Acta Informatica*, 3:347–355, 1974. doi:10.1007/BF00263588.

24 Evangelia Gergatsouli and Christos Tzamos. Weitzman's rule for pandora's box with correlations. *CoRR*, abs/2301.13534, 2023. doi:10.48550/arXiv.2301.13534.

25 J.C. Gittins and D.M. Jones. A dynamic allocation index for the sequential design of experiments. *Progress in Statistics*, pages 241–266, 1974.

26 Ashish Goel, Sudipto Guha, and Kamesh Munagala. Asking the right questions: model-driven optimization using probes. In *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, USA*, pages 203–212, 2006. doi:10.1145/1142351.1142380.

27 Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *J. Artif. Intell. Res.*, 42:427–486, 2011. doi:10.1613/jair.3278.

28 Daniel Golovin and Andreas Krause. Adaptive submodularity: A new approach to active learning and stochastic optimization. *CoRR*, abs/1003.3967, 2017. arXiv:1003.3967.

29 Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal bayesian active learning with noisy observations. In John D. Lafferty, Christopher K. I. Williams, John Shawe-Taylor, Richard S. Zemel, and Aron Culotta, editors, *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*, pages 766–774. Curran Associates, Inc., 2010. URL: https://proceedings.neurips.cc/paper/2010/hash/1e6e0a04d20f50967c64dac2d639a577-Abstract.html.

30 Andrew Guillory and Jeff A. Bilmes. Average-case active learning with costs. In Ricard Gavaldà, Gábor Lugosi, Thomas Zeugmann, and Sandra Zilles, editors, *Algorithmic Learning Theory, 20th International Conference, ALT 2009, Porto, Portugal, October 3-5, 2009. Proceedings*, volume 5809 of *Lecture Notes in Computer Science*, pages 141–155. Springer, 2009. doi:10.1007/978-3-642-04414-4_15.

31 Anupam Gupta, Haotian Jiang, Ziv Scully, and Sahil Singla. The markovian price of information. In *Integer Programming and Combinatorial Optimization – 20th International Conference, IPCO 2019, Ann Arbor, MI, USA, May 22-24, 2019, Proceedings*, pages 233–246, 2019. doi:10.1007/978-3-030-17953-3_18.

32 Anupam Gupta and Amit Kumar. Sorting and selection with structured costs. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 416–425, 2001. doi:10.1109/SFCS.2001.959916.

33 Anupam Gupta and Viswanath Nagarajan. A stochastic probing problem with applications. In *Integer Programming and Combinatorial Optimization – 16th International Conference, IPCO 2013, Valparaíso, Chile, March 18-20, 2013. Proceedings*, pages 205–216, 2013. doi:10.1007/978-3-642-36694-9_18.

34 Anupam Gupta, Viswanath Nagarajan, and R. Ravi. Approximation algorithms for optimal decision trees and adaptive TSP problems. *Math. Oper. Res.*, 42(3):876–896, 2017. doi:10.1287/moor.2016.0831.

35 Anupam Gupta, Viswanath Nagarajan, and Sahil Singla. Algorithms and adaptivity gaps for stochastic probing. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1731–1747, 2016. doi:10.1137/1.9781611974331.ch120.

36 Anupam Gupta, Viswanath Nagarajan, and Sahil Singla. Adaptivity gaps for stochastic probing: Submodular and XOS functions. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1688–1702, 2017. doi:10.1137/1.9781611974782.111.

37 Noam Hazon, Yonatan Aumann, Sarit Kraus, and David Sarne. Physical search problems with probabilistic knowledge. *Artif. Intell.*, 196:26–52, 2013. doi:10.1016/j.artint.2012.12.003.

38 Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is np-complete. *Inf. Process. Lett.*, 5(1):15–17, 1976. doi:10.1016/0020-0190(76)90095-8.

39 Sungjin Im, Viswanath Nagarajan, and Ruben van der Zwaan. Minimum latency submodular cover. *ACM Trans. Algorithms*, 13(1):13:1–13:28, 2016. doi:10.1145/2987751.

40 Su Jia, Viswanath Nagarajan, Fatemeh Navidi, and R. Ravi. Optimal decision tree with noisy outcomes. 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, December 8-14, 2019, Vancouver, BC, Canada*, pages 3298–3308, 2019. URL: https://proceedings.neurips.cc/paper/2019/hash/85f007f8c50dd25f5a45fca73cad64bd-Abstract.html.

41 Prabhanjan Kambadur, Viswanath Nagarajan, and Fatemeh Navidi. Adaptive submodular ranking. In *Integer Programming and Combinatorial Optimization – 19th International Conference, IPCO 2017, Waterloo, ON, Canada, June 26-28, 2017, Proceedings*, pages 317–329, 2017. doi:10.1007/978-3-319-59250-3_26.

42 S. Rao Kosaraju, Teresa M. Przytycka, and Ryan S. Borgstrom. On an optimal split tree problem. In Frank K. H. A. Dehne, Arvind Gupta, Jörg-Rüdiger Sack, and Roberto Tamassia, editors, *Algorithms and Data Structures, 6th International Workshop, WADS '99, Vancouver, British Columbia, Canada, August 11-14, 1999, Proceedings*, volume 1663 of *Lecture Notes in Computer Science*, pages 157–168. Springer, 1999. doi:10.1007/3-540-48447-7_17.

43 Ray Li, Percy Liang, and Stephen Mussmann. A tight analysis of greedy yields subexponential time approximation for uniform decision tree. 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 102–121. SIAM, 2020. doi:10.1137/1.9781611975994.7.

44 Zhen Liu, Srinivasan Parthasarathy, Anand Ranganathan, and Hao Yang. Near-optimal algorithms for shared filter evaluation in data stream systems. In Jason Tsong-Li Wang, editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 133–146. ACM, 2008. doi:10.1145/1376616.1376633.

45 Donald W. Loveland. Performance bounds for binary testing with arbitrary weights. *Acta Informatica*, 22(1):101–114, 1985. doi:10.1007/BF00290148.

46 Kamesh Munagala, Shivnath Babu, Rajeev Motwani, and Jennifer Widom. The pipelined set cover problem. In Thomas Eiter and Leonid Libkin, editors, *Database Theory – ICDT 2005, 10th International Conference, Edinburgh, UK, January 5-7, 2005, Proceedings*, volume 3363 of *Lecture Notes in Computer Science*, pages 83–98. Springer, 2005. doi:10.1007/978-3-540-30570-5_6.

47 Feng Nan and Venkatesh Saligrama. Comments on the proof of adaptive stochastic set cover based on adaptive submodularity and its implications for the group identification problem in "group-based active query selection for rapid diagnosis in time-critical situations". *IEEE Trans. Inf. Theory*, 63(11):7612–7614, 2017. doi:10.1109/TIT.2017.2749505.

48 Krishna R. Pattipati and Mahesh Dontamsetty. On a generalized test sequencing problem. *IEEE Trans. Syst. Man Cybern.*, 22(2):392–396, 1992. doi:10.1109/21.148415.

49 Vili Podgorelec, Peter Kokol, Bruno Stiglic, and Ivan Rozman. Decision trees: An overview and their use in medicine. *Journal of medical systems*, 26:445–63, November 2002. doi:10.1023/A:1016409317640.

50 Sahil Singla. The price of information in combinatorial optimization. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2523–2532, 2018. doi:10.1137/1.9781611975031.161.

51 Martin Skutella and David P. Williamson. A note on the generalized min-sum set cover problem. *Oper. Res. Lett.*, 39(6):433–436, 2011. doi:10.1016/j.orl.2011.08.002.

52 Martin L Weitzman. Optimal Search for the Best Alternative. *Econometrica*, 47(3):641–654, May 1979.

A Proofs from Section 4

▷ Claim 7. If there exists an $\alpha(n,m)$-approximation algorithm for PB then there exists a $\alpha(n,m)$-approximation for MSSC$_f$.

Proof of Claim 7. Let $\mathcal{I}$ be an instance of MSSC$_f$. We create an instance $\mathcal{I}'$ of PB the following way: for every set s_j of $\mathcal{I}$ that gives feedback f_{ij} when element e_i is selected, we create a scenario s_j with the same probability and whose value for box i, is either 0 if $e_i \in s_j$ or $\infty_{f_{ij}}$ otherwise, where $\infty_{f_{ij}}$ denotes an extremely large value which is different for different values of the feedback f_{ij}. Observe that any solution to the PB instance gives a solution to the MSSC$_f$ at the same cost and vice versa. ◁

▷ Claim 9. If there exists an $\alpha(n, m)$ approximation algorithm for UMSSC_f then there exists an $3\alpha(n + m, m^2)$-approximation for $\text{UPB}_{\leq T}$.

Before formally proving this claim, recall the correspondence of scenarios and boxes of PB-type problems, to elements and sets of MSSC-type problems. The idea for the reduction is to create T copies of sets for each scenario in the initial $\text{PB}_{\leq T}$ instance and one element per box, where if the price a box gives for a scenario i is $< T$ then the corresponding element belongs to all T copies of the set i. The final step is to "simulate" the outside option T, for which we we create T elements where the k'th one belongs only to the k'th copy of each set.

Proof of Claim 9. Given an instance $\mathcal{I}$ of $\text{UPB}_{\leq T}$ with outside cost box b_T, we construct the instance $\mathcal{I}'$ of UMSSC_f as follows.

Construction of the instance. For every scenario s_i in the initial instance, we create T sets denoted by s_{ik} where $k \in [T]$. Each of these sets has equal probability $p_{ik} = 1/(mT)$. We additionally create one element e^B per box B, which belongs to every set s_{ik} for all k iff $v_{Bi} < T$ in the initial instance, otherwise gives feedback v_{Bi}. In order to simulate box b_T without introducing an element with non-unit cost, we use a sequence of T *outside option* elements e^T_k where $e^T_k \in s_{ik}$ for all $i \in [m]$ i.e. element e^T_{ik} belongs to "copy k" of every set [8].

Construction of the policy. We construct policy $\pi_{\mathcal{I}}$ by ignoring any outside option elements that $\pi_{\mathcal{I}'}$ selects until $\pi_{\mathcal{I}'}$ has chosen at least $T/2$ such elements, at which point $\pi_{\mathcal{I}}$ takes the outside option box b_T. To show feasibility we need that for every scenario either b_T is chosen or some box with $v_{ij} \leq T$. If b_T is not chosen, then less than $T/2$ isolating elements were chosen, therefore in instance of UMSSC_f some sub-sets will have to be covered by another element e^B, corresponding to a box. This corresponding box however gives a value $\leq T$ in the initial $\text{UPB}_{\leq T}$ instance.

Approximation ratio. Let s_i be any scenario in $\mathcal{I}$. We distinguish between the following cases, depending on whether there are outside option tests on s_i's branch.

1. **No outside option tests** on s_i's branch: scenario s_i contributes equally in both policies, since absence of *isolating elements* implies that all copies of scenario s_i will be on the same branch (paying the same cost) in both $\pi_{\mathcal{I}'}$ and $\pi_{\mathcal{I}}$
2. **Some outside option tests** on i's branch: for this case, from Lemma 22 we have that $c(\pi_{\mathcal{I}}(s_i)) \leq 3c(\pi_{\mathcal{I}'}(s_i))$.

   Putting it all together we get

   $$c(\pi_{\mathcal{I}}) \leq 3c(\pi_{\mathcal{I}'}) \leq 2\alpha(n + m, m^2)c(\pi^*_{\mathcal{I}'}) \leq 3\alpha(n + m, m^2)c(\pi^*_{\mathcal{I}}),$$

where the second inequality follows since we are given an α approximation and the last inequality since if we are given an optimal policy for $\text{UPB}_{\leq T}$, the exact same policy is also feasible for any $\mathcal{I}'$ instance of UDT, which has cost at least $c(\pi^*_{\mathcal{I}'})$. We also used that $T \leq m$, since otherwise the initial policy would never take the outside option. ◁

▶ **Lemma 22.** *Let $\mathcal{I}$ be an instance of $\text{UPB}_{\leq T}$, and $\mathcal{I}'$ the instance of UMSSC_f constructed by the reduction of Claim 9. For a scenario s_i, if there is at least one outside option test run in $\pi_{\mathcal{I}}$, then $c(\pi_{\mathcal{I}}(s_i)) \leq 3c(\pi_{\mathcal{I}'}(s_i))$.*

[8] Observe that there are exactly T possible options for k for any set. Choosing all these elements costs T and covers all sets thus simulating b_T.

Proof. For the branch of scenario s_i, denote by M the box elements chosen before there were $T/2$ *outside option* elements, and by N the number of *outside option* elements in $\pi_{\mathcal{I}'}$. Note that the smallest cost is achieved if all the outside option elements are chosen first[9]. The copies of scenario s_i can be split into two groups; those that were isolated **before** $T/2$ *outside option* elements were chosen, and those that were isolated **after**. We distinguish between the following cases, based on the value of N.

1. $N \geq T/2$: in this case each of the copies of s_i that are isolated after pays at least $M + T/2$ for the initial box elements and the initial sequence of *outside option* elements. For the copies isolated before, we lower bound the cost by choosing all *outside option* elements first.
   The cost of all the copies in $\pi_{\mathcal{I}'}$ then is at least
   $$\begin{aligned}\sum_{j=1}^{K_i}\sum_{k=1}^{T/2}\frac{cp_\ell}{T}k + \sum_{j=1}^{K_i}\sum_{k=T/2+1}^{T}\frac{cp_\ell}{T}(T/2+M) &= cp_i\frac{\frac{T}{2}(\frac{T}{2}+1)}{2T} + cp_i\frac{\frac{T}{2}(T/2+M)}{T}\\ &\geq cp_i(3T/8 + M/2)\\ &\geq \frac{3}{8}p_i(T+M)\end{aligned}$$
   Since $N \geq T/2$, policy $\pi_{\mathcal{I}}$ will take the outside option box for s_i, immediately after choosing the M initial boxes corresponding to the box elements. So, the total contribution s_i has on the expected cost of $\pi_{\mathcal{I}}$ is at most $p_i(M+T)$ in this case. Hence, we have that s_i's contribution in $\pi_{\mathcal{I}}$ is at most $\frac{8}{3} \leq 3$ times s_i's contribution in $\pi_{\mathcal{I}'}$.
2. $N < T/2$: policy $\pi_{\mathcal{I}}$ will only select the M boxes (corresponding to *box* elements) and this was sufficient for finding a value less than T. The total contribution of s_i on $c(\pi_{\mathcal{I}})$ is exactly $p_i M$. On the other hand, since $N < T/2$ we know that at least half of the copies will pay M for all of the box elements. The cost of all the copies is at least
   $$\sum_{j=1}^{K_i}\sum_{k=N}^{T}\frac{cp_\ell}{T}M = cp_i\frac{T-N}{T}M \geq cp_i M/2,$$
   therefore, the contribution s_i has on $c(\pi_{\mathcal{I}'})$ is at least $cp_i M/2$. Hence, we have $c(\pi_{\mathcal{I}}) \leq 3c(\pi_{\mathcal{I}'})$. ◀

A.1 Proofs from subsection 4.1.1

▶ **Lemma 15.** *Given an instance $\mathcal{I}$ of PB; an α-approximation algorithm $\mathcal{A}_T$ to PB$_{\leq T}$; and any $q < 1$ and $\beta \geq 2$, suppose that the threshold T satisfies*

$$T \geq t_{q/(\beta\alpha)} + \beta\alpha \sum_{\substack{c_s \in [t_q, t_{q/(\beta\alpha)}]\\ s \in \mathcal{S}}} c_s \frac{p_s}{q}.$$

Then if $\mathcal{A}_T$ is run on a q-sub instance of $\mathcal{I}$ with threshold T, at most a total mass of $(2/\beta)q$ of the scenarios pick the outside option box T.

[9] Since the outside option tests cause some copies to be isolated and so can reduce their cost.

Proof. Consider a policy $\pi_{\mathcal{I}_q}$ which runs $\pi^*_{\mathcal{I}}$ on the instance $\mathcal{I}_q$; and for scenarios with cost $c_s \geq t_{q/(\beta\alpha)}$ aborts after spending this cost and chooses the outside option T. The cost of this policy is:

$$c(\pi^*_{\mathcal{I}_q}) \leq c(\pi_{\mathcal{I}_q}) = \frac{T + t_{q/(\beta\alpha)}}{\beta\alpha} + \sum_{\substack{c_s \in [t_q, t_{q/(10\alpha)}] \\ s \in \mathcal{S}}} c_s \frac{p_s}{q}, \tag{3}$$

By our assumption on T, this cost is at most $2T/\beta\alpha$. On the other hand since $\mathcal{A}_T$ is an α-approximation to the optimal we have that the cost of the algorithm's solution is at most

$$\alpha c(\pi^*_{\mathcal{I}_q}) \leq \frac{2T}{\beta}$$

Since the expected cost of $\mathcal{A}_T$ is at most $2T/\beta$, then using Markov's inequality, we get that $\mathbf{Pr}\left[c_s \geq T\right] \leq (2T/\beta)/T = 2/\beta$. Therefore, $\mathcal{A}_T$ covers at least $1 - 2/\beta$ mass every time. ◀

▶ **Lemma 14** (Optimal Lower Bound). *Let $\mathcal{I}$ be the instance of PB. For any $q < 1$, any $\alpha > 1$, and $\beta \geq 2$, for the optimal policy $\pi^*_{\mathcal{I}}$ for PB it that*

$$\mathrm{cost}(\pi^*_{\mathcal{I}}) \geq \sum_{i=0}^{\infty} \frac{1}{\beta\alpha} \cdot (q)^i \, t_{q^i/\beta\alpha}.$$

Proof. In every interval of the form $\mathcal{I}_i = [t_{q^i}, t_{q^i/(\beta\alpha)}]$ the optimal policy for PB covers at least $1/(\beta\alpha)$ of the probability mass that remains. Since the values belong in the interval $\mathcal{I}_i$ in phase i, it follows that the minimum possible value that the optimal policy might pay is t_{q^i}, i.e. the lower end of the interval. Summing up for all intervals, we get the lemma. ◀

A.2 Proofs from subsection 4.1.2

Algorithm 3 **Expand**: rescales and returns an instance of UPB.

```
Input: Set of scenarios $\mathcal{S}$
Scale all probabilities by $c$ such that $c\sum_{s\in\mathcal{S}} p_s = 1$
Let $p_{\min} = \min_{s\in\mathcal{S}} p_s$
$\mathcal{S}' =$ for each $s \in \mathcal{S}$ create $p_s/p_{\min}$ copies
Each copy has probability $1/|\mathcal{S}'|$
return $\mathcal{S}'$
```

▶ **Main Lemma 10.** *Given a polynomial-time $\alpha(n,m)$-approximation algorithm for UPB$_{\leq T}$, there exists a polynomial-time $O(\alpha(n,m)\log\alpha(n,m))$-approximation for PB.*

Proof. The proof in this case follows the steps of the proof of Theorem 11, and we are only highlighting the changes. The process of the reduction is the same as Algorithm 1 with the only difference that we add two extra steps; (1) we initially remove all low probability scenarios (line 3 - remove at most c fraction) and (2) we add them back after running UPB$_{\leq T}$ (line 8). The reduction process is formally shown in Algorithm 2.

Calculating the thresholds. For every phase i we choose a threshold T_i such that $T_i = \min\{T : \mathbf{Pr}\left[c_s > T\right] \leq \delta\}$ i.e. at most δ of the probability mass of the scenarios are not covered, again using binary search as in Algorithm 1. We denote by

$\text{Int}_i = [t_{(1-c)(\delta+c)^i}, t_{(1-c)(\delta+c)^i/(\beta\alpha)}]$ the relevant interval of costs at every run of the algorithm, then by Lemma 15, we know that for remaining total probability mass $(1-c)(\delta+c)^i$, any threshold which satisfies

$$T_i \geq t_{(1-c)(\delta+c)^{i-1}/\beta\alpha} + \beta\alpha \sum_{\substack{s\in\mathcal{S}\\ c_s\in\text{Int}_i}} c_s \frac{p_s}{(1-c)(\delta+c)^i}$$

also satisfies the desired covering property, i.e. at least $(1-2/\beta)(1-c)(\delta+c)$ mass of the current scenarios is covered. Therefore the threshold T_i found by our binary search satisfies

$$T_i = t_{(1-c)(\delta+c)^{i-1}/\beta\alpha} + \beta\alpha \sum_{\substack{s\in\mathcal{S}\\ c_s\in\text{Int}_i}} c_s \frac{p_s}{(1-c)(\delta+c)^i}. \tag{4}$$

Following the proof of Theorem 11, **Constructing the final policy** and **Accounting for the values** remain exactly the same as neither of them uses the fact that the scenarios are uniform.

Bounding the final cost. Using the guarantee that at the end of every phase we cover $(\delta + c)$ of the scenarios, observe that the algorithm for $\text{PB}_{\leq T}$ is run in an interval of the form $\text{Int}_i = [t_{(1-c)(\delta+c)^i}, t_{(1-c)(\delta+c)^i/(\beta\alpha)}]$. Note also that these intervals are overlapping. Bounding the cost of the final policy $\pi_\mathcal{I}$ for all intervals we get

$$\begin{aligned}
\pi_\mathcal{I} &\leq \sum_{i=0}^{\infty}(1-c)(\delta+c)^i T_i \\
&= \sum_{i=0}^{\infty}\left((1-c)(\delta+c)^i t_{(1-c)(\delta+c)^{i-1}/\beta\alpha} + \beta\alpha \sum_{\substack{s\in\mathcal{S}\\ c_s\in\text{Int}_i}} c_s p_s\right) && \text{From equation (4)} \\
&\leq 2\cdot\beta\alpha\pi^*_\mathcal{I} + \beta\alpha\sum_{i=0}^{\infty}\sum_{\substack{s\in\mathcal{S}\\ c_s\in\text{Int}_i}} c_s p_s && \text{Using Lemma 14} \\
&\leq 2\beta\alpha\log\alpha\cdot\pi^*_\mathcal{I},
\end{aligned}$$

where the inequalities follow similarly to the proof of Theorem 11. Choosing $c = \delta = 0.1$ and $\beta = 20$ we get the theorem. ◀

Stable Approximation Algorithms for Dominating Set and Independent Set

Mark de Berg
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Arpan Sadhukhan
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Frits Spieksma
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Abstract

We study Dominating Set and Independent Set for dynamic graphs in the vertex-arrival model. We say that a dynamic algorithm for one of these problems is *k-stable* when it makes at most k changes to its output independent set or dominating set upon the arrival of each vertex. We study trade-offs between the stability parameter k of the algorithm and the approximation ratio it achieves. We obtain the following results.

- We show that there is a constant $\varepsilon^* > 0$ such that any dynamic $(1+\varepsilon^*)$-approximation algorithm for Dominating Set has stability parameter $\Omega(n)$, even for bipartite graphs of maximum degree 4.
- We present algorithms with very small stability parameters for Dominating Set in the setting where the arrival degree of each vertex is upper bounded by d. In particular, we give a 1-stable $(d+1)^2$-approximation, and a 3-stable $(9d/2)$-approximation algorithm.
- We show that there is a constant $\varepsilon^* > 0$ such that any dynamic $(1+\varepsilon^*)$-approximation algorithm for Independent Set has stability parameter $\Omega(n)$, even for bipartite graphs of maximum degree 3.
- Finally, we present a 2-stable $O(d)$-approximation algorithm for Independent Set, in the setting where the average degree of the graph is upper bounded by some constant d at all times.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Mathematics of computing → Graph algorithms

Keywords and phrases Dynamic algorithms, approximation algorithms, stability, dominating set, independent set

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.27

Category APPROX

Funding MdB, AS, and FS are supported by the Dutch Research Council (NWO) through Gravitation-grant NETWORKS-024.002.003.

1 Introduction

Given a simple, undirected graph $G = (V, E)$, a *dominating set* is a subset $D \subset V$ such that each vertex in V is either a neighbor of a vertex in D, or it is in D itself. An *independent set* is a set of vertices $I \subset V$ such that no two vertices in I are neighbors. Dominating Set (the problem of finding a minimum-size dominating set) and Independent Set (the problem of finding a maximum-size independent set) are fundamental problems in algorithmic graph theory. They have numerous applications and served as prototypical problems for many algorithmic paradigms.

We are interested in Dominating Set and Independent Set in a dynamic setting, where the graph G changes over time. In particular, we consider the well-known *vertex-arrival model.* Here one starts with an empty graph $G(0)$, and new vertices arrive one by one,

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 27; pp. 27:1–27:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

along with their edges to previously arrived vertices. In this way, we obtain a sequence of graphs $G(t)$, for $t = 0, 1, 2, \ldots$. Our algorithm is then required to maintain a valid solution – a dominating set, or an independent set – at all times. In the setting we have in mind, computing a new solution is not the bottleneck, but each change to the solution (adding or deleting a vertex from the solution) is expensive. Of course we also want that the maintained solution has a good approximation ratio. To formalize this, and following De Berg et al. [12], we say that a dynamic algorithm is a *k-stable ρ-approximation algorithm* if, upon the arrival upon each vertex, the number of changes (vertex additions or removals) to the solution is at most k and the solution is a ρ-approximation at all times. In this framework, we study trade-offs between the stability parameter k and the approximation ratio that can be achieved. Ideally, we would like to have a so-called *stable approximation scheme (SAS)*: an algorithm that, for any given yet fixed parameter $\varepsilon > 0$ is k_ε-stable and gives a $(1+\varepsilon)$ approximation algorithm, where k_ε only depends on ε and not on the size of the current instance. (There is an intimate relation between local-search PTASs and SASs; we come back to this issue in Section 2.)

The vertex-arrival model is a standard model for online graph algorithms, and our stability framework is closely related to online algorithms with bounded recourse. However, there are two important differences. First, computation time is free in our framework – for instance, the algorithm may decide to compute an optimal solution in exponential time upon each insertion – while in online algorithms with bounded recourse the update time is typically taken into account. Thus we can fully focus on the the trade-off between stability and approximation ratio. Secondly, we consider the approximation ratio of the solution, while in online algorithm one typically considers the competitive ratio. Thus we compare the quality of our solution at time t to the *static optimum*, which is simply the optimum for the graph $G(t)$. A competitive analysis, one the other hand, compares the quality of the solution at time t to the *offline optimum*: the best solution for $G(t)$ that can be computed by a dynamic algorithm that knows the sequence $G(0), \ldots, G(t)$ in advance but must still process the sequence with bounded recourse. See also the discussion in the paper by Boyar et al. [6]. Thus approximation ratio is a much stronger notion that competitive ratio. As a case in point, consider Dominating Set, and suppose that n singleton vertices arrive, followed by a single vertex with edges to all previous ones. Then the static optimum for the final graph is 1, while the offline optimum with bounded recourse is $\Omega(n)$.

Related work. We now review some of the most relevant existing literature on the online version of our problems. (Borodin and El-Yaniv [5] give a general introduction to online computation.) The classical online model, where a vertex that has been added to the dominating set or to the independent set, can never be removed from it – that is, the no-recourse setting – is e.g. considered by King and Tzeng [18] and Lipton and Tomkins [20]. They show that already for the special case of interval graphs no online algorithm has constant competitive ratio; see also De et al. [9], who study these two problems for geometric intersection graphs. For Dominating Set in the vertex-arrival model, Boyar et al. [6] give online algorithms with bounded competitive ratio for trees, bipartite graphs, bounded degree graphs, and planar graphs. They actually analyze both the competitive ratio and the approximation ratio (which, as discussed earlier, can be quite different). A crucial difference between the work of Boyar et al. [6] and ours is that they do not allow recourse: in their setting, once a vertex is added to the dominating set, it cannot be removed.

To better understand online algorithmic behavior, various ways to relax classical online models have been studied. In particular, for Independent Set, among others, Halldórsson et al. [16] consider the option for the online algorithm to maintain multiple solutions. Göbel et al. [13] analyze a stochastic setting where, among other variants, a randomly generated graph is presented in adversarial order (the prophet inequality model), and one where an adversarial graph is presented in random order. In these settings they find constant competitive algorithms for Independent Set on interval graphs, and more generally for graphs with bounded so-called inductive independence number.

One other key relaxation of the online model is to allow recourse. Having recourse can be seen as relaxing the irrevocability assumption in classical online problems, and allows to assess the impact of this assumption on the competitive ratio, see Boyar et al. [7]. The notion of (bounded) recourse has been investigated for a large set of problems. Without aiming for completeness, we mention Angelopoulos et al. [1] and Gupta et al. [15] who deal with online matching and matching assignments, Gupta et al. [14] who investigate the set cover problem, and Berndt et al. [4] who deal with online bin covering with bounded migration, and Berndt et al. [3] who propose a general framework for dynamic packing problems. For these problems, it is shown what competitive or approximation ratios can be achieved when allowing a certain amount of recourse (or migration). Notice that, in many cases, an amortized interpretation of recourse is used; then the average number of changes to a solution is bounded (instead of the maximum, as for k-stable algorithms defined above). For instance, Lsiu and Charington-Toole [21] show that, for independent set, there is an interesting trade-off between the competitive ratio and the amortized recourse cost: for any $t > 1$, they provide a t-competitive algorithm for independent set using $t - 1$ recourse cost. Their results however do not apply to our notion of stability.

Our contribution. We obtain the following results.

- In Section 2, we show that the existence of a local-search PTAS for the static version of a graph problem, implies, under certain conditions, the existence of a SAS for the problem in the vertex-arrival model (whereas the converse need not be true). This implies that for graphs with strongly sublinear separators, a SAS exists for Independent Set and for Dominating Set when the arrival degrees – that is, the degrees of the vertices upon arrival – are bounded by a constant.
- In Section 3, we consider Dominating Set in the vertex-arrival model. Let d denote the maximum arrival degree. We show (i) there does not exist a SAS even for bipartite graphs of maximum degree 4, (ii) there is a 1-stable $(d+1)^2$-approximation algorithm, and (iii) there is a 3-stable $\frac{9d}{2}$-approximation algorithm.
- In Section 4, we consider Independent Set in the vertex-arrival model. We show that there does not exist a SAS even for bipartite graphs of maximum degree 3. Further, we give a 2-stable $O(d)$-approximation algorithm for the case where the average degree of $G(t)$ is bounded by d at all times.

2 Stable Approximation Schemes versus PTAS by local search

In this section we discuss the relation between Stable Approximation Schemes (SASs) and Polynomial-Time Approximation Schemes (PTASs). Using known results on local-search PTASs, we then obtain SASs for Independent Set for Dominating Set on certain graph classes. While the results in this section are simple, they set the stage for our main results in the next sections.

The goals of a SAS and a PTAS are the same: both aim to achieve an approximation ratio $(1+\varepsilon)$, for any given $\varepsilon > 0$. A SAS, however, works in a dynamic setting with the requirement that k_ε, the number of changes per update, is a constant for fixed ε, while a

PTAS works in a static setting with the condition that the running time is polynomial for fixed ε. Hence, there are problems that admit a PTAS (or are even polynomial-time solvable) but no SAS [12].

One may think that the converse should be true: if a dynamic problem admits a SAS, then its static version admits a PTAS. Indeed, we can insert the input elements one by one and let the SAS maintain a $(1+\varepsilon)$-approximation, by performing at most k_ε changes per update. For a SAS, there is no restriction on the time needed to update the solution, but it seems we can simply try all possible combinations of at most k_ε changes in, say, $n^{O(k_\varepsilon)}$ time. This need not work, however, since there could be many ways to update the solution using at most k_ε changes. Even though we can find all possible combinations in polynomial time, we may not be able to decide which combination is the right one: the update giving the best solution at this moment may get us stuck in the long run. The SAS can avoid this by spending exponential time to decide what the right update is. Thus the fact that a problem admits a SAS does not imply that it admits a PTAS.

Notwithstanding the above, there is a close connection between SASs and PTASs and, in particular between SASs and so-called local-search PTASs: under certain conditions, the existence of a local-search PTAS implies the existence of a SAS. For simplicity we will describe this for graph problems, but the technique may be applied to other problems.

Let $G=(V,E)$ be a graph and suppose we wish to select a minimum-size (or maximum-size) subset $S \subset V$ satisfying a certain property. Problems of this type include Dominating Set, Independent Set, Vertex Cover, Feedback Vertex Set, and more. A local-search PTAS for such a graph minimization problem starts with an arbitrary feasible solution S – the whole vertex set V, say – and then it tries to repeatedly decrease the size of S by replacing a subset $S_{\text{old}} \subset S$ by a subset $S_{\text{new}} \subset V \setminus S$ such that $|S_{\text{new}}| = |S_{\text{old}}|-1$ and $(S \setminus S_{\text{old}}) \cup S_{\text{new}}$ is still feasible. (For a maximization problem we require $|S_{\text{new}}| = |S_{\text{old}}|+1$.) This continues until no such replacement can be found.[1] A key step in the analysis of a local-search PTAS is to show the following, where n is the number of vertices.

- **Local-Search Property.** If S is a feasible solution that is not a $(1+\varepsilon)$-approximation then there are subsets $S_{\text{old}}, S_{\text{new}}$ as above with $|S_{\text{old}}| \leqslant f_\varepsilon$, for some f_ε depending only on ε and not on n.

This condition indeed gives a PTAS, because we can simply try all possible pairs $S_{\text{old}}, S_{\text{new}}$, of which there are $O(n^{2f_\varepsilon})$.

Now consider a problem that has the Local-Search Property in the vertex-arrival model, possibly with some extra constraint (for example, on the arrival degrees of the vertices). Let $G(t)$ denote the graph after the arrival of the t-th vertex, and let $\textsc{opt}(t)$ denote the size of an optimal solution for for $G(t)$. We can obtain a SAS if the problem under consideration has the following properties.

- **Continuity Property.** We say that the dynamic problem (in the vertex-arrival model, possibly with extra constraints) is *d-continuous* if $|\textsc{opt}(t+1) - \textsc{opt}(t)| \leqslant d$. In other words, the size of an optimal solution should not change by more than d when a new vertex arrives. Note that the solution itself may change completely; we only require its size not to change by more than d.
- **Feasibility Property.** For maximization problems we require that any feasible solution for $G(t-1)$ is also a feasible solution for $G(t)$, and for minimization problems we require that any feasible solution for $G(t-1)$ can be turned into a feasible solution for $G(t)$ by

[1] See the paper by Antunes etal [2] for a nice exposition on problems solved using local-search PTAS.

adding the arriving vertex to the solution. (This condition can be relaxed to saying that we can repair feasibility by $O(1)$ modifications to the current solution, but for concreteness we stick to the simpler formulation above.)

Note that Independent Set, Vertex Cover, and Feedback Vertex Set are 1-continuous, and that Dominating Set is $(d-1)$-continuous when the arrival degree of the vertices is bounded by $d \geqslant 2$. Moreover, these problems all have the Feasibility Property.

For problems that have the Local-Search Property as well as the Continuity and Feasibility Properties, it is easy to give a SAS. Hence, non-existence of SAS for a problem with Continuity and Feasibility directly implies non-existence of local-search PTAS.[2] We give the pseudocode for minimization problems, but for maximization problems a similar approach works.

Algorithm 1 SAS-for-Continuous-Problems(v).

```
▷ $v$ is the vertex arriving at time $t$
$S_{\text{alg}} \leftarrow S_{\text{alg}}(t-1) \cup \{v\}$        ▷ $S_{\text{alg}}$ is feasible for $G(t)$ by the feasibility condition
while $S_{\text{alg}}$ is not a $(1+\varepsilon)$-approximation do
    Find sets $S_{\text{old}} \subset S_{\text{alg}}$ and $S_{\text{new}} \subset V(t) \setminus S_{\text{alg}}$ with $|S_{\text{old}}| \leqslant f_\varepsilon$ and $|S_{\text{new}}| = |S_{\text{old}}| - 1$
    such that $(S_{\text{alg}} \setminus S_{\text{old}}) \cup S_{\text{new}}$ is a valid solution, and set $S_{\text{alg}} \leftarrow (S_{\text{alg}} \setminus S_{\text{old}}) \cup S_{\text{new}}$.
$S_{\text{alg}}(t) \leftarrow S_{\text{alg}}$
```

▶ **Theorem 1.** *Any graph problem that has the Continuity Property, the Feasibility Property and the Local-Search Property admits a SAS in the vertex arrival model, with stability parameter $(d+1)\cdot(2f_\varepsilon - 1)+1$ for minimization problems and $d \cdot (2f_\varepsilon + 1)$ for maximization problems.*

Proof. First consider a minimization problem. By the Feasibility Property and the working of the algorithm, the solution that SAS-for-Continuous-Problems computes upon the arrival of a new vertex is feasible. Moreover, it must end with a $(1+\varepsilon)$-approximation because of the Local-Search Property. Before the while-loop we add v to S_{alg}, and in each iteration of the while-loop, at most f_ε vertices are deleted from S_{alg} and at most $f_\varepsilon - 1$ vertices are added. We claim that the number of iterations is at most $\lceil(1+\varepsilon)d\rceil + 1$. Indeed, before the arrival of v we have $|S_{\text{alg}}(t-1)| \leqslant (1+\varepsilon)\cdot|\text{OPT}(t-1)|$, and we have $|\text{OPT}(t-1) - \text{OPT}(t)| \leqslant d$ by the Local-Search Property. Hence, after $\lceil(1+\varepsilon)d\rceil + 1$ iterations we have

$$\begin{aligned} |S_{\text{alg}}| &= |S_{\text{alg}}(t-1)| + 1 - (\lceil(1+\varepsilon)d\rceil + 1) \\ &\leqslant (1+\varepsilon)\cdot|\text{OPT}(t-1)| - (1+\varepsilon)d \\ &\leqslant (1+\varepsilon)\cdot(\text{OPT}(t) + d) - (1+\varepsilon)d \\ &\leqslant (1+\varepsilon)\cdot\text{OPT}(t). \end{aligned}$$

For a maximization problem the proof is similar. The differences are that we do not add an extra vertex to S_{alg} in step 2, that the number of changes per iteration is at most $2f_\varepsilon + 1$, and that the number of iterations is at most d. (We do not get the factor $(1+\varepsilon)$ because if OPT increases, then the error that we can make, which is $\varepsilon \cdot \text{OPT}$, also increases.) ◀

This general result allows us to obtain a SAS for a variety of problems, for graph classes for which a local-search PTAS is known.

[2] Dominating Set and Independent Set, even with maximum degree bounded by 3, do not admit a PTAS assuming P ≠ NP [8, 22]. In sections 3.1 and 4.1, by proving the non-existence of a SAS for Dominating Set with maximum degree bounded by 4 and Independent Set with maximum degree bounded by 3, we thus show non-existence of local-search PTAS, independent of the assumption of P ≠ NP.

Recall that a balanced *separator* of a graph $G = (V, E)$ with n vertices is a subset $S \subset V$ such that $V \setminus S$ can be partitioned into subsets A and B with $|A| \leqslant 2n/3$ and $|B| \leqslant 2n/3$ and no edges between A and B. We say that a graph class[3] $\mathcal{G}$ has *strongly sublinear separators*, if any graph $G \in \mathcal{G}$ has a balanced separator of size $O(n^\delta)$, for some fixed constant $\delta < 1$. Planar graphs, for instance, have separators of size $O(\sqrt{n})$ [19]. A recent generalization of separators are so-called *clique-based separators*, which are separators that consist of cliques and whose size is measured in terms of the number of cliques [10]. (Actually, the cost of a separator S that is the union of cliques $C_1, \ldots, C_k$ is defined as $\sum_{i=1}^{k} \log(|C_i| + 1)$, but this refined measure is not needed here.) Disk graphs (which do not have normal separators of sublinear size) have a clique-based separator of size $O(\sqrt{n})$, for instance, and pseudo-disk graphs have a clique-based separator of size $O(n^{2/3})$ [11]. For graph classes with strongly sublinear separators there are local-search PTASs for several problems. Combining that with the technique above gives the following result.

▶ **Corollary 2.** *The following problems admit a SAS in the vertex-arrival model.*

(i) Independent Set *on graph classes with sublinear clique-based separators.*

(ii) Dominating Set *on graph class with sublinear separators, when the arrival degree of each vertex bounded by some fixed constant d.*

Proof. As noted earlier, Independent Set is 1-continuous and Dominating Set is $(d-1)$-continuous. Moreover, these problems have the Feasibility Property. It remains to check the Local-Search Property.

(i) Any graph class with a separator of size $O(n^\delta)$ has the Local-Search Property for Independent Set; see the paper by Her-Peled and Quanrud [17]. (In that paper they show the Local-Search Property for graphs of polynomial expansion – see Corollary 26 and Theorem 3.4 – and graphs of polynomial expansion have sublinear separators.) Theorem 1 thus implies the result for such graph classes. To extend this to clique-based separators, we note that (for Independent Set) we only need the Local-Search Property for graphs that are the union of two independent sets, namely the independent set S and an optimal independent set S_{opt}. Such graphs are bipartite, so the largest clique has size two. Hence, the existence of a clique-based separator of size $O(n^\delta)$ immediately implies the existence of a normal separator of size $O(n^\delta)$.

(ii) For Dominating Set on graphs with polynomial expansion (hence, on graphs with sublinear separators) the Local-Search Property holds [17, Theorem 3.15]. Theorem 1 thus implies an $O(d \cdot f_\varepsilon)$-stable $(1+\varepsilon)$-approximation algorithm, for some constant f_ε depending only on ε. Hence, if d is a fixed constant , we obtain a SAS. ◀

3 Dominating Set

In this section we study stable approximation algorithms for Dominating Set in the vertex arrival model. We first show that the problem does not admit a SAS, even when the maximum degree of the graph is bounded by 4. After that we will describe two algorithms that achieve constant approximation ratio with constant stability, in the setting where each vertex arrives with constant degree.

Let $G = (V, E)$ be a graph. For a subset $S \subset V$, we denote the open neighborhood of a subset $W \subset V$ in G by $N_G(W)$, so $N_G(W) := \{v \in V \setminus W : \text{there is a } w \in W \text{ with } (v, w) \in E\}$. The closed neighborhood $N_G[W]$ is defined as $N_G(W) \cup W$. When the graph G is clear from the context, we may omit the subscript G and simply write $N(W)$ and $N[W]$.

[3] We only consider hereditary graph classes, that is, graph classes $\mathcal{G}$ such that any induced subgraph of a graph in $\mathcal{G}$ is also in $\mathcal{G}$.

3.1 No SAS for graphs of maximum degree 4

Our lower-bound construction showing that Dominating Set does not admit a SAS – in fact, our construction will show a much stronger result, namely that there is a fixed constant $\varepsilon^* > 0$ such that any stable $(1+\varepsilon^*)$-approximation algorithm must have stability $\Omega(n)$ – is based on a certain type of expander graphs, as given by the following proposition. Note that L (the left part of the bipartition of the vertex set) is larger by a constant fraction than R (the right part of the bipartition), while the expansion property goes from L to R. The proof of the following proposition was communicated to us by Noga Alon.

▶ **Proposition 3.** *For any $\mu > 0$ and any n that is sufficiently large, there are constants $0 < \varepsilon, \delta < 1$ such that there is a bipartite graph $G_{\text{exp}}(L \cup R, E)$ with the following properties:*

- *$|L| = (1+\varepsilon)n$ and $|R| = n$.*
- *The degree of every vertex in G is at most 3.*
- *For any $S \subset L$ with $|S| \leqslant \delta n$ we have $|N(S)| \geqslant (2-2\mu)|S|$*

Proof. Let $\mu > 0$ and let t be an integer so that $t > 1/\mu$. Let $\varepsilon \leqslant 1/3^{2t+1}$ be a fixed positive number. Let $H = (A \cup B, E_H)$ be a 3-regular bipartite graph with vertex classes A and B, each of size $(1+\varepsilon)n$, in which for every subset $S \subset B$ of size at most δn we have $|N(S)| \geqslant (2-\mu)|S|$, for some fixed real number $\delta = \delta(\mu) > 0$. (It is known that random cubic bipartite graphs have this property with high probability.) Now pick a set $T \subset A$ of εn vertices so that the distance between any pair of them is larger that $2t$. Such a set exists, as one can choose its members one by one, making sure to avoid the balls of radius $2t$ around the already chosen vertices. This gives a set T of the desired size since $\varepsilon 3^{2t+1} < 1$. We define G_{exp} to be the induced subgraph of H on the classes of vertices $R = A \setminus T$ and $L = B$. It remains to show that G_{exp} has the desired properties.

We have $|L| = (1+\varepsilon)n$ and $|R| = n$ by construction, and the maximum degree of G_{exp} is clearly at most 3. Now let S be a set of at most δn vertices in L. We have to show that $N_{G_{\text{exp}}}(S)$, its neighbor set in G_{exp}, has size at least $(2-2\mu)|S|$. We can assume without loss of generality that $S \cup N_{G_{\text{exp}}}(S)$ is connected in G_{exp}, since we can apply the bound to each connected component separately and just add the inequalities. Note that this assumption implies that $S \cup N_H(S)$ is also connected in H. Observe that the neighborhood $N_H(S)$ of S in the original graph H, which is contained in B, is of size at least $(2-\mu)|S|$, by the property of H. If the set T of deleted vertices has at most $\mu|S|$ members in $N_H(S)$ then the desired inequality holds and we are done. Otherwise T has more than $\mu|S|$ vertices that belong to $N_H(S)$. But the distance between any two such vertices in H is larger than $2t$ (and so the balls of radius t around them are disjoint), and since $S \cup N_H(S)$ is connected this would imply $|S| > t|T \cap N_H(S)| > t\mu|S| > |S|$, which is a contradiction. ◀

Now consider Dominating Set for a dynamic graph in the vertex-arrival model. Let $G(t)$ denote the graph at time t, that is, after the first t insertions. Let $\varepsilon^* > 0$ be such that $\varepsilon^* < \min\left(\frac{\varepsilon}{2+\varepsilon}, \frac{0.49\delta}{2(1+\varepsilon)}\right)$, where ε and δ are the constants in the expander construction of Proposition 3. Consider a dynamic algorithm ALG for Dominating Set such that $|D_{\text{alg}}(t)| \leqslant (1+\varepsilon^*) \cdot \text{OPT}(t)$ at any time t, where $D_{\text{alg}}(t)$ is the output dominating set of ALG at time t and $\text{OPT}(t)$ is the minimum size of a dominating set for $G(t)$. Let $f_{\varepsilon^*}(n)$ denote the stability of ALG, that is, the maximum number of changes it performs on D_{alg} when a new vertex arrives, where n is the number of vertices before the arrival.

We now give a construction showing that, for arbitrarily large n, there is a sequence of n arrivals that requires $f_{\varepsilon^*}(n) \geqslant \frac{1}{6(7+6\varepsilon)}\lfloor \delta n \rfloor$. To this end, choose N large enough such that the bipartite expander graph $G_{\text{exp}} = (L \cup R, E)$ from Proposition 3 exists for $\mu = 0.005$ and

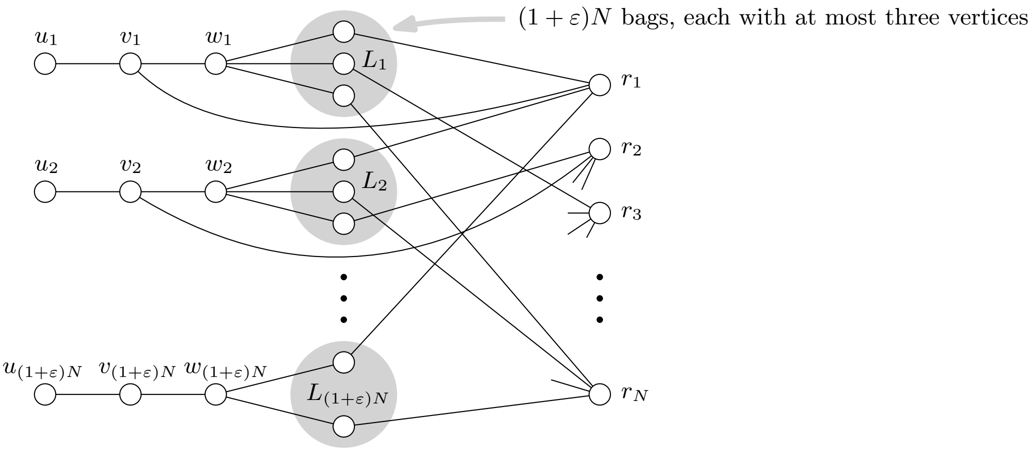


Figure 1 The lower-bound construction for Dominating Set.

$|R| = N$. Label the vertices in L as $\ell_1, \ldots, \ell_{|L|}$ and the vertices in R as $r_1, \ldots, r_{|R|}$. Our construction uses five layers of vertices, arriving one by one, as described next and illustrated in Fig. 1.

- *Layer 1:* The first layer consists of vertices $u_1, \ldots, u_{(1+\varepsilon)N}$, each arriving as a singleton.
- *Layer 2:* The second layer consists of vertices $v_1, \ldots, v_{(1+\varepsilon)N}$, where each v_i has an edge to vertex u_i from the first layer.
- *Layer 2:* The third layer consists of vertices $w_1, \ldots, w_{(1+\varepsilon)N}$, where each w_i has an edge to v_i from the second layer.
- *Layer 4:* Let $G_{\text{exp}} = (L \cup R, E)$ be the expander from Proposition 3. The fourth layer consists of $|L| = (1+\varepsilon)N$ bags, each with at most three vertices. More precisely, if $\deg(\ell_i)$ is the degree of vertex $\ell_i \in L$ in the expander G_{exp}, then bag L_i has $\deg(\ell_i)$ vertices. Each vertex in L_i has an edge to vertex w_i from the third layer.
- *Layer 5:* Finally, the fifth layer arrives. Each vertex in this layer corresponds to a vertex r_i from the bipartite expander G_{exp} and, with a slight abuse of notation, we will also denote it by r_i. If, in $G\exp$, r_i has an edge to some vertex ℓ_j in G_{exp}, then the corresponding vertex r_i in our construction will have an edge to some vertex in the bag L_j. Clearly, we can do this in such a way that each vertex in any of the bags L_i has an edge to exactly one vertex r_i. In addition to the edges to (vertices in) the bags, each vertex r_i also has an edge to the vertex v_i from the second layer.

Let t_1 be the time at which the last vertex of $L_{(1+\varepsilon)N}$ was inserted, and let t_2 be the time at which r_N was inserted. Let $G(t)$ denote the graph induced by all vertices inserted up to time t. Thus $G(t_1)$ consists of the layers 1–4, and $G(t_2)$ consists of layers 1–5.

▶ **Observation 4.** *For any t with $t_1 \leqslant t \leqslant t_2$ we have* $\text{OPT}(t) \leqslant (2+2\varepsilon)N$. *Moreover,* $\text{OPT}(t_2) \leqslant (2+\varepsilon)N$.

Proof. For any $t_1 \leqslant t \leqslant t_2$, the set $D_1 := \{v_1, \ldots, v_{(1+\varepsilon)N}\} \cup \{w_1, \ldots, w_{(1+\varepsilon)N}\}$ forms a dominating set for $G(t)$. Moreover, $D_1 := \{v_1, \ldots, v_{(1+\varepsilon)N}\} \cup \{r_1, \ldots, r_N\}$ forms a dominating set for $G(t_2)$. ◀

We call a bag L_i *fully dominated* by a set D of vertices if each vertex in L_i is dominated by some vertex in D. Observation 4 states that $\text{OPT}(t_2)$ is significantly smaller than $\text{OPT}(t_1)$, which is because the vertices in R can fully dominate all bags. This means that $D_{\text{alg}}(t_2)$ must contain most vertices of R, in order to achieve the desired approximation. Adding only a few vertices from R will be too expensive, however, since fully dominating a small number of bags will be expensive, because of the expander property of G_{exp}. Hence, if the stability parameter $f_{\varepsilon^*}(n)$ is small, then ALG cannot maintain the desired approximation ratio. Next we make this proof idea precise.

▶ **Lemma 5.** *Let $D_{\text{alg}}(t_2)$ denote the output dominating set for $G(t_2)$. Suppose $D_{\text{alg}}(t_2) \cap R$ fully dominates at most δN bags L_i. Then $|D_{\text{alg}}(t_2)| > (1+\varepsilon^*) \cdot \text{OPT}(t_2)$.*

Proof. Let $m < \delta N$ denote the number of bags fully dominated by $D_{\text{alg}}(t_2) \cap R$. Consider a bag L_i that is not fully dominated by $D_{\text{alg}}(t_2) \cap R$. Then $D_{\text{alg}}(t_2)$ must contain the vertex w_i or at least one vertex from the bag L_i. Hence, the number of vertices in $D_{\text{alg}}(t_2)$ from the third and fourth layer is at least $(1+\varepsilon)N - m$. Moreover, $D_{\text{alg}}(t_2)$ must have at least $(1+\varepsilon)N$ vertices from the first and second layer, to dominate all vertices from the first layer. Observe that in order to fully dominate a bag L_i by vertices in R, we need all vertices in R with an edge to some vertex of L_i. So if $D_{\text{alg}}(t_2) \cap R$ fully dominates $m \leqslant \delta N$ bags, then $|D_{\text{alg}}(t_2) \cap R| \geqslant 1.99m$ by the properties of the expander graph G_{exp} in Proposition 3. Hence,

$$|D_{\text{alg}}(t_2)| > (1+\varepsilon)N - m + (1+\varepsilon)N + 1.99m \geqslant (2+2\varepsilon)N.$$

Observation 4 thus implies that $\frac{|D_{\text{alg}}(t_2)|}{\text{OPT}(t_2)} \geqslant \frac{2+2\varepsilon}{2+\varepsilon} > 1+\varepsilon^*$. ◀

Lemma 5 means that, in order to achieve approximation ratio $1+\varepsilon^*$, the set $D_{\text{alg}}(t_2) \cap R$ must fully dominate more than δN bags. Next we show that this cannot be done when the stability parameter $f_{\varepsilon^*}(n)$ is $o(n)$.

▶ **Lemma 6.** *Let t^* be the first time when $D_{\text{alg}}(t^*) \cap R$ fully dominates at least δN bags. If $f_{\varepsilon^*}(n) < \frac{1}{6(7+6\varepsilon)}\lfloor \delta n \rfloor$ then $|D_{\text{alg}}(t^*)| > (1+\varepsilon^*) \cdot \text{OPT}(t^*)$.*

Proof. Let n_t denote the number of vertices of the graph $G(t)$, and observe that $n_{t^*} \leqslant (7+6\varepsilon)N$. Hence, $f_{\varepsilon^*}(n_{t^*}) \leqslant \frac{1}{6}\delta N$. By definition of t^*, we know that just before time t^* the set $D_{\text{alg}} \cap R$ fully dominates less than δN bags. Because ALG is $f_{\varepsilon^*}(n)$-stable, the number of vertices from R added to D_{alg} at time t^* is at most $f_{\varepsilon^*}(n_{t^*})$. Since these new vertices have degree at most three, they can complete the full domination of at most $3f_{\varepsilon^*}(n_{t^*})$ bags. Thus,

$$\big(\text{ number of bags fully dominated by } D_{\text{alg}}(t^*) \cap R\ \big) \quad < \quad \delta N + 3f_{\varepsilon^*}(n_{t^*}) \quad \leqslant \quad \frac{3}{2}\delta N.$$

Let L_i be a bag that is not fully dominated by $D_{\text{alg}}(t^*) \cap R$. Since $D_{\text{alg}}(t^*)$ is a dominating set, it must then contain the vertex w_i or at least one vertex from L_i. Hence, the number of vertices in $D_{\text{alg}}(t^*)$ from layers 3 and 4 is more than $(1+\varepsilon)N - \frac{3}{2}\delta N$. In addition, $D_{\text{alg}}(t^*)$ must have at least $(1+\varepsilon)N$ vertices from layers 1 and 2. Finally, in order to fully dominate a bag L_i by vertices in R, we need all the vertices in R that have an edge to some vertex of L_i. In other words, $D_{\text{alg}}(t^*) \cap R$ must contain all neighbors of the fully dominated bags. Since $D_{\text{alg}}(t^*) \cap R$ dominates at least δN bags, we know that $|D_{\text{alg}}(t^*) \cap R| \geqslant 1.99 \cdot \delta N$, by the properties of the expander graph in Proposition 3. Hence,

$$\begin{aligned} |D_{\text{alg}}(t^*)| &> (1+\varepsilon)N - \frac{3}{2}\delta N + (1+\varepsilon)N + 1.99 \cdot \delta N \\ &\geqslant \left(1 + \frac{0.49\delta}{2+2\varepsilon}\right)(2+2\varepsilon)N \\ &\geqslant (1+\varepsilon^*) \cdot \text{OPT}(t^*). \end{aligned}$$ ◀

By Lemmas 5 and 6 we obtain the following result.

▶ **Theorem 7.** *There is a constant $\varepsilon^* > 0$ such that any dynamic $(1+\varepsilon^*)$-approximation algorithm for* Dominating Set *in the vertex arrival model, must have stability parameter $\Omega(n)$, even when the maximum degree of any of the graphs $G(t)$ is bounded by 4.*

3.2 Constant-stability algorithms when the arrival degrees are bounded

In the previous section we saw that there is no SAS for Dominating Set, even when the maximum degree is bounded by 4. In this section we present stable algorithms whose approximation ratio depends on the arrival degree of the vertices. More precisely, we give a simple 1-stable algorithm with approximation ratio $(d+1)^2$ and a more complicated 3-stable algorithm with approximation ratio $9d/2$, where d is the maximum degree of any vertex upon arrival. Note we only restrict the degree upon arrival: the degree of a vertex may further increase due to the arrival of new vertices. This implies that deletions cannot be handled by a stable algorithm with bounded approximation ratio: even with arrival degree 1 we may create a star graph of arbitrarily large size, and deleting the center of the star cannot be handled in a stable manner without compromising the approximation ratio.

A 1-stable $(d+1)^2$-approximation algorithm. Recall that $G(t)$ denotes the graph after the arrival of the t-th vertex. We can turn $G(t)$ into a directed graph $\vec{G}(t)$, by directing each edge towards the older of its two incident vertices. In other words, when a new vertex arrives then its incident edges are directed away from it.

Let $N^+[v] := \{v\} \cup \{\text{out-neighbors of } v \text{ in } \vec{G}(t)\}$, where t is such that v is inserted at time t. In other words, $N^+[v]$ contains v itself plus the neighbors of v immediately after its arrival. Let $\text{OPT}_{\text{out}}(t)$ denote the minimum size of a dominating set in $\vec{G}(t)$ under the condition that every vertex v should be dominated by a vertex in $N^+[v]$. We call such a dominating set a *directed dominating set*. Note that a directed dominating set for $\vec{G}(t)$ is a dominating set in $G(t)$ as well. The following lemma states that $\text{OPT}_{\text{out}}(t)$ is not much larger than $\text{OPT}(t)$.

▶ **Lemma 8.** *At any time t we have $\text{OPT}_{\text{out}}(t) \leqslant (d+1) \cdot \text{OPT}(t)$.*

Proof. Let $D_{\text{opt}}(t)$ be a minimum dominating set for $G(t)$. Let $D := \bigcup_{v \in D_{\text{opt}}(t)} N^+[v]$. Observe that D is a directed dominating set for $\vec{G}(t)$. Since every vertex arrives with degree at most d, we have $|N^+[v]| = d+1$. The result follows. ◀

We call two vertices u, v *unrelated* if $N^+[u] \cap N^+[v] = \emptyset$, otherwise u, v are *related*. The following lemma follows immediately from the definition of $\text{OPT}_{\text{out}}(t)$.

▶ **Lemma 9.** *Let $U(t)$ be a set of pairwise unrelated vertices in $\vec{G}(t)$. Then $\text{OPT}_{\text{out}}(t) \geqslant |U(t)|$.*

Our algorithm will maintain a directed dominating set $D_{\text{alg}}(t)$ for $\vec{G}(t)$ and a set $U(t)$ of pairwise unrelated vert. Since the initial graph is empty, we initialize $D_{\text{alg}}(0) := \emptyset$ and $U(0) := \emptyset$. When a new vertex v arrives at time t, we proceed as follows.

Algorithm 2 Directed-DomSet(v).

```
▷ $v$ is the vertex arriving at time $t$
if $N^+[v] \cap D_{\text{alg}}(t-1) \neq \emptyset$ then                ▷ $v$ is already dominated
    Set $D_{\text{alg}}(t) \leftarrow D_{\text{alg}}(t-1)$ and $U(t) \leftarrow U(t-1)$
else
    if $v$ is unrelated to all vertices $u \in U(t-1)$ then
        Set $U(t) \leftarrow U(t-1) \cup \{v\}$ and $D_{\text{alg}}(t) \leftarrow D_{\text{alg}}(t-1) \cup \{v\}$
    else
        Let $u$ be a vertex related to $v$, that is, with $N^+[u] \cap N^+[v] \neq \emptyset$.
        Pick an arbitrary vertex $w \in N^+[u] \cap N^+[v]$.
        Set $D_{\text{alg}}(t) \leftarrow D_{\text{alg}}(t-1) \cup \{w\}$ and $U(t) \leftarrow U(t-1)$.
```

This leads to the following theorem.

▶ **Theorem 10.** *There is a* 1*-stable* $(d+1)^2$*-approximation algorithm for* Dominating Set *in the vertex-arrival model, where* d *is the maximum arrival degree of any vertex.*

Proof. Clearly the set D_{alg} maintained by Directed-DomSet is a directed dominating set. Moreover the algorithm is 1-stable as after each arrival, it either adds a single vertex to D_{alg} or does nothing. It is easily checked that the set U maintained by the algorithm is always a set of pairwise unrelated vertices, and that all vertices in D_{alg} are an out-neighbor of a vertex in U or are in U themselves. Hence, by Lemmas 8 and 9, at any time t we have

$$|D_{\text{alg}}(t)| \leqslant (d+1) \cdot |U(t)| \leqslant (d+1) \cdot \text{OPT}_{\text{out}}(t) \leqslant (d+1)^2 \cdot \text{OPT}(t),$$

which finishes the proof. ◀

In Appendix A we show that the approximation ratio $\Theta(d^2)$ is tight for this algorithm.

A 3-stable $(9d/2)$-approximation algorithm. The algorithm presented above has optimal stability, but its approximation ratio is $\Omega(d^2)$. We now present an algorithm whose stability is still very small, namely 3, but whose approximation ratio is only $O(d)$. This is asymptotically optimal, since, as is easy to see, any algorithm with constant stability must have approximation ratio $\Omega(d)$. Our approach is somewhat similar to that of Liu and Toole-Charignon [21], but a key difference is that we obtain a worst-case bound on the stability and they obtain an amortized bound. Our algorithm works in *phases*, as explained next. Suppose we start a new phase at time t and let $D_{\text{alg}}(t-1)$ be the output dominating set at time $t-1$. The algorithm then computes a minimum dominating set $D_{\text{opt}}(t)$ for the graph $G(t)$, which we call the *target dominating set*. The algorithm will then slowly migrate from $D_{\text{alg}}(t-1)$ to $D_{\text{opt}}(t)$, by first adding the vertices in $D^+ := D_{\text{opt}}(t) \setminus D_{\text{alg}}(t-1)$ and then removing the vertices in $D^- := D_{\text{alg}}(t-1) \setminus D_{\text{opt}}(t)$. This is done in $\lceil |D^+ \cup D^-|/2 \rceil$ steps. Vertices that arrive in the meantime are also added to the dominating set, to ensure that the output remains a dominating set at all time. After all vertices in D^+ and D^- have been added and deleted, respectively, the next phase starts. Next we describe and analyze the algorithm in detail.

At the start of the whole algorithm, at time $t = 0$, we initialize $D_{\text{alg}}(0) := \emptyset$, and $D(0)^+ = \emptyset$ and $D(0)^- = \emptyset$.

🟨 **Algorithm 3** Set-and-Achieve-Target(v).

```
▷ $v$ is the vertex arriving at time $t$ and $G(t)$ is the graph after arrival of $v$
$D_{\text{alg}} \leftarrow D_{\text{alg}}(t-1) \cup \{v\}$
if $D^+(t-1) = \emptyset$ and $D^-(t-1) = \emptyset$ then        ▷ start a new phase
    Let $D_{\text{opt}}(t)$ be a minimum dominating set for $G(t)$.
    Set $D^+(t) \leftarrow D_{\text{opt}}(t) \setminus D_{\text{alg}}(t-1)$ and $D^-(t) \leftarrow D_{\text{alg}}(t-1) \setminus D_{\text{opt}}(t)$.
else
    Set $D^+(t) \leftarrow D^+(t-1)$ and $D^-(t) \leftarrow D^-(t-1)$
Set $m^+ \leftarrow \min(2, |D^+(t)|)$. Delete $m^+$ vertices from $D^+(t)$ and add them to $D_{\text{alg}}$.
Set $m^- \leftarrow \min(2 - m^+, |D^-(t)|)$. Delete $m^-$ vertices from $D^-(t)$ and delete the same
    vertices from $D_{\text{alg}}$.
$D_{\text{alg}}(t) \leftarrow D_{\text{alg}}$
```

The algorithm defined above is 3-stable, as it adds one vertex to D_{alg} in step 2 and then makes two more changes to D_{alg} in steps 8 and 9. Next we prove that its approximation ratio is bounded by $9d/2$. Note that the size of a minimum dominating set can reduce over time, due to the arrival of new vertices. The next lemma shows that this reduction is bounded. Let $\textsc{max-opt}(t) := \max(\textsc{opt}(1), \textsc{opt}(2), \ldots, \textsc{opt}(t))$ denote the maximum size of any of the optimal solutions until (and including) time t.

▶ **Lemma 11.** *For any time t we have $\textsc{max-opt}(t) \leqslant d \cdot \textsc{opt}(t)$, where d is the maximum arrival degree of any vertex.*

Proof. Let $t^* \leqslant t$ be such that $\textsc{max-opt}(t) = \textsc{opt}(t^*)$. Let $D_{\text{opt}}(t)$ be an optimal dominating set for $G(t)$ and define $V(t^*)$ to be the set of vertices of $G(t^*)$. Let D be the set of vertices in $D_{\text{opt}}(t)$ that were not yet present at time t^*, and define $D^* := (D_{\text{opt}}(t) \setminus D) \cup N_{t^*}(D)$, where $N_{t^*}(D)$ contains the neighbors of D in $V(t^*)$. Then D^* is a dominating set for $G(t^*)$ since any vertex in $V(t^*)$ that is not dominated by a vertex in $D_{\text{opt}}(t^*) \setminus D$ is in D^* itself. Moreover, $|D^*| \leqslant d \cdot \textsc{opt}(t)$, since each vertex in D has at most d neighbors in $V(t^*)$. ◀

We first bound the size of D_{alg} at the start of each phase. Note that in the proofs below, $D^+(t)$ and $D^-(t)$ refer to the situation before the execution of line 8 and 9 in the algorithm SET-AND-ACHIEVE-TARGET.

▶ **Lemma 12.** *If a new phase starts at time t, then $D_{\text{alg}}(t-1) \leqslant 3 \cdot \textsc{max-opt}(t-1)$.*

Proof. We proceed by induction on t. The lemma trivially holds at the start of the first phase, when $t = 1$. Now consider the start of some later phase, at time t, and let t_{prev} be the previous time at which a new phase started. Recall that $D_{\text{alg}}(t) = D_{\text{opt}}(t_{\text{prev}}) \cup \{\text{vertices arriving at times } t_{\text{prev}}, t_{\text{prev}}+1, \ldots, t-1\}$.

Moreover,

$$|D^+(t_{\text{prev}}) \cup D^-(t_{\text{prev}})| \leqslant |D_{\text{alg}}(t_{\text{prev}}-1)| + |D_{\text{opt}}(t_{\text{prev}})| \leqslant 3 \cdot \textsc{max-opt}(t_{\text{prev}}-1) + \textsc{opt}(t_{\text{prev}}),$$

where the last inequality uses the induction hypothesis. From time t_{prev} up to time $t-1$, the vertices from $D^+(t_{\text{prev}}) \cup D^-(t_{\text{prev}})$ are added/deleted in pairs, so

$$t - t_{\text{prev}} = \left\lceil \frac{3 \cdot \textsc{max-opt}(t_{\text{prev}}-1) + \textsc{opt}(t_{\text{prev}})}{2} \right\rceil \leqslant \left\lceil \frac{4 \cdot \textsc{max-opt}(t_{\text{prev}})}{2} \right\rceil = 2 \cdot \textsc{max-opt}(t_{\text{prev}}).$$

Hence,

$$\begin{aligned} D_{\text{alg}}(t-1) &\leqslant \textsc{opt}(t_{\text{prev}}) + (t - t_{\text{prev}}) \\ &\leqslant \textsc{max-opt}(t_{\text{prev}}) + 2 \cdot \textsc{max-opt}(t_{\text{prev}}) \\ &\leqslant 3 \cdot \textsc{max-opt}(t-1) \end{aligned}$$ ◀

The previous lemma bounds $|D_{\text{alg}}|$ just before the start of each phase. Next we use this to bound $|D_{\text{alg}}|$ during each phase.

▶ **Lemma 13.** *For any time t we have $|D_{\text{alg}}(t)| \leqslant (9/2) \cdot \textsc{max-opt}(t)$.*

Proof. Consider a time t. If a new phase starts at time $t+1$ then the lemma follows directly from Lemma 12. Otherwise, let $t_{\text{prev}} \leqslant t$ be the last time at which a new phase started, and let t_{next} be the next time at which a new phase starts. Furthermore, let $t^* := \max\{t' : t_{\text{prev}} \leqslant t' < t_{\text{next}} \text{ and } D^+(t') \neq \emptyset\}$. In other words, t^* is the last time step in the interval $[t_{\text{prev}}, t_{\text{next}})$ at which we still add vertices from D^+ to D_{alg}. If $D^+(t_{\text{prev}})$ is empty then let $t^* = t_{\text{prev}}$. It is easy to see from the algorithm that $|D_{\text{alg}}(t)| \leqslant |D_{\text{alg}}(t^*)|$. Note that $D_{\text{alg}}(t^*)$ contains the vertices from $D_{\text{alg}}(t_{\text{prev}} - 1)$, plus the vertices from $D_{\text{opt}}(t_{\text{prev}})$, plus the vertices that arrived from time t_{prev} to time t^*. Hence,

$$\begin{aligned} |D_{\text{alg}}(t^*)| &\leqslant |D_{\text{alg}}(t_{\text{prev}} - 1)| + \text{OPT}(t_{\text{prev}}) + (t^* - t_{\text{prev}} + 1) \\ &\leqslant |D_{\text{alg}}(t_{\text{prev}} - 1)| + \text{OPT}(t_{\text{prev}}) + \tfrac{\text{OPT}(t_{\text{prev}})}{2} \\ &\leqslant 3 \cdot \text{MAX-OPT}(t_{\text{prev}} - 1) + \text{MAX-OPT}(t_{\text{prev}}) + \tfrac{\text{OPT}(t_{\text{prev}})}{2} \\ &\leqslant (9/2) \cdot \text{MAX-OPT}(t). \end{aligned}$$

Note that from the first to the second line we replaced $(t^* - t_{\text{prev}} + 1)$ by $\text{OPT}(t_{\text{prev}})/2$, which we can do because we add vertices from $D_{\text{opt}}(t_{\text{prev}})$ in pairs. It may seem that we should actually write $\left\lceil \frac{\text{OPT}(t_{\text{prev}})}{2} \right\rceil$ here. When $\text{OPT}(t_{\text{prev}})$ is odd, however, then the algorithm can already remove a vertex in D^- from D_{alg} when the last vertex from D^+ is added to D_{alg}. (This is not true in the special case when $D^-(t_{\text{prev}}) = \emptyset$, but in that case the second term is an over-estimation. Indeed, when $D^-(t_{\text{prev}}) = \emptyset$ then $D_{\text{alg}}(t_{\text{prev}} - 1) \subseteq D_{\text{opt}}(t_{\text{prev}})$ and so $|D^+(t_{\text{prev}})| = |D_{\text{opt}}(t_{\text{prev}}) \setminus D_{\text{alg}}(t_{\text{prev}} - 1)| < |\text{OPT}(t_{\text{prev}} - 1)|$.) This finishes the proof. ◀

Putting Lemmas 11 and 13 together, we obtain the following theorem.

▶ **Theorem 14.** *There is a 3-stable $(9d/2)$-approximation algorithm for* DOMINATING SET *in the vertex-arrival model, where d is the maximum arrival degree of any vertex.*

4 Stable Approximation Algorithms for INDEPENDENT SET

In this section we first show that INDEPENDENT SET does not admit a SAS, even when restricted to graphs of maximum degree 3. Then we give an 2-stable $O(d)$-approximation algorithm for graphs whose average degree is bounded by d.

4.1 No SAS for graphs of maximum degree 3

We prove our no-SAS result for INDEPENDENT SET in a similar (but simpler) way as for DOMINATING SET. Thus we actually prove the stronger result that there is a constant $\varepsilon^* > 0$ such that any dynamic $(1+\varepsilon^*)$-approximation algorithm for INDEPENDENT SET in the vertex arrival model, must have stability parameter $\Omega(n)$, in this case even when the maximum degree of any of the graphs $G(t)$ is bounded by 3.

Let $\varepsilon^* > 0$ be a real number less than $\min\left(\frac{0.82\delta}{1-0.82\delta}, \varepsilon\right)$. Let ALG be an algorithm that maintains an independent set I_{alg} such that $\text{OPT}(t) \leqslant (1+\varepsilon^*) \cdot |I_{\text{alg}}(t)|$ at all times. Let $f_{\varepsilon^*}(n)$ denote the stability of ALG, that is, the maximum number of changes it performs on I_{alg} when a new vertex arrives, where n is the number of vertices before the arrival. We will show that, for arbitrarily large n, there is a sequence of n arrivals that requires $f_{\varepsilon^*}(n) \geqslant \frac{1}{6(2+\varepsilon)} \lfloor \delta n \rfloor$. As before, choose N large enough such that the bipartite expander graph $G_{\text{exp}} = (L \cup R, E)$ from Proposition 3 exists for $\mu = 0.005$ and $|R| = N$. In our no-SAS construction for INDEPENDENT SET, we only need G_{exp}, not additional layers are needed. Thus the construction is simply as follows.

- First the vertices $r_1, \ldots, r_N$ from the set R arrive one by one, as singletons.
- Next the vertices $\ell_1, \ldots, \ell_{(1+\varepsilon)N}$ from L arrive one by one (in any order), along with their incident edges in G_{exp}.

▶ **Lemma 15.** *Let t^* be the first time when $|I_{\text{alg}}(t) \cap L| \geqslant \delta N$. If $f_{\varepsilon^*}(n) < \frac{1}{6(2+\varepsilon)} \lfloor \delta n \rfloor$ then* $\text{OPT}(t^*) > (1+\varepsilon^*) \cdot |I_{\text{alg}}(t^*)|$.

Proof. Let n_t denote the number of vertices of the graph $G(t)$, and observe that $n_{t^*} \leqslant (2+\varepsilon)N$. Hence, $f_{\varepsilon^*}(n_{t^*}) \leqslant \frac{1}{6}\delta N$. By definition of t^*, we know that just before time t^* we have $|I_{\text{alg}} \cap L| < \delta N$. Because ALG is $f_{\varepsilon^*}(n)$-stable, we have

$$|I_{\text{alg}}(t^*) \cap L| \leqslant \delta N + f(n_{t_f}) \leqslant \frac{7}{6}\delta N.$$

Since $\mu = 0.005$, from Proposition 3 we have $|N(I_{\text{alg}}(t^*) \cap L| \geqslant 1.99 \cdot \delta N$. Hence $|I_{\text{alg}}(t^*) \cap R| \leqslant N - 1.99 \cdot \delta N$, and so

$$|I_{\text{alg}}(t^*)| = |I_{\text{alg}}(t^*) \cap R| + |I_{\text{alg}}(t^*) \cap L| \leqslant \frac{7}{6}\delta N + N - 1.99 \cdot \delta N < N - 0.82\,\delta N$$

. We also have $|\text{OPT}(t^*)| \geqslant N$. Hence, $\text{OPT}(t^*) > \frac{N}{N-0.82\delta N}|I_{\text{alg}}(t^*)| > (1+\varepsilon^*) \cdot |I_{\text{alg}}(t^*)|$. ◀

▶ **Theorem 16.** *There is a constant $\varepsilon^* > 0$ such that any dynamic $(1+\varepsilon^*)$-approximation algorithm for* INDEPENDENT SET *in the vertex arrival model, must have stability parameter $\Omega(n)$, even when the maximum degree of any of the graphs $G(t)$ is bounded by 3.*

Proof. By Proposition 3 the maximum degree of the graph is always bounded by 3. Let $t = 2N + \varepsilon N$ and let $|I_{\text{alg}}(t) \cap L| = M$. We know by Lemma 15 that if ALG is a $(1+\varepsilon^*)$-approximation and if $f_{\varepsilon^*}(n) < \frac{1}{6(2+\varepsilon)}\lfloor \delta n \rfloor$ then $M \leqslant \delta n$. Hence $|I_{\text{alg}}(t) \cap R| \leqslant N - 1.99M$. So we have

$$|I_{\text{alg}}(t)| = |I_{\text{alg}}(t) \cap R| + |I_{\text{alg}}(t) \cap L| \leqslant N - 1.99M + M \leqslant N.$$

But we have $\text{OPT}(t) = (1+\varepsilon)N$. Hence the approximation ratio at time $t = 2N + \varepsilon N$ is greater than or equal to $\frac{(1+\varepsilon)N}{N} = 1 + \varepsilon > 1 + \varepsilon^*$ which is a contradiction. This finishes the proof. ◀

4.2 Constant-stability algorithms when the average degree is bounded

A 2-stable $O(d)$-approximation algorithm. In this section we consider the setting where the average degree of $G(t)$ is upper bounded by some constant d at all times. It is easy to observe that in this setting, if we allow just one change after each vertex arrival, then it's not possible to get a bounded approximation ratio. However, we are able to get a bounded approximation ratio with only two changes per arrival.

It is not hard to see that if the maximum degree is bounded by some constant d^*, then a simple greedy 1-stable algorithm maintains a $O(d^*)$ approximation. Our idea is to maintain an induced subgraph with a number of vertices that is linear in the number of vertices of $G(t)$, and whose maximum degree (rather than average degree) is bounded. We then use the induced subgraph to generate an independent set. Below we make the idea precise.

First we define a (trivial) subroutine algorithm below which takes in an independent set I^* and a subset W^* of vertices as an input, and tries to add a vertex v from $W^* \setminus I^*$ to I^* such that $I^* \cup \{v\}$ is still an independent set.

Algorithm 4 GREEDY-ADDITION(I^*, W^*).

```
if there exist a vertex $v \in W^* \setminus I^*$ such that $I^* \cup \{v\}$ is an independent set then
    Set $I^* = I^* \cup \{v\}$
```

Next we move on to describe our main algorithm, which uses GREEDY-ADDITION as a subroutine. Let $\Delta(G)$ denote the maximum degree of a graph $G = (V, E)$. For a subset $W \subset V$, define $G[W]$ to be the subgraph of G induced by W. Let $V(t)$ denote the set of vertices of $G(t)$.

Observe that by ordering the vertices of $G(t)$ in increasing order of their degree and by taking the first $\lceil \frac{99}{100}|V(t)| \rceil$ vertices, we can construct a set $V^*(t) \subseteq V$ such that $|V^*(t)| \geqslant \frac{99}{100}|V(t)|$ and $\Delta(G[V^*(t)]) \leqslant 100d$. (The number 100 has no special significance – it can be chosen much smaller – but we use it for convenience.) The idea of our algorithm is to maintain a vertex set $W(t) \subseteq V(t)$ such that $\Delta(G[W(t)]) \leqslant 100d$ and the size of $W(t)$ is linear in $|V(t)|$. In order to maintain such a subset, we work in phase, as before: at the start of each phase, the algorithm sets itself a target vertex set $V^*(t)$ of large size and with $\Delta(G[V^*(t)]) \leqslant 100d$. At time t, $W^+(t)$ and $W^-(t)$ denote the vertices that need to be added and removed respectively from $W(t)$ in order to achieve the target. This is done by the algorithm presented next, where we initialize $W^+(t), W^-(t), W(t) = \emptyset$.

Algorithm 5 SET-ACHIEVE-AND-USE-TARGET(v).

```
▷ $v$ is the vertex arriving at time $t$ and $G(t)$ is the graph after arrival of $v$
if $W^+(t-1) = \emptyset$ and $W^-(t-1) = \emptyset$ then                ▷ start a new phase
    Choose $V^*(t) \subseteq V$ such that $|V^*(t)| \geqslant \frac{99}{100}|V(t)|$ and $\Delta(G[V^*(t)]) \leqslant 100d$.
    Set $W^+(t) \leftarrow V^*(t) \setminus W(t-1)$ and $W^-(t) \leftarrow W(t-1) \setminus V^*(t)$.
else
    Set $W^+(t) \leftarrow W^+(t-1)$ and $W^-(t) \leftarrow W^-(t-1)$
Set $m^- \leftarrow \min(1, |W^-(t)|)$. Delete $m^-$ vertices from $W^-(t)$ and delete the same vertices from $W(t)$. Call the vertex deleted(if any) as $v^*$.
Set $I_{\text{alg}}(t) \leftarrow I_{\text{alg}}(t-1) \setminus \{v^*\}$
Set $m^+ \leftarrow \min(1 - m^-, |W^+(t)|)$. Delete $m^+$ vertices from $W^+(t)$ and add them to $W(t)$.
GREEDY-ADDITION($I_{\text{alg}}(t), W(t)$)
```

▶ **Lemma 17.** *At the start of each phase – that is, at a time t such that $W^+(t-1) = \emptyset$ and $W^-(t-1) = \emptyset$ – we have $|W(t-1)| \geqslant \frac{495}{1000} \cdot |V(t)|$.*

Proof. We proceed by induction on t. The lemma trivially holds at the start of the first phase, when $t = 1$. Now consider the start of some later phase, at time t, and let t_{prev} be the previous time at which a new phase started. Since the start of a phase is just after the end of the previous phase we have $|W(t-1)| \geqslant \frac{99}{100}|V(t_{\text{prev}})|$.

Observe that $W^+(t_{\text{prev}})$ and $W^-(t_{\text{prev}})$ are disjoint subsets of the vertices of $V(t_{\text{prev}})$. Hence $|W^+(t_{\text{prev}}) \cup W^-(t_{\text{prev}})| \leqslant |V(t_{\text{prev}})|$. So $(t - t_{\text{prev}}) \leqslant |V(t_{\text{prev}})|$, since we make one change at a time. So $|V(t)| \leqslant 2|V(t_{\text{prev}})|$ and we have

$$|W(t-1)| \geqslant \frac{99}{100} \cdot |V(t_{\text{prev}})| \geqslant \frac{495}{1000} \cdot 2 \cdot |V(t_{\text{prev}})| \geqslant \frac{495}{1000} \cdot |V(t)|$$

This finishes the proof of the lemma. ◀

The previous lemma gives lower bound of $|W(t)|$ at the start of each phase. Next we use this to give a lower bound of $|W(t)|$ at any time point t.

▶ **Lemma 18.** *For any time t, we have $|W(t)| \geqslant \frac{485}{1010} \cdot |V(t)|$.*

Proof. Consider a time t. If a new phase starts at time t then the lemma follows directly from Lemma 17. Otherwise, let t_{prev} be the previous time at which a new phase started, and let t_{next} be the next time at which a new phase starts.

Let $t^* := \max\{t' : t_{\text{prev}} \leqslant t' < t_{\text{next}}$ and $W^-(t') \neq \emptyset\}$. In other words, t^* is the last time step in the interval $[t_{\text{prev}}, t_{\text{next}})$ at which we still delete vertices from $W(t)$. It is easy to see from the algorithm that in the interval $[t_{\text{prev}}, t_{\text{next}})$, the value $\frac{|W(t)|}{|V(t)|}$ is minimum at $t = t^*$. Observe that $|W^-(t_{\text{prev}})| \leqslant \frac{1}{100} \cdot |V(t_{\text{prev}})|$, which implies $(t^* - t_{\text{prev}}) \leqslant \frac{1}{100} \cdot |V(t_{\text{prev}})|$. Hence, $|V(t^*)| \leqslant \frac{101}{100}|V(t_{\text{prev}})|$ and so by Lemma 17 we have

$$|W(t^*)| \geqslant \frac{495}{1000} \cdot |V(t_{\text{prev}})| - \frac{1}{100} \cdot |V(t_{\text{prev}})| = \frac{485}{1010} \cdot \frac{101}{100} \cdot |V(t_{\text{prev}})| \geqslant \frac{485}{1010} \cdot |V(t^*)|$$

This finishes the proof of the lemma. ◀

▶ **Lemma 19.** *For any time t we have $\frac{|W(t)|}{|I_{\text{alg}}(t)|} \leqslant c \cdot d$ for some constant c.*

Proof. Observe that at any time point t the maximum degree of $G[W(t)]$ is bounded by $100d$. Choose a constant c such that $c \cdot d > 100d + 1$. Hence if $I \subseteq W(t)$ is an independent set with $|I| < \frac{|W(t)|}{c \cdot d}$, then there always exist a vertex $v \in W(t) \setminus I$ such that $\{v\} \cup I$ is a independent set. If not then all vertices in $W(t) \setminus I$ have an edge to I. Now $|W(t) \setminus I| \geqslant \frac{(c \cdot d - 1)}{c \cdot d}|W(t)|$, so the average degree of I in $G[W(t)]$ is greater than $c \cdot d - 1 > 100d$, which is a contradiction.

Also observe that $I_{\text{alg}}(t) \subset W(t)$. Now we proceed by induction on t. The lemma holds trivially for $t = 0$. Now suppose the lemma holds for $t = k$. There are two cases.

Case 1: At t=k+1, a vertex v^* is deleted from $W(t)$

In this case initially $I_{\text{alg}}(t) = I_{\text{alg}}(t-1) \setminus \{v^*\}$. Then the subroutine Greedy-Addition tries to add a new vertex to $I_{\text{alg}}(t)$. If $\frac{|W(t)|}{|I_{\text{alg}}(t)|} \leqslant c \cdot d$ before Greedy-Addition is initiated then we are done. Else by the arguments above, we know that if $|I_{\text{alg}}| < \frac{|W(t)|}{c \cdot d}$ then the subroutine Greedy-Addition can always add a vertex. In that case $|I_{\text{alg}}(t)| \geqslant |I_{\text{alg}}(t-1)|$ and $|W(t)| = |W(t-1)| - 1$, so $\frac{|W(t)|}{|I_{\text{alg}}(t)|} \leqslant \frac{|W(t-1)|}{|I_{\text{alg}}(t-1)|} \leqslant c \cdot d$ by induction.

Case 2: At t=k+1, a vertex v^* is added to $W(t)$

If $\frac{|W(t)|}{|I_{\text{alg}}(t)|} \leqslant c \cdot d$ before Greedy-Addition is initiated then we are done. Else we know that if $|I_{\text{alg}}| < \frac{|W(t)|}{c \cdot d}$ then the subroutine Greedy-Addition can always add a vertex. In that case $|I_{\text{alg}}(t)| = |I_{\text{alg}}(t-1)| + 1$ and $|W(t)| = |W(t-1)| + 1$, so $\frac{|W(t)|}{|I_{\text{alg}}(t)|} \leqslant \frac{|W(t-1)|}{|I_{\text{alg}}(t-1)|} \leqslant c \cdot d$ by induction. ◀

▶ **Theorem 20.** *There is a 2-stable $O(d)$-approximation algorithm for* Independent Set *in the vertex-arrival model where the average degree of $G(t)$ is bounded by d at all times.*

Proof. We know that the size of the maximum independent set at time t is trivially bounded by $|V(t)|$. Now by Lemmas 18 and 19, we have,

$$\frac{|V(t)|}{|I_{\text{alg}}(t)|} \leqslant \frac{1010}{485} \frac{|W(t)|}{|I_{\text{alg}}(t)|} \leqslant \frac{1010}{485} \cdot c \cdot d$$

Hence the algorithm set-achieve-and-use-target is an $O(d)$ approximation. Also observe that the maximum number of changes to $I_{\text{alg}}(t)$ occurs when there is a induced deletion of a single vertex due to the deletion of a vertex from $W(t)$ and then adding a vertex during the execution of the subroutine algorithm Greedy-Addition. So clearly the stabilty of the algorithm set-achieve-and-use-target is bounded by 2. ◀

5 Concluding remarks

We studied the stability of dynamic algorithms for DOMINATING SET and INDEPENDENT SET in the vertex-arrival model. For both problems we showed that a SAS does not exist. For INDEPENDENT SET this even holds when the degrees of all vertices are bounded by 3 at all times. This is clearly tight, since a SAS is easily obtained on graphs of maximum degree 2. For DOMINATING SET the no-SAS result holds for degree-4 graphs. A challenging open problem is whether a SAS exists for DOMINATING SET for degree-3 graphs. We also gave algorithms whose approximation ratio and/or stability depends on the (arrival or average) degree. An interesting open problem here is: Is there a 1-stable $O(d)$-approximation algorithm for DOMINATING SET, when the arrival degree is at most d? Finally, we believe the concept of stability for dynamic algorithms, which purely focuses on the change in the solution (rather than computation time) is interesting to explore for other problems as well.

References

1 Spyros Angelopoulos, Christoph Dürr, and Shendan Jin. Online maximum matching with recourse. In *Proc. 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 117 of *LIPIcs*, pages 8:1–8:15, 2018. doi:10.4230/LIPIcs.MFCS.2018.8.

2 Daniel Antunes, Claire Mathieu, and Nabil H. Mustafa. Combinatorics of local search: An optimal 4-local Hall's theorem for planar graphs. In *Proc. 25th Annual European Symposium on Algorithms (ESA)*, volume 87 of *LIPIcs*, pages 8:1–8:13, 2017. doi:10.4230/LIPIcs.ESA.2017.8.

3 Sebastian Berndt, Valentin Dreismann, Kilian Grage, Klaus Jansen, and Ingmar Knof. Robust online algorithms for certain dynamic packing problems. In *Proc. 17th International Workshop on Approximation and Online Algorithms (WAOA)*, volume 11926 of *Lecture Notes in Computer Science*, pages 43–59, 2019. doi:10.1007/978-3-030-39479-0_4.

4 Sebastian Berndt, Leah Epstein, Klaus Jansen, Asaf Levin, Marten Maack, and Lars Rohwedder. Online bin covering with limited migration. In *Proc. 27th Annual European Symposium on Algorithms (ESA)*, volume 144 of *LIPIcs*, pages 18:1–18:14, 2019. doi:10.4230/LIPIcs.ESA.2019.18.

5 Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.

6 Joan Boyar, Stephan J. Eidenbenz, Lene M. Favrholdt, Michal Kotrbcík, and Kim S. Larsen. Online dominating set. *Algorithmica*, 81(5):1938–1964, 2019. doi:10.1007/s00453-018-0519-1.

7 Joan Boyar, Lene M. Favrholdt, Michal Kotrbcík, and Kim S. Larsen. Relaxing the irrevocability requirement for online graph algorithms. *Algorithmica*, 84(7):1916–1951, 2022. doi:10.1007/s00453-022-00944-w.

8 Miroslav Chlebík and Janka Chlebíková. Approximation hardness of dominating set problems in bounded degree graphs. *Inf. Comput.*, 206(11):1264–1275, 2008. doi:10.1016/j.ic.2008.07.003.

9 Minati De, Sambhav Khurana, and Satyam Singh. Online dominating set and independent set. *CoRR*, abs/2111.07812, 2021. arXiv:2111.07812.

10 Mark de Berg, Hans L. Bodlaender, Sándor Kisfaludi-Bak, Dániel Marx, and Tom C. van der Zanden. A framework for Exponential-Time-Hypothesis-tight algorithms and lower bounds in geometric intersection graphs. *SIAM J. Comput.*, 49:1291–1331, 2020. doi:10.1137/20M1320870.

11 Mark de Berg, Sándor Kisfaludi-Bak, Morteza Monemizadeh, and Leonidas Theocharous. Clique-based separators for geometric intersection graphs. In *Proc. 32nd International Symposium on Algorithms and Computation (ISAAC)*, volume 212 of *LIPIcs*, pages 22:1–22:15, 2021. doi:10.4230/LIPIcs.ISAAC.2021.22.

12 Mark de Berg, Arpan Sadhukhan, and Frits C. R. Spieksma. Stable approximation algorithms for the dynamic broadcast range-assignment problem. In *Proc. 18th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, volume 227 of *LIPIcs*, pages 15:1–15:21, 2022. doi:10.4230/LIPIcs.SWAT.2022.15.

13 Oliver Göbel, Martin Hoefer, Thomas Kesselheim, Thomas Schleiden, and Berthold Vöcking. Online independent set beyond the worst-case: Secretaries, prophets, and periods. In *Proc. 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 8573 of *Lecture Notes in Computer Science*, pages 508–519, 2014. doi:10.1007/978-3-662-43951-7_43.

14 Anupam Gupta, Ravishankar Krishnaswamy, Amit Kumar, and Debmalya Panigrahi. Online and dynamic algorithms for set cover. In *Proc. 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 537–550, 2017. doi:10.1145/3055399.3055493.

15 Anupam Gupta, Amit Kumar, and Cliff Stein. Maintaining assignments online: Matching, scheduling, and flows. In Chandra Chekuri, editor, *Proc. 2th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 468–479, 2014. doi:10.1137/1.9781611973402.35.

16 Magnús M. Halldórsson, Kazuo Iwama, Shuichi Miyazaki, and Shiro Taketomi. Online independent sets. *Theor. Comput. Sci.*, 289(2):953–962, 2002. doi:10.1016/S0304-3975(01)00411-X.

17 Sariel Har-Peled and Kent Quanrud. Approximation algorithms for polynomial-expansion and low-density graphs. In *Proc. 23rd Annual European Symposium on Algorithms (ESA)*, volume 9294 of *Lecture Notes in Computer Science*, pages 717–728. Springer, 2015. doi:10.1007/978-3-662-48350-3_60.

18 Gow-Hsing King and Wen-Guey Tzeng. On-line algorithms for the dominating set problem. *Inf. Process. Lett.*, 61(1):11–14, 1997. doi:10.1016/S0020-0190(96)00191-3.

19 Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math*, 36(2):177–189, 1977. doi:doi/10.1137/0136016.

20 Richard J. Lipton and Andrew Tomkins. Online interval scheduling. In *Proc. 5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 302–311, 1994. URL: http://dl.acm.org/citation.cfm?id=314464.314506.

21 Hsiang-Hsuan Liu and Jonathan Toole-Charignon. The power of amortized recourse for online graph problems. *CoRR*, abs/2206.01077, 2022. doi:10.48550/arXiv.2206.01077.

22 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proc. 38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 681–690, 2006. doi:10.1145/1132516.1132612.

A Lower-bound example for Directed-DomSet

Fig. 2 shows that the Directed-DomSet has approximation ratio $\Omega(d^2)$. First, vertex a_1 arrives, which will be put into U and D_{alg}. Then a_2 and $v_1, \dots, v_{d^2}$ arrive, with an outgoing edge to a_1. These vertices are dominated by a_1, so U and D_{alg} are not modified. Next, $w_1, \dots, w_d$ arrive. Vertex w_1 arrives with neighbors $v_1, \dots, v_d$. We have $U = D_{\text{alg}} = \{a_1\}$, so w_1 is not dominated by any node in D_{alg} and it is not related to any node in D_{alg}. Hence, w_1 will be put into U and D_{alg}. Then vertex w_2 arrives with neighbors $v_{d+1}, \dots, v_{2d}$. We have $U = D_{\text{alg}} = \{a_1, w_1\}$, so w_1 is not dominated by any node in D_{alg} and it is not related to any node in D_{alg}. Hence, w_2 will be put into U and D_{alg}. Similarly, $w_3, \dots, w_d$ will all be put into U and D_{alg}. Then $x_1, \dots, x_{d(d-1}$ arrive, each with one outgoing edge, to a unique node among $v_1, \dots, v_{d^2}$, as shown in the figure. They are all related to a node in U, namely one of the w_i's, so their out-neighbors will be out into D_{alg}. Finally, z arrives, which is also put into D_{alg}. Hence, at the end of the algorithm $|D_{\text{alg}}| = d^2 + 2$, but $\text{OPT} = 3$ since $\{a_1, a_2, z\}$ is a dominating set.

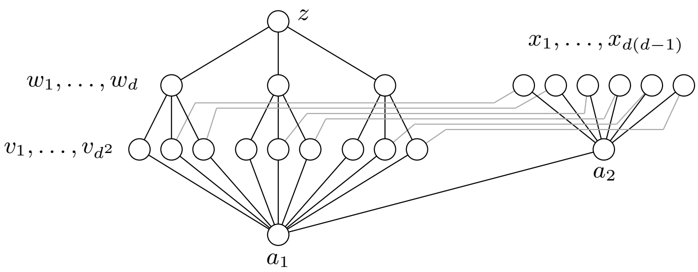


Figure 2 Example showing that DIRECTED-DOMSET has approximation ratio $\Omega(d^2)$.

Scalable Auction Algorithms for Bipartite Maximum Matching Problems

Quanquan C. Liu
Northwestern University, Evanston, IL, USA

Yiduo Ke
Northwestern University, Evanston, IL, USA

Samir Khuller
Northwestern University, Evanston, IL, USA

Abstract

Bipartite maximum matching and its variants are well-studied problems under various models of computation with the vast majority of approaches centering around various methods to find and eliminate augmenting paths. Beginning with the seminal papers of Demange, Gale and Sotomayor [DGS86] and Bertsekas [Ber81], bipartite maximum matching problems have also been studied in the context of *auction algorithms*. These algorithms model the maximum matching problem as an auction where one side of the bipartite graph consists of bidders and the other side consists of items; as such, these algorithms offer a very different approach to solving this problem that do not use classical methods. Dobzinski, Nisan and Oren [DNO14] demonstrated the utility of such algorithms in distributed, interactive settings by providing a simple and elegant $O(\log n/\varepsilon^2)$ round *maximum cardinality bipartite matching (MCM)* algorithm that has small round and communication complexity and gives a $(1-\varepsilon)$-approximation for any (not necessarily constant) $\varepsilon > 0$. They leave as an open problem whether an auction algorithm, with similar guarantees, can be found for the *maximum weighted bipartite matching (MWM)* problem. Very recently, Assadi, Liu, and Tarjan [ALT21] extended the utility of auction algorithms for MCM into the semi-streaming and massively parallel computation (MPC) models, by cleverly using maximal matching as a subroutine, to give a new auction algorithm that uses $O(1/\varepsilon^2)$ rounds and achieves the state-of-the-art bipartite MCM results in the streaming and MPC settings.

In this paper, we give new auction algorithms for maximum weighted bipartite matching (MWM) and maximum cardinality bipartite b-matching (MCbM). Our algorithms run in $O\left(\log n/\varepsilon^8\right)$ and $O\left(\log n/\varepsilon^2\right)$ rounds, respectively, in the distributed setting. We show that our MWM algorithm can be implemented in the distributed, interactive setting using $O(\log^2 n)$ and $O(\log n)$ bit messages, respectively, directly answering the open question posed by Demange, Gale and Sotomayor [DNO14]. Furthermore, we implement our algorithms in a variety of other models including the the semi-streaming model, the shared-memory work-depth model, and the massively parallel computation model. Our semi-streaming MWM algorithm uses $O(1/\varepsilon^8)$ passes in $O(n \log n \cdot \log(1/\varepsilon))$ space and our MCbM algorithm runs in $O(1/\varepsilon^2)$ passes using $O\left(\left(\sum_{i\in L} b_i + |R|\right)\log(1/\varepsilon)\right)$ space (where parameters b_i represent the degree constraints on the b-matching and L and R represent the left and right side of the bipartite graph, respectively). Both of these algorithms improves *exponentially* the dependence on ε in the space complexity in the semi-streaming model against the best-known algorithms for these problems, in addition to improvements in round complexity for MCbM. Finally, our algorithms eliminate the large polylogarithmic dependence on n in depth and number of rounds in the work-depth and massively parallel computation models, respectively, improving on previous results which have large polylogarithmic dependence on n (and exponential dependence on ε in the MPC model).

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases auction algorithms, maximum weight bipartite matching, maximum b-matching, distributed blackboard model, parallel work-depth model, streaming model, massively parallel computation model

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.28

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 28; pp. 28:1–28:24
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Category APPROX

Related Version *Full Version*: https://arxiv.org/abs/2307.08979

Funding We gratefully acknowledge support from NSF-Award 2216970 (IDEAL Institute).

1 Introduction

One of the most basic problems in combinatorial optimization is that of bipartite matching. This central problem has been studied extensively in many fields including operations research, economics, and computer science and is the cornerstone of many algorithm design courses and books. There is an abundance of existing classical and recent theoretical work on this topic [25, 12, 13, 19, 20, 27, 29, 30, 6, 11, 31]. Bipartite maximum matching and its variants are commonly taught in undergraduate algorithms courses and are so prominent to be featured regularly in competitive programming contests. In both of these settings, the main algorithmic solutions for maximum cardinality matching (MCM) and its closely related problems of maximum weight matching (MWM) are the Hungarian method using augmenting paths and reductions to maximum flow. Although foundational, such approaches are sometimes difficult to generalize to obtain efficient solutions in other *scalable models of computation*, e.g. distributed, streaming, and parallel models.

Although somewhat less popularly known, the elegant and extremely simple *auction-based maximum cardinality and maximum weighted matching algorithms* of Demange, Gale, and Sotomayor [10] and Bertsekas [8] solve the maximum cardinality/weighted matching problems in bipartite graphs. Their MCM auction algorithms denote vertices on one side of the bipartite input as *bidders* and the other side as *items*. Bidders are maintained in a queue and while the queue is not empty, the first bidder from the queue *bids* on an item with minimum price (breaking ties arbitrarily) from its neighbors. This bidder becomes the new owner of the item. Each time an item is reassigned to a new bidder, its price increases by some (not necessarily constant) $\varepsilon > 0$. If the assigned item still has price less than 1, the bidder is added again to the end of the queue. Setting $\varepsilon = \frac{1}{n+1}$ results in an algorithm that gives an exact maximum cardinality matching in $O(mn)$ time, where m and n refer to the number of edges and vertices respectively. Such an algorithm intuitively takes advantage of the fact that bidders prefer items in low demand (smaller price); naturally, such items should also be matched in a maximum cardinality matching.

One of the bottlenecks in the original auction algorithm is the need to maintain bidders in a queue from which they are selected, one at a time, to bid on items. Such a bottleneck is a key roadblock to the scalability of such algorithms. More recently, Dobzinski, Nisan, and Oren [11] extended this algorithm to the approximation setting for any (not necessarily constant) $\varepsilon > 0$. They give a simple and elegant randomized $(1 - \varepsilon)$-approximation algorithm for bipartite MCM in $O\left(\frac{\log n}{\varepsilon^2}\right)$ rounds of communication for any $\varepsilon > 0$. Furthermore, they illustrate an additional advantage for this algorithm beyond its simplicity. They show that in a distributed, interactive, blackboard setting, their auction MCM multi-round interactive algorithm uses less communication bits than traditional algorithms for this problem. This interactive setting is modeled via simultaneous communication protocols where agents simultaneously send a single message in each round to a central coordinator and some state is computed by the central coordinator after each round of communication. The goal in this model is to limit the total number of bits sent in all of the agents' messages throughout the duration of the algorithm. They leave as an open question whether an interactive, approximation auction algorithm that uses approximately the same number of rounds and bits of communication can be found for the maximum *weighted* bipartite matching problem.

Such an approach led to the recent simple and elegant paper of Assadi, Liu, and Tarjan [6] that adapted their algorithm to the semi-streaming setting and removed the $\log n$ factor in the semi-streaming setting from the number of passes to give an algorithm that finds an $(1-\varepsilon)$-approximate maximum cardinality matching in $O\left(1/\varepsilon^2\right)$ passes, where in each pass a maximal matching is found. Furthermore, they showed implementations of their algorithm in the massively parallel computation (MPC) model, achieving the best-known bounds in both of these settings. In this paper, we extend their algorithm to other variants of the problem on bipartite graphs, including maximum weight matching and maximum cardinality b-matching and achieve novel improvements in a variety of *scalable models*. The maximum cardinality b-matching problem (MCbM) is a well-studied generalization of MCM. In MCbM, each vertex is given an integer budget b_v where each vertex can be matched to at most b_v of their neighbors; a matching of maximum cardinality contains the maximum possible number of edges in the matching. The b-matching problem generalizes a number of real-life allocation problems such as server to client request serving, medical school residency matching, ad allocation, and many others. Although the problem is similar to MCM, often obtaining efficient algorithms for this problem requires non-trivial additional insights. As indicated in Ghaffari et al [16] b-matching problems can be considerably harder than matching.

Summary of Results. In this paper, we specifically give the following results. Our auction algorithms and their analyses are described in detail in Section 3 and Section 4.

▶ **Theorem 1** (Maximum Weight Bipartite Matching). *There exists an auction algorithm for maximum weight bipartite matching (MWM) that gives a $(1-\varepsilon)$-approximation for any $\varepsilon > 0$ and runs in $O\left(\frac{\log n \cdot \log(1/\varepsilon)}{\varepsilon^8}\right)$ rounds of communication (with high probability) and with $O\left(\log^2 n\right)$ bits per message. This algorithm can be implemented in the multi-round, semi-streaming model using $O\left(n \cdot \log n \cdot \log(1/\varepsilon)\right)$ space and $O\left(\frac{\log(1/\varepsilon)}{\varepsilon^7}\right)$ passes. This algorithm can be implemented in the work-depth model in $O\left(\frac{m \cdot \log(1/\varepsilon)}{\varepsilon^6}\right)$ work and $O\left(\frac{\log n \cdot \log(1/\varepsilon)}{\varepsilon^8}\right)$ depth. Finally, our algorithm can be implemented in the MPC model using $O(\log(1/\varepsilon)/\varepsilon^7)$ rounds, $O(n)$ space per machine, and $O\left(\frac{m \log(1/\varepsilon) \log n}{\varepsilon}\right)$ total space.*

The best-known algorithms in the semi-streaming model for the maximum weight bipartite matching problem are the $(1/\varepsilon)^{O(1/\varepsilon^2)}$ pass, $O\left(n \operatorname{poly}(\log(n)) \operatorname{poly}(1/\varepsilon)\right)$ space algorithm of Gamlath et al. [15] and the $O\left(\frac{\log(1/\varepsilon)}{\varepsilon^2}\right)$ pass, $O\left(\frac{n \log n}{\varepsilon^2}\right)$ space algorithm of Ahn and Guha [1]. To the best of our knowledge, our result is the first to achieve sub-polynomial dependence on $1/\varepsilon$ in the space for the MWM problem in the semi-streaming model. Thus, we improve the space bound exponentially compared to the previously best-known algorithms in the streaming model. The best-known algorithms in the distributed and work-depth models required $\operatorname{poly}(\log n)$ in the number of rounds and depth, respectively [21]; in the MPC setting, the best previously known algorithms have exponential dependence on ε [15]. We eliminate such dependencies in our paper and our algorithm is also simpler. A summary of previous results and our results can be found in Table 1.

▶ **Theorem 2** (Maximum Cardinality Bipartite b-Matching). *There exists an auction algorithm for maximum cardinality bipartite b-matching (MCbM) that gives a $(1-\varepsilon)$-approximation for any $\varepsilon > 0$ and runs in $O\left(\frac{\log n}{\varepsilon^2}\right)$ rounds of communication. This algorithm can be implemented in the multi-round, semi-streaming model using $O\left(\left(\sum_{i\in L} b_i + |R|\right)\log(1/\varepsilon)\right)$ space and $O\left(\frac{1}{\varepsilon^2}\right)$ passes. Our algorithm can be implemented in the shared-memory work-depth model in $O\left(\frac{\log^3 n}{\varepsilon^2}\right)$ depth and $O\left(\frac{m \log n}{\varepsilon^2}\right)$ total work.*

The best-known algorithms for maximum cardinality bipartite b-matching in the semi-streaming model is the $O\left(\frac{\log n}{\varepsilon^3}\right)$ pass, $\widetilde{O}\left(\frac{\sum_{i \in L \cup R} b_i}{\varepsilon^3}\right)$ space algorithm of Ahn and Guha [1]. In the general, non-bipartite setting (a harder setting than what we consider), a very recent $(1-\varepsilon)$-approximation algorithm of Ghaffari, Grunau, and Mitrović [16] runs in $\exp\left(2^{O(1/\varepsilon)}\right)$ passes and $\widetilde{O}\left(\sum_{i \in L \cup R} b_i + \text{poly}(1/\varepsilon)\right)$ space. Here, we also improve the space exponentially in $1/\varepsilon$ and, in addition, improve the number of passes by an $O(\log n)$ factor. More details comparing our results to other related works are given in Section 1.1. Due to the space constraints, most proofs in the following sections are deferred to the full version of our paper. Our results as well as comparisons with previous work are given in Table 1.

Table 1 We assume the ratio between the largest weight edge and smallest weight edge in the graph is $\text{poly}(n)$. Results for general graphs are labeled with (general); results that are specifically for bipartite graphs do not have a label. Upper bounds are given in terms of $O(\cdot)$ and lower bounds are given in terms of $\Omega(\cdot)$. "Space p.m." stands for space per machine. The complexity measures for the "blackboard distributed" setting is the total communication (over all rounds and players) in bits.

Model		Previous Results		Our Results	
Blackboard Distributed	MWM	$\Omega(n \log n)$ (trivial)	[11]	$O\left(\frac{n \log^3(n) \cdot \log(1/\varepsilon)}{\varepsilon^9}\right)$	Theorem 11
	MCbM	$\Omega(nb \log n)$	trivial	$O\left(\frac{nb \log^2 n}{\varepsilon^2}\right)$	Theorem 28
Streaming	MWM	$O\left(\frac{\log(1/\varepsilon)}{\varepsilon^2}\right)$ pass $O\left(\frac{n \log n}{\varepsilon^2}\right)$ space	[1]	$O\left(\frac{\log(1/\varepsilon)}{\varepsilon^7}\right)$ pass $O\left(n \cdot \log n \cdot \log(1/\varepsilon)\right)$ space	Theorem 13
	MCbM	$O(\log n/\varepsilon^3)$ pass $\widetilde{O}\left(\frac{\sum_{i \in L \cup R} b_i}{\varepsilon^3}\right)$ space	[2]	$O\left(\frac{1}{\varepsilon^2}\right)$ pass $O\left(\left(\sum_{i \in L} b_i + \lvert R \rvert\right) \log(1/\varepsilon)\right)$ space	Theorem 30
MPC	MWM	$O_\varepsilon(\log \log n)$ rounds $O_\varepsilon(n \,\text{poly}(\log n))$ space p.m.	[15] (general)	$O\left(\frac{\log(1/\varepsilon) \cdot \log \log n}{\varepsilon^7}\right)$ rounds $O(n)$ space p.m.	Theorem 17
Parallel	MWM	$O\left(m \cdot \text{poly}\left(1/\varepsilon, \log n\right)\right)$ work $O\left(\text{poly}\left(1/\varepsilon, \log n\right)\right)$ depth	[21] (general)	$O\left(\frac{m \log(1/\varepsilon)}{\varepsilon^6}\right)$ work $O\left(\frac{\log n \cdot \log(1/\varepsilon)}{\varepsilon^8}\right)$ depth	Theorem 15
	MCbM	N/A	N/A	$O\left(\frac{m \log n}{\varepsilon^2}\right)$ work $O\left(\frac{\log^3 n}{\varepsilon^2}\right)$ depth	Theorem 31

Concurrent, Independent Work. In concurrent, independent work, Zheng and Henzinger [34] study the maximum weighted matching problem in the sequential and dynamic settings using auction-based algorithms. Their simple and elegant algorithm makes use of a sorted list of items (by utility) for each bidder and then matches the bidders one by one individually (in round-robin order) to their highest utility item. They also extend their algorithm to give dynamic results. Due to the sequential nature of their matching procedure, they do not provide any results in scalable models such as the streaming, MPC, parallel, or distributed models.

1.1 Other Related Works

There has been no shortage of work done on bipartite matching. In addition to the works we discussed in the introduction, there has been a number of other relevant works in this general area of research. Here we discuss the additional works not discussed in Section 1. These include a plethora of results for $(1-\varepsilon)$-approximate maximum cardinality matching as well as some additional results for MWM and b-matching. Most of these works use various methods to find augmenting paths with only a few works focusing on auction-based techniques. We hope that our paper further demonstrates the utility of auction-based approaches in this setting and will lead to additional works in this area in the future. Although our work focuses on the bipartite matching problem, we also provide the best-known bounds for the matching problem on general graphs here, although this is a harder problem than our setting. We separate these results into the bipartite matching results, the general matching results, and lower bounds.

General Matching. A number of works have considered MCM in the streaming setting, providing state-of-the-art bounds in this setting. Fischer et al. [14] gave a deterministic $(1-\varepsilon)$-approximate MWM algorithm in general graphs in the semi-streaming model that uses $\text{poly}(1/\varepsilon)$ passes, improving exponentially on the number of passes of Lotker et al. [28]. Very recently, Assadi et al. [4] provided a semi-streaming algorithm in optimal $O(n)$ space and $O(\log n \log(1/\varepsilon)/\varepsilon)$ passes. They also provide a MWM algorithm that also runs in $O(n)$ space but requires $\widetilde{\Omega}(n/\varepsilon)$ passes. Please refer to these papers and references therein for older results in this area. Ahn and Guha [2] also considered the general weighted non-bipartite maximum matching problem in the semi-streaming model and utilize linear programming approaches for computing a $(2/3-\varepsilon)$-approximation and $(1-\varepsilon)$-approximation that uses $O(\log(1/\varepsilon)/\varepsilon^2)$ passes, $O\left(n \cdot \left(\frac{\log(1/\varepsilon)}{\varepsilon^2} + \frac{\log n/\varepsilon}{\varepsilon}\right)\right)$ space, and $O\left(\frac{\log n}{\varepsilon^4}\right)$ passes, $O\left(\frac{n\log n}{\varepsilon^4}\right)$ space, respectively.

Bipartite Matching. Ahn and Guha [2] also extended their results to the bipartite MWM and b-Matching settings with small changes. Specifically, in the MWM setting, they give a $O(\log(1/\varepsilon)/\varepsilon^2)$ pass, $O(n\cdot((\log(1/\varepsilon))/\varepsilon^2 + (\log n/\varepsilon)/\varepsilon))$ space algorithm. For maximum cardinality b-matching, they give a $O(\log n/\varepsilon^3)$ pass and $\widetilde{O}\left(\frac{\sum_{i\in L\cup R} b_i}{\varepsilon^3}\right)$ space algorithm. For exact bipartite MWM in the semi-streaming model, Liu et al. [26] gave the first streaming algorithm to break the n-pass barrier in the exact setting; it uses $\tilde{O}(n)$ space and $\tilde{O}(\sqrt{m})$ passes using interior point methods, SDD system solvers, and various other techniques to output the optimal matching with high probability. Work on bipartite MWM prior to [26] either required $\Omega(n\log n)$ passes[22] or only found approximate solutions [1, 2, 23].

Lower Bounds. Several papers have looked at matching problems from the lower bound side. Konrad et al. [24] considered the communication complexity of graph problems in a blackboard model of computation (for which the simultaneous message passing model of Dobzinski et al. [11] is a special variant). Specifically, they show that any non-trivial graph problem on n vertices require $\Omega(n)$ bits [24] in communication complexity. In a similar model called the *demand query model*, Nisan [32] showed that any *deterministic* algorithm that runs in $n^{o(1)}$ rounds where in each round at most $n^{1.99}$ demand queries are made, cannot find a MCM within a $n^{o(1)}$ factor of the optimum. This is in contrast to *randomized* algorithms which can make such an approximation using only $O(\log n)$ rounds. For streaming matching algorithms, Assadi [3] provided a conditional lower bound ruling out the possibilities of small constant factor approximations for two-pass streaming algorithms that solve the MCM problem. Such a lower bound also necessarily extends to MWM and MCbM. Goel et al. [17] provided a $n^{1+\Omega(1/\log\log n)}$ lower bound for the one-round message complexity of bipartite $(2/3+\varepsilon)$-approximate MCM (this also naturally extends to a space lower bound). For older papers on these lower bounds, please refer to references cited within each of the aforementioned cited papers. Finally, Assadi et al. [5] showed that any streaming algorithm that approximates MCM requires either $n^{\Omega(1)}$ space or $\Omega(\log(1/\varepsilon))$ passes.

Unweighted to Weighted Matching Transformations. Current transformations for transforming unweighted to weighted matchings all either:

- lose a factor of 2 in the approximation factor [18, 33], or
- increase the running time of the algorithm by an exponential factor in terms of $1/\varepsilon$, specifically, a factor of $\varepsilon^{-O(1/\varepsilon)}$ [7].

Thus, we cannot use such default transformations from unweighted matchings to weighted matchings in our setting since all of the complexity measures in this paper have only *polynomial* dependence on ε and all guarantee $(1-\varepsilon)$-approximate matchings. However, we do make use of *weighted to weighted matching transformations* provided our original weighted matching algorithms have only *polylogarithmic* dependence on the maximum ratio between edge weights in the graph. Such transformations from weighted to weighted matchings do not increase the approximation factor and also allows us to *eliminate the polylogarithmic dependence on the maximum ratio of edge weights.*

2 Preliminaries

This paper presents algorithms for bipartite matching under various settings. The input consists of a bipartite graph $G = (L \cup R, E)$. We denote the set of neighbors of any $i \in L, j \in R$ by $N(i), N(j)$, respectively. We present $(1-\varepsilon)$-approximation algorithms where $\varepsilon \in (0, 1)$ is our approximation parameter. All notations used in all of our algorithms in this paper are given in Table 2. The specified weight of an edge (i, j) will become the valuation of the bidder i for item j. Due to space constraints, we defer most of our proofs to the full version of our paper.

2.1 Scalable Model Definitions

In addition, we consider a number of scalable models in our paper including the ***blackboard distributed*** model, the ***semi-streaming*** model, the ***massively parallel computation (MPC)*** model, and the ***parallel shared-memory work-depth*** model.

Blackboard distributed model. We use the blackboard distributed model as defined in [11]. There are n *players*, one for each vertex of the left side of our bipartite graph (we assume wlog that the left side of the graph contains more vertices). The players engage in a fixed communication protocol using messages sent to a central coordinator. In other words, players write on a common "blackboard." Each players can receive a (not necessarily identical) message in each round from the coordinator. Players communicate using *rounds* of communication where in each round the player sends a message (of some number of bits) to the central coordinator. In every round, players choose to send messages depending solely on the contents of the blackboard and their private information. Termination of the algorithm and the final matching are determined by the central coordinator and the contents of the blackboard. The measure of complexity is the number of rounds of the algorithm and the message size sent by each player in each round. One can also measure the total number of bits send by all messages by multiplying the two.

Table 2 Table of Notations.

Symbol	Meaning
ε	approximation parameter
L, R	bidders, items, resp. WLOG $\lvert L\rvert \leq \lvert R\rvert$
i, j, i', j'	$i \in L, j \in R, i' \in L', j' \in R'$, i' (resp. j') indicates copy of i (resp. j)
p_j	current price of item j
D_i	demand set of bidder i
(i, a_i)	bidder $i \in L$ and currently matched item a_i
o_i	the item matched to bidder i in OPT
u_i	the utility of bidder i which is calculated by $1 - p_{a_i}$
$v_i(j)$	the valuation of bidder i for item j, i.e. the weight of edge (i, j)
C_i, C_j	copies of bidder $i \in L$, copies of item $j \in R$, resp.
L', R'	$L' = \bigcup_{i \in L} C_i$, $R' = \bigcup_{j \in R} C_j$
E', G'	$E' = \{(i^{(k)}, j^{(l)}) \mid (i, j) \in E, k \in [b_i], l \in [b_j]\}$, $G' = (L' \cup R', E')$
$c_{i'}$	price cutoff for bidder i'
N	ratio of the maximum weighted edge over the minimum weighted edge
v_i	the valuation function for bidder i
G_d	induced subgraph consisting of $(\cup_{i \in L} D_i) \cup L$
$\widehat{M}_d$	a non-duplicate maximal matching in G'_d
M'_d, M_d	produced matching in G', corresponding matching in G, resp.
$M_{\max}$	matching with largest cardinality produced

Semi-streaming model. In this paper, we use the semi-streaming model with arbitrary ordered edge insertions. Edges are arbitrarily (potentially adversarially) ordered in the stream. For this paper, we only consider insertion-only streams. The space usage for semi-streaming algorithms is bounded by $\widetilde{O}(n)$. The relevant complexity measures in this model are the number of passes of the algorithm and the space used.

Massively parallel computation (MPC) model. The massively parallel computation (MPC) model is a distributed model where different machines communicate with each other via a communication network. There are M machines, each with S space, and these machines communicate with each using Q rounds of communication. The initial graph is given in terms of edges and edges are partitioned arbitrarily across the machines. The relevant complexity measures are the total space usage $(M \cdot S)$, space per machine S, and number of rounds of communication Q.

Parallel shared-memory work-depth model. The parallal shared-memory work-depth model is a parallel model where different processors can process instructures in parallel and read and write from the same shared-memory. The relevant complexity measures for an algorithm in this model are the *work* which is the total amount of computation performed by the algorithm and the *depth* which is the longest chain of sequential dependencies in the algorithm.

3 An Auction Algorithm for $(1 - \varepsilon)$-Approximate Maximum Weighted Bipartite Matching

We present the following auction algorithm for maximum (weighted) bipartite matching (MWM) that is a generalization of the simple and elegant algorithm of Assadi et al. [6] to the weighted setting. Our generalization requires several novel proof techniques and recovers the round guarantee of Assadi et al. [6] in the maximum cardinality matching setting when the weights of all edges are 1. Furthermore, we answer an open question posed by Dobzinski et al. [11] for developing a $(1-\varepsilon)$-approximation auction algorithm for maximum *weighted* bipartite matching for which no prior algorithms are known. Throughout this section, we denote the maximum ratio between two edge weights in the graph by R. Our algorithm can also be easily extended into algorithms in various scalable models:

- a semi-streaming algorithm which uses $O\left(n \cdot \log n \cdot \log(1/\varepsilon)\right)$ space (the number of vertices in the bipartite graph) and which requires $O(\log(1/\varepsilon)/\varepsilon^7)$ rounds,
- a shared-memory parallel algorithm using $O\left(\frac{n \log n}{\varepsilon^9}\right)$ work and $O\left(\frac{1}{\varepsilon^8}\right)$ depth, and
- an MPC algorithm using $O(\log(1/\varepsilon)/\varepsilon^7)$ rounds, $O(n)$ space per machine, and $O\left(\frac{n \log n}{\varepsilon}\right)$ total space.

In contrast, the best-known semi-streaming MWM algorithm of Ahn and Guha [1] requires $\widetilde{O}(\log(1/\varepsilon)/\varepsilon^2)$ passes and $\widetilde{O}\left(\frac{n \log n}{\varepsilon^2}\right)$ space. Our paper shows a $\widetilde{O}(1/\varepsilon^5)$ round algorithm that instead uses $O(n \log(1/\varepsilon))$ space. Since $\varepsilon = \Omega(1/n)$ (or otherwise we obtain an exact maximum weight matching), our algorithm works in the semi-streaming model for all possible values of ε whereas Ahn and Guha [1] no longer works in semi-streaming when ε is small enough.

Our algorithm follows the general framework given in [6]. However, both our algorithm and our analysis require additional techniques. The main hurdle we must overcome is the fact that the weights may be much *larger* than the number of bidders and items. In that case, if we use the MCM algorithm trivially in this setting, the number of rounds can be very large, proportional to $\frac{w_{\max}}{\varepsilon^2}$ where $w_{\max}$ is the maximum weight of the edge. We avoid this problem in our algorithm, instead obtaining only poly $\log n$ and ε dependence in the number of rounds. Our main result in this section is the following (recall from Section 1).

▶ **Theorem 1** (Maximum Weight Bipartite Matching). *There exists an auction algorithm for maximum weight bipartite matching (MWM) that gives a $(1-\varepsilon)$-approximation for any $\varepsilon > 0$ and runs in $O\left(\frac{\log n \cdot \log(1/\varepsilon)}{\varepsilon^8}\right)$ rounds of communication (with high probability) and with $O\left(\log^2 n\right)$ bits per message. This algorithm can be implemented in the multi-round, semi-streaming model using $O\left(n \cdot \log n \cdot \log(1/\varepsilon)\right)$ space and $O\left(\frac{\log(1/\varepsilon)}{\varepsilon^7}\right)$ passes. This algorithm can be implemented in the work-depth model in $O\left(\frac{m \cdot \log(1/\varepsilon)}{\varepsilon^6}\right)$ work and $O\left(\frac{\log n \cdot \log(1/\varepsilon)}{\varepsilon^8}\right)$ depth. Finally, our algorithm can be implemented in the MPC model using $O(\log(1/\varepsilon)/\varepsilon^7)$ rounds, $O(n)$ space per machine, and $O\left(\frac{m \log(1/\varepsilon) \log n}{\varepsilon}\right)$ total space.*

Before we give our algorithm, we give some notation used in this section.

Notation. The input bipartite graphs is represented by $G = (L \cup R, E)$ where L is the set of bidders and R is the set of items. Let $N(v)$ denote the neighbors of node $v \in L \cup R$. We use the notation $i \in L$ to denote bidders and $j \in R$ to denote items. For a bidder $i \in L$, the *valuation* of i for items in R is defined as the function $v_i : R \to \mathbb{Z}_{\geq 0}$ where the function outputs a non-negative integer. If $v_i(j) > 0$, for any $j \in R$, then $j \in N(i)$. Each bidder can match to at most one item. We denote the bidder item pair by (i, a_i) where a_i is the matched item and $a_i = \perp$ if i is not matched to any item. For any agent i where $a_i \neq \perp$, the *utility* of a bidder i given its matched item a_i is $u_i \triangleq v_i(a_i) - p_{a_i}$ where p_{a_i} is the current price of item a_i. For an agent i where $a_i = \perp$, the utility of agent i is 0. We denote an optimum matching by OPT. We use the notation $i \in \text{OPT}$ to denote a bidder who is matched in OPT and o_i to denote the item matched to bidder i in OPT.

Input Specifications. In this section, we assume all weights are poly(n) where $n = |L| + |R|$. We additionally assume the following characteristics about our inputs because we can perform a simple pre-processing of our graph to satisfy these specifications. Provided an input graph $G = (L \cup R, E)$ with weights $v_i(j)$ for every edge $(i, j) \in E$, we find the maximum weight among all the weights of the edges, $w_{\max} = \max_{(i,j)\in E} (v_i(j))$. We rescale the weights of all the edges by $\frac{1}{w_{\max}}$ and remove all edges with rescaled weight $< \varepsilon^{\lceil \log_\varepsilon m \rceil + 1}$. This upper bound of $\varepsilon^{\lceil \log_\varepsilon m \rceil + 1}$ is crucial in our analysis.

In other words, we create a new graph $G' = (L \cup R, E')$ with the same set of bidders L and items R. We associate the new weight functions v'_i with each bidder $i \in L$ where $(i, j) \in E'$ if $v_i(j) \geq w_{\max} \cdot \varepsilon^{\lceil \log_\varepsilon m \rceil + 1}$ and $v'_i(j) = v_i(j)/w_{\max}$ for each $(i, j) \in E'$. Provided that finding the maximum weight edge can be done in $O(1)$ rounds in the blackboard distributed and MPC models, $O(1)$ passes in the streaming model, and $O(n + m)$ work and $O(1)$ depth in the parallel model, we assume the input to our algorithms is G' (instead of the original graph G). In other words, we assume all inputs $G = (V, E)$ to our algorithm have scaled edge weights and $v_i(j)$ for $i \in L, j \in R$ are functions that return the scaled edge weights in the rest of this section.

3.1 Detailed Algorithm

We now present our auction algorithm for maximum weighted bipartite matching in Algorithm 1. The algorithm works as follows. Recall that we also assume the input to our algorithm is the scaled graph. This means that the maximum weight of the scaled edges is 1 and there exists at least one edge with weight 1; hence, the maximum weight matching will have value at least 1. We also initialize the tuples that keep track of matched items. Initially, no items are assigned to bidders (Algorithm 1) and the prices of all items are set to 0 (Algorithm 1).

We perform $\lceil \frac{\log^2(N)}{\varepsilon^4} \rceil$ phases of bidding (Algorithm 1). In each phase, we form the *demand set* D_i of each unmatched bidder i. The demand set is defined to be the set of items with non-zero utility which have *approximately* the maximum utility value for bidder i (Algorithm 1). This procedure is different from both MCM and MCbM (where no slack is needed in creating the demand set) but we see in the analysis that we require this slack in the maximum utility value to ensure that enough progress is made in each round. Then, we create the induced subgraph consisting of all unmatched bidders and their demand sets (Algorithm 1). We find an arbitrary maximal matching in this created subgraph (Algorithm 1) by first finding the maximal matching in order of decreasing buckets (from highest – bucket with the largest weights – to lowest). This means that we call our maximal matching algorithm $O(\log(N))$ times first on the induced subgraph consisting of the highest bucket, removing the matches, and then on the induced subgraph of the remaining edges plus the next highest bucket, and so on. We use the folklore distributed maximal matching algorithm where in each round, a bidder uniformly-at-random picks a neighbor to match; this algorithm is also used in [11] for the maximal matching step. This simple algorithm terminates in $O(\log n)$ rounds with high probability using $O(\log n)$ communication complexity. Such randomization is necessary to obtain $O(\log n)$ rounds using $O(\log n)$ communication complexity.

We rematch items according to the new matching (Algorithm 1). We then increase the price of each rematched item. The price increase depends on the weight of the matched edge to the item; higher weight matched edges have larger increases in price than smaller weight edges. Specifically, the price is increased by $\varepsilon \cdot v_i(a_i)$ where $v_i(a_i)$ is the weight of the newly matched edge between i and a_i (Algorithm 1). The intuition behind this price increase is that we want to increase the price proportional to the weight gained from the matching since

Algorithm 1 Auction Algorithm for Maximum Weighted Bipartite Matching.

```
Input: A scaled graph $G = (L \cup R, E)$, parameter $0 < \varepsilon < 1$, and the scaling factor $w_{\max}$.
Output: An $(1-6\varepsilon)$-approximate maximum weight bipartite matching.
For each bidder $i \in L$, set $(i, a_i)$ to $a_i = \perp$.
For each item $j \in R$, set $p_j = 0$.
for $d = 1, \dots, \lceil \frac{\log^2(N)}{\varepsilon^4} \rceil$ do
    for each unmatched bidder $i \in L$ do
        Let $U_i \triangleq \max_{j \in N(i), v_i(j) - p_j > 0} (v_i(j) - p_j)$.
        Let $D_i \triangleq \{j \in R \mid p_j < v_i(j), v_i(j) - p_j \geq U_i - \varepsilon \cdot v_i(j)\}$.
    Create the subgraph $G_d$ as the subgraph consisting of $\left(\bigcup_{i \in L} D_i\right) \cup L$ and all edges.
    Find any arbitrary maximal matching $M_d$ of $G_d$ in order of highest bucket to lowest.
    for $(i, j) \in M_d$ do
        match $j$ to $i$ by setting $a_i = j$ and $a_{i'} = \perp$ for the previous owner $i'$ of $j$.
        Increase the price of $j$ to $p_j \leftarrow p_j + \varepsilon \cdot v_i(j)$.
    Let $M'$ be the matched edges in this current iteration.
Return the matching $M = \arg\max_{M'} \left(w_{\max} \cdot \sum_{i \in L} v_i(a_i)\right)$ as the approximate maximum
weight matching and $(i, a_i) \in M$ as the matched edges.
```

the price increase takes away from the overall utility of our matching. If not much weight is gained from the matching, then the price should not increase by much; otherwise, if a large amount of weight is gained from the matching, then we can afford to increase the price by a larger amount. We see later on in our analysis that this allows us to bucket the items according to their matched edge weight into $\lceil \log_{(1/\varepsilon)} m \rceil$ buckets. Such bucketing is useful in ensuring that we have sufficiently many happy bidders with a sufficiently large total matched weight. Finally, we return all matched items and bidders as our approximate matching and the sum of the weights of the matched items as the approximate weight. Obtaining the maximum weight of the matching in the original, unscaled graph is easy. We multiply the edge weights by $w_{\max}$ and the sum of these weights is the total weight of our approximate matching (Algorithm 1).

3.2 Analysis

In this section, we prove the approximation factor and round complexity of our algorithm. We use the same definition of happy that is defined in [6].

▶ **Definition 3** (ε-Happy [6]). *A bidder i is ε-happy if $u_i \geq v_i(j) - p_j - \varepsilon$ for every $j \in R$.*

▶ **Definition 4** (Unhappy). *A bidder i is **unhappy** at the end of round d if they are unmatched and their demand set is non-empty.*

Note that a happy bidder may never be unhappy and vice versa. For this definition, we assume that the demand set of a bidder can be computed at any point in time (not only when the Algorithm computers it).

Approach. The main challenge we face in our MWM analysis is that it is no longer sufficient to just show at least $(1-\varepsilon)$-fraction of bidders in OPT are happy in order to obtain the desired approximation. Consider this simple example. Suppose a given instance has an optimum solution OPT with six matched bidders where one bidder is matched to an item via a weight-1 edge. It also has five additional bidders matched to items via weight-$\frac{1}{\sqrt{n}}$ edges. Suppose we set $\varepsilon = 1/6$ to be a constant. Then, requiring 5/6-fraction of the bidders in OPT to be happy is not sufficient to get a 5/6-factor approximation. Suppose the five bidders matched with edges of weight $\frac{1}{\sqrt{n}}$ are the happy bidders. This is sufficient to satisfy the condition that 5/6-fraction of the bidders in OPT are happy. However, the total combined weight of the matching in this case is $\frac{5}{\sqrt{n}}$ while the weight of the optimum matching is $\left(1 + \frac{5}{\sqrt{n}}\right)$. The returned matching is then a $\frac{5}{\sqrt{n}}$-approximate MWM and for large n, this is much less than the desired 5/6-factor approximation.

Instead, we require a specific fraction of the total *weight* of the optimum solution, W_{OPT}, to be matched in our returned matching. We ensure this new requirement by considering two types of unhappy bidders. Type 1 unhappy bidders are bidders who are unhappy in round $k-1$ and remain unmatched in round k. Type 2 unhappy bidders are bidders who are unhappy in round $k-1$ and become matched in round k. We show that there exists a round where the following two conditions are satisfied:

1. We bucket the bidders in OPT according to the weight of their matched edge in OPT such that bidders matched with similar weight edges are in the same bucket; there exists a round where at most (ε^2)-fraction of the bidders in each bucket are Type 1 unhappy.
2. We charge the weight a Type 2 unhappy bidder obtains in round k to the bidder in round $k-1$; there exists a round $k-1$ where a total of at most $\varepsilon \cdot W_{\text{OPT}}$ weight is charged to Type 2 unhappy bidders.

Simultaneously satisfying both of the above conditions is enough to obtain our desired approximation. The rest of this section is devoted to showing our precise analysis using the above approach.

Detailed Analysis. Recall that we defined the utility of agent i to be the value of the item matched to her minus its price $u_i = v_i(a_i) - p_{a_i}$. In this section, we use the definition of ε-happy from Definition 3.

A similar observation to the observation made in [6] about the happiness of matched bidders can also be made in our case; however, since we are dealing with edge weights, we need to be careful to increment our prices in terms of the newly matched edge weight. In other words, two different bidders could be ε_1-happy and ε_2-happy after incrementing the price of their respective items by ε_1 and ε_2 where $\varepsilon_1 \neq \varepsilon_2$; the incremented prices ε_1 and ε_2 depend on the matched edge weights of the items assigned to the bidders. We prove the correct happiness guarantees given by our algorithm below.

▶ **Observation 5.** *At the end of every round, matched bidder i with matched edge (i, a_i) where a_i is priced at p_{a_i} are $(2\varepsilon \cdot v_i(a_i))$-happy. At the end of every round, unmatched bidders with empty demand sets D_i are ε-happy.*

For the weighted case, we need to consider what we call *weight buckets*. We define these weight buckets with respect to the optimum matching OPT. Recall our notation where $i \in \text{OPT}$ is a bidder who is matched in OPT and o_i is the matched item of the bidder in OPT. Bidder i is in the b-th weight bucket if $\varepsilon^{b-1} \leq v_i(o_i) < \varepsilon^{b-2}$.

▶ **Observation 6.** *All bidders $i \in OPT$ in bucket b satisfy $\varepsilon^{b-1} \leq v_i(o_i) < \varepsilon^{b-2}$.*

We now show that if a certain number of bidders in OPT are happy in our matching, then we obtain a matching with sufficiently large enough weight. However, our guarantee is somewhat more intricate than the guarantee provided in [6]. We show that in $\lceil \frac{\log^2(N)}{\varepsilon^4} \rceil$

rounds, there exists one round d where a set of sufficient conditions are satisfied to obtain our approximation guarantee. To do this, we introduce two types of unhappy bidders. Specifically, Type 1 and Type 2 unhappy bidders.

Each *unhappy* bidder results in some loss of total matched weight. However, at the end of round $k-1$ it is difficult to determine the exact amount of weight lost to unhappy bidders. Thus, in our analysis, we determine the amount of weight lost to unhappy bidders at the end of round $k-1$ in round k. The way that we determine the weight lost in round $k-1$ is by *retroactively* categorizing an unhappy bidder in round $k-1$ as a Type 1 or Type 2 unhappy bidder *depending on what happens in round* k. Thus, for our analysis, we categorize the bidders into categories of unhappy bidders for the *previous* round.

A ***Type* 1** unhappy bidder in round $k-1$ is a bidder i that remains unmatched at the end of round k. In other words, a Type 1 unhappy bidder was unhappy in round $k-1$ and either remains unhappy in round k or becomes happy because it does not have any demand items anymore (and remains unmatched). A ***Type* 2** unhappy bidder i in round $k-1$ is a bidder who was unhappy in round $k-1$ but is matched to an item in round k. Thus, a Type 2 unhappy bidder i in round $k-1$ becomes happy in round k because a new item is matched to i. Both types of bidders are crucial to our analysis given in the proof of Lemma 7 since they contribute differently to the potential amount of value that could be matched by our algorithm.

In the following lemma, let OPT be the optimum matching in graph G and $W_{\text{OPT}} = \sum_{i\in\text{OPT}} v_i(o_i)$. Let B_b be the set of bidders $i \in \text{OPT}$ in weight bucket b. If a Type 2 unhappy bidder i gets matched to a_i in round k, we say the weight $v_i(a_i)$ is ***charged*** to bidder i in round $k-1$. We denote this charged weight as $c_i(a_i)$ when performing calculations for round $k-1$.

▶ **Lemma 7.** *Provided $G = (L \cup R, E)$ and an optimum weighted matching OPT with weight $W_{OPT} = \sum_{i\in OPT} v_i(o_i)$, if in some round d of Algorithm 1 of Algorithm 1 both of the following are satisfied,*

1. *at most $\varepsilon^2 \cdot |B_b|$ of the bidders in each bucket b are Type 1 unhappy and*
2. *at most $\varepsilon \cdot W_{OPT}$ weight is charged to Type 2 unhappy bidders,*

then the matching in G has weight at least $(1-6\varepsilon)\cdot W_{OPT}$.

Proof. In such an iteration r, let Happy denote the set of all happy bidders. For any bidder $i \in \text{Happy} \cap \text{OPT}$, by Definition 3 and Observation 5, $u_i \geq v_i(o_i) - p_{o_i} - 2\varepsilon \cdot v_i(a_i)$ where o_i is the item matched to i in OPT and a_i is the item matched to i from our matching.

Before we go to the core of our analysis, we first make the observation that we can, in general, disregard prices of the items in our analysis. Let M be our matching. The sum of the utility of every matched bidder in our matching can be upper and lower bounded by the following expression:

$$\sum_{i\in M} (v_i(a_i) - p_{a_i}) \geq \sum_{i\in\text{OPT}\cap\text{Happy}} u_i \geq \sum_{i\in\text{OPT}\cap\text{Happy}} (v_i(o_i) - p_{o_i} - 2\varepsilon \cdot v_i(a_i)).$$

As in the maximum cardinality matching case, all items with non-zero price are matched to a bidder. We can then simplify the above expression to give

$$\sum_{i\in M} v_i(a_i) - \sum_{j\in R} p_j \geq \sum_{i\in \mathrm{OPT}\cap\textsc{Happy}} v_i(o_i) - \sum_{i\in \mathrm{OPT}\cap\textsc{Happy}} p_{o_i} - \sum_{i\in \mathrm{OPT}\cap\textsc{Happy}} 2\varepsilon\cdot v_i(a_i) \tag{1}$$

$$\sum_{i\in M\setminus(\mathrm{OPT}\cap\textsc{Happy})} v_i(a_i) + \sum_{i\in \mathrm{OPT}\cap\textsc{Happy}} (1+2\varepsilon)v_i(a_i) - \sum_{j\notin\{o_i \mid i\in \mathrm{OPT}\cap\textsc{Happy}\}} p_j \geq \sum_{i\in \mathrm{OPT}\cap\textsc{Happy}} v_i(o_i) \tag{2}$$

$$\sum_{i\in M}(1+2\varepsilon)v_i(a_i) - \sum_{j\notin\{o_i \mid i\in \mathrm{OPT}\cap\textsc{Happy}\}} p_j \geq \sum_{i\in \mathrm{OPT}\cap\textsc{Happy}} v_i(o_i). \tag{3}$$

Equation (1) follows from the fact that all non-zero priced items are matched. Equation (2) follows from separating $\mathrm{OPT}\cap\textsc{Happy}$ from the left hand side and moving the summation of the $2\varepsilon\cdot v_i(a_i)$ values over $\mathrm{OPT}\cap\textsc{Happy}$ from the right hand side to the left hand side. Finally, Equation (3) follows because $\sum_{i\in M}(1+2\varepsilon)v_i(a_i)$ upper bounds the left hand side expression for $\sum_{i\in M\setminus(\mathrm{OPT}\cap\textsc{Happy})} v_i(a_i) + \sum_{i\in\mathrm{OPT}\cap\textsc{Happy}}(1+2\varepsilon)v_i(a_i)$.

Let $\textsc{Unhappy}_1$ denote the set of Type 1 unhappy bidders and $\textsc{Unhappy}_2$ denote the set of Type 2 unhappy bidders. We let $c_i(a_i)$ be the weight charged to bidder i in $\textsc{Unhappy}_2$ in the next round. Recall that each bidder in $\textsc{Unhappy}_2$ is matched in the next round.

For each bucket, b, we can show the following using our assumption that at most $\varepsilon^2\cdot|B_b|$ of the bidders in bucket b are Type 1 unhappy,

$$\sum_{i\in B_b\cap\textsc{Happy}} v_i(o_i) \geq \sum_{i\in B_b\setminus\textsc{Unhappy}_2} v_i(o_i) - \varepsilon^2\cdot\varepsilon^{b-2}\cdot|B_b| \tag{4}$$

$$\geq \sum_{i\in B_b\setminus\textsc{Unhappy}_2} v_i(o_i) - \varepsilon\cdot\varepsilon^{b-1}\cdot|B_b| \tag{5}$$

$$\geq \sum_{i\in B_b\setminus\textsc{Unhappy}_2} v_i(o_i) - \sum_{i\in B_b}\varepsilon\cdot v_i(o_i). \tag{6}$$

Equation (4) shows that one can lower bound the sum of the optimum values of all happy bidders in bucket b by the sum of the optimum values of all bidders who are not Type-2 unhappy minus some factor. First, $\sum_{i\in B_b\setminus UNHAPPY_2} v_i(o_i)$ is the sum of the optimum values of all bidders in bucket b *except* for the Type-2 unhappy bidders. Now, we need to subtract the maximum sum of values given to the Type-1 unhappy bidders. We know that bucket b has at most $\varepsilon^2\cdot|B_b|$ Type-1 unhappy bidders. Each of these bidders could be assigned an optimum item with value at most ε^{b-2} (by Observation 6). Thus, the maximum value lost to Type-1 unhappy bidders is $\varepsilon^2\cdot\varepsilon^{b-2}\cdot|B_b|$, leading to Equation (4). Thus, the maximum value of weight lost to all Type 1 unhappy bidders in bucket b is $\varepsilon^2\cdot\varepsilon^{b-2}\cdot|B_b|$. Then, Equation (6) follows because $v_i(o_i)\geq\varepsilon^{b-1}$ for all $i\in B_b$. This means that $\sum_{i\in B_b} v_i(o_i)\geq\varepsilon^{b-1}\cdot|B_b|$.

Summing Equation (6) over all buckets b we obtain

$$\sum_{i\in \mathrm{OPT}\cap\textsc{Happy}} v_i(o_i) \geq \sum_{i\in \mathrm{OPT}\setminus\textsc{Unhappy}_2} v_i(o_i) - \sum_{i\in\mathrm{OPT}}\varepsilon\cdot v_i(o_i). \tag{7}$$

We now substitute our expression obtained in Equation (7) into Equation (3),

$$\sum_{i\in M}(1+2\varepsilon)v_i(a_i) - \sum_{j\notin\{o_i \mid i\in \mathrm{OPT}\cap\textsc{Happy}\}} p_j \geq \sum_{i\in \mathrm{OPT}\setminus\textsc{Unhappy}_2} v_i(o_i) - \sum_{i\in\mathrm{OPT}}\varepsilon\cdot v_i(o_i). \tag{8}$$

The last thing that we need to show is a bound on the weight lost due to bidders in $\text{OPT} \cap \textsc{Unhappy}_2$. We now consider our second assumption which states that at most $\varepsilon \cdot W_{\text{OPT}}$ weight is charged to Type 2 unhappy bidders. Since all bidders $i \in \textsc{Unhappy}_2$ become happy in the next round, we can bound the weights charged to the Type 2 unhappy bidders using Observation 5 by

$$\sum_{i \in \text{OPT} \cap \textsc{Unhappy}_2} c_i(a_i) \geq \sum_{i \in \text{OPT} \cap \textsc{Unhappy}_2} \left(v_i(o_i) - p_{o_i} - 2\varepsilon \cdot c_i(a_i)\right). \tag{9}$$

Note first that $\sum_{j \notin \{o_i \mid i \in \text{OPT} \cap \textsc{Happy}\}} p_j \geq \sum_{i \in \text{OPT} \cap \textsc{Unhappy}_2} p_{o_i}$ since $\text{OPT} \setminus (\text{OPT} \cap \textsc{Happy})$ includes $\text{OPT} \cap \textsc{Unhappy}_2$ so we can remove the prices from these bounds in Equation (10). We add Equation (9) to Equation (8) and use our assumptions to obtain

$$\sum_{i \in M} (1+2\varepsilon) v_i(a_i) + \sum_{i \in \text{OPT} \cap \textsc{Unhappy}_2} c_i(a_i) \geq \sum_{i \in \text{OPT}} (1-\varepsilon) \cdot v_i(o_i) - \sum_{i \in \text{OPT} \cap \textsc{Unhappy}_2} 2\varepsilon \cdot c_i(a_i) \tag{10}$$

$$\sum_{i \in M} (1+2\varepsilon) v_i(a_i) \geq \sum_{i \in \text{OPT}} (1-\varepsilon) \cdot v_i(o_i) - \sum_{i \in \text{OPT} \cap \textsc{Unhappy}_2} (1+2\varepsilon) \cdot c_i(a_i) \tag{11}$$

$$\geq \left(\sum_{i \in \text{OPT}} (1-\varepsilon) \cdot v_i(o_i) \right) - (1+2\varepsilon) \cdot \varepsilon \cdot W_{\text{OPT}} \tag{12}$$

$$\sum_{i \in M} v_i(a_i) \geq \frac{(1-4\varepsilon)}{(1+2\varepsilon)} \cdot \sum_{i \in \text{OPT}} v_i(o_i) \tag{13}$$

$$\sum_{i \in M} v_i(a_i) \geq (1-6\varepsilon) W_{\text{OPT}}. \tag{14}$$

Equation (10) follows from summing $\sum_{i \in \text{OPT}} (1-\varepsilon) \cdot v_i(o_i) = \sum_{i \in \text{OPT} \setminus \textsc{Unhappy}_2} v_i(o_i) + \sum_{i \in \text{OPT} \cap \textsc{Unhappy}_2} v_i(o_i) - \sum_{i \in \text{OPT}} \varepsilon \cdot v_i(o_i) = \sum_{i \in \text{OPT}} (1-\varepsilon) \cdot v_i(o_i)$. Equation (11) follows from moving $\sum_{i \in \text{OPT} \cap \textsc{Unhappy}_2} c_i(a_i)$ to the right hand side. Equation (12) follows from substituting our assumption that $\sum_{i \in \text{OPT} \cap \textsc{Unhappy}_2} c_i(a_i) \leq \varepsilon \cdot W_{\text{OPT}}$. Equation (13) follows from simple manipulations and since $W_{\text{OPT}} = \sum_{i \in \text{OPT}} v_i(o_i)$. Finally, Equation (14) follows because $\frac{(1-4\varepsilon)}{(1+2\varepsilon)} \geq (1-6\varepsilon)$ for all $\varepsilon > 0$ and gives the desired approximation given in the lemma statement. ◀

We show that the conditions of Lemma 7 are satisfied for at least one round if the algorithm is run for at least $\lceil \frac{\log^2(N)}{\varepsilon^4} \rceil$ rounds. We prove this using potential functions similar to the potential functions used for MCM. We first bound the maximum value of these potential functions.

▶ **Lemma 8.** *Define the potential function $\Phi_{items} \triangleq \sum_{j \in R} p_j$. Then the upper bound for this potential is $\Phi_{items} \leq W_{OPT}$.*

Proof. We show that the potential function Φ_{items} is always upper bounded by W_{OPT} via a simple proof by contradiction. Suppose that $\Phi_{items} > W_{\text{OPT}}$, then, we show that the matching obtained by our algorithm has weight greater than W_{OPT}, a contradiction. For a bidder/item pair, (i, a_i), the weight of edge (i, a_i) is at least $p_{a_i} - 2\varepsilon \cdot v_i(a_i)$. Let p'_{a_i} be the price of a_i before the last reassignment of a_i to i. Furthermore, since i picked a_i, it must mean that $v_i(a_i) > p'_{a_i}$ since a_i would not be included in D_i otherwise. This means that the sum of the weights of all the matched edges is at least $\sum_{(i,a_i)} v_i(a_i) > \sum_{(i,a_i)} p'_{a_i} \geq \Phi_{items} > W_{\text{OPT}}$ by our assumption that $\Phi_{items} > W_{\text{OPT}}$. Thus, we obtain that we get a matching with greater weight than the optimum weight matching, a contradiction. ◀

▶ **Lemma 9.** *There exists a phase $d \leq \frac{\log^2(N)}{\varepsilon^4}$ wherein* both *of the following statements are satisfied:*

1. *At most $\varepsilon^2 \cdot |B_b|$ bidders in bucket b are Type 1 unhappy for all buckets b;*
2. *The set of all Type 2 unhappy bidders results in a loss of less than $\varepsilon \cdot W_{OPT}$ weight (in charged weight) where W_{OPT} is the optimum weight attainable by the matching.*

Recall that we assign each bidder to a weight bucket using the weight assigned to the bidder in OPT.

Using the above lemmas, we can prove our main theorem that our algorithm gives a $(1-7\varepsilon)$-approximate maximum weight bipartite matching in $O\left(\frac{\log^3(N)\cdot\log(n)}{\varepsilon^4}\right)$ distributed rounds using $O(\log n)$ communication complexity.

▶ **Theorem 10.** *Algorithm 1 returns a $(1-7\varepsilon)$-approximate maximum weight bipartite matching M in $O\left(\frac{\log^3(N)\cdot\log n}{\varepsilon^4}\right)$ rounds whp using $O(\log n)$ bits of communication per message in the broadcast model.*

Reducing the Round Complexity. We can use the following transformation from Gupta-Peng [18] to reduce the round complexity at an increase in the communication complexity. For completeness, we give the theorem for the transformation the full version of our paper.

▶ **Theorem 11.** *There exists a $(1-\varepsilon)$-approximate distributed algorithm for maximum weight bipartite matching that runs in either:*

- *$O\left(\log n \cdot \frac{\log(1/\varepsilon)}{\varepsilon^7}\right)$ rounds of communication using $O\left(\frac{\log^2 n}{\varepsilon}\right)$ bits of communication, or*
- *$O\left(\log n \cdot \frac{\log(1/\varepsilon)}{\varepsilon^8}\right)$ rounds of communication using $O\left(\log^2 n\right)$ bits of communication*

where we assume the maximum ratio between weights of edges in the input graph is $\text{poly}(n)$. In the blackboard model, this requires a total of $O\left(\frac{n\log^3 n\log(1/\varepsilon)}{\varepsilon^8}\right)$ bits of communication.

3.3 Semi-Streaming Implementation

The implementation of this algorithm in the semi-streaming model is very similar to the implementation of the MCM algorithm of Assadi et al. [6].

▶ **Lemma 12.** *Given a weighted graph $G = (V, E)$ as input in an arbitrary edge-insertion stream where all weights are at most $\text{poly}(n)$, there exists a semi-streaming algorithm which uses $O\left(\frac{\log^2(N)}{\varepsilon^4}\right)$ passes and $O\left(n \cdot \log n \cdot \log(1/\varepsilon)\right)$ space that computes a $(1-\varepsilon)$-approximate maximum weight bipartite matching for any $\varepsilon > 0$.*

Reducing the Number of Passes. We use the transformation of [18] as stated the full version of our paper to eliminate our dependence on n within our number of rounds. The transformation is as follows. For each instance of $(1+\varepsilon)$-MWM, we maintain the prices in our algorithm for each of the nodes involved in each of the copies of our algorithm. When an edge arrives in the stream, we first partition it into the relevant level of the appropriate copy of the structure.

▶ **Theorem 13.** *There exists a $(1-\varepsilon)$-approximate streaming algorithm for maximum weight bipartite matching that uses $O\left(\frac{\log(1/\varepsilon)}{\varepsilon^7}\right)$ passes in $O\left(n \cdot \log n \cdot \log(1/\varepsilon)\right)$ space.*

3.4 Shared-Memory Parallel Implementation

The implementation of this algorithm in the shared-memory work-depth model follows almost directly from our auction algorithm. We show the following lemma when directly implementing our auction algorithm.

▶ **Lemma 14.** *Given a weighted graph $G = (V, E)$ as input where all weights are at most* $\mathrm{poly}(n)$*, there exists a shared-memory parallel algorithm which uses* $O\left(\frac{m\log^2 n}{\varepsilon^4}\right)$ *work and* $O\left(\frac{\log^4 n}{\varepsilon^4}\right)$ *depth that computes a* $(1-\varepsilon)$*-approximate maximum weight bipartite matching for any* $\varepsilon > 0$*.*

Using the transformations, we can reduce the number of rounds for our shared-memory parallel algorithms.

▶ **Theorem 15.** *Given a weighted graph $G = (V, E)$ as input where all weights are at most* $\mathrm{poly}(n)$*, there exists a shared-memory parallel algorithm which uses* $O\left(\frac{m\log(1/\varepsilon)}{\varepsilon^6}\right)$ *work and* $O\left(\frac{\log(n)\cdot\log(1/\varepsilon)}{\varepsilon^8}\right)$ *depth that computes a* $(1-\varepsilon)$*-approximate maximum weight bipartite matching for any* $\varepsilon > 0$*.*

3.5 MPC Implementation

We implement our auction algorithm in the MPC model below.

▶ **Lemma 16.** *Given a weighted graph $G = (V, E)$ as input where all weights are at most* $\mathrm{poly}(n)$*, there exists a MPC algorithm using* $O\left(\frac{\log^2(N)\cdot\log\log n}{\varepsilon^4}\right)$ *rounds,* $O(n)$ *space per machine, and* $O\left(n\log(1/\varepsilon)+m\right)$ *total space that computes a* $(1-\varepsilon)$*-approximate maximum weight bipartite matching for any* $\varepsilon > 0$*.*

As before, we can improve the complexity of our MPC algorithm using the transformations the full version of our paper.

▶ **Theorem 17.** *Given a weighted graph $G = (V, E)$ as input where all weights are at most* $\mathrm{poly}(n)$*, there exists a MPC algorithm using* $O\left(\frac{\log(1/\varepsilon)\cdot\log\log n}{\varepsilon^7}\right)$ *rounds,* $O(n)$ *space per machine, and* $O\left(\frac{m\log(1/\varepsilon)\log n}{\varepsilon}\right)$ *total space that computes a* $(1-\varepsilon)$*-approximate maximum weight bipartite matching for any* $\varepsilon > 0$*.*

4 A $(1-\varepsilon)$-approximation Auction Algorithm for b-Matching

We show in this section that we also obtain an auction-based algorithm for MCBM by extending the auction-based algorithm of [6]. This algorithm also leads to better streaming algorithms for this problem. We use the techniques introduced in the auction-based MCM algorithm of Assadi, Liu, and Tarjan [6] as well as new techniques developed in this section to obtain a $(1-\varepsilon)$-approximation algorithm for bipartite maximum cardinality b-matching. The maximum cardinality b-matching problem is defined in Definition 18.

▶ **Definition 18** (Maximum Cardinality Bipartite b-Matching (MCBM))**.** *Given an undirected, unweighted, bipartite graph $G = (L \cup R, E)$ and a set of values $\{b_v \leq |R| \mid v \in L \cup R\}$, a* ***maximum cardinality*** b***-matching*** *(MCBM) finds a matching of maximum cardinality between vertices in L and R where each vertex $v \in L \cup R$ is matched to at most b_v other vertices.*

The key difference between our algorithm for b-matching and the MCM algorithm of [6] is that we have to account for when more than one item is assigned to each bidder in L; in fact, up to b_i items in R can be assigned to any bidder $i \in L$. This one to many relationship calls for a different algorithm and analysis. The crux of our algorithm in this section is to create b_i copies of each bidder i and b_j copies of each item j. Then, copies of items maintain their own prices and copies of bidders can each choose at most one item. We define some notation to describe these copies. Let C_i be the set of copies of bidder i and C_j be the set of copies of item j. Then, we denote each copy of i by $i^{(k)} \in C_i$ for $k \in [b_i]$ and each copy of j by $j^{(k)} \in C_j$ for $k \in [b_j]$. As before, we denote a bidder and their currently matched item by $\left(i^{(k)}, a_{i^{(k)}}\right)$.

In MCbM, we require that the set of all items chosen by different copies of the same bidder to include at most one copy of each item. In other words, we require if $j^{(k)} \in \bigcup_{i' \in C_i} a_{i'}$, then no other $j^{(l)} \in \bigcup_{i' \in C_i} a_{i'}$ for any $j^{(k)}, j^{(l)} \in C_j$ and $k \neq l$. This *almost* reduces to the problem of finding a maximum cardinality matching in a $\sum_{i \in L} b_i + \sum_{j \in R} b_j$ sized bipartite graph but *not quite*. Specifically, the main challenge we must handle is when multiple copies of the same bidder want to be matched to copies of the *same* item. In this case, we cannot match any of these bidder copies to copies of the same item and thus must somehow handle the case when there exist items of lower price but we cannot match them.

In addition to handling the above hard case, as before, the crux of our proof relies on a variant of the ε-happy definition and the definitions of appropriate potential functions.

Recall from the MCM algorithm of [6] that an ε-happy bidder has utility that is at least the utility gained from matching to any other item (up to an additive ε). Such a definition is insufficient in our setting since it may be the case that matching to a copy of an item that is already matched to a different copy of the same bidder results in lower cost. However, such a match is not helpful since any number of matches between copies of the same bidder and copies of the same item contributes a value of one to the cardinality of the eventual matching.

Our algorithm solves all of the above challenges and provides a $(1-\varepsilon)$-approximate MCbM in asymptotically the same number of rounds as the MCM algorithm of [6]. We describe our auction based algorithm for MCbM next and the precise pseudocode is given in Algorithm 2. Our algorithm uses the parameters defined in Table 2. We show the following results using our algorithm. We discuss semi-streaming implementations of our algorithm in Section 4.3. Let L be the half with fewer numbers of nodes.

▶ **Theorem 2** (Maximum Cardinality Bipartite b-Matching). *There exists an auction algorithm for maximum cardinality bipartite b-matching (MCbM) that gives a $(1-\varepsilon)$-approximation for any $\varepsilon > 0$ and runs in $O\left(\frac{\log n}{\varepsilon^2}\right)$ rounds of communication. This algorithm can be implemented in the multi-round, semi-streaming model using $O\left(\left(\sum_{i \in L} b_i + |R|\right)\log(1/\varepsilon)\right)$ space and $O\left(\frac{1}{\varepsilon^2}\right)$ passes. Our algorithm can be implemented in the shared-memory work-depth model in $O\left(\frac{\log^3 n}{\varepsilon^2}\right)$ depth and $O\left(\frac{m \log n}{\varepsilon^2}\right)$ total work.*

4.1 Algorithm Description

The algorithm works as follows. We assign to each bidder, i, b_i unmatched slots and the goal is to fill all slots (or as many as possible). For each bidder $i \in L$ and each item $j \in R$, we create b_i and b_j copies, respectively, and assign these copies to new sets L' and R', respectively (Algorithm 2). This step of the algorithm changes slightly in our streaming implementation. For each bidder and item with an edge between them $(i, j) \in E$, we create

Algorithm 2 Auction Algorithm for Bipartite b-Matching.

```
Input: Graph $G = (L \cup R, E)$ and parameter $0 < \varepsilon < 1$.
Output: An $(1 - \varepsilon)$-approximate maximum cardinality bipartite $b$-matching.
Create $L', R', E'$ and graph $G'$. (Defined in Table 2.)
For each $i' \in L'$, set $(i', \perp)$ where $a_{i'} = \perp$, $c_{i'} \leftarrow 0$.
For each $j' \in R'$, set $p_{j'} = 0$.
Set $M_{\max} \leftarrow \emptyset$.
for $d = 1, \dots, \lceil \frac{2}{\varepsilon^2} \rceil$ do
    For each unmatched bidder $i' \in L'$, find $D_{i'} \leftarrow \text{FindDemandSet}(G', i', c_{i'})$ [Algorithm 3].
    Create $G'_d$.
    Find any arbitrary non-duplicate maximal matching $\widehat{M_d}$ of $G_d$.
    for $(i', j') \in \widehat{M_d}$ do
        Set $a_{i'} = j'$ and $a_{i_{prev}} = \perp$ for the previous owner $i_{prev}$ of $j'$.
        Increase $p_{j'} \leftarrow p_{j'} + \varepsilon$.
    For each $i' \in L'$ with $D_{i'} \neq \emptyset$ and $a_{i'} = \perp$, increase $c_{i'} \leftarrow c_{i'} + \varepsilon$.
    Using $M'_d$ compute $M_d$ where for each $(i', j') \in M'_d$, add $(i, j)$ to $M_d$ if $(i, j) \notin M_d$.
    If $|M_d| > |M_{\max}|$, $M_{\max} \leftarrow M_d$.
Return $M_{\max}$.
```

Algorithm 3 $\text{FindDemandSet}(G' = (L' \cup R', E'), i', c_{i'})$.

```
Let $N'(i') = \{j' \in R' \mid j^{(l)} \neq a_{i^{(k)}} \forall i^{(k)} \in C_i, \forall j^{(l)} \in C_j \wedge p_{j'} \geq c_{i'} \forall j' \in C_j\}$.
$D_{i'} \leftarrow \arg\min_{j' \in N'(i'), p_{j'} < 1} (p_{j'})$.
Return $D_{i'}$.
```

a biclique between C_i and C_j; the edges of all created bicliques is the set of edges E'. The graph $G' = (L' \cup R', E')$ is created as the graph consisting of nodes in $L' \cup R'$ and edges in E'. As before, we initialize each bidder's assigned item to $\perp$ (Algorithm 2). Then, we set the price for each copy in R' to 0 (Algorithm 2).

In our MCBM algorithm, we additionally set a price cutoff for each bidder $c_{i'}$ initialized to 0 (Algorithm 2). Such a cutoff helps us to prevent bidding on lower price items previously not bid on because they were matched to another copy of the same bidder. More details on how the cutoff prevents bidders from bidding against themselves can be found in the proof of Lemma 25. We maintain the maximum cardinality matching we have seen in $M_{\max}$ (Algorithm 2). We perform $\lceil \frac{2}{\varepsilon^2} \rceil$ rounds of assigning items to bidders (Algorithm 2). For each round, we first find the demand set for each unmatched bidder $i' \in L'$ using Algorithm 3 (Algorithm 2). The demand set is defined with respect to the cutoff price $c_{i'}$ and the set of items assigned to other copies of bidder i. The demand set considers all items $j' \in R'$ that are neighbors of i' where no copy of j, $j^{(k)} \in C_j$, is assigned to any copies of i and $p_{j'} \geq c_{i'}$ (Algorithm 3, Algorithm 3). From this set of neighbors, the returned demand set is the set of item copies with the minimum price in $N'(i')$ (Algorithm 3).

Using the induced subgraph of $\left(\bigcup_{i' \in L'} D_{i'}\right) \cup L'$ (Algorithm 2), we greedily find a maximal matching while avoiding assigning copies of the same item to copies of the same bidder (Algorithm 2). We call such a maximal matching that does not assign more than one copy of the same item to copies of the same bidder to be a ***non-duplicate*** maximal matching. This greedy matching prioritizes the unmatched items by first matching the unmatched items and then matching the matched items. We can perform a greedy matching by matching an edge if the item is unmatched and no copies of the bidder it will match to is matched to another copy of the item. For each newly matched item (Algorithm 2), we rematch the item to the newly matched bidder (Algorithm 2). We increase the price of the newly matched item (Algorithm 2). For each remaining unmatched bidder, we increase the cutoff price by ε (Algorithm 2).

We compute the corresponding matching in the original graph using M'_d (Algorithm 2) by including one edge (i, j) in the matching if and only if there exists at least one bidder copy $i' \in C_i$ matched to at least one copy of the item $j' \in C_j$. Finally, we return the maximum cardinality $M_{\max}$ matching from all iterations as our $(1-\varepsilon)$-approximate maximum cardinality b-matching (Algorithm 2).

4.2 Analysis

In this section, we analyze the approximation error of our algorithm and prove that it provides a $(1 - \varepsilon)$-approximate maximum cardinality b-matching.

Approach. We first provide an intuitive explanation of the approach we take to perform our analysis and then we give our precise analysis. Here, we describe both the challenges in performing the analysis and explain our choice of certain methods in the algorithm to facilitate our analysis. We especially highlight the parts of our algorithm and analysis that differ from the original MCM algorithm of [6]. First, in order to show the approximation factor of our algorithm, we require that the utility obtained by a large number of matched bidders from our algorithm is greater than the corresponding utility from switching to the optimum items in the optimum matching. For b-matching, any combination of matched items and bidder copies satisfy this criteria. Furthermore, matching multiple item copies of the same item to bidder copies of the same bidder does not increase the utility of the bidder. Thus, we look at matchings where at most one copy of each bidder is matched to at most one copy of each item. Recall our definition of ε-happy given in Definition 3 and we let Happy be the set of bidders satisfying that definition.

For b-matching, each bidder i is matched to a set of at most b_i items. Let $(i, O_i) \in \text{OPT}$ denote the set of items $O_i \subseteq R$ matched to bidder i in OPT. Recall [6] that the proof requires $u_i \geq 1 - p_{o_i} - \varepsilon$ for every bidder $i \in \textsc{Happy} \cap \text{OPT}$ to show that $\sum_{i \in L} u_i \geq \sum_{i \in \textsc{Happy} \cap \text{OPT}} 1 - p_{o_i} - \varepsilon$. Using our bidder copies, C_i, the crux of our analysis proof is to show that for every $(i, O_i) \in \text{OPT}$, we can *assign* the items in O_i to the set of happy bidder copies in C_i such that each happy bidder copy receives a unique item, denoted by $r_{i'}$, and $c_{i'} \leq p_{\min, r_{i'}}$ where $p_{\min, r_{i'}}$ is the price of the minimum priced copy of $r_{i'}$. Using this assignment, we are able to show once again that $\sum_{i' \in L'} u_{i'} \geq \sum_{i' \in \textsc{Happy} \cap \text{OPT}} 1 - p_{\min, r_{i'}} - \varepsilon$. This requires a precise definition of $\textsc{Happy} \cap \text{OPT}$. Let $S_i \subseteq C_i$ be the set of all happy bidders in C_i. Recall that the optimum solution gives a matching between a bidder $i \in L$ and potentially multiple items in R; we turn this matching into an optimum matching in G'. If $|S_i| \leq |O_i|$, then all happy copies in S_i are in OPT; otherwise, we pick an arbitrary set of $|O_i|$ happy bidder copies in S_i to be in OPT. Then, the summation is determined based on this set of happy bidder copies in $\textsc{Happy} \cap \text{OPT}$.

Once we have shown this, the only other remaining part of the proof is to show that in the $\lceil \frac{2}{\varepsilon^2} \rceil$ rounds that we run the algorithm the potential increases by ε for every unhappy bidder in OPT for each round that the bidder is unhappy. As in the case for MCM, the price of an item increases whenever it becomes re-matched. Hence, Π_{items} increases by ε each time a bidder who was happy becomes unhappy. To ensure that $\Pi_{bidders}$ increases by ε

for each bidder who was unhappy and remains unhappy, we set a ***cutoff*** price that increases by ε for each round where a bidder remains unhappy. Thus, this cutoff guarantees that $\Pi_{bidders}$ increases by ε each time.

Detailed Analysis. Now we show our detailed analysis that formalizes our approach described above. We first show that our algorithm maintains both Invariant 20 and Invariant 21. We also show our algorithm obeys the following invariant.

▶ **Invariant 19.** *The set of matched items of all copies of any bidder $i \in L$ contains at most one copy of each item. In other words, $\left|\bigcup_{i' \in C_i} a_{i'} \cap C_j\right| \leq 1$ for all $j \in R$.*

We restate two invariants used in [6] below. We prove that our Algorithm 2 also maintains these two invariants.

▶ **Invariant 20** (Non-Zero Price Matched [6]). *Any item j with positive price $p_j > 0$ is matched.*

▶ **Invariant 21** (Maximum Utility [6]). *The total utility of all bidders is at most the cardinality of the matching minus the total price of the items.*

▶ **Lemma 22.** *Algorithm 2 maintains Invariant 19, Invariant 20, and Invariant 21.*

We follow the style of analysis outlined in [6] by defining appropriate definitions of ε-happy and appropriate potential functions Π_{items} and $\Pi_{bidders}$. In the case of b-matching, we modify the definition of ε-happy in this setting to be the following.

▶ **Definition 23** ((ε, c)-Happy)**.** *A bidder $i' \in L'$ is (ε, c)-happy (at the end of a round) if $u_{i'} \geq 1 - p_{j'} - \varepsilon$ for all neighbors in the set $N'(i')$ where $N'(i')$ is as defined in Algorithm 3 of Algorithm 3 (i.e. contains all neighboring items j' where $p_{j'} \geq c_{i'}$ and no copy of the neighbor is matched to another copy of i').*

At the end of each round, it is easy to show that all matched i' and i' whose demand sets $D_{i'}$ are empty are $(\varepsilon, c_{i'})$-happy.

▶ **Lemma 24.** *At the end of any round, if bidder i' is matched or if their demand set is empty, $D_{i'} = \emptyset$, then i' is $(\varepsilon, c_{i'})$-happy.*

In addition to the new definition of happy, we require another crucial observation before we prove our approximation guarantee. Specifically, we show that for any set of bidder copies C_i and any set of $|C_i|$ items $I \subseteq R$, Lemma 24 is sufficient to imply there exists at least one assignment of items in I to happy bidders in S_i such that each item is assigned to at most one bidder and each happy bidder is assigned at least one item where the minimum price of the item is at least the cutoff price of the bidder.

▶ **Lemma 25.** *For a set of bidder copies C_i and any set $I \subseteq R$ of $|C_i|$ items where $(i, j) \in E$ for all items $j \in I$, there exists at least one assignment of items in I to bidders in C_i, where we denote the item assigned to copy i' by $r_{i'}$, that satisfy the following conditions:*

1. *The assignment is a one-to-one mapping between bidders in C_i and items in I.*
2. *Any item j matched to i' is assigned to i'.*
3. *Let $r^*_{i'}$ be the lowest cost copy of item $r_{i'}$, $r^*_{i'} = \arg\min_{j' \in C_{r_{i'}}} (p_{j'})$; then $p_{r^*_{i'}} \geq c_{i'}$ for all $i' \in C_i$.*

We now perform the approximation analysis. Suppose as in the case of MCM, we have at least $(1-\varepsilon)|\mathrm{OPT}|$ happy bidders in OPT (i.e. $|\textsc{Happy} \cap \mathrm{OPT}| \geq (1-\varepsilon)|\mathrm{OPT}|$), then we show that we can obtain a $(1-\varepsilon)$-approximate MCbM. Let OPT be an optimum MCbM matching and $|\mathrm{OPT}|$ be the cardinality of this matching.

▶ **Lemma 26.** *Assuming $|\textsc{Happy} \cap OPT| \geq (1-\varepsilon)|OPT|$, then we obtain a $(1-2\varepsilon)$-approximate MCbM.*

The potential argument proof is almost identical to that for MCM provided our use of $c_{i'}$. Specifically, as in the case for MCM, we use the same potential functions and using these potential functions, we show that our algorithm terminates in $O\left(\frac{1}{\varepsilon^2}\right)$ rounds. The key difference between our proof and the proof of MCM explained in [6] is our definition of $\Pi_{bidders}$ which is precisely defined in the proof of Lemma 27 below.

▶ **Lemma 27.** *In $\lceil \frac{2}{\varepsilon^2} \rceil$ rounds, there exists at least one round where $|OPT \cap \textsc{Happy}| \geq (1-\varepsilon)|OPT|$.*

Using the above lemmas, we can prove the round complexity of Theorem 2 to be $O\left(\frac{1}{\varepsilon^2}\right)$ by Lemma 26 and Lemma 27.

▶ **Theorem 28.** *There exists an auction algorithm for maximum cardinality bipartite b-matching (MCbM) that gives a $(1-\varepsilon)$-approximation for any $\varepsilon > 0$ and runs in $O\left(\frac{\log n}{\varepsilon^2}\right)$ rounds of communication using $O(b \log n)$ bits per message in the blackboard distributed model. In total, the number of bits used by the algorithm is $O\left(\frac{nb\log^2 n}{\varepsilon^2}\right)$.*

4.3 Semi-Streaming Implementation

We now show an implementation of our algorithm to the semi-streaming setting and show the following lemma which proves the semi-streaming portion of our result in Theorem 2. We are guaranteed $\varepsilon \geq \frac{1}{2n^2}$; otherwise, an exact matching is found. In order to show the space bounds, we use an additional lemma below that upper and lower bounds the prices of any copies of the same item in R'.

▶ **Lemma 29.** *For any $j \in R$, let $j_{\min}$ be the minimum priced copy in C_j and $j_{\max}$ be the maximum priced copy in C_j. Then, $p_{j_{\max}} - p_{j_{\min}} \leq \varepsilon$.*

Using the above, we prove our desired bounds on the number of passes and the space used.

▶ **Theorem 30.** *There exists a semi-streaming algorithm for maximum cardinality bipartite b-matching that uses $O\left(\frac{1}{\varepsilon^2}\right)$ rounds and $\widetilde{O}\left(\left(\sum_{i \in L} b_i + |R|\right)\log(1/\varepsilon)\right)$ space where L is the side with the smaller number of nodes in the input graph.*

We note that the space bound is necessary in order to report the solution. (There exists a given input where reporting the solution requires $\widetilde{O}\left(\left(\sum_{i \in L} b_i + |R|\right)\log(1/\varepsilon)\right)$ space.) Thus, our algorithm is tight with respect to this notion.

4.4 Shared-Memory Parallel Implementation

We now show an implementation of our algorithm to the shared-memory parallel setting. The main challenge for this setting is obtaining an algorithm for obtaining non-duplicate maximal matchings. To obtain non-duplicate maximal matchings, we just need to modify the maximal matching algorithm of [9] to obtain a maximal matching with the non-duplicate characteristic. Namely, the modification we make is to consider all copies of a node to be neighbors of each other. Since there can be at most n copies of a node, this increases the degree of each node by at most n. Hence, the same analysis as the original algorithm still holds in this new setting.

▶ **Theorem 31.** *There exists a shared-memory parallel algorithm for maximum cardinality bipartite b-matching that uses $O\left(\frac{\log^3 n}{\varepsilon^2}\right)$ depth and $O\left(\frac{m \log n}{\varepsilon^2}\right)$ total work where L is the side with the smaller number of nodes in the input graph.*

References

1 Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. In *Proceedings of the 38th International Conference on Automata, Languages and Programming - Volume Part II*, ICALP'11, pages 526–538, Berlin, Heidelberg, 2011. Springer-Verlag.

2 Kook Jin Ahn and Sudipto Guha. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. *ACM Trans. Parallel Comput.*, 4(4), January 2018. doi:10.1145/3154855.

3 Sepehr Assadi. A two-pass (conditional) lower bound for semi-streaming maximum matching. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 708–742. SIAM, 2022. doi:10.1137/1.9781611977073.32.

4 Sepehr Assadi, Arun Jambulapati, Yujia Jin, Aaron Sidford, and Kevin Tian. Semi-streaming bipartite matching in fewer passes and optimal space. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 627–669. SIAM, 2022.

5 Sepehr Assadi, Gillat Kol, Raghuvansh R. Saxena, and Huacheng Yu. Multi-pass graph streaming lower bounds for cycle counting, max-cut, matching size, and other problems. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 354–364, 2020. doi:10.1109/FOCS46700.2020.00041.

6 Sepehr Assadi, S. Cliff Liu, and Robert E. Tarjan. *An Auction Algorithm for Bipartite Matching in Streaming and Massively Parallel Computation Models*, pages 165–171. SIAM, 2021. doi:10.1137/1.9781611976496.18.

7 Aaron Bernstein, Aditi Dudeja, and Zachary Langley. A framework for dynamic matching in weighted graphs. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 668–681, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3406325.3451113.

8 Dimitri P Bertsekas. A new algorithm for the assignment problem. *Mathematical Programming*, 21(1):152–171, 1981.

9 Guy E. Blelloch, Jeremy T. Fineman, and Julian Shun. Greedy sequential maximal independent set and matching are parallel on average. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 308–317, 2012.

10 Gabrielle Demange, David Gale, and Marilda Sotomayor. Multi-item auctions. *Journal of political economy*, 94(4):863–872, 1986.

11 Shahar Dobzinski, Noam Nisan, and Sigal Oren. Economic efficiency requires interaction. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 233–242, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2591796.2591815.

12 Jack Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of Research of the National Bureau of Standards B*, 69:125–130, 1965.

13 Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

14 Manuela Fischer, Slobodan Mitrović, and Jara Uitto. Deterministic $(1+\varepsilon)$-approximate maximum matching with poly$(1/\varepsilon)$ passes in the semi-streaming model and beyond. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 248–260, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519935.3520039.

15 Buddhima Gamlath, Sagar Kale, Slobodan Mitrovic, and Ola Svensson. Weighted matchings via unweighted augmentations. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, pages 491–500, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3293611.3331603.

16 Mohsen Ghaffari, Christoph Grunau, and Slobodan Mitrović. Massively parallel algorithms for b-matching. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2022.

17 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. *On the communication and streaming complexity of maximum bipartite matching*, pages 468–485. SIAM, 2012. doi:10.1137/1.9781611973099.41.

18 Manoj Gupta and Richard Peng. Fully dynamic (1+ e)-approximate matchings. In *FOCS*, pages 548–557. IEEE Computer Society, 2013.

19 Nicholas J. A. Harvey. Algebraic structures and algorithms for matching and matroid problems. *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 531–542, 2006.

20 John E. Hopcroft and Richard M. Karp. A n5/2 algorithm for maximum matchings in bipartite. In *12th Annual Symposium on Switching and Automata Theory (swat 1971)*, pages 122–125, 1971. doi:10.1109/SWAT.1971.1.

21 Shang-En Huang and Hsin-Hao Su. (1-ϵ)-approximate maximum weighted matching in poly(1/ϵ, log n) time in the distributed and parallel settings. *CoRR*, abs/2212.14425, 2022. doi:10.48550/arXiv.2212.14425.

22 Arun Jambulapati, Yang P. Liu, and Aaron Sidford. Parallel reachability in almost linear work and square root depth. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1664–1686, 2019. doi:10.1109/FOCS.2019.00098.

23 Michael Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1679–1697. SIAM, 2013.

24 Christian Konrad, Peter Robinson, and Viktor Zamaraev. Robust lower bounds for graph problems in the blackboard model of communication. *CoRR*, abs/2103.07027, 2021. arXiv:2103.07027.

25 D. König. Über graphen und ihre anwendung auf determinantentheorie und mengenlehre. *Mathematische Annalen*, 77:453–465, 1916. URL: http://eudml.org/doc/158740.

26 S Cliff Liu, Zhao Song, and Hengjie Zhang. Breaking the n-pass barrier: A streaming algorithm for maximum weight bipartite matching. *arXiv preprint*, 2020. arXiv:2009.06106.

27 Yang P. Liu and Aaron Sidford. *Faster Energy Maximization for Faster Maximum Flow*, pages 803–814. Association for Computing Machinery, New York, NY, USA, 2020. doi:10.1145/3357713.3384247.

28 Zvi Lotker, Boaz Patt-Shamir, and Seth Pettie. Improved distributed approximate matching. *J. ACM*, 62(5), November 2015. doi:10.1145/2786753.

29 Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 253–262, 2013. doi:10.1109/FOCS.2013.35.

30 Silvio Micali and Vijay V. Vazirani. An $o(\sqrt{|v|} \cdot |e|)$ algorithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)*, pages 17–27, 1980. doi:10.1109/SFCS.1980.12.

31 M. Mucha and P. Sankowski. Maximum matchings via gaussian elimination. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 248–255, 2004. doi:10.1109/FOCS.2004.40.

32 Noam Nisan. The demand query model for bipartite matching. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 592–599. SIAM, 2021.

33 Daniel Stubbs and Virginia Vassilevska Williams. Metatheorems for dynamic weighted matching. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, volume 67 of *LIPIcs*, pages 58:1–58:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ITCS.2017.58.

34 Da Wei Zheng and Monika Henzinger. Multiplicative auction algorithm for approximate maximum weight bipartite matching. *arXiv preprint*, 2023. arXiv:2301.09217.

Giant Components in Random Temporal Graphs

Ruben Becker
Ca' Foscari University of Venice, Italy

Arnaud Casteigts
University of Geneva, Switzerland

Pierluigi Crescenzi
Gran Sasso Science Institute, L'Aquila, Italy

Bojana Kodric
Ca' Foscari University of Venice, Italy

Malte Renken
Technical University of Berlin, Germany

Michael Raskin
LaBRI, CNRS, University of Bordeaux, France

Viktor Zamaraev
University of Liverpool, UK

Abstract

A temporal graph is a graph whose edges appear only at certain points in time. In these graphs, reachability among nodes relies on paths that traverse edges in chronological order (*temporal paths*). Unlike standard paths, temporal paths may not be composable, thus the reachability relation is *not transitive* and connected components (i.e., sets of pairwise temporally connected nodes) do not form equivalence classes, a fact with far-reaching consequences.

Recently, Casteigts et al. [FOCS 2021] proposed a natural temporal analog of the seminal Erdős-Rényi random graph model, based on the same parameters n and p. The proposed model is obtained by randomly permuting the edges of an Erdős-Rényi random graph and interpreting this permutation as an ordering of presence times. Casteigts et al. showed that the well-known single threshold for connectivity in the Erdős-Rényi model fans out into multiple phase transitions for several distinct notions of reachability in the temporal setting.

The second most basic phenomenon studied by Erdős and Rényi in static (i.e., non-temporal) random graphs is the emergence of a *giant connected component*. However, the existence of a similar phase transition in the temporal model was left open until now. In this paper, we settle this question. We identify a sharp threshold at $p = \log n/n$, where the size of the largest temporally connected component increases from $o(n)$ to $n - o(n)$ nodes. This transition occurs significantly later than in the static setting, where a giant component of size $n - o(n)$ already exists for any $p \in \omega(1/n)$ (i.e., as soon as p is larger than a constant fraction of n). Interestingly, the threshold that we obtain holds for both *open* and *closed* connected components, i.e., components that allow, respectively forbid, their connecting paths to use external nodes – a distinction arising from the absence of transitivity.

We achieve these results by strengthening the tools from Casteigts et al. and developing new techniques that provide means to decouple dependencies between past and future events in temporal Erdős-Rényi graphs, which could be of general interest in future investigations of these objects.

2012 ACM Subject Classification Theory of computation → Random network models; Mathematics of computing → Random graphs

Keywords and phrases random temporal graph, Erdős–Rényi random graph, sharp threshold, temporal connectivity, temporal connected component, edge-ordered graph

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.29

Category RANDOM

Related Version *Full Version*: https://arxiv.org/abs/2205.14888v3

Funding Some of the authors are supported by the German Research Foundation (DFG) under the project MATE (NI 369/17). Some of the authors are supported by the French ANR, project ANR-22-CE48-0001 (TEMPOGRAL). Some of the authors received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 787367 (PaVeS).

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 29; pp. 29:1–29:17
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Many real-world networks vary with time, as exemplified by the dynamic nature of today's social media, telecommunication, transportation, and interaction in general in a complex network. Indeed, the examination of specific applications illustrates how networks endowed with temporal information enable more accurate and effective analysis of real-world systems compared to static networks [33].

This insight has motivated plethora of studies focusing on network modeling approaches that incorporate the time dimension [24, 25, 27]. A widely used model for these networks is given by *temporal graphs* (sometimes also called *time-varying graphs*, *evolving graphs*, or other names). A temporal graph is a pair $\mathcal{G} = (G, \lambda)$, where $G = (V, E)$ is an *underlying* (static) graph, and λ is an *edge labeling function* that assigns to every edge $e \in E$ a set of time labels $\lambda(e) \subseteq \mathbb{N}$ indicating when this edge is present. This definition, although simple, already captures two important aspects that determine temporal networks. Namely, (a) the topology of the network defined by the underlying graph G; and (b) the schedule of edge availabilities represented by the labeling function λ.

Even though this model has gained much traction recently, the available tools for analyzing temporal graphs are still nowhere near the level of tools that have been developed for understanding static networks. One of the main challenges is the fundamentally changed notion of *reachability*. In temporal graphs, reachability is naturally based on paths that traverse edges in ascending time, a.k.a. *temporal paths*. A first difference with standard paths is that temporal paths are inherently directed, regardless of whether the graph itself is directed, due to the arrow of time. Even more significantly, temporal reachability is *not transitive*, i.e., the fact that node u can reach node v and node v can reach node w does not imply that u can reach w. The resulting non-composability is a source of complication for structural studies, as well as a frequent source of computational hardness. In fact, many problems related to reachability are hard in temporal graphs, even when their classical analogs are polynomial time solvable – see, for instance, the seminal paper by Kempe, Kleinberg, and Kumar [27] on k-disjoint temporal paths (and many further examples appearing in more recent works [1, 8, 13–15, 20, 22]). As observed by Bhadra and Ferreira [6], the fact that *(temporally) connected components* do not form equivalence classes and intersect in non trivial ways implies, among other consequences, that finding one of maximum size is NP-hard.

Random Models of Temporal Graphs

One of the most important tools in (static) network theory are random network models [30]. They allow reproducing characteristics of real networks and studying their statistical properties. The random perspective enables prediction of properties, anomaly detection, identification of phase transitions, and other conclusions about the nature of typical networks.

The cornerstone of random network theory is the Erdős-Rényi random graph model [3]. It has proven tremendously useful as a source of insight into the structure of networks [32]. An Erdős-Rényi random graph $G_{n,p}$ is obtained by placing an edge between each distinct pair of n vertices[1] independently with probability p. The study of this model was sparked by a series of seminal papers published by Erdős and Rényi starting in 1959 [16–19]. Since then, an important number of articles and books have been devoted to this model. These

[1] We use the terms *vertex* and *node* interchangeably.

results laid a solid foundation for the development of other models of more practical interest, including the configuration model [28,29,31], the small-world model [34], and the preferential attachment model [4].

The number of models of random *temporal* networks proposed in the literature is still limited and no systematic foundations are available [24]. In establishing such foundations, a natural question is: *What is the temporal analog of the Erdős-Rényi random graph model?* The answer to this question is not unique, as the time dimension can be incorporated in different ways [32]. Some candidates considered in the literature consider a sequence of independent Erdős-Rényi graphs, some others incorporate some dependencies in such a sequence (see for example [2,5,10–12,23,35]).

Temporal Erdős-Rényi Random Graphs

Recently, another natural and more direct temporal analog of the Erdős-Rényi random graphs was proposed by Casteigts et al. [9], based on the same parameters n and p. In this model, which we refer to as the *temporal Erdős-Rényi random graph model*, a random temporal graph is obtained from an Erdős-Rényi random graph $G_{n,p}$ by assigning to each edge a unique label (presence time) according to a uniformly random permutation of its edges. The main motivation is to obtain a temporal graph model whose properties (such as threshold values) can be directly compared to the classical Erdős-Rényi model, thereby highlighting the qualitative impact of the time dimension. A systematic study of this model may also set a benchmark for practical models.

As already remarked, the time dimension leads to a number of distinctions between static and temporal graphs. Many of them come from the conceptual difference between the notions of path and temporal paths. The reachability of a temporal graph is not symmetric (even in the undirected case) and not transitive, which is in stark contrast with static graphs. Indeed, the results of [9] revealed that even the notion of connectivity translates to a rich spectrum of phase transitions in the temporal setting. Namely, at $p = \log n/n$, any *fixed* pair of vertices can asymptotically almost surely (a.a.s.) reach each other; at $2\log n/n$, at least one vertex (and in fact, any fixed vertex) can a.a.s. reach all the others; and at $3\log n/n$, all the vertices can a.a.s. reach all others, i.e., the graph is temporally connected.

Connected Components in Temporal Erdős-Rényi Random Graphs

Perhaps the most investigated aspects of Erdős-Rényi random graphs is the emergence of a “giant” connected component [7,21], which culminates in connectivity itself. The analogous question in a temporal setting is therefore natural. Interestingly, the lack of transitivity makes the very definition of temporal components ambiguous. If the vertices of the component need temporal paths traveling *outside* the component in order to reach each other, then the component is *open*; otherwise, it is *closed* [6].

Analyzing the emergence of (both types of) *temporally* connected components in the above model presents technical challenges that cannot be overcome by the only tools developed in [9]. These technical challenges and the importance of understanding connected components in temporal Erdős-Rényi random graphs motivated the present work.

1.1 Contributions

In this paper, we analyze the evolution of the largest connected component in a temporal Erdős-Rényi random graph with parameters n and p, as p increases (with $n \to \infty$). Our main result is that, in contrast to static graphs, the phase transition occurs at $p = \log n/n$. At this point, the size of the largest component jumps from $o(n)$ to $n - o(n)$.

▶ **Main Theorem** (informal). *There exists a function $\varepsilon(n) \in o(\log n/n)$ such that the size of a largest temporally connected component in a temporal Erdős-Rényi random graph is*

(i) *$o(n)$ a.a.s., if $p < \frac{\log n}{n} - \varepsilon(n)$; and*

(ii) *$n - o(n)$ a.a.s., if $p > \frac{\log n}{n} + \varepsilon(n)$.*

Notably, the same threshold holds for both open and closed connected components, although showing the latter requires more effort. We achieve these results by developing new techniques and combining them with strengthened versions of the tools from [9]. Informally, the new tools enable us to effectively contain the dependencies that exist between different time slices. Thus they facilitate building graph structures witnessing a desired property in *multiple independent phases.*

1.2 Significance of the Results & Techniques

Results. Our main result reveals a qualitative difference between the evolution of connected components in static random graphs and temporal random graphs. The emergence of a giant component in (static) Erdős–Rényi graphs follows a well-known pattern of events [19]. Below a critical probability $p_0 = 1/n$, almost all the components are trees, and no component is larger than $O(\log n)$. Then, at p_0, a single "giant" component of size $\Theta(n^{2/3})$ arises. Then, at $p = c/n > 1/n$, this component contains a constant fraction $1 - x/c$ of all vertices (with $0 < x < 1$ being defined through $xe^{-x} = ce^{-c}$). As soon as $p \in \omega(1/n)$, the component contains all but $o(n)$ vertices. The case of directed static graphs is similar. Namely, for $p = c/n < 1/n$, a.a.s. all strongly connected components have size less than $3c^{-2}\log n$, and when $p = c/n > 1/n$, the graph contains a strongly connected component of size approximately $(1 - x/c)^2\, n$ (with x as above) [21, 26], which implies that this component contains all but $o(n)$ vertices when $p \in \omega(1/n)$.

In the temporal setting, we show that the phase transition occurs at $p = \log n/n$. Namely, all components are of size $o(n)$ before that threshold and there is one component of size $n - o(n)$ afterwards. The fact that this transition occurs later in the temporal setting is not surprising, as the thresholds for *connectivity* is already known to be significantly smaller in the static setting than in the temporal setting; namely, connectivity occurs at $p = \log n/n$ in the static case (for both directed and undirected graphs) versus $p = 3\log n/n$ for temporal connectivity [9]. However, while these thresholds for connectivity are within a multiplicative constant of each other, our results show that in the case of connected components the static and the temporal threshold are of distinct asymptotic orders.

Techniques. In the temporal Erdős-Rényi model, the unicity of presence times for the edges causes delicate dependencies between past and future events. To contain these dependencies, we introduce a multiphase analysis that consists of splitting the time interval into several phases where these dependencies are decoupled. We believe that many further temporal graph properties will require such a multiphase analysis and could benefit from the tools developed here. In constrast, the techniques from [9] are well suited for analyzing single-phase processes, where temporal paths do not interact across different time intervals (e.g. through composition).

In particular, our switch from a fixed base graph $G = K_n$ to an arbitrary graph of high minimum degree provides the possibility to "encapsulate" all dependencies on events occurring in some fixed "short" phase into the choice of base graph, effectively eliminating the need to deal with these dependencies individually. As an unsurprising but quite useful technical extension, we study also the behaviour of sets of journeys starting from any of a set of source vertices. Furthermore, Lemma 5.2 is of independent interest for "bootstrapping" various such multiphase analysis attempts; it essentially proves that in the very early regime, there is only a small number of poorly connected vertices, and that these can be removed without compromising the connectivity of the remaining temporal graph.

Although our techniques handle specific dependencies of temporal Erdős-Rényi graphs, they remain general enough to be adaptable to models with less dependencies, such as models where several appearances of an edge is possible and these appearances follow an exponential distribution (Poisson process). The reasons for this are exactly the same as in [9]. Note, however, that weaker tools could suffice for such models, as past and future appearances of an edge are independent.

1.3 Organization

In Section 2, we provide all necessary definitions, and introduce the random temporal graph models used in the paper. In Section 3, we present the algorithm for constructing a foremost forest. We also state a core technical theorem (Theorem 3.3) concerned with reachability between two sets of nodes in a temporal graphs. The full proof of that theorem is deferred to the full version due to space restrictions, as are several other proofs and intermediate results. Using this theorem, we then prove in Section 4 that at $p = \log n/n$ the size of the largest *open* connected component jumps from $o(n)$ to $n - o(n)$. This also serves as a stepping stone towards Section 5, where we extend our technique to also apply to *closed* connected components. The proof is slightly more involved than for open components, as it requires further subdivisions of the phases. However, we show that both variants undergo essentially the same phase transition.

2 Preliminaries

In this paper, $[k]$ denotes the set of integers $\{1, \dots, k\}$, and $[a, b]$ denotes either the discrete interval from a to b, or the continuous interval from a to b, the distinction being clear from the context. All graphs are simple, i.e., without loops or multiple edges. For a graph G, we denote by $V(G)$ and $E(G)$ its vertex set and edge set respectively. We denote by $\delta(G)$ and $\Delta(G)$ the minimum and the maximum vertex degree of G respectively. As usual, K_n denotes the complete n-vertex graph.

2.1 Temporal Graphs

A *temporal graph* is a pair (G, λ), where $G = (V, E)$ is a *static* graph and λ is a function that assigns to every edge $e \in E$ a finite set of numbers, interpreted as presence times. The graph G is called the *underlying graph* of the temporal graph and the elements of $\lambda(e)$ are called the *time labels* of e. We will denote temporal graphs by calligraphic letters, e.g., by $\mathcal{G}$. Instead of (G, λ) we will sometimes use the notation (V, E, λ) to denote the same temporal graph. In most cases, time labels will be elements of the real unit interval $[0, 1]$. Furthermore, in this paper, we restrict our consideration only to *simple* temporal graphs[2], i.e., temporal graphs in which every edge $e \in E$ is only present at a single point in time, i.e., $|\lambda(e)| = 1$. We sometimes write $V(\mathcal{G})$ and $E(\mathcal{G})$ for the node and edge set of a temporal graph $\mathcal{G}$ respectively.

[2] We remark that all our results can be directly transferred to another, closely related model of non-simple temporal graphs; see Section 6.1.2 in [9].

A temporal graph $\mathcal{H} = (V_{\mathcal{H}}, E_{\mathcal{H}}, \lambda_{\mathcal{H}})$ is a *temporal subgraph* of a temporal graph $\mathcal{G} = (V_{\mathcal{G}}, E_{\mathcal{G}}, \lambda_{\mathcal{G}})$, if $V_{\mathcal{H}} \subseteq V_{\mathcal{G}}$, $E_{\mathcal{H}} \subseteq E_{\mathcal{G}}$ and $\lambda_{\mathcal{H}}(e) = \lambda_{\mathcal{G}}(e)$ for all $e \in E_{\mathcal{H}}$. For a vertex set $S \subseteq V(\mathcal{G})$, we denote by $\mathcal{G}[S]$ a temporal subgraph of $\mathcal{G}$ induced by S. We use $\mathcal{G}_{[a,b]}$ to denote the temporal subgraph of $\mathcal{G}$ with the same node set $V_{\mathcal{G}}$, the edge set $E' := \{e \in E_{\mathcal{G}} : \lambda_{\mathcal{G}}(e) \in [a, b]\}$, and the time labeling function $\lambda_{\mathcal{G}}|_{E'}$ which is the restriction of $\lambda_{\mathcal{G}}$ to E'.

A *temporal* (u, v)*-path* in $\mathcal{G} = (V, E, \lambda)$ between two nodes $u, v \in V$ is a sequence $u = u_0, u_1, \ldots, u_\ell = v$ such that $e_i = \{u_{i-1}, u_i\} \in E$ for each $i \in [\ell]$, and time labels are increasing, i.e., $\lambda(e_1) < \ldots < \lambda(e_\ell)$. We call $\lambda(e_\ell)$ the *arrival time* of the path. A temporal (u, v)-path is called *foremost* (u, v)*-path* if it has the earliest arrival time among all temporal (u, v)-paths. If there exists a temporal (u, v)-path, we say that u can reach v (every vertex reaches itself). A set $S \subseteq V$ is said to *reach* v if at least one of its elements reaches v. In that case, a *foremost* (S, v)*-path* in $\mathcal{G}$ is a temporal (u, v)-path with earliest arrival time among all $u \in S$.

A vertex $u \in V$ is called *temporal source* in $\mathcal{G} = (V, E, \lambda)$ if there exists a temporal (u, v)-path for each $v \in V$. Similarly, a vertex $v \in V$ is called *temporal sink* in $\mathcal{G}$ if there exists a temporal (u, v)-path for each $u \in V$.

A temporal graph $\mathcal{G} = (V, E, \lambda)$ is *temporally connected* if all nodes are temporal sources. We note that this also implies that all nodes are temporal sinks. An *open temporally connected component* or simply *connected component* in $\mathcal{G}$ is an inclusion-wise maximal set $Z \subseteq V$ of nodes such that for every ordered pair of vertices $u, v \in Z$, there exists a temporal (u, v)-path in $\mathcal{G}$. We stress that such a temporal (u, v)-path can contain nodes from $V \setminus Z$. If for every ordered pair $u, v \in Z$, there exists a temporal (u, v)-path in $\mathcal{G}[Z]$, then Z is called *closed connected component*.

2.2 Random Temporal Graph Models

The model of *temporal Erdős-Rényi random graphs* was introduced in [9][3] as a natural temporal generalization of the classical Erdős–Rényi model $G_{n,p}$ of random graphs. An n-vertex temporal Erdős-Rényi random graph with the parameter $p \in [0, 1]$ is obtained by first drawing a *static* random Erdős–Rényi $G_{n,p}$ and then defining a temporal order on its edges by ordering them according to a uniformly random permutation. An equivalent and technically more convenient way of defining the temporal order on the edges is to draw, for every edge e, independently and uniformly at random a time label $\lambda(e)$ from the unit interval $[0, 1]$. Since the event that two edges get the same time label happens with probability 0, all edge orderings induced by such random time labels are equiprobable. Therefore, as long as the absolute values of time labels are irrelevant (which is the case for the questions studied in [9] and in the present paper), the two models are indeed equivalent. This latter model is denoted as $\mathcal{F}_{n,p}$. A possible way of generating objects from $\mathcal{F}_{n,p}$ is to first draw a temporal graph $\mathcal{G} = (G, \lambda)$ from $\mathcal{F}_{n,1}$ (thus the underlying graph G is complete), and to then consider $\mathcal{G}' = (G', \lambda') = (G, \lambda)|_{[0,p]}$, i.e., the temporal graph obtained from $\mathcal{G}$ by removing edges with time labels greater than p. Observe that $G' \sim G_{n,p}$ and each time label $\lambda(e)$ is uniformly distributed on $[0, p]$. Hence, $\mathcal{G}'$ is distributed according to $\mathcal{F}_{n,p}$ up to multiplying all labels by a factor of $\frac{1}{p}$, which we can ignore as it neither changes the relative order of time labels nor the absolute values of time labels are of any importance to us. For similar reasons, for any $0 \leq a \leq b \leq 1$, up to rescaling time labels, the temporal subgraph $\mathcal{G}|_{[a,b]}$ is distributed according to $\mathcal{F}_{n,q}$, where $q = b - a$.

[3] In [9], this model was called Random Simple Temporal Graphs (RSTGs)

In order to overcome some technical challenges caused by interdependence of different temporal subgraphs, we define and study a natural generalization of $\mathcal{F}_{n,p}$ that we describe next. For an n-vertex graph G and a real value $p \in [0,1]$, we denote by $\mathcal{F}_p(G)$ the following random temporal graph model. A random temporal graph $\mathcal{G} = (V, E, \lambda) \sim \mathcal{F}_p(G)$ is obtained by (1) independently and uniformly sampling a time label $\lambda'(e) \in [0,1]$ for every $e \in E(G)$, and (2) setting $V = V(G)$, $E := \{e \in E(G) : \lambda(e) \leq p\}$ and $\lambda(e) = \lambda'(e)$ for every $e \in E$. We call G the *base graph* of $\mathcal{F}_p(G)$. We observe that the $\mathcal{F}_{n,p}$ model is obtained when choosing the base graph to be the complete n-vertex graph K_n.

In what follows we sometimes implicitly assume that $n = |V|$ is sufficiently large without restating this assumption. We note that some of our estimates hold only for rather large values of n. We did not attempt to reduce these bounds, but instead focused on achieving best possible readability.

At this point we refer the interested reader to Appendix A of the full version, where, as a warm-up, we give a simple upper bound on p which guarantees that $\mathcal{G} \sim \mathcal{F}_p(G)$ is temporally connected a.a.s.

3 The Foremost Forest Algorithm

The main aim of this section is to present an algorithm for constructing a foremost forest and to prove a property of this algorithm.

Foremost forests play a crucial role in the development of our main technical tool: for a fixed set of vertices S and a given number k, the estimation of the minimum value of p such that the vertices in S can reach k vertices in $\mathcal{G} = (V, E, \lambda) \sim \mathcal{F}_p(G)$ a.a.s.

We obtain such an estimation by examining the evolution of a foremost forest for S in $\mathcal{G}$ via analysis of the execution of the formost forest algorithm on random temporal graphs. To elaborate on this approach, let us consider $v \in V \setminus S$. We would like to estimate the probability that S reaches v in $\mathcal{G}$. For this, we follow an approach similar to the one used in [9]. Let $\mathcal{G}' \sim \mathcal{F}_1(G)$ and observe that the probability that S can reach v in $\mathcal{G}$ is equal to the probability that the temporal subgraph $\mathcal{G}'_{[0,p]}$ contains a temporal (u, v)-path P for some node $u \in S$. This again is equivalent to the arrival time of P in $\mathcal{G}'$ being at most p. Therefore, the estimation of the parameter p for which some node from S can reach v can be reduced to the estimation of the minimum arrival time of a foremost temporal path from S to v in $\mathcal{G}' \sim \mathcal{F}_1(G)$. A *foremost forest for S in $\mathcal{G}$* is a minimal temporal subgraph that preserves foremost reachabilities from S to all other vertices reachable from S in $\mathcal{G}$. We proceed with the necessary formal definitions.

▶ **Definition 3.1.** *Let $\mathcal{G} = (V, E, \lambda)$ be a temporal graph and let $S \subseteq V$ be a set of vertices. The graph $\mathcal{G}_F = (V_F, E_F, \lambda_F)$ is an* increasing temporal forest for S, *if*

(a) *$\mathcal{G}_F$ is a temporal subgraph of $\mathcal{G}$,*

(b) *the graph $F = (V_F, E_F)$ is a forest (i.e. acyclic graph) with $|S|$ components,*

(c) *for each $s \in S$ there is a connected component T_s of F such that s reaches all vertices of T_s in $\mathcal{G}_F$.*

We are now ready to define (partial) foremost forests.

▶ **Definition 3.2.** *Let $\mathcal{G} = (V, E, \lambda)$ be a temporal graph, let $S \subseteq V$ be a set of vertices and let $\mathcal{G}_F = (V_F, E_F, \lambda_F)$ be an increasing temporal forest for S.*

1. *Then $\mathcal{G}_F$ is a* partial foremost forest for S, *if, for all $v \in V_F \setminus S$, the unique temporal (S, v)-path in $\mathcal{G}_F$ is a foremost (S, v)-path in $\mathcal{G}$.*
2. *A partial foremost forest for S is a* foremost forest for S *if V_F contains all vertices reachable from S in $\mathcal{G}$, i.e., $V_F = \{v \in V : \exists (u, v)\text{-temporal path in } \mathcal{G} \text{ for some } u \in S\}$.*
3. *A (partial) foremost forest for $\{v\}$ is a* (partial) foremost tree for v.

The Algorithm

Next, we present an algorithm that, given a temporal graph $\mathcal{G} = (V, E, \lambda)$ and a set of nodes $S \subseteq V$ constructs a foremost forest $\mathcal{G}_F$ for S. This algorithm is a straightforward generalization of the foremost tree algorithm from [9], where the input set S is assumed to be singleton.

The idea of the algorithm similar to Prim's algorithm for minimum spanning trees in static graphs: Starting from $\mathcal{G}_F = (V_F, E_F, \lambda_F) = (S, \emptyset, \emptyset)$, which is trivially a partial foremost forest for S, the algorithm iteratively adds one node and one edge to V_F and E_F, λ_F, respectively, until $\mathcal{G}_F$ becomes a foremost forest for S. The main difference to Prim's algorithm is that, in every iteration, the next edge to be added is chosen as the edge of minimum time label among all edges that extend the current increasing temporal forest. For brevity, we introduce the following notation. We write $\mathcal{G}_F \cup e$ for adding the edge $e = \{u, v\}$ to $\mathcal{G}_F$, i.e., the result is the temporal graph $(V_F \cup \{u, v\}, E_F \cup \{e\}, \lambda_F \cup \{(e, \lambda(e))\})$. The set of edges that extend the current partial forest can then be defined as

$$\operatorname{ext}(\mathcal{G}_F) := \{e = \{u, v\} \in E : u \in V_F, v \in V \setminus V_F, \text{ and } \mathcal{G}_F \cup e \text{ is an increasing temporal forest for } S\}.$$

We now are ready to state the algorithm.

Algorithm 1 Foremost Forest.

```
Input   : Simple temporal graph $\mathcal{G} = (V, E, \lambda)$; set of nodes $S \subseteq V$.
Output : Foremost forest for $S$.
$k = |S| - 1$, $\mathcal{G}_F^k = (S, \emptyset, \emptyset)$
while $\operatorname{ext}(\mathcal{G}_F^k) \neq \emptyset$ do
    $k := k + 1$
    $e_k := \arg\min\{\lambda(e) \mid e \in \operatorname{ext}(\mathcal{G}_F^{k-1})\}$
    $\mathcal{G}_F^k := \mathcal{G}_F^{k-1} \cup e_k$
return $\mathcal{G}_F^k$
```

In Appendix B of the full version we prove that Algorithm 1 in fact builds a foremost forest.Furthermore, one of our main technical results is the following theorem which, for two given sets of nodes S and T, quantifies the probability that a foremost forest grown from set S reaches T.

▶ **Theorem 3.3** (Foremost Forest Target Set Reachability). *Let*

- *G be a graph of minimum degree $\delta(G) \geq n - (\log n)^a$ for some $a \in \mathbb{N}$,*
- *let S and T be two sets of nodes in G of cardinalities $s \in [(\log n)^{13}, n/2]$ and t, respectively,*
- *let $z = z(n)$ be a function with $\varepsilon \leq z(n) \leq 1 - \varepsilon$ for some constant $\varepsilon \in (0, 1)$, and*
- *let $\mathcal{G} \sim \mathcal{F}_p(G)$ with $p \geq \frac{z \log n - \log s}{n} + \frac{3 \log \log n}{n}$.*

Then the foremost forest algorithm from S on $\mathcal{G}$ reaches T with probability at least $1 - \frac{5}{2} n^{-\log \log n} - e^{-\frac{t}{2n}(n^z - s)}$.

The formal proof of Theorem 3.3 is one of the technically more involved portions of this work. It is divided into a number of lemmas and has to be deferred to Appendix C of the full version due to lack of space; for improved accessibility, a high level overview of the proof structure is depicted in Figure 1. We proceed with a short proof sketch.

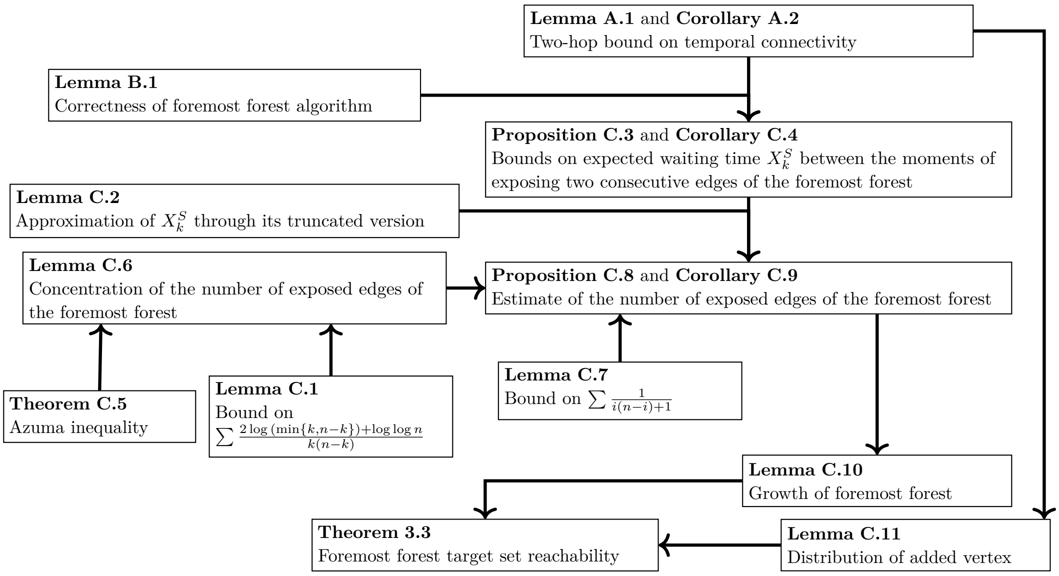

Lemma A.1 and Corollary A.2
Two-hop bound on temporal connectivity
Lemma B.1
Correctness of foremost forest algorithm
Proposition C.3 and Corollary C.4
Bounds on expected waiting time X_k^S between the moments of exposing two consecutive edges of the foremost forest
Lemma C.2
Approximation of X_k^S through its truncated version
Lemma C.6
Concentration of the number of exposed edges of the foremost forest
Proposition C.8 and Corollary C.9
Estimate of the number of exposed edges of the foremost forest
Lemma C.7
Bound on $\sum \frac{1}{i(n-i)+1}$
Theorem C.5
Azuma inequality
Lemma C.1
Bound on $\sum \frac{2\log(\min\{k,n-k\})+\log\log n}{k(n-k)}$
Lemma C.10
Growth of foremost forest
Theorem 3.3
Foremost forest target set reachability
Lemma C.11
Distribution of added vertex

Figure 1 Overview of the proof of Theorem 3.3.

Proof Sketch. The theorem is deduced from Lemma C.10 and Lemma C.11 that can be found in Appendix C of the full version. Lemma C.10 essentially constitutes a generalization of the foremost tree growth analysis from [9], which estimates the number of vertices that a given vertex (referred to as a *source*) reaches by specific time in $\mathcal{F}_{n,p}$. Besides the difference that in Lemma C.10 we need to consider a fixed *set of source* vertices, the main technical challenge here is that we have to consider the $\mathcal{F}_p(G)$ model rather than the basic $\mathcal{F}_{n,p}$ model, resulting in fewer edges per node. While Lemma C.10 merely gives a statement over the number of nodes that are reached from a given source set, Lemma C.11 gives the second crucial ingredient for proving Theorem 3.3. It states that every new vertex reached by the foremost forest grown from S (i.e., every new vertex added to the foremost forest) is distributed almost uniformly on the vertices that are not reached yet and this allows us to estimate the probability that the forest reaches the target set T. ◀

4 Sharp Threshold for Giant Open Connected Component

In this section, we report on our first main result.

▶ **Theorem 4.1** (Main Result for Open Components)**.** *The function $\frac{\log n}{n}$ is a sharp threshold for Giant Open Connected Component. More specifically, there exists a function $\varepsilon(n) \in o\Big(\frac{\log n}{n}\Big)$, such that the size of a largest open temporally connected component in $\mathcal{G} \in \mathcal{F}_{n,p}$ is*

(i) *$o(n)$ a.a.s., if $p < \frac{\log n}{n} - \varepsilon(n)$; and*
(ii) *$n - o(n)$ a.a.s., if $p > \frac{\log n}{n} + \varepsilon(n)$.*

We prove the lower bound on the threshold (i.e. Theorem 4.1 (i)) in Section 4.1. The proof of this bound is a straightforward consequence of a result on foremost *tree* growth in $\mathcal{F}_p(K_n)$ from [9]. The upper bound (i.e. Theorem 4.1 (ii)) on the threshold is proved in Section 4.2 and is significantly more involved. In particular, it relies on Theorem 3.3 to measure foremost *forest* growth in $\mathcal{F}_p(G)$, where G is chosen to contain all edges that did *not* occur within some particular time window.

4.1 Lower Bound on the Threshold

We state the lower bound in form of the following theorem which says that a.a.s. there is no linear size component before time $\log n/n$. This theorem can be derived rather easily from results of Casteigts et al. [9]; we refer to Appendix D of the full version for the details.

▶ **Theorem 4.2** (Lower Bound in Theorem 4.1). *Let $\mathcal{G} \sim \mathcal{F}_p(K_n)$ with $p < \frac{\log n}{n} - \frac{3(\log n)^{0.8}}{n}$. Then, for any constant $c \in (0,1)$, the graph $\mathcal{G}$ does not contain a temporally connected component of size at least $c \cdot n$ with probability at least $1 - 2n^{-\sqrt{\log n}}$.*

4.2 Upper Bound on Threshold

Next, we present the first, weaker version of our main result, stating that an open temporally connected components containing almost all vertices appears already around time $\log n/n$.

▶ **Theorem 4.3** (Upper Bound in Theorem 4.1). *Let $\mathcal{G} \sim \mathcal{F}_p(K_n)$ with $p \geq (1 + \varepsilon(n)) \cdot \frac{\log n}{n}$. Then, the graph $\mathcal{G}$ contains a temporally connected component of size $n - o(n)$ a.a.s.*

We begin by giving a sketch of the proof idea.

Proof Sketch. The strategy is as follows, see also Figure 2. We split the time interval $[0, p]$ into three intervals I_1, I_2, and I_3 of equal duration $p/3$, and reveal the edges of the graph in two phases.

In Phase 1, we reveal the edges whose time labels are in one of the intervals I_1 and I_3. Using a result from [9] (Lemma D.2), we can conclude that there are $n - o(n)$ nodes (call them X), each of which a.a.s. reaches at least $\sqrt[3]{n} \log n$ vertices during I_1, and there are at least $n - o(n)$ nodes (call them Y) that a.a.s. is reached by at least $\sqrt[3]{n} \log n$ vertices during I_3.

In Phase 2, we reveal the edges appearing during the middle interval I_2. We show that for every ordered pair of nodes x, y in the set $Z := X \cap Y$ (which is our intended connected component), the set of vertices that x can reach during I_1, can reach during I_2 at least one vertex in the set of vertices that reach y during I_3; thus implying that x can reach y during $[0, p]$. For this purpose we can employ Theorem 3.3 with S being the set that x can reach during I_1 and T being the set of vertices that can reach y during I_3. Note that the analysis of this phase is what requires us to develop the generalization $\mathcal{F}_p(G)$ of the model $\mathcal{F}_{n,p}$. In fact, the static base graph G used in the application of Theorem 3.3 is the graph obtained from K_n by removing the edges that appeared during either I_1 or I_3. Finally a union bound over all pairs of nodes x and y yields the result. ◀

The remainder of this section is dedicated to proving Theorem 4.3. Throughout, we denote $\varepsilon(n) := \frac{1}{\log \log n}$.

Let $p = (1 + \varepsilon(n)) \cdot \frac{\log n}{n}$ and $\mathcal{G} \sim \mathcal{F}_p(K_n)$. We will prove Theorem 4.3 only for this value of p as it will then clearly follow for any larger value. Our strategy is to split the interval $[0, p]$ into three sub-intervals $[p_0, p_1]$, $[p_1, p_2]$, $[p_2, p_3]$, where $p_i := \frac{i}{3}(1 + \varepsilon(n))\frac{\log n}{n}$ for $i \in [0, 3]$. We now first deduce the following corollary about the connectivity of the subgraphs $\mathcal{G}_{[p_i, p_{i+1}]}$ for $i \in [0, 2]$ of $\mathcal{G}$ from Lemma D.2.

▶ **Corollary 4.4.** *For $i \in [0, 2]$, the number of vertices* reached by *(resp.* reaching*) a fixed vertex in $\mathcal{G}_{[p_i, p_{i+1}]}$ lies within $[n^{1/3} \log n, n^{1/3 + \varepsilon(n)}]$ with probability at least $1 - \frac{10}{\log n}$.*

For space reasons, the proof of Corollary 4.4 is found in Appendix E of the full version.

Using Markov's inequality we can obtain that, a.a.s., almost all nodes can reach (resp. be reached by) the above number of nodes.

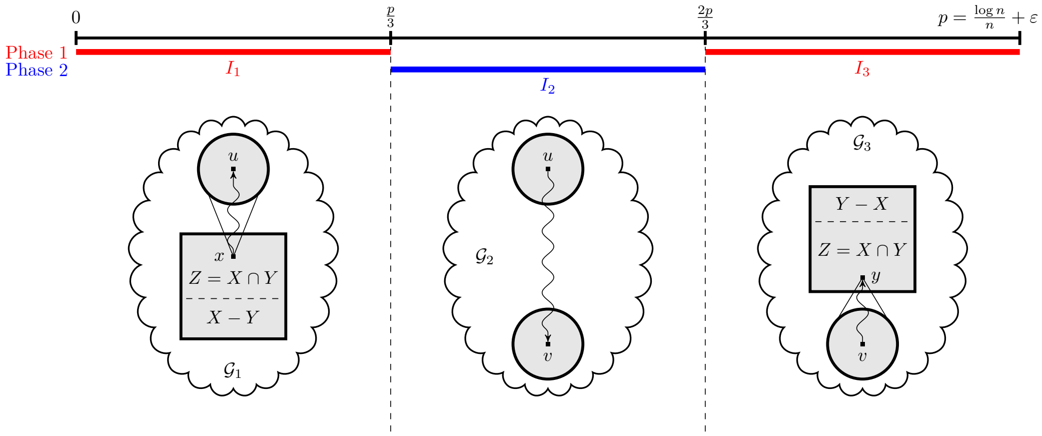


Figure 2 General strategy for upper bounding the value of p in the case of open components. Here, $\mathcal{G}_i$ denotes the restriction of the temporal graph to subinterval I_i. Wavy lines denote temporal paths. We show that any node $x \in Z$ can reach any other node $y \in Z$ by reaching a node u in $\mathcal{G}_1$, then a node v in $\mathcal{G}_2$, and finally y in $\mathcal{G}_3$.

▶ **Lemma 4.5.** *Let $i \in \{0, 1, 2\}$. The number of vertices that can reach (resp. be reached by) at least $n^{1/3} \log n$ and at most $n^{1/3+\varepsilon(n)}$ vertices in $\mathcal{G}_{[p_i,p_{i+1}]}$ is at least $n - \frac{n}{\log \log n}$ with probability at least $1 - \frac{10 \log \log n}{\log n}$.*

Proof. Let $\bar{X}$ denote the number of nodes in $\mathcal{G}_{[p_i,p_{i+1}]}$ that can reach (resp. be reached by) less than $n^{1/3} \log n$ or more than $n^{1/3+\varepsilon(n)}$ vertices in $\mathcal{G}_{[p_i,p_{i+1}]}$. Then $\mathbb{E}[\bar{X}] \leq 10n/\log n$ by Corollary 4.4. Using Markov's inequality $\mathbb{P}\left[\bar{X} \geq \frac{n}{\log \log n}\right] \leq \frac{10 \log \log n}{\log n}$. ◀

We now denote by X the set of nodes that can reach at least $n^{1/3} \log n$ and at most $n^{1/3+\varepsilon(n)}$ vertices in $\mathcal{G}_{[0,p_1]}$ and by Y the set of nodes that are reached by at least $n^{1/3} \log n$ and at most $n^{1/3+\varepsilon(n)}$ vertices in $\mathcal{G}_{[p_2,p_3]}$. Furthermore, we denote by $Z = X \cap Y$ their intersection. According to Lemma 4.5, it holds that $|Z| \geq n - \frac{2n}{\log \log n}$ with probability at least $1 - \frac{20 \log \log n}{\log n}$. The hardest part of our proof is to now show that, for a fixed ordered pair $x, y \in Z$, the probability that there is a temporal path from x to y is so large that we can take a union bound over all ordered pairs. To this end, let $A(x)$ be the set of nodes that x can reach in $\mathcal{G}_{[0,p_1]}$ and let $B(y)$ be the set of nodes that can reach y in $\mathcal{G}_{[p_2,p_3]}$. Furthermore, for $x \in X$, let

$$A'(x) := \{v \in V : \exists a \in A(x) \text{ s.t. } a \text{ reaches } v \text{ in } \mathcal{G}_{[p_1,p_2]}\}$$

be the set of nodes that x can reach in $\mathcal{G}_{[0,p_2]}$. Notice that x reaches y if and only if $A'(x)$ intersects $B(y)$.

Let $G'' = (V, E'')$ with $E'' = \{e \in \binom{V}{2} \mid \lambda(e) \in [0, p_1] \cup [p_2, p_3]\}$ be the graph containing all edges appearing in $\mathcal{G}_{[0,p_1]}$ or $\mathcal{G}_{[p_2,p_3]}$, and let $G' = (V, E')$ with $E' = \binom{V}{2} \setminus E''$ contain all other edges. Then we observe that the distribution of the set $A'(x)$ conditioned on the information about the edges appearing in $\mathcal{G}_{[0,p_1]}$ and $\mathcal{G}_{[p_2,p_3]}$ is identical to the node set of a foremost forest grown from $S := A(x)$ in $\mathcal{H} \sim \mathcal{F}_{p'}(G')$, where $p' = \frac{1}{3}(1 + \varepsilon(n))\frac{\log n}{n}$. Furthermore, G'' is distributed as an Erdős-Rényi graph $G'' \sim G_{n,p}$ with $p := \frac{2}{3}(1+\varepsilon(n))\frac{\log n}{n}$. From a standard result regarding the maximum degree in $G_{n,p}$ we can thus conclude the following fact.

▶ **Observation 4.6.** *It holds that $\Delta(G'') \leq 4\log n$ a.a.s. and, thus, $\delta(G') \geq n - (\log n)^2$ a.a.s.*

Proof. Recall that G'' is distributed according to $G_{n,p}$ with $p := \frac{2}{3}(1+\varepsilon(n))\frac{\log n}{n}$. Following [7, Corollary 3.13], with $m = 1$ and $\omega(n) = \log n$, we have that a.a.s.

$$\Delta(G'') \leq pn + \sqrt{2pn\log n} + \log n\sqrt{\frac{pn}{\log n}} \leq \log n + \sqrt{2(\log n)^2} + \log n \leq 4\log n.$$

The observation about the minimum degree now follows immediately for sufficiently large n. ◀

Thus, in order to lower bound the probability that $A'(x)$ intersects $B(y)$, we can use the following corollary of Theorem 3.3.

▶ **Corollary 4.7.** *Let*

- *G be a graph of minimum degree $\delta(G) \geq n - (\log n)^a$ for some $a \in \mathbb{N}$,*
- *let S and T be two sets of nodes in G, each of cardinality at least $n^{1/3}\log n$, and*
- *let $\mathcal{G} \sim \mathcal{F}_p(G)$ with $p \geq \frac{1}{3}(1+\varepsilon(n))\frac{\log n}{n}$.*

Then, the foremost forest algorithm from S on $\mathcal{G}$ reaches T with probability at least $1 - 3n^{-\log\log n}$.

Proof. Set $s := |S|$, $t := |T|$. Without loss of generality, we may assume $s \leq n^{1/3+\varepsilon(n)}$. Note that for large enough n it holds that

$$\begin{aligned}
p &\geq \frac{1}{3}\Big(1 + \frac{1}{\log\log n}\Big)\frac{\log n}{n} \\
&\geq \frac{\frac{1}{3}\log n + 4\log\log n}{n} \\
&= \frac{\frac{2}{3}\log n + 2\log\log n - \frac{1}{3}\log n - \log\log n}{n} + \frac{3\log\log n}{n} \\
&\geq \frac{z\log n - \log s}{n} + \frac{3\log\log n}{n},
\end{aligned}$$

for $z = \frac{2}{3} + \frac{2\log\log n}{\log n}$. From Theorem 3.3 it then follows that the foremost forest algorithm from S reaches T with probability at least

$$\begin{aligned}
1 - \frac{5}{2}n^{-\log\log n} - e^{-\frac{t}{2n}(n^z - s)} &\geq 1 - \frac{5}{2}n^{-\log\log n} - e^{-\frac{n^{1/3}\log n}{2n}(n^{2/3}(\log n)^2 - n^{1/3+\varepsilon(n)})} \\
&\geq 1 - \frac{5}{2}n^{-\log\log n} - e^{-\frac{(\log n)^3}{4}} \geq 1 - 3n^{-\log\log n},
\end{aligned}$$

completing the proof. ◀

Using the above stated corollary, we can finally prove our first main result.

Proof of Theorem 4.3. Let $p = (1+\varepsilon(n)) \cdot \frac{\log n}{n}$ and $\mathcal{G} \sim \mathcal{F}_p(K_n)$. As above, let X be the nodes that can reach between $n^{1/3}\log n$ and $n^{1/3+\varepsilon(n)}$ vertices in $\mathcal{G}_{[0,p_1]}$ and let Y be the nodes that are reached by between $n^{1/3}\log n$ and $n^{1/3+\varepsilon(n)}$ vertices in $\mathcal{G}_{[p_2,p_3]}$. Furthermore, let $Z = X \cap Y$ be their intersection and recall that $|Z| \geq n - \frac{2n}{\log\log n}$ with probability at least $1 - \frac{20\log\log n}{\log n}$ according to Lemma 4.5. Now, conditioned on the information about the edges appearing in $\mathcal{G}_{[0,p_1]}$ and $\mathcal{G}_{[p_2,p_3]}$, let $G' = (V, E')$ be the static graph with the same node set as $\mathcal{G}$ and the edge set $E' = \{e \in \binom{V}{2} : \lambda(e) \notin [p_0,p_1] \cup [p_2,p_3]\}$, where λ is the time label function of $\mathcal{G}$. Note that according to Observation 4.6 the minimum degree in G' a.a.s. is at

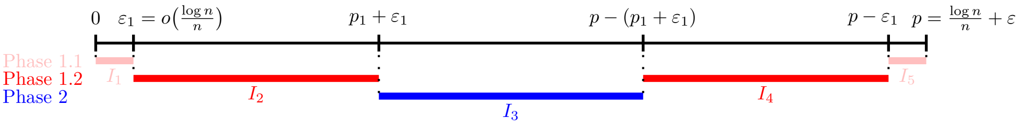

0
$\varepsilon_1 = o(\frac{\log n}{n})$
$p_1 + \varepsilon_1$
$p - (p_1 + \varepsilon_1)$
$p - \varepsilon_1$
$p = \frac{\log n}{n} + \varepsilon$
Phase 1.1
Phase 1.2
Phase 2
I_1
I_2
I_3
I_4
I_5

Figure 3 Illustration of the three different phases in our proof for the case of closed components. Here, the length p_1 of I_2 and I_4 and the length p_2 of I_3 are each roughly $\frac{1}{3}\frac{\log n}{n}$. In Phase 1.1, we reveal edges in I_1 and I_5 and identify our target closed connected component, a set of n' nodes V' each of which can reach (be reached by) poly-logarithmically many vertices within V' during I_1 (I_5) via temporal paths in V'. For Phase 1.2 (consisting of intervals I_2 and I_4) we show that every vertex in $v \in V'$ reaches (is reached by) polynomially many vertices in V' during $I_1 \cup I_2$ ($I_4 \cup I_5$). We then show that during Phase 2 (consisting of I_3), for each ordered pair of vertices $u, w \in V'$, the set of vertices reached by u during $I_1 \cup I_2$ can reach the set of vertices that reach w during $I_4 \cup I_5$, implying that u can reach w during $[0, p]$.

least $n - (\log n)^2$. Now, let $x, y \in Z$ be a fixed ordered pair of vertices. Applying Corollary 4.7 to $\mathcal{H} \sim \mathcal{F}_{\frac{p}{3}}(G')$ with $S = A(x)$, $a = 2$, and $T = B(y)$, we can conclude that $A'(x) \cap B(y) \neq \emptyset$ with probability at least $1 - 3n^{-\log\log n}$, and, thus, x reaches y with at least that probability. Hence, after a union bound over all ordered pairs, we get that all nodes in Z reach each other with probability at least $1 - 3n^{-\log\log n + 2}$. Therefore, $\mathcal{G}$ has a temporally connected component of size at least $n - \frac{2n}{\log\log n} = n - o(n)$ a.a.s. ◀

5 Sharp Threshold for Giant Closed Connected Component

In this section we report on the result that $\frac{\log n}{n}$ is also a sharp threshold for the existence of a giant *closed* connected component. We first sketch the general proof idea; the formal proof given subsequently is based upon a lemma proven in Appendix F of the full version.

▶ **Theorem 5.1** (Main Result for Closed Components). *The function $\frac{\log n}{n}$ is a sharp threshold for Giant Closed Connected Component. More precisely, there exists a function $\varepsilon(n) \in o\left(\frac{\log n}{n}\right)$, such that the size of a largest closed temporally connected component in $\mathcal{G} \sim \mathcal{F}_{n,p}$ is*

(i) *$o(n)$ a.a.s., if $p < \frac{\log n}{n} - \varepsilon(n)$; and*
(ii) *$n - o(n)$ a.a.s., if $p > \frac{\log n}{n} + \varepsilon(n)$.*

Proof Sketch. The lower bound of Theorem 5.1 is obviously a trivial consequence of the lower bound in Theorem 4.1. Thus, it remains to prove the upper bound. We start from our strategy of splitting the time into three intervals. We do not need to make any changes to our approach in the middle one (Phase 2), which previously required the most effort. However, we now need to do additional work in the first and last interval (Phase 1), which is the main technical contribution of this part. Recall that in the proof of Theorem 4.3, we only required that $n - o(n)$ vertices can all reach (resp. be reached by) at least $n^{1/3} \log n$ vertices within each of the three intervals from Figure 2. Now, we will need to prove that there exists a set V' of $n - o(n)$ vertices, such that every vertex in this set can reach (resp. be reached by) at least $n^{1/3} \log n$ vertices *via temporal paths that use only vertices in V'*. Once this is achieved, we can use the same approach as in the case of open components for Phase 2.

In order to obtain the set of vertices V' mentioned above, we have to insert an additional Phase 1.1, which looks only at a short time interval I_1 at the very beginning (and symmetrically I_5 at the very end). The purpose of this new phase is to "bootstrap" the closed component by identifying a set V' of $n' = n - o(n)$ vertices, which each reach at least *poly-logarithmically* many vertices by paths that are contained in V'. Lemma 5.2 formalizes this result.

A technical difficulty in Phase 1.1 is the need to control possible cascading effects, where removing low-degree vertices from the graph can cause further vertices to become low-degree vertices, etc. We overcome this difficulty by partitioning the vertices into sectors $V_1, \dots, V_C$ and removing vertices from each sector V_i solely on the base of whether they have too few neighbors in the *next sector* V_{i+1}. This ensures that the sets of vertices removed from each sector are determined independently of each other. On this base, we are then able to prove that no cascading effects occur a.a.s. Subsequently, we show that after these removals, every remaining vertex can reach a poly-logarithmic number of others by considering *clocked* paths, which essentially march in lockstep, traversing the sectors in circular order (see Figure 4).

Subsequently, in Phase 1.2, we reveal edges that appear during I_2 or I_4. We use the foremost forest technique developed earlier to show that, conditioned on the edges revealed in Phase 1.1, for every vertex v in V' the poly-logarithmic set of vertices reached by v during I_1 reaches *polynomially many* (by which we mean n^p for some fixed $p < 1$) vertices during I_2. (Similarly, the set of vertices that reach v during I_5 is reached by polynomially many vertices during I_4.) ◀

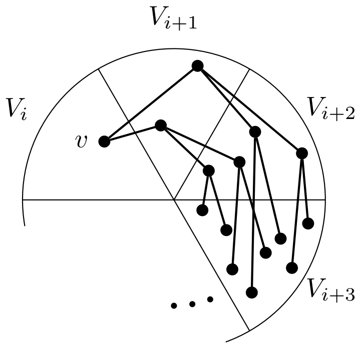


Figure 4 Example of a temporal tree formed by clocked paths starting at a vertex $v \in V_i$. By restricting the edges used between sectors V_{i+j} and V_{i+j+1} to an appropriate time interval I_{j-i}, we ensure that the time labels of all these paths are monotonically clockwise increasing.

▶ **Lemma 5.2.** *Let $C \geq 3$, $\frac{1}{2} < \gamma < \alpha < 1$, and let $\mathcal{G} \sim \mathcal{F}_{n,p}$, where $p = 2C^2 \frac{(\log n)^\alpha}{n}$. Then a.a.s. $\mathcal{G}$ contains a set V' of $n - o(n)$ vertices, such that, denoting $\mathcal{G}' := \mathcal{G}[V']$, every vertex in V' reaches at least $(\log n)^{(C-3)\gamma}$ vertices in $\mathcal{G}'_{[0,p/2]}$ and is reached by at least the same number of vertices in $\mathcal{G}'_{[p/2,p]}$.*

In the rest of this section we prove Theorem 5.1 using the above lemma, whose proof is found in Appendix F of the full version.

Proof of Theorem 5.1. Let $\gamma = 0.7$, $\alpha = 0.9$, $C = 30$, and let $n'(n) \in n - o(n)$ be the size of the vertex set guaranteed by Lemma 5.2. Set $\varepsilon_1 := C^2 \frac{(\log n)^\alpha}{n} \in o\Big(\frac{\log n}{n}\Big)$, $\varepsilon_2 := 4\frac{\log\log n'}{\log n'} \in o(1)$, and $\varepsilon_3 := \frac{1}{3\log\log n} \in o(1)$. Set also $p_1 := \left(\frac{1}{3} + \varepsilon_2\right) \frac{\log n'}{n'}$ and $p_2 := \left(\frac{1}{3} + \varepsilon_3\right) \frac{\log n'}{n'}$. Finally, define $p := 2\varepsilon_1 + 2p_1 + p_2$, which is equal to $\frac{\log n}{n} + \varepsilon$ for some $\varepsilon \in o\Big(\frac{\log n}{n}\Big)$.

Let $\mathcal{G} \sim \mathcal{F}_{n,p}$. We split $[0,p]$ into a total of five intervals I_i, $i \in [5]$. The first and the last interval are short and each has length ε_1, i.e., $I_1 = [0, \varepsilon_1]$ and $I_5 = [p - \varepsilon_1, p]$. The three middle intervals are long and have lengths p_1, p_2, and p_1, respectively, i.e., $I_2 = [\varepsilon_1, \varepsilon_1 + p_1]$, $I_3 = [\varepsilon_1 + p_1, \varepsilon_1 + p_1 + p_2]$, $I_4 = [p - (\varepsilon_1 + p_1), p - \varepsilon_1]$. We will reveal the edges of the graph in three phases (Phase 1.1, Phase, 1.2, and Phase 2), as was graphically summarized in Figure 3 in the introduction, and in each phase we condition on the edges revealed in the previous phases. In Phase 1.1 we reveal edges in the intervals I_1 and I_5 and apply Lemma 5.2 to identify a large set of nodes V', each of which can reach poly-logarithmically many vertices in V' during the first interval and can be reached by poly-logarithmically many vertices in V' during the last interval via temporal paths that use only nodes from V'. In the subsequent phases we restrict our attention to the subgraph induced by V', which is the target giant closed connected component. In Phase 1.2, we reveal edges appearing in the intervals I_2 and I_4. Because in Phase 1.1 a.a.s. we revealed poly-logarithmic number of edges for every vertex, we can use Lemma C.10 to argue that for every vertex $v \in V'$ the poly-logarithmic set of vertices reached by v during I_1 can reach polynomially many vertices during I_2. Similarly, during I_4 polynomially many vertices can reach the poly-logarithmic set of vertices that reach v during I_5. The main outcome of this phase is that every vertex in $v \in V'$ reaches polynomially many vertices in V' during $I_1 \cup I_2$ and is reached by at least as many vertices in $I_4 \cup I_5$. Finally, in Phase 2, because in the previous phases a.a.s. at most poly-logarithmically many edges were revealed for every vertex, we can apply Corollary 4.7 to prove that for each ordered pair of vertices $u, w \in V'$ the set of vertices reached by u during $I_1 \cup I_2$ can reach during I_3 the set of vertices that reach w during $I_4 \cup I_5$, implying that u can reach w during $[0, p]$. We now proceed with the formal details.

Phase 1.1. Let $\mathcal{G}_1$ be the temporal subgraph of $\mathcal{G}$ formed by the edges with time labels in the intervals $I_1 \cup I_5$. Note that, up to shifting the time labels in the interval I_5 by $p - 2\varepsilon_1$, $\mathcal{G}_1$ is distributed according to $\mathcal{F}_{n,2\varepsilon_1}$. Thus, by Lemma 5.2, a.a.s. there is a set $V' \subseteq V(\mathcal{G})$ containing n' vertices such that, denoting $\mathcal{G}' := \mathcal{G}[V']$, every vertex $v \in V'$ reaches a set $R_1(v)$ of at least $(\log n)^{(C-3)\gamma}$ vertices in $\mathcal{G}'_{I_1}$ and is reached by a set $R'_1(v)$ of at least $(\log n)^{(C-3)\gamma}$ vertices in $\mathcal{G}'_{I_5}$.

Phase 1.2. Let G_1 be the underlying graph of $\mathcal{G}_1$. Since G_1 is distributed as an Erdős-Rényi graph $G_{n,\,2\varepsilon_1}$, similarly to Observation 4.6, we have that $\Delta(G_1) < 4\log n$ a.a.s. Hence, in the graph $G'_2 = \left(V', \binom{V'}{2} \setminus E(G_1)\right)$ the minimum degree is at least $n' - 4\log n \geq n' - (\log n')^2$. Observe that, up to shifting time labels, $\mathcal{G}'_{I_2} \sim \mathcal{F}_{p_1}(G'_2)$ when conditioning on the knowledge about all edges seen in $I_1 \cup I_5$. Therefore, since $|R_1(v)| \geq (\log n')^{13}$ for every vertex $v \in V'$, by applying Lemma C.10 to $\mathcal{G}'_{I_2}$ and $R_1(v)$ (with parameter $z = 1/3 + \frac{\log\log n'}{\log n'}$), we conclude that the vertices in $R_1(v)$ reach in $\mathcal{G}'_{I_2}$ at least $r := (n')^{\frac{1}{3}+\frac{\log\log n'}{\log n'}} = (n')^{1/3}\log n'$ vertices with probability at least $1 - 2(n')^{-\log\log n'}$. By the union bound, we have that with probability at least $1 - 2(n')^{1-\log\log n'} \in 1 - o(1)$, every vertex $v \in V'$ can reach in $\mathcal{G}'_{I_1 \cup I_2}$ a set $R_2(v)$ of at least r vertices. Symmetrically, with probability at least $1 - 2(n')^{1-\log\log n'} \in 1 - o(1)$, every vertex $v \in V'$ is reached in $\mathcal{G}'_{I_4 \cup I_5}$ by a set $R'_2(v)$ of at least r vertices.

Phase 2. Let G'_3 be the static graph defined by the vertex set V' and all edges appearing in $\mathcal{G}'_{I_1 \cup I_2}$ and $\mathcal{G}'_{I_4 \cup I_5}$. As in Phase 1.2, we can argue that the maximum degree of G'_3 is at most $4\log n'$ a.a.s., and therefore the minimum degree of the graph $G'_4 = \left(V', \binom{V'}{2} \setminus E(G'_3)\right)$ is at least $n' - (\log n')^2$. Hence, up to shifting time labels, $\mathcal{G}'_{I_3}$ is distributed according to $\mathcal{F}_{p_2}(G'_4)$ when conditioned on the knowledge of all edges revealed in $I_1 \cup I_2 \cup I_4 \cup I_5$. Thus, by Corollary 4.7, a given set of at least $(n')^{1/3}\log n'$ vertices in $\mathcal{G}'_{I_3}$ can reach another given set of at least as many vertices with probability at least $1 - 3(n')^{-\log\log n'}$. Applying this to all ordered pairs of sets $(R_2(v), R'_2(w))$, $v, w \in V'$ and using the union bound, we conclude that the probability that all these pairs of sets reach each other in $\mathcal{G}'_{I_3}$ is at least $1 - 3(n')^{2-\log\log n'} \in 1 - o(1)$.

Putting all together, we conclude that a.a.s. in $\mathcal{G}' = \mathcal{G}[V']$ any vertex reaches every other vertex. Thus, V' is, as desired, a giant closed connected component. ◀

References

1 Eleni C. Akrida, Leszek Gasieniec, George B. Mertzios, and Paul G. Spirakis. The complexity of optimal design of temporally connected graphs. *Theory of Computing Systems*, 61(3):907–944, 2017. doi:10.1007/s00224-017-9757-x.

2 Eleni C. Akrida, George B. Mertzios, Sotiris E. Nikoletseas, Christoforos L. Raptopoulos, Paul G. Spirakis, and Viktor Zamaraev. How fast can we reach a target vertex in stochastic temporal graphs? *Journal of Computer and System Sciences*, 114:65–83, 2020. doi:10.1016/j.jcss.2020.05.005.

3 Albert-László Barabási. *Network Science.* Cambridge University Press, 2016.

4 Albert-Lázló Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999. doi:10.1126/science.286.5439.509.

5 Hervé Baumann, Pierluigi Crescenzi, and Pierre Fraigniaud. Parsimonious flooding in dynamic graphs. *Distributed Computing*, 24(1):31–44, 2011. doi:10.1007/s00446-011-0133-9.

6 Sandeep Bhadra and Afonso Ferreira. Complexity of connected components in evolving graphs and the computation of multicast trees in dynamic networks. In *Proceedings of the 2nd International Conference on Ad-Hoc Networks and Wireless (ADHOC-NOW)*, pages 259–270, 2003. doi:10.1007/978-3-540-39611-6_23.

7 Béla Bollobás. *Random Graphs.* Cambridge University Press, 2 edition, 2001. doi:10.1017/CBO9780511814068.

8 Arnaud Casteigts, Anne-Sophie Himmel, Hendrik Molter, and Philipp Zschoche. Finding temporal paths under waiting time constraints. *Algorithmica*, 83(9):2754–2802, 2021. doi:10.1007/s00453-021-00831-w.

9 Arnaud Casteigts, Michael Raskin, Malte Renken, and Viktor Zamaraev. Sharp thresholds in random simple temporal graphs. In *Proceedings of the 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 319–326, 2021. doi:10.1109/FOCS52979.2021.00040.

10 Augustin Chaintreau, Abderrahmen Mtibaa, Laurent Massoulie, and Christophe Diot. The diameter of opportunistic mobile networks. In *Proceedings of the 3rd International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, pages 1–12, 2007. doi:10.1145/1364654.1364670.

11 Andrea Clementi, Angelo Monti, Francesco Pasquale, and Riccardo Silvestri. Broadcasting in dynamic radio networks. *Journal of Computer and System Sciences*, 75(4):213–230, 2009. doi:10.1016/J.JCSS.2008.10.004.

12 Andrea E. F. Clementi, Claudio Macci, Angelo Monti, Francesco Pasquale, and Riccardo Silvestri. Flooding time of edge-markovian evolving graphs. *SIAM Journal on Discrete Mathematics*, 24(4):1694–1712, 2010. doi:10.1137/090756053.

13 Alessio Conte, Pierluigi Crescenzi, Andrea Marino, and Giulia Punzi. Enumeration of s-d separators in DAGs with application to reliability analysis in temporal graphs. In *Proceedings of the 45th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 25:1–25:14, 2020. doi:10.4230/LIPIcs.MFCS.2020.25.

14 Jessica Enright, Kitty Meeks, George B. Mertzios, and Viktor Zamaraev. Deleting edges to restrict the size of an epidemic in temporal networks. *Journal of Computer and System Sciences*, 119:60–77, 2021. doi:10.1016/j.jcss.2021.01.007.

15 Jessica Enright, Kitty Meeks, and Fiona Skerman. Assigning times to minimise reachability in temporal graphs. *Journal of Computer and System Sciences*, 115:169–186, 2021. doi:10.1016/j.jcss.2020.08.001.

16 P. Erdős and A. Rényi. On random graphs I. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959. URL: https://publi.math.unideb.hu/load_doc.php?p=2769&t=pap.

17 P. Erdős and A. Rényi. On the strength of connectedness of a random graph. *Acta Mathematica Academiae Scientiarum Hungarica*, 12(1):261–267, 1964. doi:10.1007/BF02066689.

18 P. Erdős and A. Rényi. On the existence of a factor of degree one of a connected random graph. *Acta Mathematica Academiae Scientiarum Hungarica*, 17:359–368, 1966. doi:10.1007/BF01894879.

19 P. Erdős and A. Rényi. On the evolution of random graphs. *A Magyar Tudományos Akadémia. Matematikai Kutató Intézetének Közleményei*, 5:17–61, 1960. URL: `http://www.renyi.hu/~p_erdos/1960-10.pdf`.

20 Till Fluschnik, Hendrik Molter, Rolf Niedermeier, Malte Renken, and Philipp Zschoche. Temporal graph classes: A view through temporal separators. *Theoretical Computer Science*, 806:197–218, 2020. `doi:10.1016/j.tcs.2019.03.031`.

21 Alan Frieze and Michał Karoński. *Introduction to Random Graphs*. Cambridge University Press, 2015. `doi:10.1017/CBO9781316339831`.

22 Emmanuel Godard and Dorian Mazauric. Computing the dynamic diameter of non-deterministic dynamic networks is hard. In *Proceedings of the 10th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (ALGOSENSORS)*, pages 88–102, 2014. `doi:10.1007/978-3-662-46018-4_6`.

23 Peter Grindrod and Desmond J. Higham. Evolving graphs: dynamical models, inverse problems and propagation. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 466(2115):753–770, 2010. `doi:10.1098/rspa.2009.0456`.

24 Petter Holme and Jari Saramäki. Temporal networks. *Physics Reports*, 519(3):97–125, 2012. `doi:10.1016/j.physrep.2012.03.001`.

25 Petter Holme and Jari Saramäki, editors. *Temporal Network Theory*. Springer, 2019. `doi:10.1007/978-3-030-23495-9`.

26 Richard M. Karp. The transitive closure of a random digraph. *Random Structures & Algorithms*, 1(1):73–93, 1990. `doi:10.1002/rsa.3240010106`.

27 David Kempe, Jon Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. *Journal of Computer and System Sciences*, 64(4):820–842, 2002. `doi:10.1006/jcss.2002.1829`.

28 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. `doi:10.1002/RSA.3240060204`.

29 Michael Molloy and Bruce Reed. The size of the giant component of a random graph with a given degree sequence. *Combinatorics, Probability and Computing*, 7(3):295–305, 1998. `doi:10.1017/S0963548398003526`.

30 M. E. J. Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003. `doi:10.1137/S003614450342480`.

31 M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64(2):026118, 2001. `doi:10.1103/PhysRevE.64.026118`.

32 Mark Newman. *Networks*. Oxford University Press, 2018. `doi:10.1093/oso/9780198805090.001.0001`.

33 John Tang, Ilias Leontiadis, Salvatore Scellato, Vincenzo Nicosia, Cecilia Mascolo, Mirco Musolesi, and Vito Latora. Applications of temporal graph metrics to real-world networks. In *Temporal Networks*, pages 135–159. Springer, 2013. `doi:10.1007/978-3-642-36461-7_7`.

34 D. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998. `doi:10.1038/30918`.

35 Xiao Zhang, Cristopher Moore, and Mark Newman. Random graph models for dynamic networks. *The European Physical Journal B*, 90(10):1–14, 2017. `doi:10.1140/epjb/e2017-80122-8`.

On Connectivity in Random Graph Models with Limited Dependencies

Johannes Lengler
Department of Computer Science,
ETH Zürich, Switzerland

Anders Martinsson
Department of Computer Science,
ETH Zürich, Switzerland

Kalina Petrova
Department of Computer Science,
ETH Zürich, Switzerland

Patrick Schnider
Department of Computer Science,
ETH Zürich, Switzerland

Raphael Steiner
Department of Computer Science,
ETH Zürich, Switzerland

Simon Weber
Department of Computer Science,
ETH Zürich, Switzerland

Emo Welzl
Department of Computer Science,
ETH Zürich, Switzerland

Abstract

For any positive edge density p, a random graph in the Erdős-Rényi $G_{n,p}$ model is connected with non-zero probability, since all edges are mutually independent. We consider random graph models in which edges that do not share endpoints are independent while incident edges may be dependent and ask: what is the minimum probability $\rho(n)$, such that for any distribution $\mathcal{G}$ (in this model) on graphs with n vertices in which each potential edge has a marginal probability of being present at least $\rho(n)$, a graph drawn from $\mathcal{G}$ is connected with non-zero probability?

As it turns out, the condition "edges that do not share endpoints are independent" needs to be clarified and the answer to the question above is sensitive to the specification. In fact, we formalize this intuitive description into a strict hierarchy of five independence conditions, which we show to have at least three different behaviors for the threshold $\rho(n)$. For each condition, we provide upper and lower bounds for $\rho(n)$. In the strongest condition, the *coloring model* (which includes, e.g., random geometric graphs), we show that $\rho(n) \to 2 - \phi \approx 0.38$ for $n \to \infty$, proving a conjecture by Badakhshian, Falgas-Ravry, and Sharifzadeh. This separates the coloring models from the weaker independence conditions we consider, as there we prove that $\rho(n) > 0.5 - o(n)$. In stark contrast to the coloring model, for our weakest independence condition – pairwise independence of non-adjacent edges – we show that $\rho(n)$ lies within $O(1/n^2)$ of the threshold $1 - 2/n$ for completely arbitrary distributions.

2012 ACM Subject Classification Mathematics of computing → Random graphs; Theory of computation → Randomness, geometry and discrete structures

Keywords and phrases Random Graphs, Independence, Dependency, Connectivity, Threshold, Probabilistic Method

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.30

Category RANDOM

Related Version *Full Version*: https://arxiv.org/abs/2305.02974

Funding *Kalina Petrova*: Swiss National Science Foundation under grant no. CRSII5 173721.
Raphael Steiner: ETH Zürich Postdoctoral Fellowship.
Simon Weber: Swiss National Science Foundation under project no. 204320.

Acknowledgements This research started at the joint workshop of the *Combinatorial Structures and Algorithms* and *Theory of Combinatorial Algorithms* groups of ETH Zürich held in Stels, Switzerland, January 2023. We thank the organizers for providing a very pleasant and inspiring working atmosphere. We thank Victor Falgas-Ravry for pointing out an abundance of related work, thus helping tremendously to put our results in perspective.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques
(APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 30; pp. 30:1–30:22
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

The probabilistic method is an important tool in theoretical computer science, graph theory and combinatorics [3]. With this method, one proves that a random construction has some desirable property with positive probability, and concludes that some objects with this property must exist. Early examples of the probabilistic method include simple constructions of expander graphs [7], or graphs with high girth and large chromatic number [16]. Often, it is possible to express the property as the intersection of very simple events $A_1, \dots, A_k$. For example, we can express the property that a set S is a clique as the intersection of the events "u is adjacent to v" for all pairs $u, v \in S$. In many cases it is clear that each of the A_i has a large probability to occur, but this is generally not enough to conclude that the intersection of all A_i has positive probability to occur. Of course, if the probability space were a *product space* in which all components are *independent* of each other, then this would be trivial.

However, for many applications it is too limiting to restrict oneself to product spaces. The probabilistic method was significantly extended by the realization that it could still be applied to settings without perfect independence, provided there is some bound on the amount of dependence in the system. The seminal result was the Lovász Local Lemma (LLL), which we give here in the slightly stronger version due to Shearer [28]: for events $A_1, \dots, A_k$ that all occur with probability at least p, if each of the A_i depends on at most d other A_j, where $p \geq 1 - 1/(ed)$, then there is a positive probability that all of the A_i occur simultaneously.[1] In this context, the dependencies are captured by a *dependency graph*, a graph with vertex set $\{A_1, \dots, A_k\}$ such that each vertex is independent from all but its neighbors. There are many situations in which the dependency graph can be restricted, and we give some examples in Section 1.1 below.

The LLL allows to re-introduce a product space through the backdoor, because the LLL condition $p \geq 1 - 1/(ed)$ allows to couple the process to a product space. This was already implicit in the inductive proof of the LLL [29], and was made explicit by Liggett, Schonmann and Spacey [23]. They also generalized this coupling to a countably infinite number of variables and showed tightness of Shearer's condition (of the precise version in footnote 1). For a finite number of variables, coupling the probability space to a product space implies trivially that the all-one event has positive probability.

Unfortunately, the LLL scales badly in d, i.e., p needs to be very close to one if d is large. However, in some cases the degree d may grow. In particular, in this paper we will study the situation that the variables are associated with the edges of a complete graph on n vertices, and dependencies only run between adjacent edges (edges which share a common endpoint). This is a common situation, and examples are given in Section 1.1. In this case, the number of dependencies per edge is $2(n-2)$, and thus grows with n. Thus, if every edge is present with constant probability $p < 1$, then it is not possible to couple the probability space with a product space for large n, and it is not true that the complete graph appears with positive probability [23]. However, in this paper we will show that some weaker global properties *can* be guaranteed. Specifically, we will focus on connectivity because this is arguably the most fundamental global graph property. We will study the question:

> Consider a random graph in which every potential edge is inserted with probability at least p. Assume that non-adjacent edges are independent. For which values of p can we guarantee that the graph is connected with positive probability?

[1] The precise condition is $p > 1 - \frac{(d-1)^{(d-1)}}{d^d}$ for $d > 1$ and $p > 1/2$ for $d = 1$. The first expression can be simplified by the estimate $\frac{(d-1)^{(d-1)}}{d^{d-1}} \geq e^{-1}$, which becomes tight in the limit $d \to \infty$.

It turns out that the answer depends heavily on what exactly we mean by independence. Surprisingly, there are at least five different ways to interpret the innocent-looking condition that non-adjacent edges are independent, which we define next in order of increasing strength. For an edge $e = \{u, v\}$, let X_e be the event that e is present in the random graph.

▶ **Definition 1** (Pairwise independence). *For every pair of non-adjacent edges e, f, the random variables X_e and X_f are independent.*

▶ **Definition 2** (Matching independence). *For any set M of pairwise non-adjacent edges (also called a* matching*), $\{X_e \mid e \in M\}$ are mutually independent.*

▶ **Definition 3** (Edge-subgraph independence). *For any edge $e = \{u, v\}$, any vertex set W with $u, v \notin W$, and any choice of graph H_W on W, we have for the random graph G sampled from the distribution that*

$$Pr[G[W] = H_W \text{ and } X_e] = Pr[G[W] = H_W] \cdot Pr[X_e].$$

▶ **Definition 4** (Subgraph independence). *For any two disjoint subsets V, W of vertices, and any choice of graphs H_V and H_W on V and W, respectively, we have for the random graph G sampled from the dsitribution that*

$$Pr[G[V] = H_V \text{ and } G[W] = H_W] = Pr[G[V] = H_V] \cdot Pr[G[W] = H_W].$$

▶ **Definition 5** (Coloring model). *The graph distribution is given by a set of probability spaces (Ω_v, Pr_v), one for each vertex $v \in V$, and a set of deterministic functions $f_{u,v} : \Omega_u \times \Omega_v \to \{0, 1\}$ computing $X_{\{u,v\}}$, one for each edge $\{u, v\} \in \binom{V}{2}$. If every probability space (Ω_v, Pr_v) is finite, we call $\max_v |\Omega_v|$ the* number of colors *of the coloring model.*

We remark that the standard proof of LLL requires edge-subgraph independence. In the paper [23] that makes the coupling to product spaces explicit, the authors describe subgraph independence as the required property, but inspecting the proof shows that they only use the weaker condition of edge-subgraph independence.

Note that some of our models have been previously studied under different names. We summarize the bounds on the connectivity thresholds obtained before or simultaneously to this paper in Figure 1, and discuss this related work in more detail in Section 1.1. Our main results for large n are summarized in Table 1 and Figure 2. We give a refined exposition of our results for *all* n in Theorem 7. We do not have matching upper and lower bounds in all cases, but as can be seen in Figure 2, our bounds imply that there are at least three different thresholds among the five independence conditions discussed above, and four

Table 1 Lower and upper bounds for the threshold ρ s.t. every constant edge probability $> \rho$ guarantees connectivity with positive probability for all sufficiently large n, while a constant edge probability $< \rho$ does not guarantee connectivity with positive probability for infinitely many n.

Independence condition	Lower bound	Upper bound
Coloring model (2 colors)	1/4	1/4
Coloring model (general)	0.381966 . . .	0.381966 . . .
Subgraph independence	1/2 [14, Thm. 16]	1/2 [14, Thm. 16]
Edge-subgraph independence	1/2	3/4
Matching independence	1/2	1
Pairwise independence	1	1

different thresholds if we also include the coloring models with only 2 colors. Notably, our proofs give different intervals for the thresholds of all six independence conditions, although this does not imply that the thresholds are all different.

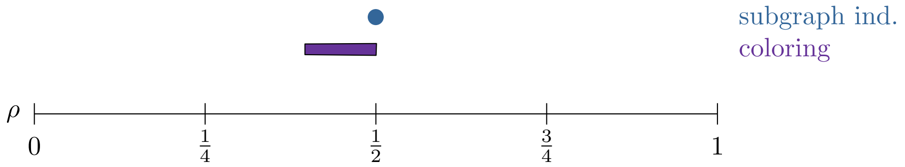


Figure 1 An illustration of the upper and lower bounds on the thresholds known from related work. The bounds for subgraph independence can be found in [14, Thm. 16]. The bounds on the coloring model have been obtained simultaneously and independently to ours [4, Thm. 1.7].

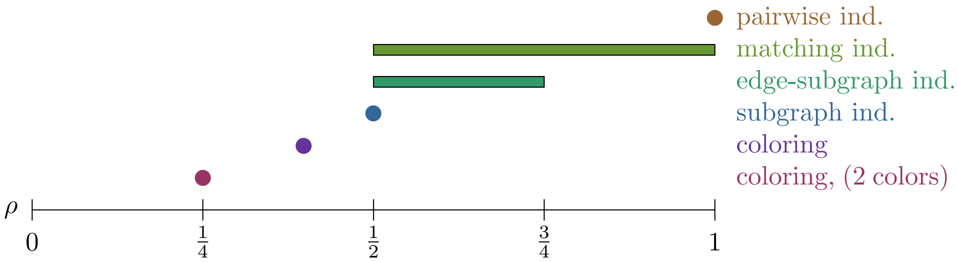


Figure 2 An illustration of the upper and lower bounds mentioned in Table 1 and Theorem 7. As can be seen, there must be at least four thresholds ρ among the six independence conditions.

Moreover, let us write $\mathcal{G}_{\text{pw}}(n,p)$ for the class of all random graph distributions on n vertices, with marginal edge probabilities at least p and with the property "pairwise independence". We write $\mathcal{G}_{\text{pw}} := \bigcup_{n\in\mathbb{N}} \mathcal{G}_{\text{pw}}(n,0)$ for the same class without restrictions on the number of vertices and the marginal probabilities. Likewise, we write $\mathcal{G}_{\text{mat}}$, $\mathcal{G}_{\text{esub}}$, $\mathcal{G}_{\text{sub}}$ and $\mathcal{G}_{\text{col}}$ for the random graph distributions satisfying matching independence, edge-subgraph independence, subgraph independence and the coloring models respectively. We show that those five models form a strict hierarchy in Appendix A.

▶ **Theorem 6.** $\mathcal{G}_{col} \subsetneq \mathcal{G}_{sub} \subsetneq \mathcal{G}_{esub} \subsetneq \mathcal{G}_{mat} \subsetneq \mathcal{G}_{pw}$.

1.1 Related Work and Examples

There are numerous applications in graph theory, theoretical computer science, and combinatorics in which dependency graphs are bounded or otherwise restricted. For example, if we want to find a vertex coloring in a hypergraph without monochromatic hyperedges, and we color the vertices independently, then any two disjoint hyperedges are independent. Therefore, often times LLL type arguments apply to coloring (hyper)graphs of bounded maximum degree. For instance, the seminal results by Johansson [20] and Molloy [24] stating that every graph of maxmimum degree Δ has (list) chromatic number $O(\frac{\Delta}{\log \Delta})$ were proved using this method. Similarly, if we want to find a satisfying assignment to a SAT-formula in conjunctive normal form, and assign the variables independently, then any two clauses are independently satisfied if they do not share a common literal, see for instance [18] for more detail and background. These are classical applications of LLL and fall into the class of coloring models.

Another important source of applications comes from percolation theory, which was also the main motivation for [23]. There, an important technique to study locally generated geometric graphs is *rescaling*: the geometric space is covered by boxes that may partially overlap, and boxes are called *good* under some conditions that depend only on the subgraph induced by the box. It is shown recursively that the probability for a box to be good increases with the box size. The goodness of two boxes is independent if they don't overlap, and typically every box only overlaps with constantly many other boxes.

So far, we gave general examples of bounded dependence. We now turn to examples where the random variables are naturally tied to the edge set of a complete graph. An important class of examples come from random graph models that are more complex than Erdős-Rényi graphs, in particular models that capture graph properties of real-world networks like clustering, communities, or heavy-tailed degree distributions. Many such models are generated by drawing some information for each vertex, and connecting two vertices based on this information. There is a large number of such random graph models: in Chung-Lu or Norros-Reittu random graphs, the vertices draw weights which determine their expected degrees [12, 27]; in Random Geometric Graphs (RGG) or Hyperbolic Random Graphs (HRG) the vertices draw some location in a geometric space [22, 26]; in scale-free percolation (SFP) the vertices lie on a grid and also draw random weights [15]; in Geometric Inhomogeneous Random Graphs (GIRG), they draw both a position and a weight [9]. In the Stochastic Block Model (SBM) they draw the community to which they should belong [21]. Many more applications of similar flavour that cannot all be listed here can be found in the literature. All these models fall in the class of coloring models.[2]

The class of coloring models is the most important one.[3] Models that do not fall into this class can arise when the graphs drawn from the distribution must fulfill some global property. For example, consider the following distribution for an *odd* number of vertices $n \geq 5$. Every vertex chooses a color from red and blue independently and uniformly at random. The resulting graph consists only of a clique on the color chosen an even number of times. We show later (in the proof of Lemma 38) that this distribution is matching independent, but not edge-subgraph independent (and thus also not a coloring model). As a second example, consider the Erdős-Rényi model $G_{n,1/2}$ with the additional side constraint that the total number of edges $|E|$ is divisible by three.[4] One can show quite easily that this distribution is not even pairwise independent. On the other hand, if we instead use the side constraint that $|E|$ is divisible by *two*, the distribution is actually a coloring model, indicating that independence conditions are surprisingly fickle. A proof of both of these facts can be found in the full version of this paper.

Two of our independence conditions have been previously considered under different names. Firstly, subgraph independent distributions have been studied under the name of 1-*independent* random graphs. More generally, a k-independent graph distribution is a distribution where any two sets of edges E, F are independent if the minimum distance

[2] The formulation of many models involves a coin flip for each edge, where the probability of inclusion is a function of the information of the two endpoints. In this formulation, the event X_e is not a *deterministic* function of the information. But one can simply define the coin flip as part of the random experiments of one of the endpoints.

[3] But note that it is not the model that arises naturally for the LLL. This raises the question whether stronger LLL-type results can be obtained for the coloring model.

[4] More formally, we consider the conditional probability distribution obtained by conditioning the $G_{n,1/2}$ distribution on the event that $|E| \equiv_3 0$.

between any vertex incident to an edge in E and any vertex incident to an edge in F is at least k. These k-independent graph distributions have been studied extensively in the context of percolation theory, for example on the infinite integer grid [5]. In [14], Day, Falgas-Ravry, and Hancock consider 1-independent random *finite* graphs. Among other results, they pin down the precise value of the connectivity threshold for 1-independent (i.e., subgraph independent) random graphs, as mentioned above in Table 1 and Figure 1.

Coloring models have been studied under two different names. First off, they have been studied as *vertex-based measures* in the context of percolation in [14]. Second off, a deterministic perspective on coloring models lies at the core of the well-studied *density Turán problem for multipartite graphs* [6, 8, 13, 17, 25]: instead of a random graph distribution, the density Turán problem for multipartite graphs considers $|V(G)|$-partite blow-ups G^* of a graph G, and the minimum density needed between parts of G^* to guarantee the existence of a graph H as a *transversal*, i.e., as a subgraph of G^* that picks precisely one vertex per part of G^*. This problem has also been considered for the more general case where we are not considering the occurrence of a single graph H as a transversal, but of any graph from some collection $\mathcal{H}$. Furthermore, one can consider the *weighted* case, where the vertices in each part of G^* each get a positive weight such that all the weights per part add up to 1. The density between two parts is then also computed in a weighted fashion. Now, considering G as the complete graph K_n, and $\mathcal{H}$ as the family of spanning trees on n vertices, one can see that the weighted density Turán problem for multipartite graphs is equivalent to the problem of determining the minimum edge probability needed to guarantee connectivity with non-zero probability in coloring models (except that the weighted density Turán setting only allows for modeling finitely many colors, which turns out to be without loss of generality, as we discuss in the following subsection). In [4], which appeared simultaneously and independently from this paper, Badakhshian, Falgas-Ravry, and Sharifzadeh consider exactly this question; they prove the same lower bound as us [4, Thm. 1.7], and propose our upper bound as a conjecture [4, Conj. 1.9].

Last but not least, let us mention that in a conceptually similar (but concretely rather different) direction, Alon and Nussboim [2] studied thresholds for the connectivity of random graph models in which only edge-sets of size at most k for a fixed parameter k are required to be independent, but dependencies between edge-sets may occur from size $k+1$ and up.

1.2 Detailed Results

In this section we give the main theorem with more detailed results. We denote by

$$\rho_{\mathrm{pw}}(n) := \inf\{p \in [0,1] \mid \forall \mathcal{D} \in \mathcal{G}_{\mathrm{pw}}(n,p), \Pr[G_n \sim \mathcal{D} \text{ is connected}] > 0\}$$

the smallest (infimum) marginal edge probability that guarantees a positive probability for G_n being connected. The quantities $\rho_{\mathrm{mat}}(n)$, $\rho_{\mathrm{esub}}(n)$, $\rho_{\mathrm{sub}}(n)$, and $\rho_{\mathrm{col}}(n)$ are defined similarly. Furthermore, let $\mathcal{G}_{\mathrm{col},k}$ denote the coloring models with at most k colors, and let $\rho_{\mathrm{col},k}(n)$ be the corresponding threshold.

▶ **Theorem 7.** *We have the following bounds on ρ:*

Pairwise independence: *For all $n \in \mathbb{N}$,*

$$1 - 2/n - \Theta(1/n^2) \leq \rho_{pw}(n) \leq 1 - 2/n.$$

Matching independence: *For all $n \in \mathbb{N}$,*

$$\frac{1}{2}(1 - \tan^2 \frac{\pi}{2n}) \leq \rho_{mat}(n) \leq 1 - 2/n.$$

Moreover, for any $k \in \mathbb{N}$,

$$\frac{1}{2} \leq \rho_{mat}(8k).$$

Edge-subgraph independence: *For all $n \in \mathbb{N}$,*

$$\frac{1}{2}(1 - \tan^2 \frac{\pi}{2n}) \leq \rho_{esub}(n) \leq 3/4.$$

Subgraph independence: *[14, Thm 16] For all $n \geq 2$,*

$$\rho_{sub}(n) = \frac{1}{2}(1 - \tan^2 \frac{\pi}{2n}).$$

Coloring model: *For all $n \geq 3$, we have $1/4 \leq \rho_{col}(n) \leq 1/2$. Moreover, let $\phi = \frac{1}{2}(1 + \sqrt{5})$ be the golden ratio. Then*

$$\lim_{n \to \infty} \rho_{col}(n) = 2 - \phi \approx 0.381966.$$

Coloring model, two colors: *For all $n \geq 3$, $\rho_{col,2}(n) = 1/4$.*

To the best of our knowledge, this question had not been considered so far for the pairwise independence, matching independence, or edge-subgraph independence models. For the subgraph independence model, the threshold was known precisely [14], but we include it in our theorem for completeness of the hierarchy of our models. Our results for the coloring model confirm a conjecture of Badakhshian, Falgas-Ravry, and Sharifzadeh [4].

We remark that, while $\mathcal{G}_{\text{col}}(n, p)$ includes models with infinitely many outcomes of the random experiment at each vertex, it has been shown[5] (in [8] for $n = 3$ and in [25, Lemma 2.1] for all n) that $\rho_{\text{col},k}(n) \geq \rho_{\text{col}}(n)$ for $k = n - 1$, which combined with the trivial $\rho_{\text{col},k}(n) \leq \rho_{\text{col}}(n)$ for any k gives $\rho_{\text{col},n-1}(n) = \rho_{\text{col}}(n)$. The proofs of $\rho_{\text{col},n-1}(n) \geq \rho_{\text{col}}(n)$ take a model which has probability 0 of being connected and modify it to use at most $n - 1$ many colors while preserving the fact that it is never connected and increasing or keeping the same its marginal edge probabilities. However, the probability distribution over all graphs on n vertices that the modified model gives is usually different from the original one. Here, we prove the even stronger statement that $\mathcal{G}_{\text{col}}(n, p) = \mathcal{G}_{\text{col},k}(n, p)$ for some $k = k(n)$, showing that finitely many colors suffice to model any probability distribution over all graphs that can be represented as a coloring model. The proof of Lemma 8 can be found in Appendix C.

▶ **Lemma 8.** *Consider a coloring model on n vertices along with its corresponding probability distribution $\mathcal{D}$ over the graphs on n vertices. Then there is a coloring model on n vertices with at most $2^{\binom{n}{2}} + 1$ colors per vertex which results in the same distribution $\mathcal{D}$.*

1.3 Open Questions

There are many interesting questions that remain open. Most obviously, for large n there are gaps between the upper and lower bounds for matching independence and edge-subgraph independence. For the coloring model we do have matching upper and lower bounds for

[5] In fact, these results are stated only for finitely many colors as they are in the density Turán setting, but the proofs go through in our more general set-up as well.

an unbounded number of colors and for $k = 2$, but not for other constant values of k. Moreover, for fixed n it is unclear how much richer the coloring model gets by adding more colors. As mentioned above, we know that $\mathcal{G}_{\text{col}}(n,p) = \mathcal{G}_{\text{col},k}(n,p)$ with $k = 2^{\binom{n}{2}} + 1$ and $\rho_{\text{col}}(n) = \rho_{\text{col},n-1}(n)$. But what is still an open question is the behavior of the functions $k^{\mathcal{G}}_{min}(n) = \min\{k \mid \mathcal{G}_{\text{col}}(n,p) = \mathcal{G}_{\text{col},k}(n,p)\}$ and $k^{\rho}_{min}(n) = \min\{k \mid \rho_{\text{col}}(n,p) = \rho_{\text{col},k}(n,p)\}$, that is, how many colors are enough to be able to express all coloring models on n vertices, respectively to capture the behaviour of the connectivity threshold of the coloring models on n vertices?

Finally, in this paper we focus on connectivity because that is arguably the most fundamental global property of a graph. We only see this as a starting point and we would find it interesting to explore analogous questions for other global properties. A natural extension might be to study the size of the largest connected component that can be achieved with non-zero probability, but the same questions arise for any other global graph properties such as Hamiltonicity, the chromatic number, and many more.

1.4 Proof Techniques

We prove the strict hierarchical structure of our considered independence conditions (Theorem 6) by giving concrete examples of distributions that fulfill the weaker independence condition, but not the stronger one. All of these examples are simple to describe and quite illustrative.

The lower bounds on the thresholds ρ are shown by concrete series of graph distributions on disconnected graphs. Here, the lower bound on ρ_{pw} (pairwise ind.) uses the same distribution as the distribution used to show $\mathcal{G}_{\text{pw}} \neq \mathcal{G}_{\text{mat}}$. The lower bounds on $\rho_{\text{mat}}, \rho_{\text{esub}}$, and ρ_{sub} all use the same construction, since we were not able to make use of the additional freedom available in the edge-subgraph or even the matching independence condition (except for the case when n is divisible by 8 in the matching independence model). This lower bound on ρ_{sub} has been previously proven in [14, Thm. 16]. While we do not restate the proof here, note that this bound uses a distribution that can be seen conceptually as a version of a coloring model with *complex probabilities*. The proof that the distribution fulfills subgraph independence follows from the fact that the coloring models are subgraph independent, and from the fundamental theorem of algebra. The lower bounds on ρ_{col} and $\rho_{\text{col},2}$ once again are simple constructions.

For the upper bounds on the thresholds ρ we use very different proof techniques depending on the independence condition. The upper bound for pairwise and matching independence does not make use of these independence conditions, and just combines a linearity of expectation argument with the maximum number of edges in a disconnected graph. As mentioned above, edge-subgraph independence is the first independence condition which allows us to apply Lovász Local Lemma to achieve a constant bound on ρ. For the coloring models, we give concrete strategies on how to pick a color for each vertex such that the graph is connected. For the two color case, this strategy is rather simple, while the strategy for the general case is based on adjusting a random coloring, which fulfills useful properties with high probability for large n. The proof of this bound (Theorem 21) is by far the most technically involved proof in this paper.

2 Preliminaries

We make use of the following version of the Lovász Local Lemma.

▶ **Lemma 9** ([28]). *Let $A_1, \dots A_k$ be a sequence of events, such that each event occurs with probability at least p and is independent of all the other events, except at most d of them. Then, if*

$$p > \begin{cases} 1 - \frac{(d-1)^{d-1}}{d^d} & \text{for } d \geq 2, \\ 1/2 & \text{for } d = 1, \end{cases}$$

there is a non-zero probability that all of the events occur.

The following lemma is a direct consequence of Markov's inequality.

▶ **Lemma 10.** *Let X be a random variable such that $Pr[X \leq u] = 1$. Then for $\ell < u$,*

$$Pr[X \geq \ell] \geq \frac{\mathbb{E}[X] - \ell}{u - \ell}.$$

Proof. Note that the random variable $u - X$ is non-negative and has expectation $u - \mathbb{E}[X]$. We can thus apply Markov's inequality.

$$Pr[X \geq \ell] = Pr[u - X \leq u - \ell] = 1 - Pr[u - X > u - \ell] \geq 1 - \frac{u - \mathbb{E}[X]}{u - \ell} = \frac{\mathbb{E}[X] - \ell}{u - \ell}. \blacktriangleleft$$

3 Bounds on ρ

In this section we prove Theorem 7, showing lower and upper bounds on ρ for our various models of independence. By the definition of ρ as in Section 1.2, any lower bound on ρ_X for some independence condition X also holds for ρ_Y for some weaker independence condition Y, i.e., one such that $\mathcal{G}_\mathrm{X} \subseteq \mathcal{G}_\mathrm{Y}$. Conversely, any upper bound on ρ_Y also holds for ρ_X.

3.1 Pairwise Independence

▶ **Lemma 11.** *For even $n \geq 4$, we have $\rho_{pw}(n) \geq 1 - \frac{2}{n} - \Theta(\frac{1}{n^2})$.*

Proof. Consider the distribution $CM(n, q)$ as defined in Definition 40 with $q = 1 - \Theta\left(\frac{1}{n^2}\right)$ chosen such that $CM(n, q)$ is pairwise independent, which can be done by Claim 41. The marginal edge probability p in $CM(n, q)$ is then $p = 1 - \frac{2}{n} - \Theta\left(\frac{1}{n^2}\right)$. Note that the probability of $G \sim CM(n, q)$ being connected is 0, since in the clique regime there is always an isolated vertex, and in the matching regime G only consists of a perfect matching. Thus, the threshold ρ_{pw} for non-zero probability of connectivity in the pairwise independence model is at least $p = 1 - \frac{2}{n} - \Theta\left(\frac{1}{n^2}\right)$. ◀

▶ **Lemma 12.** *For any n, we have $\rho_{pw}(n) \leq 1 - \frac{2}{n}$.*

Proof. If the minimum marginal edge probability p in any graph distribution (not even necessarily pairwise independent) is larger than $1 - \frac{2}{n}$, the expected number of edges $\mathbb{E}[\sum_e X_e]$ is larger than $(1 - \frac{2}{n})\binom{n}{2} = \binom{n-1}{2}$. Since any disconnected graph contains at most $\binom{n-1}{2}$ edges, the graph must be connected with non-zero probability. ◀

3.2 Matching Independence

We have no general bounds specifically holding for matching independence. Matching independence is a condition that is surprisingly difficult to exploit. We thus state the following corollary, following directly from Lemma 12 and Lemma 17.

▶ **Corollary 13.** *For any $n \geq 2$, we have $\frac{1}{2}(1 - \tan^2 \frac{\pi}{2n}) \leq \rho_{mat}(n) \leq 1 - \frac{2}{n}$.*

The lower bound in Corollary 13 can be improved slightly when n is divisible by 8, which in particular shows that $\rho_{mat}(n) \neq \rho_{sub}(n)$ for infinitely many n, illustrating again the different behavior of the connectivity thresholds for different independence conditions. The proof of this statement can be found in the full version.

▶ **Lemma 14.** *For any $n \in \mathbb{N}$ that is divisible by 8, we have $\rho_{mat}(n) \geq \frac{1}{2}$.*

3.3 Edge-Subgraph Independence

We have no lower bound specifically for edge-subgraph independence. We inherit this bound from the stricter subgraph independence, i.e., Lemma 17.

▶ **Corollary 15.** *For any $n \geq 2$, we have $\rho_{esub}(n) \geq \frac{1}{2}(1 - \tan^2 \frac{\pi}{2n})$.*

▶ **Lemma 16.** *For any n, we have $\rho_{esub}(n) \leq \frac{3}{4}$.*

Proof. We show that any edge-subgraph independent distribution with minimum marginal edge probability $p > \frac{3}{4}$ is connected with non-zero probability. To achieve this, we use the Lovász Local Lemma, as stated in Lemma 9.

We pick the edges of any Hamiltonian path $e_1, \ldots, e_{n-1}$ of K_n and consider their corresponding events $X_{e_1}, \ldots, X_{e_{n-1}}$. Since we have edge-subgraph independence, each of these events depends on only at most $d = 2$ of the other events (the ones corresponding to the neighboring edges). Furthermore, the probability of each of these events is at least $p > \frac{3}{4}$. By Lemma 9, since $p > \frac{3}{4} = 1 - \frac{1^1}{2^2}$, with non-zero probability all of the events $X_{e_1}, \ldots, X_{e_{n-1}}$ happen, all edges of the Hamiltonian path are present, and thus the graph is connected. ◀

3.4 Subgraph Independence

The exact behavior of ρ_{sub} has been determined in [14, Thm. 16]. For a proof, we point the reader to [14]. We also include a proof of the lower bound part of this statement in the full version of this paper.

▶ **Lemma 17** ([14, Thm. 16])**.** *For any $n \geq 2$, we have $\rho_{sub}(n) = \frac{1}{2}(1 - \tan^2 \frac{\pi}{2n})$.*

3.5 Coloring Models

We first consider coloring models with only 2 colors, since we can find matching lower and upper bounds for that case. Other restrictions of the coloring model could also be interesting to investigate, such as other bounded numbers of colors, or the case where every vertex must pick *uniformly* among its colors. We show that the threshold probability is exactly 1/4 in this case. The proof can be found in Appendix B.

▶ **Lemma 18.** *For all $n \geq 3$, we have $\rho_{col,2}(n) = 1/4$.*

We now consider the full generality of coloring models, with an unbounded or even infinite (although by Lemma 8 this can w.l.o.g. be excluded) number of outcomes of the local experiment at each vertex.

▶ **Lemma 19.** *For all $n \geq 3$, we have $\rho_{col}(n) \geq (n-2)\frac{3n-4-\sqrt{5n^2-16n+12}}{2(n-1)^2}$.*

Note that for $n = 3$, this bound is equal to $1/4$ (which matches the previous subsection, since our construction only uses 2 colors in this case), and for $n \to \infty$, it tends to $2 - \phi \approx 0.381966$. We remark that Lemma 19 was obtained simultaneously and independently in [4, Thm 1.7] with the exact same construction.

Proof. We show that for each number of vertices $n \geq 3$, there exists a coloring model with marginal edge probability $p(n) = (n-2)\frac{3n-4-\sqrt{5n^2-16n+12}}{2(n-1)^2}$ which is connected with probability 0.

We define the distribution $CS(n)$ for any $n \geq 3$. A fixed vertex v picks as its local experiment one other vertex $v' \in [n] \setminus \{v\}$ uniformly at random. All other vertices pick the color red with probability q (to be determined later), and blue with probability $1 - q$. Then, v connects to all vertices in $V \setminus \{v, v'\}$ that are colored blue. Between the vertices in $V \setminus \{v\}$, an edge is present if both endpoints are colored red. The resulting graph is clearly not connected: every red vertex is only connected to other red vertices, and in the case that all vertices in $V \setminus \{v\}$ are blue, only $n - 2$ edges exist.

For every edge e not incident to v we have $Pr[X_e] = q^2$, and for every edge e' incident to v we have $Pr[X_{e'}] = (1-q)\frac{n-2}{n-1}$. We pick q maximizing the minimum of these probabilities. Since q^2 is increasing in q for $q \geq 0$, and $(1-q)\frac{n-2}{n-1}$ is decreasing in q, the minimum is maximized when $q^2 = (1-q)\frac{n-2}{n-1}$. We can thus solve

$$q^2 + \frac{n-2}{n-1}q - \frac{n-2}{n-1} = 0$$

to get

$$q = \frac{2 - n \pm \sqrt{5n^2 - 16n + 12}}{2(n-1)}.$$

Only one of these solutions fulfills $q > 0$, namely the one with "+". Since the marginal edge probability $p(n)$ is equal to $(1-q)\frac{n-2}{n-1}$, we get the claimed bound. ◀

Let us now consider upper bounds for $\rho_{\mathrm{col}}(n)$. We first state the weaker constant bound holding for all n. The proof can be found in the full version of this paper. Note that the proof of Theorem 1.7 in [4], which came out independently and simultaneously, uses essentially the same techniques as the ones we employ in the proof of the lemma below, except that they optimize the calculations to get a bound of $\frac{1}{2} - \frac{1}{4n-6}$.

▶ **Lemma 20.** *For every n, we have $\rho_{col}(n) \leq \frac{1}{2}$.*

Next, we present the stronger upper bound holding for large enough n. This bound tends to $2 - \phi$ for $n \to \infty$ and thus matches the lower bound for large enough n, as conjectured in [4].

▶ **Theorem 21.** *For any $\varepsilon > 0$, there exists an $n_0(\varepsilon)$, such that for any coloring model distribution on graphs on $n \geq n_0(\varepsilon)$ vertices with minimum marginal edge probability at least $p' = 2 - \phi + \varepsilon$, the graph is connected with non-zero probability.*

To prove this theorem, we will need the following notation and setup. We write p for $2 - \phi$.

▶ **Observation 22.** *For $p = 2 - \phi$, we have the nice identities $\frac{1}{p} - 2 = 1 - p$ and $(1-p)^2 = p$.*

To begin, we color each vertex independently with a random color according to its own distribution.[6] For this fixed coloring, we write $N(v, c)$ for the set of neighbors of v, if v is recolored to the color c.

▶ **Lemma 23.** *In a random coloring, with high probability it holds that $\mathbb{E}_c[|N(v, c)|] > (p + \varepsilon/2)n$ for all v simultaneously.*

The proof of this lemma uses standard concentration bounds and can be found in the full version.

Thus, from now on, we condition on the high probability event of Lemma 23. This allows us to argue that we can recolor any vertex such that it has a large neighborhood. Furthermore, using Lemma 10, we can also argue that for every vertex, relatively large neighborhoods must exist with a somewhat large probability. This allows us to use union bounds to show that large neighborhoods must exist at the same time as some fixed edges.

Our general strategy for showing the graph is connected with positive probability will be to mostly rely on the random coloring of all vertices that we start with, but recolor some vertices as necessary, using some large neighborhoods to connect vertices to.

Since we will recolor some vertices, the true neighborhoods in the final graph may be different (possibly also smaller) than in the random coloring for which we get these bounds. We use the error term $C := C'(\varepsilon) \log_2 n$ for an upper bound on the number of vertices that we recolor and aim to connect to (we may recolor more vertices, but we do not argue that we connect to these vertices, thus recoloring them does not matter). If we can guarantee a desired intersection of neighborhoods (or of a neighborhood with a fixed set) to contain at least $C + 1$ vertices in the random coloring, we know that in the actual coloring at the end the intersection is non-empty.

Let (r, c_r) be a vertex-color pair which maximizes $|N(r, c_r)|$. We recolor r to c_r. If every vertex outside of $N(r, c_r)$ can be recolored to some color such that it has an edge to a vertex in $N(r, c_r)$, the graph can be connected. Otherwise, we pick a vertex-color pair (b, c_b) which maximizes $|N(b, c_b)|$ among all vertex-color pairs for which $b \notin N(r, c_r) \cup \{r\}$, and $|N(r, c_r) \cap N(b, c_b)| \leq C$. We recolor b to c_b.

In the following, we use the shorthands

$$R := N(r, c_r),\ B := N(b, c_b),\ \alpha := |R|/n,\ \text{and}\ \beta := |B|/n.$$

The vertices r and b, along with their respective neighborhoods R and B, will be central to our argument, and we will find various ways to connect all other vertices to those sets.

The next lemma is fairly straightforward and its proof can be found in the full version.

▶ **Lemma 24.** *$p < \alpha < 1 - p$, and $p < \beta \leq \alpha$.*

Note that if we have $R \cap B \neq \varnothing$, we can easily connect the graph: if we recolor each vertex $v \notin R \cup B \cup \{r, b\}$ to some color c such that $|N(v, c)| > pn$, $N(v, c)$ must intersect $R \cup B$. This follows from $|R \cup B| = |R| + |B| - |R \cap B| \geq \alpha n + \beta n - C \geq 2pn - C$, and $(2pn - C) + pn > n$. We thus assume in the following that $R \cap B = \varnothing$.

▶ **Lemma 25.** *$\beta \leq \min(1 - \alpha, 1/2)$.*

Proof. This follows directly from $R \cap B = \varnothing$ and thus $\alpha + \beta \leq 1$. ◀

[6] We assume that we have a finite number of colors, which we can do by Lemma 8.

To connect the graph in the remaining cases, we need to introduce some more definitions. Intuitively, we say that any vertex v is *obligate red*, if it cannot robustly be connected to the blue set B. More formally, we say that a vertex $v \notin B \cup \{b\}$ is obligate red, if there exists no color c, such that $|N(v,c) \cap B| > C$. Similarly, we say that a vertex $v \notin R \cup \{r\}$ is *obligate blue*, if there exists no color c such that $|N(v,c) \cap R| > C$. Let OR be the set of obligate red vertices, and OB the set of obligate blue vertices.

Note that no vertex can be both obligate blue and obligate red. When a vertex is neither obligate blue nor obligate red, we say it is *non-obligate*.

We start with giving some guarantees on the sizes of neighborhoods of obligate blue and obligate red vertices.

▶ **Lemma 26.** *Let $u \in OB, v \in OR$. Then, over the distribution of the color of u (v), the following hold for the neighborhood of u (v).*

1. $Pr[|N(u,c)| \geq \frac{p+p\beta-\beta}{p}n + \varepsilon n/2] \geq 1-p$.
2. $Pr[|N(v,c)| \geq \frac{p+p\alpha-\alpha}{p}n + \varepsilon n/2] \geq 1-p$.
3. $Pr[|N(u,c)| \geq (1-p)(1-\beta)n + \varepsilon n/2] \geq 1-\sqrt{p}$.
4. $Pr[|N(v,c)| \geq (1-p)(1-\alpha)n + \varepsilon n/2] \geq 1-\sqrt{p}$.

Proof (sketch). We can upper bound the size of a neighborhood of any vertex by αn. Furthermore, we can upper bound the size of a neighborhood of any obligate blue vertex u by βn. The lemma follows from applying Lemma 10 and simplifying. The full calculations are found in the full version. ◀

The next two corollaries follow from Lemma 26 and some calculations that can be found in the full version.

▶ **Corollary 27.** *Let S be a vertex set of size $|S| \leq C$. Let $u \in OB$. Then,*

$$Pr_c[N(u,c) \cap (B \setminus S) \neq \varnothing] \geq 1-p.$$

▶ **Corollary 28.** *Let S be a vertex set of size $|S| \leq C$. Let $v \in OR$. Then,*

$$Pr_c[N(v,c) \cap (R \setminus S) \neq \varnothing] \geq 1-p.$$

We will now prove Theorem 21 by case distinction on the number of obligate red and obligate blue vertices.

▶ **Lemma 29.** *If there are either no obligate red or no obligate blue vertices, the graph can be connected.*

Proof. If there are no obligate red vertices, we recolor every vertex $v \notin B \cup \{b\}$ to a color c such that $N(v,c)$ intersects B. Every vertex is thus connected to b either directly or through a vertex in B, proving that the graph is connected. Symmetrically, if there are no obligate blue vertices, all vertices can be connected to r or R. ◀

▶ **Lemma 30.** *If $|OR| > C$ or $|OB| > C$, the graph can be connected.*

The full proof of this lemma can be found in the full version.

Proof (sketch). Suppose without loss of generality that $|OB| > |OR|$ (the other case is symmetric). Let $OB' \subset OB \setminus \{b\}$ be a subset of C obligate blue vertices. We recolor all vertices in OB' as well as all non-obligate vertices such that they connect into $B \setminus OB'$. Crucially, each vertex in OB' has a color that allows it to connect to $B \setminus OB'$ as well as a

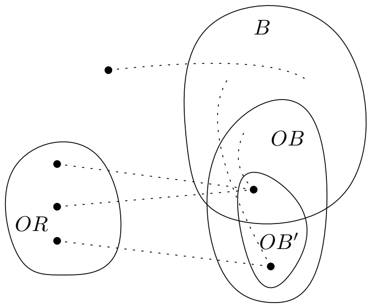


Figure 3 How to connect the graph when $|OB| > C$.

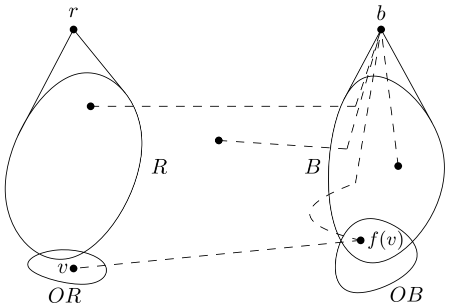


Figure 4 How to connect the graph given an injective mapping f from OR to $OB \setminus \{b\}$.

constant proportion of the remaining vertices in OR. Since C is logarithmic in n, we are able to cover all vertices in OR in this way, and thus connect the graph. An illustration of this can be seen in Figure 3. ◀

We thus assume in the following that the numbers of obligate red and obligate blue vertices are more than 0 and at most C.

▶ **Lemma 31.** *If there is an injective mapping from OR to $OB \setminus \{b\}$, or an injective mapping from OB to $OR \setminus \{r\}$, the graph can be connected.*

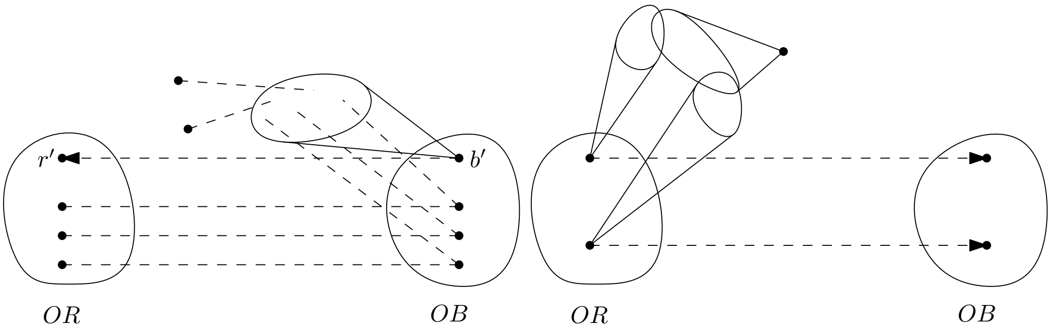
Proof. We first assume an injective mapping f from OR to $OB \setminus \{b\}$ exists. The opposite case works symmetrically. In this case we aim to connect every vertex to b, by connecting each obligate red vertex v to the vertex $f(v)$. These obligate blue vertices, as well as the non-obligate vertices outside $B \setminus \{b\}$ are then connected to some vertex in $B \setminus OB$. Since these vertices are connected to b, the whole graph is connected. This is shown in Figure 4. Note that we recolor the vertices in OB and outside B, but not those in $B \setminus OB$.

We let $v \in OR$ and $f(v) \in OB$ pick a color independently at random from their respective distributions. We have that $Pr[vf(v) \in E(G)] \geq p' = p + \varepsilon$. Furthermore, by Corollary 27, $Pr[N(f(v), c) \cap (B \setminus OB) \neq \varnothing] \geq (1 - p)$. Thus, letting $Y_{v,f(v)}$ be the indicator random variable that is 1 if and only if $vf(v) \in E(G)$ and $N(f(v), c) \cap (B \setminus OB) \neq \varnothing$, we have

$$Pr[Y_{v,f(v)} = 1] \geq p' + (1 - p) - 1 \geq \varepsilon.$$

Thus, we can pick colors for each pair of vertices $(v, f(v))$, for $v \in OR$, such that v connects to $f(v)$ and $f(v)$ connects to $B \setminus OB$. ◀

We can enhance the previous lemma by considering non-obligate vertices which also have a large probability (at least $1 - p$) to robustly connect to either of the two sets R and B (as obligate vertices are guaranteed to, by Corollary 27 and Corollary 28, respectively):



(a) Case 1: At least one edge is oriented towards OR. **(b)** Case 2: All edges are oriented towards OB.

Figure 5 Connecting the graph in Lemma 35.

▶ **Lemma 32.** *If there exists a non-obligate vertex $v \notin \{r, b\}$ such that for $S = R$ or $S = B$, it holds that $Pr_c[|N(v,c) \cap S| > C] \geq 1 - p$, then the graph can be connected.*

Proof. If the conditions of Lemma 30 or Lemma 31 are fulfilled, we can connect the graph. Otherwise, $|OR| = |OB| \leq C$ and $r \in OR$ and $b \in OB$. Assume there is a vertex v such that $Pr_c[|N(v,c) \cap B| > C] \geq 1 - p$ (the case $S = R$ works symmetrically). We can make a bijective mapping from OR to $(OB \cup \{v\}) \setminus \{b\}$. Since we have that $Pr_c[N(v,c) \cap (B \setminus (OB \cup \{v\} \setminus \{b\})) \neq \varnothing] \geq 1 - p$, the proof of Lemma 31 also works for this mapping. ◀

Furthermore, if there exists a vertex that can be used to join the red and blue sets, we can also connect the graph:

▶ **Lemma 33.** *If there exists a vertex $v \notin \{r, b\}$ and a color c, such that $N(v,c) \cap R \neq \varnothing$ and $N(v,c) \cap B \neq \varnothing$, we can connect the graph.*

Proof. We give the vertex v the color c. Every vertex $w \notin R \cup B \cup \{r, b, v\}$ is recolored to some color c_w such that $|N(w, c_w)| > pn$. Since $p > 1 - \alpha - \beta$, w is then connected to some vertex in R or B. After doing this for all w, the graph is connected. ◀

Finally, we will consider the cases which were not covered by the previous lemmas. For this, we need the following lower bounds regarding non-obligate vertices, the proofs of which can be found in the full version.

▶ **Lemma 34.** *For any non-obligate vertex $v \notin \{r, b\}$, we define*

$$p_v^r := \max_{c:|N(v,c)\cap B| \leq C} |N(v,c)|, \quad \textit{and} \quad p_v^b := \max_{c:|N(v,c)\cap R| \leq C} |N(v,c)|.$$

Assuming there exists no vertex fulfilling the conditions of either Lemma 32 or Lemma 33, we have $p_v^r > \frac{\beta p - \beta + p}{p} n$, and $p_v^b > \frac{\alpha p - \alpha + p}{p} n$.

Finally, we cover the only remaining case, whose proof can be found in Appendix B. The idea of the proof is sketched in Figure 5. We consider the potential edges $b'r'$ between $b' \in OB$ and $r' \in OR$. For each such edge we determine the vertex with a higher probability of picking a color such that the other vertex can be connected to it. We then orient the edge from this vertex to the other. If at least one edge is oriented from OB to OR, we can use this fact to connect every vertex to $B \cup OB$. Otherwise, we connect every vertex to $R \cup OR$.

▶ **Lemma 35.** *If $|OB| = |OR|$, and there exists no vertex as in Lemma 32 or Lemma 33, we can connect the graph.*

Proof of Theorem 21. Lemmas 29–33 and 35 cover all possible cases. Thus, we can always connect the graph, proving the theorem. ◀

References

1 Jon Aaronson, David Gilat, Michael Keane, and Vincent de Valk. An algebraic construction of a class of one-dependent processes. *The annals of probability*, pages 128–143, 1989.

2 Noga Alon and Asaf Nussboim. k-wise independent random graphs. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 813–822, 2008. doi:10.1109/FOCS.2008.61.

3 Noga Alon and Joel H. Spencer. *The probabilistic method.* John Wiley & Sons, 2016. doi:10.1002/9780470277331.

4 Leila Badakhshian, Victor Falgas-Ravry, and Maryam Sharifzadeh. On density conditions for transversal trees in multipartite graphs. *arXiv preprint*, 2023. doi:10.48550/arXiv.2305.05713.

5 Paul Balister, Béla Bollobás, and Mark Walters. Random transceiver networks. *Advances in Applied Probability*, 41(2):323–343, 2009. doi:10.1239/aap/1246886613.

6 Paul Balister, Tom Johnston, Michael Savery, and Alex Scott. Improved bounds for 1-independent percolation on $\mathbb{Z}^n$. *arXiv preprint*, 2022. doi:10.48550/arXiv.2206.12335.

7 Béla Bollobás. The isoperimetric number of random regular graphs. *European Journal of Combinatorics*, 9(3):241–244, 1988. doi:10.1016/S0195-6698(88)80014-3.

8 Adrian Bondy, Jian Shen, Stéphan Thomassé, and Carsten Thomassen. Density conditions for triangles in multipartite graphs. *Combinatorica*, 26(2):121–131, 2006. doi:10.1007/s00493-006-0009-y.

9 Karl Bringmann, Ralph Keusch, and Johannes Lengler. Geometric inhomogeneous random graphs. *Theoretical Computer Science*, 760:35–54, 2019. doi:10.1016/j.tcs.2018.08.014.

10 Robert M. Burton, Marc Goulet, and Ronald Meester. On 1-dependent processes and k-block factors. *The Annals of Probability*, 21(4):2157–2168, 1993. doi:10.1214/aop/1176989014.

11 Constantin Carathéodory. Über den variabilitätsbereich der koeffizienten von potenzreihen, die gegebene werte nicht annehmen. *Mathematische Annalen*, 64(1):95–115, 1907. doi:10.1007/BF01449883.

12 Fan Chung and Linyuan Lu. Connected components in random graphs with given expected degree sequences. *Annals of combinatorics*, 6(2):125–145, 2002. doi:10.1007/PL00012580.

13 Péter Csikvári and Zoltán L. Nagy. The density Turán problem. *Combinatorics, Probability and Computing*, 21(4):531–553, 2012. doi:10.1017/S0963548312000016.

14 A. Nicholas Day, Victor Falgas-Ravry, and Robert Hancock. Long paths and connectivity in 1-independent random graphs. *Random Structures & Algorithms*, 57(4):1007–1049, 2020. doi:10.1002/rsa.20972.

15 Maria Deijfen, Remco van der Hofstad, and Gerard Hooghiemstra. Scale-free percolation. *Annales de l'IHP Probabilités et statistiques*, 49(3):817–838, 2013. doi:10.1214/12-AIHP480.

16 Paul Erdős. Graph theory and probability. *Canadian Journal of Mathematics*, 11:34–38, 1959. doi:10.4153/CJM-1959-003-9.

17 Victor Falgas-Ravry and Vincent Pfenninger. 1-independent percolation on $\mathbb{Z}^2 \times K_n$. *Random Structures & Algorithms*, 2022. doi:10.1002/rsa.21129.

18 Heidi Gebauer, Robin A. Moser, Dominik Scheder, and Emo Welzl. The Lovász Local Lemma and satisfiability. In Susanne Albers, Helmut Alt, and Stefan Näher, editors, *Efficient Algorithms: Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*, pages 30–54. Springer, Berlin, Heidelberg, 2009. doi:10.1007/978-3-642-03456-5_3.

19 Alexander E. Holroyd and Thomas M. Liggett. Finitely dependent coloring. In *Forum of Mathematics, Pi*, volume 4, page e9. Cambridge University Press, 2016. doi:10.1017/fmp.2016.7.

20 Anders Johansson. Asymptotic choice number for triangle free graphs. *Technical Report, DIMACS*, pages 91–95, 1996.

21 Brian Karrer and Mark E. J. Newman. Stochastic blockmodels and community structure in networks. *Physical review E*, 83(1):016107, 2011. doi:10.1103/PhysRevE.83.016107.

22 Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010. doi:10.1103/PhysRevE.82.036106.

23 Thomas M. Liggett, Roberto H. Schonmann, and Alan M. Stacey. Domination by product measures. *The Annals of Probability*, 25(1):71–95, 1997. doi:10.1214/aop/1024404279.

24 Michael Molloy. The list chromatic number of graphs with small clique number. *Journal of Combinatorial Theory, Series B*, 134(1):264–284, 2019. doi:10.1016/j.jctb.2018.06.007.

25 Zoltán L. Nagy. A multipartite version of the Turán problem – density conditions and eigenvalues. *The Electronic Journal of Combinatorics*, 18(1):P46, 2011. doi:10.37236/533.

26 Mathew Penrose. *Random Geometric Graphs*. Oxford University Press, 2003. doi:10.1093/acprof:oso/9780198506263.001.0001.

27 Hannu Reittu and Ilkka Norros. Random graph models of communication network topologies. In *Unifying Themes in Complex Systems VII: Proceedings of the Seventh International Conference on Complex Systems*, pages 214–221. Springer, 2012. doi:10.1007/978-3-642-18003-3_21.

28 James B. Shearer. On a problem of Spencer. *Combinatorica*, 5(3):241–245, 1985. doi:10.1007/BF02579368.

29 Joel Spencer. Asymptotic lower bounds for Ramsey functions. *Discrete Mathematics*, 20:69–76, 1977. doi:10.1016/0012-365X(77)90044-9.

A A Strict Hierarchy

This section is dedicated to proving Theorem 6:

▶ **Theorem 6.** $\mathcal{G}_{col} \subsetneq \mathcal{G}_{sub} \subsetneq \mathcal{G}_{esub} \subsetneq \mathcal{G}_{mat} \subsetneq \mathcal{G}_{pw}$.

We will prove each strict inclusion as its own lemma, from left to right. The first among these lemmas shows that each coloring model is subgraph independent but some subgraph independent distributions are not coloring models.

▶ **Lemma 36.** *$\mathcal{G}_{col} \subsetneq \mathcal{G}_{sub}$, i.e., every coloring model is subgraph independent, but there are some subgraph independent distributions that are not coloring models.*

Proof. We first prove $\mathcal{G}_{\text{col}} \subseteq \mathcal{G}_{\text{sub}}$. Consider some graph distribution $D \in \mathcal{G}_{\text{col}}$ and consider any two disjoint subsets of vertices V, W. The resulting graph within V depends only on elementary experiments on V, and the resulting graph within W depends only on elementary experiments on W. Since the elementary experiments are mutually independent, we have independence of the resulting graphs within V and W. Thus, $D \in \mathcal{G}_{\text{sub}}$.

The fact that $\mathcal{G}_{\text{col}} \neq \mathcal{G}_{\text{sub}}$ is well-known from prior work [1, 10, 14, 19]. ◀

▶ **Lemma 37.** *$\mathcal{G}_{sub} \subsetneq \mathcal{G}_{esub}$, i.e., every subgraph independent distribution is edge-subgraph independent, but there are some edge-subgraph independent distributions that are not subgraph independent.*

Proof. Clearly $\mathcal{G}_{\text{sub}} \subseteq \mathcal{G}_{\text{esub}}$ as subgraph independence implies edge-subgraph independence by definition, since a single edge is also a subgraph.

To prove $\mathcal{G}_{\text{sub}} \neq \mathcal{G}_{\text{esub}}$, we consider the following graph distribution $CC(n)$ on $n \geq 6$ vertices: with probability 1/2, the distribution returns a graph drawn from the Erdős-Rényi distribution $G_{n,1/2}$. Otherwise, the distribution picks a uniformly random decomposition of the vertex set $[n]$ into two sets A and B, and returns the graph consisting of the union of cliques on A and B. We first show that $CC(n)$ is edge-subgraph independent. In both the Erdős-Rényi regime as well as the two-cliques regime, for any edge e we have $Pr[X_e] = 1/2$. This holds even when we condition on the outcome within any subgraph disjoint from e, thus $CC(n)$ is edge-subgraph independent. On the other hand, we show that $CC(n)$ is not subgraph independent. To this end, we decompose the vertex set $[n]$ into the sets $P = \{u, v, w\}$ and $Q = [n] \setminus P$. We consider the two events $\mathcal{P}$:= "There are exactly two edges within P" and $\mathcal{Q}$:= "Q is a clique". Clearly, we have

$$Pr[\mathcal{Q}] = \frac{1}{2} \cdot 2^{-(n-4)} + \frac{1}{2} \cdot 2^{-\binom{n-3}{2}} > 2^{-\binom{n-3}{2}}.$$

On the other hand, $\mathcal{P}$ implies that we are in the Erdős-Rényi regime, and thus

$$Pr[\mathcal{Q}|\mathcal{P}] = 2^{-\binom{n-3}{2}}.$$

We conclude that $Pr[\mathcal{Q}] > Pr[\mathcal{Q}|\mathcal{P}]$ and thus $\mathcal{P}$ and $\mathcal{Q}$ are not independent, showing that $CC(n)$ is not subgraph independent. ◀

The graph distribution $CC(n)$ that we built in the proof above is a *convex combination* of its two regimes, as defined in the full version of this paper.

▶ **Lemma 38.** *$\mathcal{G}_{esub} \subsetneq \mathcal{G}_{mat}$, i.e., every edge-subgraph independent distribution is matching independent, but there are some matching independent distributions that are not edge-subgraph independent.*

Proof. To prove $\mathcal{G}_{\text{esub}} \subseteq \mathcal{G}_{\text{mat}}$, consider some edge-subgraph independent distribution. To prove that it is also matching independent, let $M = \{e_1, \ldots, e_k\}$ be some matching. We show for all $i \leq k$ that $Pr[X_{e_i}|X_{e_1}, \ldots, X_{e_{i-1}}] = Pr[X_{e_i}]$. This implies

$$Pr[X_{e_1} \text{ and } X_{e_2} \text{ and } \ldots \text{ and } X_{e_k}] = \prod_{i=1}^{k} Pr[X_{e_i}],$$

and thus implies matching independence. Let V' be the set of endpoints of the edges $e_1, \ldots, e_{i-1}$. Furthermore, let $\mathcal{G}(V')$ be the set of all possible graphs on V'. Then, by the law of total probability, we have

$$Pr[X_{e_i}|X_{e_1}, \ldots, X_{e_{i-1}}] = \sum_{G \in \mathcal{G}(V')} Pr[X_{e_i}|G] Pr[G|X_{e_1}, \ldots, X_{e_{i-1}}].$$

Due to edge-subgraph independence, $Pr[X_{e_i}|G] = Pr[X_{e_i}]$, and we thus have

$$Pr[X_{e_i}|X_{e_1}, \ldots, X_{e_{i-1}}] = Pr[X_{e_i}] \sum_{G \in \mathcal{G}(V')} Pr[G|X_{e_1}, \ldots, X_{e_{i-1}}] = Pr[X_{e_i}],$$

proving the desired claim.

For the second part of the statement, $\mathcal{G}_{\text{esub}} \neq \mathcal{G}_{\text{mat}}$, we consider the following graph distribution $SC(n)$ on n vertices for odd $n \geq 5$: every vertex independently and uniformly picks from the two colors red and blue. Since n is odd, one color was picked by an even number of vertices. The vertices with that color form a clique, and no other edges are present.

We first show that $SC(n)$ is matching independent: every edge individually occurs with probability $1/4$, since first both endpoints need to have the same color, and second this color must be the color picked by an even number of vertices. We prove that a matching of k edges occurs with probability $(1/4)^k$. For the matching to occur, all $2k$ vertices must pick the same color, which happens with probability $(1/2)^{2k-1}$. Second, that color must end up to be the color picked by an even number of vertices, which has probability $1/2$. Altogether, we have probability $(1/2)^{2k-1+1} = (1/4)^k$.

Finally, we show that $SC(n)$ is not edge-subgraph independent. Let W be a set of $n-2$ vertices, and e be the edge between the remaining 2 vertices. If W is a clique, e cannot be present. Since both W being a clique and e being present have non-zero marginal probabilities, edge-subgraph independence cannot hold. ◀

▶ **Lemma 39.** *$\mathcal{G}_{mat} \subsetneq \mathcal{G}_{pw}$, i.e., every matching independent distribution is pairwise independent, but there are some pairwise independent distributions that are not matching independent.*

To prove this final strict inclusion lemma, we define the following distribution. This distribution will also be useful later to prove a lower bound on ρ_{pw}.

▶ **Definition 40.** *$CM(n,q)$ for n even is the following distribution over the graphs on n vertices: to sample $G \sim CM(n,q)$, with probability q we sample from the* clique regime, *and otherwise from the* matching regime. *In the clique regime, we pick a vertex $x \in [n]$ uniformly at random and add all edges e with $x \notin e$ to G. In the matching regime, we pick a perfect matching on $[n]$ uniformly at random and add its edges to G.*

▷ Claim 41. There exists $0 \leq q(n) \leq 1$ with

$$q(n) = 1 - \Theta\Big(\frac{1}{n^2}\Big)$$

such that $CM(n, q(n))$ for n even is pairwise independent but not matching independent, and the probability of each edge is

$$p(n) = 1 - \frac{2}{n} - \Theta\Big(\frac{1}{n^2}\Big).$$

The proof of this claim is straightforward and just requires some calculations. It can be found in the full version.

Proof of Lemma 39. Clearly $\mathcal{G}_{\mathrm{mat}} \subseteq \mathcal{G}_{\mathrm{pw}}$ as matching independence implies pairwise independence by definition, since two vertex-disjoint edges e, f are a matching.

To see that $\mathcal{G}_{\mathrm{mat}} \neq \mathcal{G}_{\mathrm{pw}}$, recall that by Claim 41, there exists some $q(n)$ such that $CM(n,q)$ is pairwise independent but not matching independent. ◀

Proof of Theorem 6. Theorem 6 now follows directly from Lemmas 36–39. ◀

B Omitted Proofs from Section 3.5

Proof of Lemma 18. We first prove $\rho_{col,2}(n) \geq 1/4$ by defining the following coloring model which is always disconnected. Decompose the vertex set into three non-empty sets A, B, C. Every vertex chooses a color uniformly among the colors red and blue. Two vertices in the same set are connected if they are both red. A vertex $a \in A$ is connected to $b \in B$, if a is red and b is blue. The same goes for $b \in B$ and $c \in C$, as well as $c \in C$ and $a \in A$. Clearly, the marginal edge probability of every edge is 1/4. We now show that every connected component of a graph sampled from this distribution is a subset of the union of at most two of the sets A, B, C. To see this, we pick some edge $\{a, b\}$ for $a \in A$ and $b \in B$ to be present, and try to grow its connected component. Since the edge $\{a, b\}$ is present, a must be red and b blue. Now, a can only be connected to other red vertices in A, and to blue vertices in B. Similarly, b can only be connected to other red vertices in A. No red vertex in A or blue vertex in B can be connected to a vertex in C, thus the connected component containing the edge $\{a, b\}$ is contained in $A \cup B$. Symmetrically this holds for any pair of sets, and we conclude that the graph must be disconnected, proving the lower bound.

Next, we prove $\rho_{col,2}(n) \leq 1/4$ by showing that if every marginal edge probability is strictly larger than 1/4, we can always find a coloring that connects the graph. We say that a vertex v *covers* vertex w, if for both colors at vertex w, there exists a color at v, such that the edge $\{v, w\}$ is present under this coloring. We now remove vertices one by one, by repeatedly removing a vertex which covers some remaining vertex, until no more such vertices exist. If the graph on the remaining vertices V' can be connected using some coloring, we can add back the removed vertices in reverse order, and connect them to the vertex they cover, thus connecting the whole graph. We thus only have to show that the graph on V' can be connected. Since in this graph no vertex covers any other, each edge is only present under exactly one of the four possible color combinations of its endpoints.

We pick a vertex $v' \in V'$ which maximizes $\max(Pr[v \text{ is red}], Pr[v \text{ is blue}])$ among all $v \in V'$. We give this vertex v' the color which is more likely, let p be the probability of this color. One can see that for any other vertex $w \in V' \setminus \{v'\}$, there must be a color such that the edge $\{v', w\}$ is present. Otherwise, the color combination making $\{v', w\}$ present would have probability of at most $p \cdot (1 - p)$, which is at most 1/4. Thus, we can connect V' by simply coloring each vertex with the correct color to connect to v'. This concludes the proof of the upper bound, and thus the whole lemma. ◀

Proof of Lemma 35. We consider the potential edges between OR and OB. Each such edge $r'b'$ can be oriented in some direction, by considering the following two probabilities:

$$p_{r' \to b'} := Pr_{c_{r'}}[\exists c_{b'} \text{ such that } r'b' \in E(G) \text{ in coloring } c(r') = c_{r'}, c(b') = c_{b'}]$$
$$p_{b' \to r'} := Pr_{c_{b'}}[\exists c_{r'} \text{ such that } r'b' \in E(G) \text{ in coloring } c(r') = c_{r'}, c(b') = c_{b'}]$$

Note that $p_{r' \to b'} \cdot p_{b' \to r'} \geq Pr[r'b' \in E(G)] \geq p'$, and thus $\max(p_{r' \to b'}, p_{b' \to r'}) \geq \sqrt{p'}$. We now direct the potential edge $r'b'$ from r' to b' if $p_{r' \to b'} > p_{b' \to r'}$, and from b' to r' otherwise. We pick an arbitrary perfect matching M among the potential edges between OR and OB, using $|OR| = |OB|$.

Case 1: There exists an arc $(b', r') \in M$ directed towards $r' \in OR$. In this case we will connect the graph as shown in Figure 5a. Every obligate red vertex is connected to its matching obligate blue vertex. The vertex b' must have a large neighborhood size that can be guaranteed with probability $\geq 1 - \sqrt{p}$, so that it can simultaneously be connected to r'. Every obligate blue and every non-obligate vertex connects to this neighborhood.

By Lemma 26, part 3, we have

$$Pr[|N(b', c)| \geq (1 - p)(1 - \beta)n + \varepsilon n/2] \geq 1 - \sqrt{p}.$$

Thus, with positive probability over the choice of color for b', the size of the neighborhood of b' is at least $(1 - p)(1 - \beta)n + \varepsilon n/2$ and there is a choice of color for r' such that $r'b'$ is an edge, since $1 - \sqrt{p} + \sqrt{p'} > 0$.

Similarly, for all other obligate blue vertices u, by Lemma 26, part 1, we have

$$Pr[|N(u,c)| \geq \frac{p+p\beta-\beta}{p}n + \varepsilon n/2] \geq 1-p,$$

so with positive probability there is an edge between u and its matching vertex in OR and u's neighborhood has size at least $\frac{p+p\beta-\beta}{p}n + \varepsilon n/2$.

We need to prove that these guaranteed sizes of neighborhoods intersect outside of $OB \cup OR$. Let $S := (1-p)(1-\beta)n + \varepsilon n/2$ and $T := \frac{p+p\beta-\beta}{p}n+\varepsilon n/2$. Since both neighborhoods can intersect R in at most C vertices, we need to prove that $(S-3C)+(T-3C) > n-|R|$, i.e.,

$$S + T + \alpha n - n - 6C \overset{!}{>} 0. \tag{1}$$

Additionally, to guarantee that each non-obligate vertex u outside the neighborhood of b' can connect to that neighborhood, we need to guarantee that u has a neighborhood which intersects that of b' outside $OB \cup OR$. Thus, we check that $(S-3C)+(p_u^b-3C) > n-|R|$, i.e.,

$$S + p_u^b + \alpha n - n - 6C \overset{!}{>} 0. \tag{2}$$

Since our lower bound on p_u^b is smaller than T (as $\alpha \geq \beta$), (2) implies (1), so it suffices to show (2). The proof of (2) can be found in the full version.

Case 2: All edges in M are oriented towards OB. In this case we will connect the graph as shown in Figure 5b. We give each obligate red vertex a large neighborhood such that we can still guarantee to be able to connect the obligate blue vertices to their matching partner. Then, we give each non-obligate vertex which is not yet connected directly to any of the obligate vertices a large neighborhood not intersecting B (as guaranteed by Lemma 34). These neighborhoods must intersect all neighborhoods of the obligate red vertices due to their sizes, thus the graph is connected.

Let $Y := (1-p)(1-\alpha)n + \varepsilon n/2$. For any obligate red vertex v, by Lemma 26, part 4, we have $Pr[|N(v,c_v)| \geq Y] \geq 1-\sqrt{p}$, so with probability $1-\sqrt{p}+\sqrt{p'} > 0$ there is a color for v's obligate blue partner vertex so that they are connected by an edge and also $|N(v,c_v)| \geq Y$.

We now check that every non-obligate vertex u has a neighborhood which intersects with every neighborhood of size at least Y of the obligate red vertices. Recall that by Lemma 34, every u has a neighborhood not robustly intersecting B of size p_u^r. Note that we need to show that these neighborhoods intersect outside of $OR \cup OB$ since these vertices may change their colors. To this end, it suffices to show that $(p_u^r - 3C) + (Y - 3C) > n - |B|$, i.e.,

$$p_u^r + Y + \beta n - n - 6C \overset{!}{>} 0,$$

which is easy to verify as can be seen in the full version. ◀

C Finitely Many Colors Suffice for the Coloring Model

Proof of Lemma 8. Let $\mathcal{D}$ be a random graph distribution given as a coloring model. We will reduce the number of colors for each vertex one by one, preserving the probability distribution $\mathcal{D}$. By a slight abuse of notation, here we refer to any outcome of the experiment at some vertex as its *color*, even if there are infinitely many of them. Suppose we're considering vertex v and for each graph H on n vertices and color c of v, let $p_{H,c}$ be the probability that H is sampled from $\mathcal{D}$ conditioned on vertex v having color c, and let p_H be the unconditional probability of H being sampled from $\mathcal{D}$. Now for each color c of v, consider the vector $\vec{p}_{*,c}$ in $2^{\binom{n}{2}}$-dimensional space that consists of all the $p_{H,c}$ for all possible graphs H on n vertices in some canonical order. Consider also the vector $\vec{p}_*$ of the same dimension with the p_H for all possible graphs H on n vertices as entries in the same canonical order. By the law of total probability, $\vec{p}_*$ is contained in the convex hull of all the $\vec{p}_{*,c}$ vectors (of which there may be infinitely many). It follows by Carathéodory's theorem [11] that $\vec{p}_*$ is a convex combination of some $2^{\binom{n}{2}} + 1$ many vectors among $\{\vec{p}_{*,c} \mid c \text{ is a color of vertex } v\}$. Thus we can pick the corresponding $2^{\binom{n}{2}} + 1$ colors for v, use the coefficients given by that convex combination as probabilities for these colors in the experiment at vertex v, and we end up with the same distribution $\mathcal{D}$. We repeat this process for each vertex, completing the proof. ◀

Synergy Between Circuit Obfuscation and Circuit Minimization

Russell Impagliazzo
Department of Computer Science, University of California San Diego, La Jolla, CA, USA

Valentine Kabanets
School of Computing Science, Simon Fraser University, Burnaby, Canada

Ilya Volkovich
Computer Science Department, Boston College, Chestnut Hill, MA, USA

Abstract

We study close connections between Indistinguishability Obfuscation (IO) and the Minimum Circuit Size Problem (MCSP), and argue that efficient algorithms/construction for MCSP and IO create a *synergy*[1]. Some of our main results are:

- If there exists a perfect (imperfect) IO that is computationally secure against nonuniform polynomial-size circuits, then for all $k \in \mathbb{N}$: $\mathsf{NP} \cap \mathsf{ZPP}^{\mathsf{MCSP}} \not\subseteq \mathsf{SIZE}[n^k]$ ($\mathsf{MA} \cap \mathsf{ZPP}^{\mathsf{MCSP}} \not\subseteq \mathsf{SIZE}[n^k]$).
- In addition, if there exists a perfect IO that is computationally secure against nonuniform polynomial-size circuits, then $\mathsf{NEXP} \cap \mathsf{ZPEXP}^{\mathsf{MCSP}} \not\subseteq \mathsf{P/poly}$.
- If $\mathsf{MCSP} \in \mathsf{BPP}$, then statistical security and computational security for IO are equivalent.
- If computationally-secure perfect IO exists, then $\mathsf{MCSP} \in \mathsf{BPP}$ iff $\mathsf{NP} = \mathsf{ZPP}$.
- If computationally-secure perfect IO exists, then $\mathsf{ZPEXP} \neq \mathsf{BPP}$.

To the best of our knowledge, this is the first consequence of strong circuit lower bounds from the existence of an IO. The results are obtained via a construction of an optimal *universal distinguisher*, computable in randomized polynomial time with access to the MCSP oracle, that will distinguish any two circuit-samplable distributions with the advantage that is the statistical distance between these two distributions minus some negligible error term. This is our main technical contribution. As another immediate application, we get a simple proof of the result by Allender and Das (*Inf. Comput.*, 2017) that $\mathsf{SZK} \subseteq \mathsf{BPP}^{\mathsf{MCSP}}$.

2012 ACM Subject Classification Theory of computation → Cryptographic primitives; Theory of computation → Complexity classes; Theory of computation → Circuit complexity

Keywords and phrases Minimal Circuit Size Problem (MCSP), Circuit Lower Bounds, Complexity Classes, Indistinguishability Obfuscation, Separation of Classes, Statistical Distance

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.31

Category RANDOM

Related Version *Full Version*: https://eccc.weizmann.ac.il/report/2023/079/

Acknowledgements The authors would like to thank the anonymous referees for their detailed comments and suggestions on the previous version of the paper.

[1] Synergy refers to the phenomenon where the combination of X and a "waek" version of Y gives rise to a "stronger" version of Y.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 31; pp. 31:1–31:21
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Circuit Obfuscation

The main purpose of program obfuscation is to transform a given program into an "unintelligible" one, while preserving the program's original functionality. A natural way to represent a program is via a Boolean circuit. Given that, the most common notion of obfuscation is the notion of *indistinguishability obfuscation*, introduced in [8]. Roughly speaking, a (potentially) randomized procedure IO is an *indistinguishability obfuscator*, if the obfuscations of two circuits C_1 and C_2 of the same size and functionality are "indistiguishable". In other words, no algorithm can "distinguish" between the outputs of $\mathsf{IO}(C_1)$ and $\mathsf{IO}(C_2)$ with a "noticeable" advantage.

The kind of security provided by the IO is defined by the class of the allowed distinguishing algorithms. More formally, consider a particular class of algorithms $\mathcal{A}$ and ask whether IO is "secure against" $\mathcal{A}$. For example, if $\mathcal{A}$ is the class of *all* (possibly inefficient) algorithms, we say that IO is *statistically* secure. On the other hand, if $\mathcal{A}$ is the class of *efficient* (i.e. randomized polynomial-time) algorithms, we say that IO is *computationally* secure.

The correctness of an IO procedure is called *perfect* if the functionality of the input circuit is preserved with probability one (over the internal randomness of the IO), or *imperfect* if the functionality is preserved with high probability only.

Circuit obfuscation turned out to be a very useful tool in many cryptographic and complexity-theoretic applications, see, e.g., [19, 43, 20, 10, 35]. The past decade saw numerous candidate constructions, culminating with the work of [29]. Yet, identifying the exact necessary and sufficient conditions for the existence of indistinguishability obfuscators remains an important open question. One reason for that is that unlike the vast majority of cryptographic primitives, obfuscators could still exist even if $\mathsf{P} = \mathsf{NP}$! In fact, in this case we get an "ultimate" obfuscator: for each circuit C, the IO will output some canonical equivalent $\hat{C}$[2].

The place of IO within the Five Worlds

Thus, in the language of Impagliazzo's Five Worlds [26], an IO exists in Algorithmica. The work of [29], on the other hand, makes a good argument that an IO may exist in Cryptomania. What about the other three worlds: Heuristica, Pessiland, and Minicrypt? It turns out that none of these remaining three worlds can accommodate an IO. The results of [43] show that an IO plus a one-way function imply public key encryption (and more), and hence IO cannot exist in Minicrypt. The results of [34] essentially show that an (even imperfect) IO cannot exist in Pessiland: if there are no one-way functions but an (imperfect) IO exists, then $\mathsf{NP} \subseteq \mathsf{io\text{-}BPP}$. This result also rules out Heuristica as a possible home for an IO. We will prove a stronger connection: if an (imperfect) IO exists in Heuristica (where $\mathsf{DistNP} \subseteq \mathsf{AvgP}$), then $\mathsf{NP} = \mathsf{P}$ (see Theorem 31 below).

So IO can exist in either Algorithmica or Cryptomania. Many of the results that we shall present in this paper can be viewed as instantiations of this fact, for various settings of parameters of IO: *If you assume* IO *exists, and assume something that threatens the existence of cryptography, then you find yourself in* Algorithmica*.*

[2] For example, given a circuit C one can find the lexicographically-smallest, equivalent circuit $\hat{C}$ in PH.

Circuit Minimization

Minimum Circuit Size Problem (MCSP) [45, 31] asks for a given truth table of an n-variate Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ and a parameter $0 \leq s \leq 2^n$, if f is computable by a Boolean circuit of size at most s. It is easy to see that $\mathsf{MCSP} \in \mathsf{NP}$. Yet, it is unknown if MCSP is NP-hard, or if MCSP is easy, say in BPP. What is known is that MCSP is powerful enough to "kill" cryptography. That is, any one-way function candidate can be efficiently inverted on average by a randomized polynomial-time algorithm with access to the MCSP oracle [41, 3]. Hence, an efficient algorithm for MCSP cannot exist in Minicrypt or Cryptomania.[3]

Interplay between IO and MCSP

By the preceding discussion, if we assume that both an IO exists and that MCSP is "easy", then we should get that NP is also "easy" (as we must be in some version of Algorithmica; see Theorems 5 & 6 for more details)[4]. In fact, we shall argue that MCSP and IO act in a *synergy*. That is, the assumed existence of an appropriate version of IO makes MCSP more powerful that it is known to be. And, on the other hand, we show results where assumed "easiness" of MCSP makes an IO stronger (i.e., more secure). We state some of our main results next.

1.1 Our Main Results

We show that the existence of an (even imperfect) IO secure against P/poly implies new circuit lower bounds.

▶ **Theorem 1.** *Suppose there exists a perfect* IO *secure against* P/poly*. Then:*
1. $\mathsf{NEXP} \cap \mathsf{ZPEXP}^{\mathsf{MCSP}} \not\subseteq \mathsf{P/poly}$.
2. *For all* $k \in \mathbb{N}$*:* $\mathsf{NP} \cap \mathsf{ZPP}^{\mathsf{MCSP}} \not\subseteq \mathsf{SIZE}[n^k]$.

▶ **Theorem 2.** *Suppose there exists an imperfect* IO *secure against* P/poly*. Then for all* $k \in \mathbb{N}$*:*
$\mathsf{MA} \cap \mathsf{ZPP}^{\mathsf{MCSP}} \not\subseteq \mathsf{SIZE}[n^k]$.

The two preceding theorems should be contrasted with the unconditional circuit lower bounds proved in [44] and [27]. There it is shown that $\mathsf{ZPEXP}^{\mathsf{MCSP}} \not\subseteq \mathsf{P/poly}$ and that, for every $k > 0$, $\mathsf{ZPP}^{\mathsf{MCSP}}/1 \not\subseteq \mathsf{SIZE}[n^k]$ and $\mathsf{MA}/1 \not\subseteq \mathsf{SIZE}[n^k]$. Although removing the extra bit of advice from the lower bounds may seem incremental, it actually has been a long standing open problem that resisted many attempts! Indeed, the same issue arises in other instances involving lower bounds for randomized complexity classes; see, e.g., [7, 18, 46, 47]. Additionally, while widely *believed* to be true, showing that $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$ seems to require techniques beyond our current reach. For a further discussion, see the seminal paper of Williams [50] where it was shown that $\mathsf{NEXP} \not\subseteq \mathsf{ACC}$ and subsequent improvements (e.g. [38]). In conclusion, the two new theorems above prove stronger circuit lower bounds, but under an assumption that a certain IO exists. One interpretation of that is that a construction of these kinds of IO will require novel techniques.

Our next result is a uniform version of Theorem 1.

[3] In fact, even an efficient one-sided average-case algorithm for MCSP (i.e., an efficiently computable natural property in the sense of [41], which is useful against exponential-size circuits) would "kill" one-way functions.

[4] This observation was made in [27] and previously a similar observation was made in [34].

▶ **Theorem 3.** *Suppose there exists a computationally-secure perfect* IO*. Then* $\mathsf{ZPEXP} \neq \mathsf{BPP}$*.*

While we do not have hierarchy theorems for randomized complexity classes, one can show that $\mathsf{ZPEXP} \neq \mathsf{ZPP}$ (see Appendix C). Yet, separating ZPEXP (or even NEXP and $\mathsf{EXP}^{\mathsf{NP}}$) from BPP appears to be a longstanding open problem (see e.g. [15, 49]). In that sense our result resolves the problem under the assumption that a computationally-secure perfect IO exists.

The following theorems are examples of results where MCSP empowers IO, and where IO empowers MCSP.

▶ **Theorem 4.** *An* IO *(both imperfect and perfect) is statistically-secure if and only if it is secure against* $\mathsf{FBPP}^{\mathsf{MCSP}}$*. Hence, assuming* $\mathsf{MCSP} \in \mathsf{BPP}$*, statistically-secure* IO *exists if and only if computationally-secure* IO *exists.*

▶ **Theorem 5.** *Let* $\Gamma \in \{\mathsf{ZPP}, \mathsf{BPP}\}$*. Suppose there exists a computationally-secure imperfect* IO*. Then* $\mathsf{MCSP} \in \Gamma$ *iff* $\mathsf{NP} \subseteq \Gamma$*.*

Note that Theorem 5 strengthens a similar result of [27] to the imperfect setting.

▶ **Theorem 6.** *Suppose there exists a computationally-secure perfect* IO*. Then* $\mathsf{MCSP} \in \mathsf{BPP}$ *iff* $\mathsf{NP} = \mathsf{ZPP}$*.*

▶ Remark 1. Note that all the results still hold true if we only have an obfuscator IO for a class of circuits $\mathcal{C}$ for which the *equivalence* problem (i.e., testing if two given circuits $C_0, C_1 \in \mathcal{C}$ agree on all inputs) is coNP-hard such as: 3-CNFs (even read-thrice 3-CNFs), read-twice depth-3 formulas, monotone depth-3 formulas[5] and others. All these circuit classes are small subsets of NC^1, which is the starting points of most candidate IO constructions (see e.g. [40, 21, 36, 29] and references within).

▶ Remark 2. Recall that the results of [34] show that if there are no one-way functions yet an imperfect IO exists, then $\mathsf{NP} \subseteq \mathsf{io}\text{-}\mathsf{BPP}$. The authors subsequently pose an open problem to get a similar result only relying on an obfuscator for 3-CNFs. While we do not solve their open problem, we believe that Theorem 5, adjusted according to the previous remark, can be viewed as partial progress towards the resolution of the problem especially in light of the recent characterizations of one-way functions in terms of "MCSP"-like problems [37, 4, 25, 24].

1.2 Our Techniques

Our main technical tool is a *universal distinguisher* that, given any two circuits C_0 and C_1 that are samplers for some distributions D_0 and D_1, will distinguish between D_0 and D_1 essentially as well as is information-theoretically possible (with the distinguishing advantage equal to the statistical distance between D_0 and D_1 minus a negligible error term). We show (see Corollary 17) that such a universal distinguisher is computable in $\mathsf{FBPP}^{\mathsf{MCSP}}$[6]. The main idea is to use a distributional inverter and a connection between one-way functions and distributional one-way functions from [28]. In particular, we argue (see Lemma 32) that a distributional inverter suffices to get a distinguisher for **any** two circuit-samplable distributions D_0 and D_1. We then use the result of [3] that allows to invert any candidate

[5] Monotone depth-3 formulas are the only class on the list for which the equivalence problem is coNP-hard ,but the satisfiability problem is trivial. See [16] for more details.

[6] $\mathsf{FBPP}^{\mathsf{MCSP}}$ denotes the class of randomized polynomial-time algorithms with MCSP oracle.

one-way (and, in fact, any polynomial-time computable) function in randomized-polynomial time given an MCSP oracle (see Lemma 15 for more details). Indeed, we generalize the inverter of [3] to get a *distributional* inverter for any candidate distributional one-way function (see Lemma 16). We believe that this extension could be of independent interest.

We note, however, that it is fairly easy to construct a universal distinguisher in $\mathsf{FBPP}^{\mathsf{SAT}}$ by approximating the "maximum likelihood" distinguisher using the well-known fact that approximate counting can be done in $\mathsf{FBPP}^{\mathsf{NP}}$ [30]. For completeness, we provide the full proof in Theorem 40 of the appendix. From this perspective, our construction constitutes another example of a computational task that can still be performed with the MCSP oracle instead of the SAT oracle. See [27] for further discussion.

With this universal distinguisher in hand, we immediately get Theorem 4. We then obtain the circuit lower bounds in Theorems 1 and 2 by a "win-win" argument on the circuit complexity of MCSP. If $\mathsf{MCSP} \notin \mathsf{P/poly}$, we are done. Otherwise (i.e., if $\mathsf{MCSP} \in \mathsf{P/poly}$) security against $\mathsf{P/poly}$ implies security against $\mathsf{FBPP}^{\mathsf{MCSP}}$ and hence, by our universal distinguisher result above, is equivalent to statistical security for our IO. Then we leverage this very secure IO to get into $\mathsf{Algorithmica}$ where NP is "easy" by extending some ideas from [23, 48]. The latter leads to certain "collapses" of high complexity classes (such as $\mathsf{NEXP}^{\mathsf{NP}}$), which are known to contain languages outside $\mathsf{P/poly}$, to smaller complexity classes (such as NEXP). Hence we get circuit lower bounds for these smaller complexity classes, as required. Theorem 6 is proved using similar ideas.

1.3 Relation to Previous Work

The results in [22, 3] imply the following: For any two samplable distribution ensembles $\{A_n\}, \{B_n\}$, we have that $\{A_n\}, \{B_n\}$ are statistically indistinguishable if and only if they are indistinguishable by $\mathsf{FBPP}^{\mathsf{MCSP}}$ algorithms. While this result says that statistical indistinguishability and $\mathsf{FBPP}^{\mathsf{MCSP}}$-computable indistinguishability are the same for efficiently *uniformly* computable distribution ensembles, we need a stronger result applicable also to efficiently *nonuniformly* computable distributions. That is, we need a *universal* distinguisher that will distinguish any two distributions given by sampler circuits, with the distinguishing advantage close to the statistical distance between these distributions.

In [39], Naor and Rothblum used similar techniques to prove a similar result, yet with **different** quantifier order: for any *uniformly* computable distribution ensembles there exist a $\mathsf{FBPP}^{\mathsf{MCSP}}$-computable distinguisher with the distinguishing advantage close to the statistical distance between these distributions. Yet, by using the MCSP oracle as a *universal* inverter, one can "extract" a universal distinguisher from their proofs. For completeness we include a self-contained proof in the universal setting.

In [23], Goldwasser and Rothblum showed that the existence of statistically-secure obfuscators IO implies that $\mathsf{NP} \subseteq \mathsf{coAM}$, which in turn results in a collapse of the polynomial hierarchy by [11]. In particular, their idea was to solve SAT in SZK[7]. This is done by leveraging the IO to reduce SAT to *Statistical Difference* (SD) - the standard SZK-complete promise problem of [42]. The result then follows from [17, 2] where it was shown that $\mathsf{SZK} \subseteq \mathsf{AM} \cap \mathsf{coAM}$. In [48] a simplified and quantified proof of the result was presented. We use some of these ideas as a part our "win-win" argument (see Lemma 22 for more details).

[7] The class of decision problems for which a "yes" answer can be verified by a statistical zero-knowledge proof protocol.

Finally, it follows from the definition that the existence of non-uniform one-way functions (i.e. secure against P/poly) already implies very strong circuit lower bounds. Namely, NP $\not\subseteq$ P/poly. However, this approach cannot be used to derive lower bounds from the existence of an IO since the very same lower bound is already required in order to obtain a one-way function from an IO! In other words, given an IO, one-way functions exist iff NP $\not\subseteq$ P/poly. Our results allow us to obtain a weaker, but still strong circuit lower bound NEXP $\not\subseteq$ P/poly from the existence of an IO, thus avoiding this circular reference.

The rest of the paper

The necessary background is given in Section 2. Our main technical contribution (a universal distinguisher) is given in Section 3. In Section 4, we give a simple proof that SZK $\subseteq$ BPP$^{\mathsf{MCSP}}$ [5]. We give some consequences for MCSP from IO assumptions (including Theorems 5 and 6) in Section 5, and those for IO from MCSP assumptions (including Theorem 4) in Section 6. We prove Theorems 1 and 2 in Section 7. In Section 8, we prove that even an imperfect IO cannot exist in Heuristica. We conclude with some open questions in Section 9. Some auxiliary results are stated in the appendix.

2 Preliminaries

2.1 Definitions

A function negl(n) is *negligible* if for any $k \in \mathbb{N}$ there exists $n_k \in \mathbb{N}$ such that, for all $n > n_k$, negl$(n) < 1/n^k$.

▶ **Definition 3** (Statistical Distance). *Let X_0 and X_1 be two random variables taking values in some finite universe $\mathcal{U}$. The* Statistical Distance *between X_0 and X_1 is defined as*

$$\Delta(X_0, X_1) \triangleq \max_{A\colon \mathcal{U}\to\{0,1\}} \left\{\mathbf{Pr}_{u\sim X_0}[A(u)=1] - \mathbf{Pr}_{u\sim X_1}[A(u)=1]\right\},$$

where $A : \mathcal{U} \to \{0,1\}$ is an arbitrary statistical test (distinguisher).[8] *Another equivalent definition is that*

$$\Delta(X_0, X_1) = (1/2) \cdot \sum_{u\in\mathcal{U}} \left|\mathbf{Pr}_{X_0}[X_0 = u] - \mathbf{Pr}_{X_1}[X_1 = u]\right|.$$

We say that X_0 and X_1 are δ-close*, if $\Delta(X_0, X_1) \le \delta$.*

▶ **Definition 4** (Indistinguishability Obfuscator [8, 34, 12]). *We say that a randomized procedure* IO$(C; r)$ *(with randomness r) is an* Indistinguishability Obfuscator *for a circuit class $\mathcal{C}$ with the following:*

1. ***(Perfect/Imperfect) Correctness:*** IO *is* ε-imperfect *if for every circuit $C \in \mathcal{C}$:*

   $$\mathbf{Pr}_r[C \equiv \mathsf{IO}(C; r)] \ge 1 - \varepsilon(|C|).$$

   If $\varepsilon = 0$, then we say that IO *is* perfect.
2. ***Polynomial slowdown:*** *There are $a, k \in \mathbb{N}$ such that, for every circuit $C \in \mathcal{C}$ and every r,*

   $$|\mathsf{IO}(C; r)| \le a \cdot |C|^k.$$

[8] Note that the maximum is attained by the statistical test A such that $A(u) = 1 \iff \mathbf{Pr}[X_0 = u] \ge \mathbf{Pr}[X_1 = u]$.

3. ***Security:***
   a. ***Statistical:*** IO *is* statistically $(1-\delta)$-secure *if for all pairs of circuits $C_1, C_2 \in \mathcal{C}$ such that $C_1 \equiv C_2$ and $|C_1| = |C_2| = s$, we have*

   $$\Delta(\mathsf{IO}(C_1; r), \mathsf{IO}(C_2; r')) \leq \delta(s),$$

   where $\mathsf{IO}(C; r)$ is a distribution over the outputs of $\mathsf{IO}(C; r)$ for random r. We say that IO *is* statistically secure, *if $\delta(s)$ is a negligible function.*

   b. ***Computational:*** *Let $\mathcal{A}$ be a class of (randomized) algorithms. We say that IO is* $(1-\delta)$-secure against $\mathcal{A}$, *if for every algorithm $A \in \mathcal{A}$, for all pairs of sufficiently large circuits $C_1, C_2 \in \mathcal{C}$ such that $C_1 \equiv C_2$ and $|C_1| = |C_2| = s$, we have*

   $$|\mathbf{Pr}_{r,A}[A(\mathsf{IO}(C_1; r)) = 1] - \mathbf{Pr}_{r,A}[A(\mathsf{IO}(C_2; r)) = 1]| \leq \delta(s),$$

   where the probabilities are over the internal randomness r of IO as well as over possible internal randomness of A. If $\delta(s)$ is negligible, we say that IO is secure against $\mathcal{A}$. *We say that IO is* computationally secure *if it is secure against the class FBPP.*

▶ Remark 5 (Efficiency of IO). By default, we assume $\mathsf{IO}(C; r)$ is computable by a randomized polynomial-time algorithm with internal randomness r. We consider IO computable in other complexity classes, e.g., $\mathsf{FBPP}^{\mathsf{MCSP}}$. In such a case, we shall explicitly say that an IO is $\mathsf{FBPP}^{\mathsf{MCSP}}$-computable.

▶ Remark 6. Some definitions in the literature also contain a security parameter. In the above definition it is incorporated in the circuit size. Any reasonable encoding scheme for Boolean circuits allows one to represent a circuit of size s as a circuit of larger size.

We will need the following definition and result for our proofs.

▶ **Definition 7** (Statistical Difference [42]). *Let $\alpha(n) : \mathbb{N} \to \mathbb{N}$ and $\beta(n) : \mathbb{N} \to \mathbb{N}$ be computable functions, such that $\alpha(n) > \beta(n)$. Then $\mathsf{SD}^{(\alpha(n)\ ,\ \beta(n))}$ is promise problem defined as $\mathsf{SD}^{(\alpha(n)\ ,\ \beta(n))} \triangleq (\mathsf{SD}_{\text{YES}}^{(\alpha(n)\ ,\ \beta(n))}, \mathsf{SD}_{\text{NO}}^{(\alpha(n)\ ,\ \beta(n))})$, where*

$$\mathsf{SD}_{\text{YES}}^{(\alpha(n)\ ,\ \beta(n))} = \{(C_0, C_1) \mid \Delta(C_0, C_1) \geq \alpha(n)\},\quad \mathsf{SD}_{\text{NO}}^{(\alpha(n)\ ,\ \beta(n))} = \{(C_0, C_1) \mid \Delta(C_0, C_1) \leq \beta(n)\}.$$

Here, C_0 and C_1 are Boolean circuits $C_0, C_1 : \{0,1\}^n \to \{0,1\}^m$ of size $\mathsf{poly}(n)$ that are samplers for some distributions D_0 and D_1, respectively.
For the standard parameters, we define $\mathsf{SD} \triangleq \mathsf{SD}^{(2/3\ ,\ 1/3)}$.
For an oracle O, we define the relativized version of the problem $\mathsf{SD}^{O\ (\alpha(n)\ ,\ \beta(n))}$ as above, when C_0 and C_1 are O-oracle circuits.

▶ **Lemma 8** ([42]). *Suppose $\alpha(n)^2 - \beta(n) \geq 1/\mathsf{poly}(n)$. Then for any oracle O, the problem $\mathsf{SD}^{O\ (\alpha(n)\ ,\ \beta(n))}$ is SZK^O-complete. In particular, SD is SZK-complete.*

2.2 Useful Lemmas

Let $\mathsf{FBPP}^{\mathsf{MCSP}}$ denote the class of randomized polynomial-time algorithms with MCSP oracle.

▶ **Lemma 9** (implicit in [27]). *If there exists an IO $(1-\delta)$-secure against $\mathsf{FBPP}^{\mathsf{MCSP}}$, for some $\delta \leq 1 - 1/n^{\ell}$ for a constant $\ell > 0$, then $\mathsf{NP} \subseteq \mathsf{ZPP}^{\mathsf{MCSP}}$ and hence $\mathsf{ZPP}^{\mathsf{NP}} = \mathsf{ZPP}^{\mathsf{MCSP}}$.*

▶ **Lemma 10** ([48]). *If there exists an IO statistically $(1-\delta)$-secure, for some $\delta < 1$, then $\mathsf{NP} \subseteq \mathsf{coNP}$ and hence $\mathsf{PH} = \mathsf{NP} \cap \mathsf{coNP}$.*

▶ **Lemma 11** ([48]). *Let IO be an ε-imperfect obfuscator and let C_1, C_2 be such that $C_1 \not\equiv C_2$. Then $\Delta(\mathsf{IO}(C_1; r), \mathsf{IO}(C_2; r')) \geq 1 - 2\varepsilon$, over the internal randomness r, r' of the IO.*

▶ **Lemma 12** ([32]). *For any $k \in \mathbb{N}$: $\mathsf{NP}^{\mathsf{NP}} \not\subseteq \mathsf{SIZE}[n^k]$. In addition, $\mathsf{NEXP}^{\mathsf{NP}} \not\subseteq \mathsf{P/poly}$.*

▶ **Lemma 13** ([13, 33]). *If $\mathsf{SAT} \in \mathsf{P/poly}$, then $\mathsf{PH} = \mathsf{ZPP}^{\mathsf{SAT}}$, and polynomial-size circuits for SAT can be constructed in $\mathsf{ZPP}^{\mathsf{SAT}}$.*

▶ **Lemma 14** ([27]). *If $\mathsf{MCSP} \in \mathsf{P/poly}$, then $\mathsf{BPP}^{\mathsf{MCSP}} = \mathsf{ZPP}^{\mathsf{MCSP}}$.*

We require the following result of [3] that allows to find preimages of functions computable in polynomial time.

▶ **Lemma 15** ([3]). *Let $f_y(x) = f(y, x)$ be a function computable uniformly in time polynomial in $|x|$. There exists a polynomial-time probabilistic oracle Turing machine M such that for any $n, K \in \mathbb{N}$ and any y:*

$$\mathbf{Pr}_{|x|=n,r}\left[f_y\left(M^{\mathsf{MCSP}}(1^K, y, f_y(x), r)\right) = f_y(x)\right] \geq 1/K,$$

where $x \in \{0,1\}^n$ is chosen uniformly at random and r denotes the internal randomness of M.

We generalize this result to get a *distributional* inverter for any candidate distributional one-way function in the sense of [28]. Roughly speaking, such a distributional inverter finds uniformly random preimages of a given polynomial-time computable function. More precisely, we have the following.

▶ **Lemma 16.** *Let $f_y(x) = f(y, x)$ be a function computable uniformly in time polynomial in $|x|$. There exists a polynomial-time probabilistic oracle Turing machine M such that, for any $n, K \in \mathbb{N}$ and any y, the following two distributions*

$$(x,\, f_y(x)) \qquad \text{and} \qquad \left(M^{\mathsf{MCSP}}(1^K, y, f_y(x), r),\, f_y(x)\right),$$

for $x \in \{0,1\}^n$ chosen uniformly at random, and r the internal uniform randomness of M, are at most $(1/K)$-far in statistical distance.

Proof. We combine Lemma 15 with the reduction from [28] showing that an inverter for candidate one-way functions can be used to get a distributional inverter for every distributional one-way function candidate $f_y(x)$ computable in polynomial time. ◀

3 From Computational to Statistical Security

Below we will argue the existence of a universal distinguisher. We will describe an algorithm $\mathcal{D}(C_0, C_1; 1^{1/\gamma})$ in $\mathsf{FBPP}^{\mathsf{MCSP}}$ that, given any pair of circuits C_0 and C_1 that are samplers for some distributions D_0 and D_1, and a parameter $0 < \gamma < 1$, will distinguish D_0 and D_1 with advantage at least $\delta - \gamma$, where δ is the statistical distance between D_0 and D_1.

▶ **Corollary 17.** *There is an $\mathsf{FBPP}^{\mathsf{MCSP}}$ algorithm $\mathcal{D}$ satisfying the following. Given any pair of circuits C_0 and C_1 that are samplers for some distributions D_0 and D_1, and given a parameter K in unary, the algorithm $\mathcal{D}(C_0, C_1; 1^K)$ will distinguish D_0 and D_1 with advantage at least $\delta - 1/K$, where δ is the statistical distance between D_0 and D_1.*

As was mentioned, a similar proof was given in [39]. We defer the proof to Section A of the appendix.

▶ Remark 18. We note that a universal distinguisher as in Corollary 17 is fairly easy to construct in $\mathsf{FBPP}^{\mathsf{SAT}}$ (using the well-known fact that approximate counting can be done in $\mathsf{FBPP}^{\mathsf{NP}}$ [30]); see Theorem 40 in Section B of the appendix. Thus, Corollary 17 is another example of a computational task that can still be performed with the MCSP oracle instead of the SAT oracle.

4 Another Proof that $\mathsf{SZK} \subseteq \mathsf{BPP}^{\mathsf{MCSP}}$

Corollary 17 can be used to give another proof of the following result by Allender and Das [5].

▶ **Theorem 19** ([5]). $\mathsf{SZK} \subseteq \mathsf{BPP}^{\mathsf{MCSP}}$.

Proof. Recall the standard SZK-complete promise problem Statistical Difference (SD) (see Definition 7): Given a pair of circuits (C_0, C_1) that are samplers for the distributions D_0 and D_1 such that either D_0 and D_1 have the statistical distance less than $1/3$, or they have the statistical distance greater than $2/3$, decide which is the case.

By Corollary 17, we get an $\mathsf{FBPP}^{\mathsf{MCSP}}$ universal distinguisher $\mathcal{D}$. Consider the distinguisher $B = \mathcal{D}(C_0, C_1; 1^{10})$. Let δ be the statistical distance between D_0 and D_1. Note that in case $\delta < 1/3$, the algorithm B (and, in fact, any algorithm) has distinguishing advantage less than $1/3$, whereas for $\delta > 2/3$, B has advantage at least $(2/3) - (1/10) = 17/30 > 1/3$. Using random sampling and the Chernoff bounds, we can estimate in $\mathsf{FBPP}^{\mathsf{MCSP}}$ the advantage of our algorithm B at distinguishing between D_0 and D_1, with high probability and sufficient accuracy. The theorem follows. ◀

▶ Remark 20. Note that the $\mathsf{BPP}^{\mathsf{MCSP}}$ algorithm for SZK in the proof of Theorem 19 works for any version of the Statistical Difference problem with a non-negligible gap between the yes- and no-instances, not just for the $1/3$ vs. $2/3$ gap.

Next, we extend the result above to the relativized version of the problem SD^O (see Definition 7) for any $O \in \mathsf{BPP}^{\mathsf{MCSP}} \cap \mathsf{P/poly}$.

▶ **Theorem 21.** *Let $O \in \mathsf{BPP}^{\mathsf{MCSP}} \cap \mathsf{P/poly}$ be any language. Then for any $\alpha(n)$ and $\beta(n)$ such that $\alpha(n) \geq \beta(n) + n^{-\ell}$, for some $\ell > 0$, we have that:*

$$\mathsf{SD}^{O\ (\alpha(n)\ ,\ \beta(n))} \in \mathsf{BPP}^{\mathsf{MCSP}}. \tag{1}$$

In particular, if $\mathsf{MCSP} \in \mathsf{P/poly}$, then

$$\mathsf{SZK}^{\mathsf{MCSP}} \subseteq \mathsf{BPP}^{\mathsf{MCSP}}. \tag{2}$$

Proof. To prove (1), we proceed exactly as in the proof of Theorem 19 above, except using Corollary 38 instead of Corollary 17, and using the observation in Remark 20. To prove (2), we use (1) for $O = \mathsf{MCSP}$ and the fact that $\mathsf{SD}^{\mathsf{MCSP}\ (2/3\ ,\ 1/3)}$ is $\mathsf{SZK}^{\mathsf{MCSP}}$-complete (by Lemma 8). ◀

5 Implications for Circuit Minimization from Obfuscation

The following lemma provides some consequences of the existence of an imperfect, computationally-secure IO, with an appropriate range of parameters. Among other things, the proof uses some ideas from [23] and [27].

▶ **Lemma 22.** *Let $\Gamma \in \{\mathsf{FBPP}, \mathsf{P/poly}\}$. Suppose there exist an ε-imperfect $\mathsf{IO}(C; r)$ that is $(1-\delta)$-secure against Γ, where $(1-2\varepsilon)^2 - \delta \geq 2/n^\ell$ for some constant $\ell > 0$. If $\mathsf{MCSP} \in \Gamma$, then:*

1. $\mathsf{NP} \subseteq \mathsf{SZK}$,
2. $\mathsf{PH} = \mathsf{MA} = \mathsf{ZPP}^{\mathsf{MCSP}}$, *and*
3. *There is a $\mathsf{ZPP}^{\mathsf{MCSP}}$ algorithm A and a constant $k > 0$, such that $A(1^n)$ outputs an $O(n^k)$-size circuit for SAT (and for MCSP) on n-bit inputs.*

Proof.

1. First, observe that since $\mathsf{MCSP} \in \Gamma$, $\mathsf{FBPP}^{\mathsf{MCSP}} \subseteq \Gamma$. Consequently, an IO that is secure against Γ is also secure against $\mathsf{FBPP}^{\mathsf{MCSP}}$. It follows from Corollary 17 that this IO is statistically $(1-\delta')$-secure, for $\delta' = \delta + 1/n^{\ell}$ (since we can make the distributional inverter's error α to be smaller than any inverse polynomial of our choice). We now use this IO to reduce SAT to the Statistical Difference problem (see Definition 7): Given a SAT instance ϕ, construct some unsatisfiable instance $\perp$ of the same size as ϕ and on the same set of input variables. Consider the distributions

$$\mathsf{IO}(\phi; r) \text{ and } \mathsf{IO}(\perp; r') \tag{3}$$

over all random strings r, r'.
We have two cases:

- If ϕ is unsatisfiable, then $\phi \equiv \perp$, and by the statistical $(1-\delta')$-security property of our IO, we get that these two distributions in (3) have statistical distance at most δ'.
- If ϕ is satisfiable, then by Lemma 11, the statistical distance between the distributions in (3) is at least $1-2\varepsilon$.

Since $(1-2\varepsilon)^2 \geq \delta + 2n^{-\ell} = \delta' + n^{-\ell}$, by Lemma 8, the resulting instance of the SD problem is SZK-complete, and so $\mathsf{NP} \subseteq \mathsf{SZK}$.[9]

2. By [17, 2], we have $\mathsf{SZK} \subseteq \mathsf{AM} \cap \mathsf{coAM}$. By [11], since $\mathsf{NP} \subseteq \mathsf{SZK} \subseteq \mathsf{coAM}$, it follows that

$$\mathsf{PH} = \mathsf{AM}. \tag{4}$$

Next, by Theorem 19, $\mathsf{SZK} \subseteq \mathsf{BPP}^{\mathsf{MCSP}}$. By Lemma 14, $\mathsf{BPP}^{\mathsf{MCSP}} = \mathsf{ZPP}^{\mathsf{MCSP}}$. Hence, we get that

$$\mathsf{NP} \subseteq \mathsf{SZK} \subseteq \mathsf{ZPP}^{\mathsf{MCSP}}. \tag{5}$$

As $\mathsf{MCSP} \in \mathsf{P/poly}$, we also get from (5) that

$$\mathsf{NP} \subseteq \mathsf{P/poly}. \tag{6}$$

By [6], (6) implies that $\mathsf{AM} = \mathsf{MA}$. So by (4), we conclude that

$$\mathsf{PH} = \mathsf{MA}.$$

Finally, (6) also implies $\mathsf{PH} = \mathsf{ZPP}^{\mathsf{NP}}$ by Lemma 13. Hence, by (5), we get that

$$\mathsf{PH} = \mathsf{ZPP}^{\mathsf{NP}} \subseteq \mathsf{ZPP}^{\mathsf{ZPP}^{\mathsf{MCSP}}} = \mathsf{ZPP}^{\mathsf{MCSP}}.$$

3. By Lemma 13, if $\mathsf{SAT} \in \mathsf{P/poly}$, then polynomial-size circuits for SAT can be found by a $\mathsf{ZPP}^{\mathsf{NP}}$ algorithm. By (6), we get that polynomial-size circuits for SAT can be found by a $\mathsf{ZPP}^{\mathsf{ZPP}^{\mathsf{MCSP}}}$ algorithm, which can be simulated by a $\mathsf{ZPP}^{\mathsf{MCSP}}$ algorithm. As $\mathsf{MCSP} \in \mathsf{NP}$ and SAT is NP-complete, polynomial-size circuits for SAT can be used to construct polynomial-size circuits for MCSP as well. ◀

[9] Note that this reduction to SZK actually allows one to solve not just SAT but an *equivalence* problem for any class of circuits that an IO can obfuscate. Thus, to conclude that $\mathsf{NP} \subseteq \mathsf{SZK}$, it suffices to pick any coNP-hard circuit equivalence problem for the class of circuits where SAT may be easy. For example, one can take the problem of testing equivalence of depth-3 *monotone* formulas, known to be coNP-complete [16]

Items (2) and (3) in the lemma should be contrasted with the result of [13, 33] that $\mathsf{SAT} \in \mathsf{P/poly}$ implies both that polynomial-size circuits for SAT can be constructed by a $\mathsf{ZPP}^{\mathsf{SAT}}$ algorithm, and that $\mathsf{PH} = \mathsf{ZPP}^{\mathsf{SAT}}$. Under an additional assumption that a $\mathsf{P/poly}$-secure imperfect IO exists, we get similar implications for MCSP instead of SAT.
The following corollary strengthens a result of [27] to the imperfect setting.

▶ **Corollary 23** (Theorem 5 re-stated). *Let $\Gamma \in \{\mathsf{ZPP}, \mathsf{BPP}\}$. Suppose there is an ε-imperfect* IO *that is $(1-\delta)$-secure against* FBPP*, where $(1-2\varepsilon)^2 \geq \delta + 2/n^\ell$ for some constant $\ell > 0$. Then* $\mathsf{MCSP} \in \Gamma$ *iff* $\mathsf{NP} \subseteq \Gamma$.

Proof. The first direction is clear since $\mathsf{MCSP} \in \mathsf{NP}$. For the other direction, by Lemma 22, $\mathsf{NP} \subseteq \mathsf{ZPP}^{\mathsf{MCSP}}$ and hence $\mathsf{NP} \subseteq \mathsf{ZPP}^{\Gamma}$. ◀

For the case of a perfect IO, we get a somewhat stronger statement.

▶ **Theorem 24** (Theorem 6 re-stated). *Suppose there is a perfect* IO *that is $(1-\delta)$-secure against* FBPP*, where $\delta \leq 1 - 2/n^\ell$ for some constant $\ell > 0$. If* $\mathsf{MCSP} \in \mathsf{BPP}$*, then* $\mathsf{NP} = \mathsf{ZPP}$.

Proof. Since $\mathsf{MCSP} \in \mathsf{BPP}$, computational $(1-\delta)$-security implies $(1-\delta)$-security against $\mathsf{FBPP}^{\mathsf{MCSP}}$. It follows by Corollary 17 that this IO is statistically $(1-\delta')$-secure, for $\delta' = \delta + 1/n^\ell \leq 1 - 1/n^\ell$.

By Lemma 10, $\mathsf{PH} = \mathsf{NP} = \mathsf{coNP}$. By Lemma 9, $\mathsf{PH} = \mathsf{NP} = \mathsf{ZPP}^{\mathsf{MCSP}} \subseteq \mathsf{BPP}$. But $\mathsf{NP} \subseteq \mathsf{BPP}$ implies that $\mathsf{NP} = \mathsf{RP}$. Since $\mathsf{coNP} = \mathsf{NP}$, we get $\mathsf{NP} = \mathsf{ZPP}$. ◀

6 Implications for Obfuscation from Circuit Minimization

▶ **Theorem 25.** *Suppose* $\mathsf{MCSP} \in \mathsf{P/poly}$*. There is an* $\mathsf{FZPP}^{\mathsf{MCSP}}$*-computable perfect* IO *that is statistically secure if and only if there is an* $\mathsf{FBPP}^{\mathsf{MCSP}}$*-computable ε-imperfect* IO *that is $(1-\delta)$ secure against* $\mathsf{P/poly}$*, for any $0 \leq \varepsilon, \delta \leq 1$ such that $1 - 2\varepsilon \geq \delta + 2/n^\ell$, for some constant $\ell > 0$.*

Proof. The interesting direction is from the right to the left. Since $\mathsf{MCSP} \in \mathsf{P/poly}$, $(1-\delta)$-security against $\mathsf{P/poly}$ implies, by Corollary 17, statistical $(1-\delta')$-security, for $\delta' = \delta + n^{-\ell}$.

▷ Claim 26. If $\mathsf{MCSP} \in \mathsf{P/poly}$ and there is an $\mathsf{FBPP}^{\mathsf{MCSP}}$-computable ε-imperfect IO that is statistically $(1-\delta')$-secure for $1 - 2\varepsilon \geq \delta' + n^{-\ell}$, for some constant $\ell > 0$, then $\mathsf{SAT} \in \mathsf{ZPP}^{\mathsf{MCSP}}$.

Proof of Claim 26. Given an instance ϕ of SAT, let $\bot$ be an unsatisfiable formula of the same size as ϕ (over the same variables). Consider the two distributions $\mathsf{IO}(\phi; r)$ and $\mathsf{IO}(\bot; r')$ over random r, r'. If $\phi \equiv \bot$, the two distributions are at most statistical distance δ' apart; if ϕ is in SAT, then the two distributions have the statistical distance at least $1 - 2\varepsilon$.

Each distribution is samplable using a polynomial-size MCSP-oracle circuit, which we can obtain from our IO algorithm. Thus, we get an $\mathsf{FP}^{\mathsf{MCSP}}$-reduction from coSAT to $\mathsf{SD}^{\mathsf{MCSP}\ (1-2\varepsilon\ ,\ \delta')}$. Since $\delta' + n^{-\ell} \leq 1 - 2\varepsilon$, we conclude by Theorem 21 that $\mathsf{SAT} \in \mathsf{BPP}^{\mathsf{MCSP}}$. By Lemma 14, $\mathsf{BPP}^{\mathsf{MCSP}} = \mathsf{ZPP}^{\mathsf{MCSP}}$, concluding the proof. ◁

Since $\mathsf{SAT} \in \mathsf{ZPP}^{\mathsf{MCSP}} \subseteq \mathsf{P/poly}$, we get by Lemma 13 that $\mathsf{PH} = \mathsf{ZPP}^{\mathsf{MCSP}}$. Given a circuit C, we can find the lexicographically smallest equivalent circuit D (of size at most that of C) in $\mathsf{FP}^{\mathsf{PH}} \subseteq \mathsf{FZPP}^{\mathsf{MCSP}}$. This gives us a perfect $\mathsf{IO}(C; r)$ that is statistically secure.[10] ◀

[10] Technically, this $\mathsf{IO}(C; r)$ outputs either a smallest equivalent circuit D, or, with a tiny probability, the "don't know" answer. We can modify it to output the input circuit C in the latter case, getting perfect correctness, and only slightly decreasing statistical security.

Along the same lines:

▶ **Corollary 27.** *Suppose* $\mathsf{MCSP} \in \mathsf{BPP}$*. There is an* ε*-imperfect* IO *with statistical security if and only if there is an* ε*-imperfect* IO *with computational* $(1-\delta)$*-security where* $1-2\varepsilon \geq \delta+2/n^{\ell}$*. (Assuming* $\mathsf{MCSP} \in \mathsf{ZPP}$*, you get a similar equivalence but for a perfect* IO *with statistical security.)*

Proof sketch. The interesting direction is from the right to the left. We first argue as in the proof of Theorem 25 to conclude that $\mathsf{SAT} \in \mathsf{ZPP}^{\mathsf{MCSP}}$. Since $\mathsf{MCSP} \in \mathsf{BPP}$, we get that $\mathsf{NP} \subseteq \mathsf{BPP}$, and hence, $\mathsf{PH} = \mathsf{BPP}$. So, given an input circuit C, we can find the lexicographically smallest equivalent circuit D (of size at most that of C), using an $\mathsf{FP}^{\mathsf{PH}} = \mathsf{FBPP}$ algorithm. This algorithm is a (negligibly) imperfect IO with statistical security. (In case of $\mathsf{MCSP} \in \mathsf{ZPP}$, we argue in a similar way, getting that $\mathsf{PH} = \mathsf{ZPP}$, and so a canonical circuit D for a given input circuit C can be found in FZPP.) ◀

7 Circuit Lower Bounds from Obfuscation

Here we prove Theorems 1 and 2, re-stated below.

▶ **Theorem 28** (Theorem 1 re-stated)**.** *Suppose there exist a perfect* IO $(1-\delta)$*-secure against* P/poly*, where* $\delta \leq 1 - 2/n^{\ell}$ *for some* $\ell > 0$*. Then:*
1. $\mathsf{NEXP} \cap \mathsf{ZPEXP}^{\mathsf{MCSP}} \not\subseteq \mathsf{P/poly}$.
2. *For all* $k \in \mathbb{N}$*,* $\mathsf{NP} \cap \mathsf{ZPP}^{\mathsf{MCSP}} \not\subseteq \mathsf{SIZE}[n^k]$*.*

Proof. The proof of all items goes by a "win-win" argument. Suppose $\mathsf{MCSP} \notin \mathsf{P/poly}$. Then both claims follow immediately since $\mathsf{MCSP} \in \mathsf{NP}$.

Now suppose $\mathsf{MCSP} \in \mathsf{P/poly}$. Then randomized polynomial-time algorithms with MCSP oracle can be simulated by polynomial-size circuits. Consequently, IO is $(1-\delta)$-secure against these algorithms. By Corollary 17, this IO is statistically $(1-\delta')$-secure, for $\delta' = \delta + n^{-\ell} \leq 1 - n^{-\ell}$. By Lemmas 9 and 10, we get that

$$\mathsf{NP}^{\mathsf{NP}} \subseteq \mathsf{PH} = \mathsf{NP} \cap \mathsf{coNP} \subseteq \mathsf{ZPP}^{\mathsf{MCSP}} \subseteq \mathsf{NP}^{\mathsf{NP}}.$$

So, $\mathsf{NP}^{\mathsf{NP}} = \mathsf{NP} \cap \mathsf{coNP} = \mathsf{ZPP}^{\mathsf{MCSP}}$. By padding, $\mathsf{NEXP}^{\mathsf{NP}} = \mathsf{NEXP} \cap \mathsf{coNEXP} = \mathsf{ZPEXP}^{\mathsf{MCSP}}$, and so both claims follow from Lemma 12. ◀

▶ **Theorem 29** (Theorem 2 re-stated)**.** *Suppose there exist an* ε*-imperfect* IO $(1-\delta)$*-secure against* P/poly*, where* $(1-2\varepsilon)^2 \geq \delta + 2/n^{\ell}$ *for some* $\ell > 0$*. Then for all* $k \in \mathbb{N}$*,* $\mathsf{MA} \cap \mathsf{ZPP}^{\mathsf{MCSP}} \not\subseteq \mathsf{SIZE}[n^k]$*.*

Proof. Again we use a "win-win" argument. If $\mathsf{MCSP} \notin \mathsf{P/poly}$, then the theorem follows. Otherwise, we get by Lemma 22 (Item 2) that $\mathsf{PH} = \mathsf{MA} = \mathsf{ZPP}^{\mathsf{MCSP}}$, which is not in $\mathsf{SIZE}[n^k]$ for any fixed $k > 0$ by Lemma 12. ◀

▶ **Theorem 30** (Theorem 3 re-stated)**.** *Suppose there is a perfect* IO *that is* $(1-\delta)$*-secure against* FBPP*, where* $\delta \leq 1 - 2/n^{\ell}$ *for some constant* $\ell > 0$*. Then* $\mathsf{ZPEXP} \neq \mathsf{BPP}$*.*

Proof of Theorem 3. Suppose for a contradiction that $\mathsf{ZPEXP} = \mathsf{BPP}$. Then, in particular, $\mathsf{MCSP} \in \mathsf{BPP}$. By Theorem 6, $\mathsf{NP} = \mathsf{ZPP}$ and hence

$$\mathsf{ZPEXP} = \mathsf{BPP} \subseteq \mathsf{ZPP}^{\mathsf{NP}} = \mathsf{ZPP}^{\mathsf{ZPP}} = \mathsf{ZPP}$$

which leads to a contradiction (see Appendix C). ◀

8 Excluding an Imperfect IO from Heuristica

Below we assume that the reader is familiar with the basic definitions of average-case complexity (in particular, the definitions of DistNP and AvgP); see, e.g., [9].

▶ **Theorem 31.** *Suppose* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$*. If an* ε*-imperfect computationally* $(1-\delta)$*-secure* IO *exists for* $1-2\varepsilon \geq \delta + 2n^{-\ell}$ *for some* $\ell > 0$*, then* $\mathsf{NP} = \mathsf{P}$.

Proof. Consider MCSP with $s = 2^{0.9n}$ under the uniform distribution over 2^n-bit inputs. This is a language in DistNP. If $\mathsf{DistNP} \subseteq \mathsf{AvgP}$, we get a language $L' \in \mathsf{P}$ that agrees with MCSP for $s = 2^{0.9n}$ on almost all instances, but may be incorrect on a tiny fraction of "no" instances of MCSP (here we use the zero-error property of problems in AvgP). Then the complement $L = \bar{L}'$ is a language in P of polynomial density (because almost all strings are very hard) such that for every $x \in L$, the circuit complexity of x (when viewed as a truth table of a boolean function) is at least $|x|^{0.9}$. All results in this paper that use MCSP as an oracle continue to hold with any such L as an oracle instead. In particular, as in the proof of Theorem 25 (see Claim 26), we conclude that $\mathsf{NP} \subseteq \mathsf{BPP}^L = \mathsf{BPP}$. Finally, by [14], if $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ then $\mathsf{BPP} = \mathsf{P}$, and so $\mathsf{NP} = \mathsf{P}$. ◀

9 Open Questions

In this paper we showed that an (even imperfect) IO secure against non-uniform polynomial-size circuits implies non-trivial circuit lower bounds. Can one prove circuit lower bounds from the assumption that a (uniform) computationally-secure IO exists?

Can we leverage the connection between one-way functions and a close relative of MCSP (time-bounded Kolmogorov complexity) [37, 4, 25, 24] to get better understanding of IO?

References

1 Leonard M. Adleman. Two theorems on random polynomial time. In *19th Annual Symposium on Foundations of Computer Science, Ann Arbor, Michigan, USA, 16-18 October 1978*, pages 75–83. IEEE Computer Society, 1978. doi:10.1109/SFCS.1978.37.

2 William Aiello and Johan Håstad. Statistical zero-knowledge languages can be recognized in two rounds. *J. Comput. Syst. Sci.*, 42(3):327–345, 1991. doi:10.1016/0022-0000(91)90006-Q.

3 Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006. doi:10.1137/050628994.

4 Eric Allender, Mahdi Cheraghchi, Dimitrios Myrisiotis, Harsha Tirumala, and Ilya Volkovich. One-way functions and a conditional variant of MKTP. In Mikolaj Bojanczyk and Chandra Chekuri, editors, *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, December 15-17, 2021, Virtual Conference*, volume 213 of *LIPIcs*, pages 7:1–7:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.FSTTCS.2021.7.

5 Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. *Inf. Comput.*, 256:2–8, 2017. doi:10.1016/j.ic.2017.04.004.

6 Vikraman Arvind, Johannes Köbler, Uwe Schöning, and Rainer Schuler. If NP has polynomial-size circuits, then MA=AM. *Theor. Comput. Sci.*, 137(2):279–282, 1995. doi:10.1016/0304-3975(95)91133-B.

7 Boaz Barak. A probabilistic-time hierarchy theorem for "slightly non-uniform" algorithms. In José D. P. Rolim and Salil P. Vadhan, editors, *Randomization and Approximation Techniques, 6th International Workshop, RANDOM 2002, Cambridge, MA, USA, September 13-15, 2002, Proceedings*, volume 2483 of *Lecture Notes in Computer Science*, pages 194–208. Springer, 2002. doi:10.1007/3-540-45726-7_16.

8 Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, 2012. doi:10.1145/2160158.2160159.

9 Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Found. Trends Theor. Comput. Sci.*, 2(1), 2006. doi:10.1561/0400000004.

10 Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. *Algorithmica*, 79(4):1233–1285, 2017. doi:10.1007/s00453-016-0242-8.

11 Ravi B. Boppana, Johan Håstad, and Stathis Zachos. Does co-np have short interactive proofs? *Inf. Process. Lett.*, 25(2):127–132, 1987. doi:10.1016/0020-0190(87)90232-8.

12 Zvika Brakerski, Christina Brzuska, and Nils Fleischhacker. On statistically secure obfuscation with approximate correctness. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016 – 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 551–578. Springer, 2016. doi:10.1007/978-3-662-53008-5_19.

13 Nader H. Bshouty, Richard Cleve, Ricard Gavaldà, Sampath Kannan, and Christino Tamon. Oracles and queries that are sufficient for exact learning. *J. Comput. Syst. Sci.*, 52(3):421–433, 1996. doi:10.1006/jcss.1996.0032.

14 Harry Buhrman, Lance Fortnow, and Aduri Pavan. Some results on derandomization. *Theory Comput. Syst.*, 38(2):211–227, 2005. doi:10.1007/s00224-004-1194-y.

15 Harry Buhrman and Leen Torenvliet. Randomness is hard. *SIAM J. Comput.*, 30(5):1485–1501, 2000. doi:10.1137/S0097539799360148.

16 Thomas Eiter and Georg Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM J. Comput.*, 24(6):1278–1304, 1995. doi:10.1137/S0097539793250299.

17 Lance Fortnow. The complexity of perfect zero-knowledge. *Adv. Comput. Res.*, 5:327–343, 1989.

18 Lance Fortnow and Rahul Santhanam. Hierarchy theorems for probabilistic polynomial time. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 316–324. IEEE Computer Society, 2004. doi:10.1109/FOCS.2004.33.

19 Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM J. Comput.*, 45(3):882–929, 2016. doi:10.1137/14095772X.

20 Sanjam Garg and Antigoni Polychroniadou. Two-round adaptively secure MPC from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography – 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 614–637. Springer, 2015. doi:10.1007/978-3-662-46497-7_24.

21 Craig Gentry, Allison Bishop Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 151–170. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.19.

22 Oded Goldreich. A note on computational indistinguishability. *Inf. Process. Lett.*, 34(6):277–281, 1990. doi:10.1016/0020-0190(90)90010-U.

23 Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. *J. Cryptol.*, 27(3):480–505, 2014. doi:10.1007/s00145-013-9151-z.

24 Shuichi Hirahara. Capturing one-way functions via np-hardness of meta-complexity. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 1027–1038. ACM, 2023. doi:10.1145/3564246.3585130.

25 Rahul Ilango, Hanlin Ren, and Rahul Santhanam. Hardness on any samplable distribution suffices: New characterizations of one-way functions by meta-complexity. *Electron. Colloquium Comput. Complex.*, TR21-082, 2021. arXiv:TR21-082.

26 Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference, Minneapolis, Minnesota, USA, June 19-22, 1995*, pages 134–147. IEEE Computer Society, 1995. doi:10.1109/SCT.1995.514853.

27 Russell Impagliazzo, Valentine Kabanets, and Ilya Volkovich. The power of natural properties as oracles. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, volume 102 of *LIPIcs*, pages 7:1–7:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CCC.2018.7.

28 Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October – 1 November 1989*, pages 230–235. IEEE Computer Society, 1989. doi:10.1109/SFCS.1989.63483.

29 Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 60–73. ACM, 2021. doi:10.1145/3406325.3451093.

30 Mark Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comput. Sci.*, 43:169–188, 1986. doi:10.1016/0304-3975(86)90174-X.

31 Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In F. Frances Yao and Eugene M. Luks, editors, *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 73–79. ACM, 2000. doi:10.1145/335305.335314.

32 Ravi Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Inf. Control.*, 55(1-3):40–56, 1982. doi:10.1016/S0019-9958(82)90382-5.

33 Johannes Köbler and Osamu Watanabe. New collapse consequences of NP having small circuits. *SIAM J. Comput.*, 28(1):311–324, 1998. doi:10.1137/S0097539795296206.

34 Ilan Komargodski, Tal Moran, Moni Naor, Rafael Pass, Alon Rosen, and Eylon Yogev. One-way functions and (im)perfect obfuscation. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 374–383. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.47.

35 Ilan Komargodski, Moni Naor, and Eylon Yogev. Secret-sharing for NP. *J. Cryptol.*, 30(2):444–469, 2017. doi:10.1007/s00145-015-9226-0.

36 Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In Marc Fischlin and Jean-Sébastien Coron, editors, *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*, volume 9665 of *Lecture Notes in Computer Science*, pages 28–57. Springer, 2016. doi:10.1007/978-3-662-49890-3_2.

37 Yanyi Liu and Rafael Pass. On one-way functions and kolmogorov complexity. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1243–1254. IEEE, 2020. doi:10.1109/FOCS46700.2020.00118.

38 Cody D. Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime from a new easy witness lemma. *SIAM J. Comput.*, 49(5), 2020. doi:10.1137/18M1195887.

39 Moni Naor and Guy N. Rothblum. Learning to impersonate. In William W. Cohen and Andrew W. Moore, editors, *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of *ACM International Conference Proceeding Series*, pages 649–656. ACM, 2006. doi:10.1145/1143844.1143926.

40 Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014 – 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 500–517. Springer, 2014. doi:10.1007/978-3-662-44371-2_28.

41 Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997. doi:10.1006/jcss.1997.1494.

42 Amit Sahai and Salil P. Vadhan. A complete problem for statistical zero knowledge. *J. ACM*, 50(2):196–249, 2003. doi:10.1145/636865.636868.

43 Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. *SIAM J. Comput.*, 50(3):857–908, 2021. doi:10.1137/15M1030108.

44 Rahul Santhanam. Circuit lower bounds for merlin–arthur classes. *SIAM J. Comput.*, 39(3):1038–1061, 2009. doi:10.1137/070702680.

45 Boris A. Trakhtenbrot. A survey of russian approaches to perebor (brute-force searches) algorithms. *IEEE Ann. Hist. Comput.*, 6(4):384–400, 1984. doi:10.1109/MAHC.1984.10036.

46 Dieter van Melkebeek and Konstantin Pervyshev. A generic time hierarchy with one bit of advice. *Comput. Complex.*, 16(2):139–179, 2007. doi:10.1007/s00037-007-0227-8.

47 Ilya Volkovich. On learning, lower bounds and (un)keeping promises. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *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 1027–1038. Springer, 2014. doi:10.1007/978-3-662-43948-7_85.

48 Ilya Volkovich. The final nail in the coffin of statistically-secure obfuscator. *Information Processing Letters*, 182:106366, 2023. doi:10.1016/j.ipl.2023.106366.

49 Ryan Williams. Towards NEXP versus bpp? In Andrei A. Bulatov and Arseny M. Shur, editors, *Computer Science – Theory and Applications – 8th International Computer Science Symposium in Russia, CSR 2013, Ekaterinburg, Russia, June 25-29, 2013. Proceedings*, volume 7913 of *Lecture Notes in Computer Science*, pages 174–182. Springer, 2013. doi:10.1007/978-3-642-38536-0_15.

50 Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014. doi:10.1145/2559903.

A A Universal Distinguisher in $\mathsf{FBPP}^{\mathsf{MCSP}}$

We will first argue that such a distinguisher for a pair of distributions sampled by circuits C_0 and C_1 can be obtained given oracle access to a distributional inverter (in the sense of Impagliazzo and Luby [28]) for a function defined in terms of C_0 and C_1 (see Lemma 32 below). Then we appeal to Lemma 16 to get a universal distributional inverter.

▶ **Lemma 32.** *There is an oracle* FBPP *algorithm $\hat{\mathcal{D}}$ satisfying the following. Let C_0 and C_1 be two circuits that are samplers for distributions D_0 and D_1 over some finite universe $\mathcal{U}$, and let δ be the statistical distance between D_0 and D_1. Let $F(b,r)$ use r to sample from D_b. Let A be a distributional inverter for F so that the distributions*

$$((b,r),\ F(b,r)) \qquad \text{and} \qquad (A(F(b,r)),\ F(b,r))$$

are at most α^2-close in statistical distance, where $0 \leq \alpha \leq \delta/28$. Then $\hat{\mathcal{D}}^A(C_0, C_1; 1^{1/\alpha})$ is a distinguisher for D_0 and D_1 with advantage at least $\delta - 14\alpha \geq \delta/2$.

Proof. Let $B(x)$ be the first bit of $A(x)$, and let $Q(x)$ be the probability that $B(x)$ is 0, i.e.,

$$Q(x) = \mathbf{Pr}_A[B(x) = 0],$$

where the probability is over the internal randomness of A. Let K be such that an empirical estimate of K iid $\{0,1\}$-valued random variables is within α of its expectation with probability $1-(\alpha/2)$; by the Chernoff bounds, we have that $K = O((\log 1/\alpha)/\alpha^2)$. Let $\tilde{Q}(x;\rho)$ be the random variable where we use randomness ρ to sample from $A(x)$ independently K times and use these to create an empirical estimate of $Q(x)$; so ρ is K times the internal randomness of A. Let $C(x;\rho)$ be the probabilistic Boolean algorithm where we accept x if $\tilde{Q}(x;\rho) \geq 1/2$. We will show that

$$\mathbf{Pr}_{x\sim D_0,\rho}[C(x;\rho)=1] - \mathbf{Pr}_{x\sim D_1,\rho}[C(x;\rho)=1] \geq \delta - 14\alpha. \tag{7}$$

Let $p_0(x)$ be the probability of x for D_0, and $p_1(x)$ that for D_1. Note that

$$q(x) = p_0(x)/(p_0(x)+p_1(x))$$

is the conditional probability that $b=0$ given that $F(b,r)=x$. Then

$$\mathbf{Pr}_{x\sim D_0,\rho}[C(x;\rho)=1] - \mathbf{Pr}_{x\sim D_1,\rho}[C(x;\rho)=1] = \mathbf{Exp}_\rho\left[\sum_{x:\ \tilde{Q}(x;\rho)\geq 1/2}(p_0(x)-p_1(x))\right], \tag{8}$$

and

$$\delta = \sum_{x:\ q(x)\geq 1/2}(p_0(x)-p_1(x)). \tag{9}$$

Note that if, for "typical" randomness ρ used by $\tilde{Q}$, we had for all $x \in \mathcal{U}$ that $\tilde{Q}(x;\rho) \geq 1/2 \Leftrightarrow q(x) \geq 1/2$, then the right-hand sides of (8) and (9) would be identical (for that randomness ρ of $\tilde{Q}$), and we would get our goal of (7) minus the error term for "atypical" randomness of $\tilde{Q}$. We formalize this argument next.

For given internal randomness ρ of $\tilde{Q}$, let the error set $E = E(\rho)$ be the set of those $x \in \mathcal{U}$ so that exactly one of $\tilde{Q}(x;\rho)$ and $q(x)$ is at least $1/2$, i.e.,

$$E(\rho) = \{x \in \mathcal{U} \mid \tilde{Q}(x;\rho) \geq 1/2 \not\Leftrightarrow q(x) \geq 1/2\}.$$

Then

$$\begin{aligned}
&\mathbf{Pr}_{x\sim D_0,\rho}[C(x;\rho)=1] - \mathbf{Pr}_{x\sim D_1,\rho}[C(x;\rho)=1] \\
&= \mathbf{Exp}_\rho\left[\sum_{x:\ \tilde{Q}(x;\rho)\geq 1/2}(p_0(x)-p_1(x))\right] \\
&= \mathbf{Exp}_\rho\left[\sum_{x\notin E(\rho):\ \tilde{Q}(x;\rho)\geq 1/2}(p_0(x)-p_1(x)) + \sum_{x\in E(\rho):\ \tilde{Q}(x;\rho)\geq 1/2}(p_0(x)-p_1(x))\right] \\
&= \mathbf{Exp}_\rho\left[\sum_{x\notin E(\rho):\ q(x)\geq 1/2}(p_0(x)-p_1(x)) + \sum_{x\in E(\rho):\ q(x)<1/2}(p_0(x)-p_1(x))\right] \\
&= \mathbf{Exp}_\rho\left[\sum_{x:\ q(x)\geq 1/2}(p_0(x)-p_1(x)) + \sum_{x\in E(\rho):\ q(x)<1/2}(p_0(x)-p_1(x)) - \sum_{x\in E(\rho):\ q(x)\geq 1/2}(p_0(x)-p_1(x))\right] \\
&\geq \delta - \mathbf{Exp}_\rho\left[\sum_{x\in E(\rho)}|p_0(x)-p_1(x)|\right],
\end{aligned}$$

where we used (9) to get the last line.

We bound the sum under the expectation in the last line above by looking at three sets whose union contains $E = E(\rho)$:

$$\begin{aligned} E_1(\rho) &= \{x \mid |\tilde{Q}(x;\rho) - Q(x)| \geq \alpha\}, \\ E_2 &= \{x \mid |Q(x) - q(x)| \geq 2\alpha, \\ E_3 &= \{x \mid |q(x) - 1/2| \leq 3\alpha\}. \end{aligned}$$

▷ Claim 33. For every ρ, $E(\rho) \subseteq E_1(\rho) \cup E_2 \cup E_3$.

Proof of Claim 33. If $\tilde{Q}(x;\rho) \geq 1/2$, and $x \notin (E_1 \cup E_2)$, then $q(x) > 1/2 - 3\alpha$. So either $q(x) \geq 1/2$, or $x \in E_3$. Similar reasoning applies if $\tilde{Q}(x;\rho) < 1/2$. So these three sets cover E. ◁

We bound the sum for E_1 just by using Chernoff bounds, the sum for E_2 by the statistical distinguishability of our distributional inverter A, and the sum for E_3 using the fact that having q close to $1/2$ means $p_0(x)$ and $p_1(x)$ are relatively close. For $E_1(\rho)$ and E_2, we will actually upperbound the summation of $p_0(x) + p_1(x)$, over x from the respective set.

▷ Claim 34. $\mathbf{Exp}_\rho\left[\sum_{x\in E_1(\rho)}(p_0(x) + p_1(x))\right] \leq \alpha$.

Proof of Claim 34. By linearity of expectation, it suffices to upperbound

$$\mathbf{Exp}_\rho\left[\sum_{x\in E_1(\rho)} p_0(x)\right] + \mathbf{Exp}_\rho\left[\sum_{x\in E_1(\rho)} p_1(x)\right].$$

The first expectation can be thought of as the probability that, if we sample x from D_0, and then perform the empirical estimate (using randomness ρ), that we are off by at least α. The second expectation is the same but for D_1. By the Chernoff bounds (our choice of K), each probability is at most $\alpha/2$. ◁

▷ Claim 35. $\sum_{x\in E_2}(p_0(x) + p_1(x)) \leq \alpha$.

Proof of Claim 35. We use the accuracy of the inverter A. The distinguishing probability between $(A(F(b,r)),\ F(b,r))$ and $((b,r),\ F(b,r))$ is at least that between any distributions computable from these. So in particular, the statistical distance between $(B(x),\ x)$ and $(b,\ x)$, for $x = F(b,r)$, is at most α^2. Using the fact that the statistical distance is the half of the ℓ_1-norm of the difference between the distributions, we get

$$\begin{aligned} \alpha^2 &\geq (1/2)\cdot\sum_x (1/2)\cdot(p_0(x)+p_1(x))\cdot(|q(x)-Q(x)| + |1-Q(x)-(1-q(x))|) \\ &= (1/2)\cdot\sum_x (p_0(x)+p_1(x))\cdot|q(x)-Q(x)|, \end{aligned}$$

Since for all x in E_2, $|q(x) - Q(x)| \geq 2\alpha$, and restricting to $x \in E_2$ only reduces the sum in the last line, we have

$$\alpha^2 \geq (1/2)\cdot\sum_{x\in E_2}(p_0(x)+p_1(x))(2\alpha),$$

or $\sum_{x\in E_2}(p_0(x) + p_1(x)) \leq \alpha$, as required. ◁

▷ Claim 36. $\sum_{x\in E_3} |p_0(x) - p_1(x)| \leq 12\alpha$.

Proof of Claim 36. If $x \in E_3$ then

$$\left|\frac{p_0(x)}{p_0(x)+p_1(x)} - \frac{1}{2}\right| \leq 3\alpha.$$

Multiplying through by $2(p_0(x) + p_1(x))$,

$$|p_0(x) - p_1(x)| \leq 6\alpha(p_0(x) + p_1(x)).$$

Thus,

$$\begin{aligned}\sum_{x\in E_3} |p_0(x) - p_1(x)| &\leq \sum_{x\in E_3} 6\alpha(p_0(x) + p_1(x))\\ &\leq 12\alpha,\end{aligned}$$

as required. ◁

Combining Claims 34–36, we get that the advantage of our probabilistic circuit C at distinguishing D_0 and D_1 is at least $\delta - 14\alpha$, as required. Given oracle access to A, our algorithm $\hat{\mathcal{D}}^A(C_0, C_1; 1^{1/\alpha})$ will construct such a circuit C in time polynomial in $1/\alpha$. ◀

We now prove Corollary 17. We repeat it here for convenience.

▶ **Corollary 37.** *There is an $\mathsf{FBPP}^{\mathsf{MCSP}}$ algorithm $\mathcal{D}$ satisfying the following. Given any pair of circuits C_0 and C_1 that are samplers for some distributions D_0 and D_1, and given a parameter K in unary, the algorithm $\mathcal{D}(C_0, C_1; 1^K)$ will distinguish D_0 and D_1 with advantage at least $\delta - 1/K$, where δ is the statistical distance between D_0 and D_1.*

Proof. Use Lemma 16 to get an $\mathsf{FBPP}^{\mathsf{MCSP}}$-computable universal distributional inverter that achieves statistical distance α^2 for $\alpha = 1/(14K)$. Define the algorithm $\mathcal{D}$ as follows. For given input circuits C_0 and C_1, run the oracle algorithm $\hat{\mathcal{D}}^A(C_0, C_1; 1^{1/\alpha})$ from Lemma 32, invoking the universal distributional inverter from Lemma 16 on every oracle query to A made by $\hat{\mathcal{D}}$. ◀

Next, we show that we can extend our universal distinguisher for distributions samplable by O-oracle circuits for languages O satisfying certain technical conditions.

▶ **Corollary 38.** *Let $O \in \mathsf{BPP}^{\mathsf{MCSP}} \cap \mathsf{P/poly}$ be any language. Then there is an $\mathsf{FBPP}^{\mathsf{MCSP}}$ algorithm $\mathcal{D}$ that, given any pair of O-oracle circuits C_0 and C_1 that are samplers for some distributions D_0 and D_1, and given a parameter K in unary, the algorithm $\mathcal{D}(C_0, C_1; 1^K)$ will distinguish D_0 and D_1 with advantage at least $\delta - 1/K$, where δ is the statistical distance between D_0 and D_1.*

Proof. Use Lemma 16 to get an $\mathsf{FBPP}^{\mathsf{MCSP}}$-computable universal distributional inverter that achieves statistical distance α^2 for $\alpha = 1/(14K)$. As in the proof of Lemma 32, we use MCSP-oracle circuit samplers for distributions D_0 and D_1 to get a circuit for distributional one-way function candidate F. We then use our universal inverter to get a distributional inverter A needed in Lemma 32 for any given input circuits C_0 and C_1. Observe that we can invert F since F is computable by a small O-oracle circuit (given the O-oracle circuits for sampling D_0 and D_1), and hence F is also computable by a circuit of polynomial size with *no* oracle gates (since by assumption $O \in \mathsf{P/poly}$). The correctness proof of the inverting algorithm relies on the fact that a small circuit for F *exists*. Yet, the inverting algorithm for

F does not need to know a small circuit for F; it just must be able to evaluate F efficiently, given a small description of F. Using the encoding of an O-oracle circuit for F works since the inverting algorithm can evaluate the circuit with probability close to 1, given access to the MCSP oracle (since $O \in \mathsf{BPP}^{\mathsf{MCSP}}$). ◀

B A Universal Distinguisher in $\mathsf{FBPP}^{\mathsf{NP}}$

▶ **Lemma 39** ([30]). *There exists a randomized algorithm that given oracle access to* NP *can approximate any function $f(x)$ in* #P *to within the multiplicative factor $(1 \pm \varepsilon)$, with probability at least $1-\gamma$, in time polynomial in $|x|$, $1/\varepsilon$, and $\log(1/\gamma)$.*

▶ **Theorem 40.** *There is an $\mathsf{FBPP}^{\mathsf{NP}}$ algorithm $\mathcal{D}$, that given circuits C_0 and C_1 that are samplers for distributions D_0 and D_1, and $K \in \mathbb{N}$ in unary, will distinguish D_0 and D_1 with the distinguishing advantage at least $\delta - 1/K$, where δ is the statistical distance between D_0 and D_1.*

Proof. Given C_0 and C_1, let $p_0(x)$ be the probability of x according to D_0, and $p_1(x)$ that according to D_1. For $0 < \gamma = \varepsilon \leq 1/2$ to be determined, consider the following probabilistic circuit $A(x;r)$: Compute the estimates $\tilde{p}_0(x) = (1 \pm \varepsilon)p_0(x)$ and $\tilde{p}_1(x) = (1 \pm \varepsilon)p_1(x)$ with probability at least $1-\gamma$ (using the algorithm from Lemma 39), and accept iff $\tilde{p}_0(x) > \tilde{p}_1(x)$.

We say that randomness r is good for x if both estimates $\tilde{p}_0(x)$ and $\tilde{p}_1(x)$ are correct within the multiplicative factor $(1 \pm \varepsilon)$. Note that by Lemma 39, for every x, r is good for x with probability at least $1 - 2\gamma$. We have

$$\begin{aligned}
&\mathbf{Pr}_{x\sim D_0,r}[A(x;r)=1] - \mathbf{Pr}_{x\sim D_1,r}[A(x;r)=1] \\
&\geq \mathbf{Pr}_{x\sim D_0,r}[A(x;r)=1 \mid \ r \text{ is good for } x] - \mathbf{Pr}_{x\sim D_1,r}[A(x;r)=1 \mid \ r \text{ is good for } x] - 2\gamma \\
&= \sum_{x:\ p_0(x)>p_1(x)} (p_0(x)-p_1(x)) - 2\gamma \\
&\quad - \sum_{x:\ p_0(x)>p_1(x)\ \wedge\ \tilde{p}_0(x)<\tilde{p}_1(x)} (p_0(x)-p_1(x)) \\
&\quad + \sum_{x:\ p_0(x)\leq p_1(x)\ \wedge\ \tilde{p}_0(x)>\tilde{p}_1(x)} (p_0(x)-p_1(x)).
\end{aligned}$$

Note that

$$\begin{aligned}
&\sum_{x:\ p_0(x)>p_1(x)\ \wedge\ \tilde{p}_0(x)<\tilde{p}_1(x)} (p_0(x)-p_1(x)) \\
&\leq \sum_{x:\ (p_0(x)>p_1(x))\ \wedge\ ((1-\varepsilon)p_0(x)<(1+\varepsilon)p_1(x))} (p_0(x)-p_1(x)) \\
&\leq \sum_x ((1+\varepsilon)/(1-\varepsilon)-1)\cdot p_1(x) \\
&= (2\varepsilon)/(1-\varepsilon).
\end{aligned}$$

Similarly,

$$\sum_{x:\ p_0(x)\leq p_1(x)\ \wedge\ \tilde{p}_0(x)>\tilde{p}_1(x)} (p_1(x)-p_0(x)) \leq (2\varepsilon)/(1-\varepsilon).$$

Putting everything together, we get

$$\mathbf{Pr}_{x\sim D_0,r}[A(x;r)=1] - \mathbf{Pr}_{x\sim D_1,r}[A(x;r)=1] \geq \delta - (2\gamma + (4\varepsilon)/(1-\varepsilon)),$$

which is at least $\delta - 10\varepsilon$. Setting $\varepsilon = 1/(10K)$ concludes the proof. ◀

C Separating ZPEXP from ZPP

▷ Claim 41. $\mathsf{ZPEXP} \neq \mathsf{ZPP}$.

Proof. Suppose for a contradiction that $\mathsf{ZPEXP} = \mathsf{ZPP}$. Then

$$\mathsf{NP}^{\mathsf{NP}} \subseteq \mathsf{EXP} \subseteq \mathsf{ZPEXP} \subseteq \mathsf{ZPP}.$$

By translation, $\mathsf{NEXP}^{\mathsf{NP}} \subseteq \mathsf{ZPEXP}$ and hence by Lemma 12, $\mathsf{ZPEXP} \not\subseteq \mathsf{P/poly}$. Yet, by Adleman's Theorem ([1]) $\mathsf{ZPP} \subseteq \mathsf{BPP} \subseteq \mathsf{P/poly}$. ◁

▶ Remark 42. Similarly, one can show that $\mathsf{BPEXP} \neq \mathsf{BPP}$. However, separating ZPEXP or even NEXP or $\mathsf{EXP}^{\mathsf{NP}}$ from BPP remains a longstanding open question. See e.g. [15, 49].

Interactive Error Correcting Codes: New Constructions and Impossibility Bounds

Meghal Gupta
University of California Berkeley, CA, USA

Rachel Yun Zhang
Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

An *interactive error correcting code (*iECC*)* is an interactive protocol with the guarantee that the receiver can correctly determine the sender's message, even in the presence of noise. It was shown in works by Gupta, Kalai, and Zhang (STOC 2022) and by Efremenko, Kol, Saxena, and Zhang (FOCS 2022) that there exist iECC's that are resilient to a larger fraction of errors than is possible in standard error-correcting codes without interaction. In this work, we improve upon these existing works in two ways:

- First, we improve upon the erasure iECC of Kalai, Gupta, and Zhang, which has communication complexity quadratic in the message size. In our work, we construct the first iECC resilient to $> \frac{1}{2}$ adversarial erasures that is also positive rate. For any $\epsilon > 0$, our iECC is resilient to $\frac{6}{11} - \epsilon$ adversarial erasures and has size $O_\epsilon(k)$.
- Second, we prove a better upper bound on the maximal possible error resilience of any iECC in the case of bit flip errors. It is known that an iECC can achieve $\frac{1}{4} + 10^{-5}$ error resilience (Efremenko, Kol, Saxena, and Zhang), while the best known upper bound was $\frac{2}{7} \approx 0.2857$ (Gupta, Kalai, and Zhang). We improve upon the upper bound, showing that no iECC can be resilient to more than $\frac{13}{47} \approx 0.2766$ fraction of errors.

2012 ACM Subject Classification Mathematics of computing → Information theory

Keywords and phrases Code, Interactive Protocol, Error Resilience

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.32

Category RANDOM

Related Version This conference publication is the merge of two works.
Previous Version: https://arxiv.org/abs/2201.11929 [10]
Previous Version: https://arxiv.org/abs/2305.04376 [11]

Funding *Meghal Gupta*: U.C. Berkeley Chancellor's Fellowship
Rachel Yun Zhang: This research was supported in part by DARPA under Agreement No. HR00112020023, an NSF grant CNS-2154149, an Akamai Presidential Fellowship, and NSF Graduate Research Fellowship 2141064.

1 Introduction

Consider the following task: Alice wishes to communicate a message to Bob such that even if a constant fraction of the communicated bits are adversarially tampered with, Bob is still guaranteed to be able to determine her message. This task motivated the prolific study of *error correcting codes*, starting with the seminal works of [16, 15]. An error correcting code encodes a message x into a longer codeword $\mathsf{ECC}(x)$, such that the Hamming distance between any two distinct codewords is a constant fraction of the length of the codewords.

An important question in the study of error correcting codes is determining the maximal possible error resilience. It is known that in the adversarial bit-flip model, any ECC can be resilient to at most $\frac{1}{4}$ corruptions, and in the adversarial erasure error model any ECC can be resilient to at most $\frac{1}{2}$ corruptions.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 32; pp. 32:1–32:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

This prompts the following natural question: *Can we achieve better error resilience if we use interaction?*

In [9], Gupta, Kalai and Zhang introduce the notion of an *interactive error correcting code (*iECC*)*, which is an interactive protocol with a fixed length and speaking order, such that Bob can correctly learn Alice's input x as long as not too large a fraction of the *total communication* is erased. They demonstrate that iECC's can in fact achieve a higher erasure resilience than standard error correcting codes. In particular, they design an iECC that is resilient to adversarial erasure of $\frac{3}{5} - \epsilon$ of the total communication.

Note that a classical error correcting code is an iECC in which Alice speaks in every round. Their result essentially shows that Bob talking occasionally *instead of Alice* actually improves the error resilience. It is not obvious that this should be the case – since Bob can only send feedback, while Alice can actually send new information, Bob's messages a priori seem a lot less valuable than Alice's. Nevertheless, they are able to leverage this to improve the erasure resilience past $\frac{1}{2}$. Later, [7] present an iECC that achieves an error (bit flip) resilience greater than $\frac{1}{4}$.

In this paper, we present two new results about iECC's. We mention that this conference version of the paper is a combination of two separate works on the arXiv: [10] and [11]. We also note the relevant context that the result of [7], which constructed a binary iECC with $> \frac{1}{4}$ error resilience, was published after the positive rate result in this paper, but before the impossbility bound in this paper.

1.1 Positive Rate iECC

One weakness of the erasure-resilient iECC presented by [9] is that the size of their protocol is quadratic in the length of Alice's original message x. This leaves open the question of whether there exists an iECC achieving $> \frac{1}{2}$ erasure resilience with size *linear* in the length of the original message. In this paper, we answer this question in the affirmative.

Specifically, we show a positive rate iECC that achieves an erasure resilience of $\frac{6}{11} - \epsilon$ over the binary erasure channel, which is larger than $\frac{1}{2}$.

▶ **Theorem 1.** *For any $\epsilon > 0$, there exists an* iECC *over the binary erasure channel resilient to $\frac{6}{11} - \epsilon$ erasures, such that the communication complexity for inputs of size n is $O_\epsilon(n)$ and the time complexity is $\text{poly}_\epsilon(n)$.*

We remark that our iECC achieves a lower erasure resilience than the quadratic sized iECC of [9], which is resilient to $\frac{3}{5} - \epsilon$ erasures. However, we believe that an iECC achieving both positive rate and $\frac{3}{5} - \epsilon$ erasure resilience can likely be constructed by combining ideas from this paper and [9]. Nevertheless, we leave open the existence of such an iECC.

▶ Remark 2. Since the original paper [10] was posted to arXiv, the work of [7] constructed a positive rate iECC resilient to $\frac{1}{4} + 10^{-5}$ bit flip errors, thereby resolving whether iECC's can achieve better error resilience in the case of bit flip errors as well! Since bit flip errors are stronger than erasures, their iECC is also a positive rate iECC resilient to $> \frac{1}{2}$ erasures. Comparing this work with theirs, theirs works in the case of bit flips errors while ours does not, but our work achieves the higher erasure resilience.

1.2 Upper Bound on Maximal Error Resilience of iECC

The current best known upper (impossibility) bound for the error resilience of an iECC (in the bit flip setting, rather than erasure setting) was given in [9], who showed that no iECC can be resilient to more than $\frac{2}{7}$ adversarial errors. This upper bound came from the combination of two natural attacks, one of which is guaranteed to work no matter how the rounds in which Alice and Bob speak are distributed.

1. Corrupt *none* of Bob's bits. Then, Bob's messages provide perfect reliable feedback, in which case works about *error-correcting codes with feedback* beginning with [2] tell us that it suffices to corrupt $\frac{1}{3}$ of Alice's bits.
2. Corrupt *half* of Bob's bits so that his messages appear random and thus essentially are useless: then Alice's communication essentially reduces to the case of a standard error-correcting code, in which case an adversary can corrupt $\frac{1}{4}$ of her bits to confuse Bob between two possible values of x.

The largest error resilience known so far is achieved by [7], who construct an iECC resilient to $\frac{1}{4} + 10^{-5}$ bit flip errors. This leaves a large gap between the best known achievable error resilience and the best known upper bound. *What is the largest possible error resilience of an iECC? Is it possible to achieve error resilience equal to this natural upper bound of $\frac{2}{7}$?*

In this work, we answer the latter question in the negative, providing a new upper bound of $\frac{13}{47} \approx 0.2766$, improving upon the previous best upper bound of $\frac{2}{7} \approx 0.2857$.

▶ **Theorem 3.** *For sufficiently small $\epsilon > 0$, there exists $k_0 = k_0(\epsilon) \in \mathbb{N}$ such that for any $k > k_0$, no iECC over the binary bit flip channel where Alice is trying to communicate $x \in \{0,1\}^k$ is resilient to $\frac{13}{47} + \epsilon$ fraction of adversarial bit flips.*

2 Related Works

In this section, we discuss previous work on interactive error-correcting codes, as well as prior work on error-correcting codes with feedback.

2.1 Interactive Error-Correcting Codes

The notion of an *interactive error-correcting code* (iECC) was first introduced in [9], who demonstrated an iECC resilient to $\frac{3}{5}$ fraction of adversarial erasures, surpassing the best possible erasure resilience of standard ECC's of $\frac{1}{2}$. They also gave an upper bound of $\frac{2}{3}$ on the erasure resilience of any iECC. In the case of bit flip errors, they proved an upper bound of $\frac{2}{7}$ on the error resilience achievable by any iECC, leaving open the problem of constructing an iECC resilient to greater than $\frac{1}{4}$ adversarial errors.

The followup work of [10] improved upon the erasure iECC of [9], giving a construction of an iECC with *positive rate* but resilient to only $\frac{6}{11}$ adversarial erasures.

In the bit flip error model, [7] answered [9]'s question in the affirmative, constructing an iECC with error resilience $\frac{1}{4} + 10^{-5}$. This narrowed the optimal error resilience of any iECC to the range $[\frac{1}{4} + 10^{-5}, \frac{2}{7}]$.

2.2 Error-Correcting Codes with Feedback

The use of interaction in the noise resilient communication of a message has been studied previously in the form of *error-correcting codes with feedback*. In an error-correcting code with feedback, Alice wishes to communicate a message to Bob in an error-resilient fashion, provided that after every message she sends she receives some feedback from Bob about what he has received. She can then use this noiseless feedback to choose the next bit that she sends. Error-correcting codes with feedback were first introduced in the Ph.D. thesis of Berlekamp [2] and have been studied in a number of followup works, including [3, 19, 17, 14, 8, 1]. Originally, this feedback was considered in the *noiseless* setting, meaning that none of Bob's messages are allowed to be corrupted, and error rate is calculated solely as a function of the number of messages Alice sends. That is, Bob's feedback is free and always correct, so that Alice can tailor her next message to specifically the bit of information Bob most needs to hear.

In the bit flip error model, [3, 19, 17, 14] showed that the maximal error resilience of an error-correcting code with noiseless feedback is $\frac{1}{3}$. [8] show this is achievable even by protocols that only send logarithmically many bits of feedback over a constant number of rounds. For explicit constant number of rounds of feedback, [4] initiated the study of the noise resilience vs. round complexity tradeoff for both erasures and errors. For larger alphabets, the maximal error resilience was studied in [1].

When the feedback is *noisy*, i.e. the feedback may be corrupted as well, much less is known. Several works such as [5, 6] considered ECC's with noisy feedback over the binary symmetric channel. [18] considers adversarial corruption, under a model which places separate corruption budgets on the forward and feedback rounds. They construct a scheme that is resilient to $\frac{1}{2}$ of the forward communication and 1 of the feedback being erased. We note that their scheme's forward erasure resilience is equal to that achievable by standard error-correcting codes.

3 Preliminaries and Definitions

Before we dive into the technical part of our paper, we present important preliminaries on classical error correcting codes, and define an iECC formally and what it means for one to be resilient to α-fraction of erasures.

Notation

In this work, we use the following notations.

- The function $\Delta(x, y)$ represents the Hamming distance between x and y.
- The interval $[a, b]$ for $a, b \in \mathbb{Z}_{\geq 0}$ denotes the integers from a to b inclusive. The interval $[n]$ denotes the integers $1, \ldots, n$.
- The symbol $\perp$ in a message represents the erasure symbol that a party might receive in the erasure model.
- When we say Bob k-decodes a message, we mean that he list decodes it to exactly k possible messages Alice could have sent in the valid message space.
- The output of an ECC is 0-indexed. All other strings are 1-indexed.

3.1 Interactive Error Correcting Codes

We formally define our notion of an *interactive error correcting code (*iECC*)*. The two types of corruptions we will be interested in are erasures and bit flips. We first start by defining a non-adaptive interactive protocol.

▶ **Definition 4** (Non-Adaptive Interactive Protocol). *A non-adaptive interactive protocol $\pi = \{\pi_n\}_{n\in\mathbb{N}}$ is an interactive protocol between Alice and Bob, where in each round a single party sends a single bit to the other party. The order of speaking, as well as the number of rounds in the protocol, is fixed beforehand. The number of rounds is denoted $|\pi|$.*

▶ **Definition 5** (Interactive Error Correcting Code). *An interactive error correcting code (*iECC*) is a non-adaptive interactive protocol $\pi = \{\pi_n\}_{n\in\mathbb{N}}$, with the following syntax:*

- *At the beginning of the protocol, Alice receives as private input some $x \in \{0, 1\}^n$.*
- *At the end of the protocol, Bob outputs some $\hat{x} \in \{0, 1\}^n$.*

We say that π is α-resilient to adversarial bit flips (resp. erasures) if there exists $n_0 \in \mathbb{N}$ such that for all $n > n_0$ and $x \in \{0, 1\}^n$, and for all online adversarial attacks consisting of flipping (resp. erasing) at most $\alpha \cdot |\pi|$ of the total communication, Bob outputs x at the end of the protocol with probability 1.

3.2 Classical Error Correcting Codes

▶ **Definition 6** (Error Correcting Code). *An error correcting code (*ECC*) is a family of maps* $\mathsf{ECC} = \{\mathsf{ECC}_n : \{0,1\}^n \to \{0,1\}^{m(n)}\}_{n\in\mathbb{N}}$. *An* ECC *has* relative distance $\alpha > 0$ *if for all* $n \in \mathbb{N}$ *and any* $x \neq y \in \{0,1\}^n$,

$$\Delta\left(\mathsf{ECC}_n(x), \mathsf{ECC}_n(y)\right) \geq \alpha m(n).$$

Binary error correcting codes with relative distance $\approx \frac{1}{2}$ are well known to exist with linear blowup in communication complexity.

▶ **Theorem 7** ([13]). *For all* $\epsilon > 0$*, there exists an explicit linear error correcting code* $\mathsf{ECC}_\epsilon = \{\mathsf{ECC}_{\epsilon,n} : \{0,1\}^n \to \{0,1\}^m\}_{n\in\mathbb{N}}$ *with relative distance* $\frac{1}{2} - \epsilon$ *and with* $m = m(n) = O_\epsilon(n)$*. Furthermore, all codewords other than* $\mathsf{ECC}_{\epsilon,n}(0^n)$ *are relative distance* $\frac{1}{2} - \epsilon$ *from* 0^m *and* 1^m *as well.*

A relative distance of $\frac{1}{2}$ is in fact optimal in the sense that as the number of codewords N approaches ∞, the maximal possible relative distance between N codewords approaches $\frac{1}{2}$. We remark, however, that for small values of N, the distance can be much larger: for $N = 2$, the relative distance between codewords can be as large as 1, e.g. the codewords 0^M and 1^M, and for $N = 4$, the relative distance can be as large as $\frac{2}{3}$, e.g. the codewords $(000)^M, (110)^M, (101)^M, (011)^M$. Our constructions leverage this fact that codes with higher relative distance exist for a small constant number of codewords.

We will also need the following important lemma about the number of shared bits between any three codewords in an error correcting code scheme that has distance $\frac{1}{2}$.

▶ **Lemma 8.** *For any error correcting code* $\mathsf{ECC}_\epsilon = \{\mathsf{ECC}_{\epsilon,n} : \{0,1\}^n \to \{0,1\}^m\}_{n\in\mathbb{N}}$ *with relative distance* $\frac{1}{2} - \epsilon$*, and any large enough* $n \in \mathbb{N}$*, any three codewords in* $\mathsf{ECC}_{\epsilon,n}$ *overlap on at most* $\left(\frac{1}{4} + \frac{3}{2}\epsilon\right) \cdot m$ *locations.*

Lemma 8 means that assuming that $< \frac{3}{4}$ of a codeword is erased, the resulting message is list-decodable to a set of size ≤ 2, at least in theory. The following theorem says such a code exists with list-decoding being polynomial time, while also satisfying a couple other properties necessary in the protocol construction in Section 4.2.

▶ **Theorem 9** ([12]). *For all* $\epsilon > 0$*, any explicit (given with its encoding matrix) linear code* $\mathsf{ECC}_\epsilon = \{\mathsf{ECC}_{\epsilon,n} : \{0,1\}^n \to \{0,1\}^m\}_{n\in\mathbb{N}}$ *with relative distance* $(\frac{1}{2} - \epsilon)$*, can be efficiently decoded and list-decoded. That is, there exists a* $\mathrm{poly}_\epsilon(n)$*-time decoding algorithm* $\mathsf{DEC}_\epsilon = \{\mathsf{DEC}_{\epsilon,n} : \{0,1\}^m \to \mathcal{P}(\{0,1\}^n)\}_{n\in\mathbb{N}}$*, such that for any* $n \in \mathbb{N}$*,* $x \in \{0,1\}^n$*, and corruption* σ *consisting of fewer than* $(\frac{1}{2} - \epsilon) \cdot m$ *erasures,*

$$x = \mathsf{DEC}_{\epsilon,n}(\sigma \circ \mathsf{ECC}_{\epsilon,n}(x)).$$

Moreover, for any corruption σ *consisting of fewer than* $(\frac{3}{4} - \frac{3}{2}\epsilon) \cdot m$ *erasures,*

$$|\mathsf{DEC}_{\epsilon,n}(\sigma \circ \mathsf{ECC}_{\epsilon,n}(x))| \leq 2, \qquad x \in \mathsf{DEC}_{\epsilon,n}(\sigma \circ \mathsf{ECC}_{\epsilon,n}(x)).$$

Our following theorem gives an ECC such that any two codewords differ on most segments of length α.

▶ **Theorem 10.** *For all $n \in \mathbb{N}, \epsilon > 0$, there exists $\alpha = \Theta_\epsilon(\log n)$ such that, there exists an explicit linear code $\mathsf{ECC}_\epsilon = \{\mathsf{ECC}_{\epsilon,n} : \{0,1\}^n \to \{0,1\}^m\}_{n\in\mathbb{N}}$ with $m = m(n) = O_\epsilon(n)$, satisfying the following property: For all $n \in \mathbb{N}, \epsilon > 0$, it holds that $\alpha | m$ and for any $x \neq x' \in \{0,1\}^n$ and $j \in \{0 \dots \frac{m}{\alpha} - 1\}$,*

$$\mathsf{ECC}_{\epsilon,n}(x)[j\alpha : (j+1)\alpha - 1] = \mathsf{ECC}_{\epsilon,n}(x')[j\alpha : (j+1)\alpha - 1]$$

for at most $\frac{\epsilon m}{\alpha}$ values of j.

▶ **Lemma 11.** *Let $\mathsf{ECC}'_\epsilon = \{\mathsf{ECC}'_{n,\epsilon} : \{0,1\}^n \to \{0,1\}^{m(n)}\}$ be a explicit linear code satisfying the properties of Theorem 10 with $\alpha = \alpha_n = \Theta_\epsilon(\log n)$. For all linear $\mathsf{ECC}_\epsilon = \{\mathsf{ECC}_{n,\epsilon} : \{0,1\}^\alpha \times \{0,1\}^\beta \to \{0,1\}^{p(n)}\}$ with relative distance $\frac{1}{2} - \epsilon$ and for all β, the code defined by*

$$C(x) = \mathsf{ECC}_\epsilon(\mathsf{ECC}'_\epsilon(x)[0, \alpha - 1], 0^\beta) || \dots || \mathsf{ECC}_\epsilon(\mathsf{ECC}'_\epsilon(x)[m - \alpha, m - 1], 0^\beta)$$

is a linear code with relative distance $\frac{1}{2} - \frac{3}{2}\epsilon$. In particular, assuming that less than $\frac{3}{4} - \frac{9}{4}\epsilon$ of $C(x)$ is erased, there is an efficient algorithm to obtain a set of size 2 containing x.

4 A New Positive Rate iECC Resilient to $6/11$ Erasures

In this section, we discuss our positive rate iECC that is resilient to $6/11$ adversarial erasures.

4.1 Overview of Ideas

We begin with an overview of the ideas that go into our positive rate iECC. We first briefly review the iECC of [9] then describe how to modify it to have a linear communication complexity.

The overarching goal of the original protocol, as well as ours, is to perform the following three steps.

1. Bob learns that Alice's value of x is one of two possible values. (This idea is known as list decoding, which achieves better noise resilience than unique decoding.)
2. Bob conveys to Alice an index i on which the two possible inputs differ.
3. Alice sends the value of her input at index i.

Summary of the Protocol of [9]

The original protocol consists of many (say $\approx \frac{n}{\epsilon}$) *chunks*, where in each chunk Alice sends a message followed by Bob's reply. The protocol is designed so that each such chunk will make *progress* towards Bob's unambiguously learning Alice's input, as long as the adversary did not invest more than $\frac{3}{5} - \epsilon$ erasures in that chunk. At a high level, in the first chunk with $< \frac{3}{5} - \epsilon$ erasures, Bob narrows down Alice's input to at most two options. In every future chunk with $< \frac{3}{5} - \epsilon$ erasures, either Alice gets closer to learning the index i on which the two options differ, or Bob fully determines x by ruling out one of the two values of x, e.g. by learning the value of $x[i]$ or by uniquely decoding Alice's message. Alice keeps track of a counter cnt initially set to 0 indicating her guess for i. The main purpose of Bob's messages is to increment Alice's counter to i.

At the beginning of the protocol, Alice sends $\mathsf{ECC}(x, \mathsf{cnt})$ to Bob in every chunk. At the first point, there are $< \frac{3}{5} - \epsilon$ erasures in a chunk, Bob will be able to list decode Alice's message to at most two options, say $(x_0, \mathsf{cnt}_0 = 0)$ and $(x_1, \mathsf{cnt}_1 = 0)$. This must happen because the relative message lengths of Alice and Bob will be such that the adversary cannot corrupt too much of Alice's message even if they corrupt none of Bob's message. Since we are in the setting of erasures, one of the two decodings must be Alice's true state, and in particular, must contain Alice's true input.

At this point, Bob begins signaling to Alice to increment cnt. His goal is to tell Alice to increment cnt until $\mathsf{cnt} = i$. He does this by only sending one of two codewords[1] every message that have relative distance 1 apart. This way, if Bob's message is not entirely erased, Alice learns what Bob tried to send. The key is that every time $< \frac{3}{5} - \epsilon$ of a chunk is corrupted, we can guarantee *both that Bob will decode Alice's message to two possible messages, and Alice uniquely decodes Bob's message*,[2] so that Alice and Bob make progress towards Alice learning i. Once Alice has discovered i, Bob signals for Alice to send the bit $x[i]$ for the rest of the protocol,[3] which allows him to distinguish whether Alice has x_0 or x_1.

Modifications to Achieve Positive Rate

The communication complexity of the above protocol is $O(n^2)$. This comes from two parts: (1) $O(n)$ chunks are necessary for Bob to communicate the index $i \in [n]$ to Alice via incrementation, and (2) Alice sends her length n input in every chunk. We show how to lessen both requirements, thus making the final protocol linear in length.

First, for Bob to communicate $i \in [n]$ to Alice, instead of incrementing cnt by 1 until it equals i. More specifically, we describe a process where Bob writes i out in binary, and then sends Alice each bit of this binary representation in sequence. This only requires $O(\log n)$ rounds of interaction, as opposed to the $O(n)$ rounds required by [9]. Designing a protocol to communicate a binary string rather than a unary string requires significant changes to the procedure used in [9].

Second, we show that instead of sending x every message, it suffices for Alice to encode a shorter string that is different than the corresponding short string for any other x' in *most* chunks. More precisely, consider an error correcting code ECC with the following property: set some α and for any $x \neq x'$,

$$\mathsf{ECC}(x)[j\alpha, (j+1)\alpha - 1] \neq \mathsf{ECC}(x')[j\alpha, (j+1)\alpha - 1]$$

for all but an ϵ fraction of values j. Then, Alice rotates through the sections, sending $\mathsf{ECC}(x)[j\alpha, (j+1)\alpha - 1]$ in the $\left(j \bmod \frac{|\mathsf{ECC}(x)|}{\alpha}\right)$'th chunk. Then, if Bob has narrowed down Alice's input to x_0 and x_1, he can simply ignore the ϵ fraction of chunks in which $\mathsf{ECC}(x_0)[j\alpha, (j+1)\alpha - 1] = \mathsf{ECC}(x_1)[j\alpha, (j+1)\alpha - 1]$. In the remainder of chunks, the segment $\mathsf{ECC}(x)[j\alpha, (j+1)\alpha - 1]$ is sufficient for Bob to distinguish between x_0 and x_1. If we were to let $\alpha \approx \log n$,[4] then our chunks are now only length $O(\log n)$.

Combining the two modifications, we see that $\Theta(n)$ communication from Alice is necessary for Bob to narrow down Alice's input to two options, and then after that, Bob can convey i to Alice in $O(\log n)$ chunks each of size $O(\log n)$. This results in an iECC with total communication $O(n + \log^2 n) = O(n)$.

[1] In our protocol, Bob will send one of four codewords each message. This contributes to the lower erasure resilience of $\frac{6}{11} - \epsilon$.

[2] It is also possible that instead Bob uniquely decodes Alice's message, but then he will have uniquely learned x.

[3] The reader familiar with the iECC of [9] may recall that in the case that the two Alices Bob sees have different values of cnt, Bob may instruct Alice to send a different bit for the rest of the protocol, but we do not address this for now.

[4] $\alpha = \Theta(\log n)$ is necessary since Alice also sends her current guess of i each message, which has length $\log n$.

We remark that our protocol has erasure resilience $\frac{6}{11} - \epsilon$. The limiting factor is in the construction of a protocol in which Bob builds i bit by bit: our protocol requires Bob sending 4 codewords with distance $\frac{2}{3}$, rather than 2 codewords with distance $\frac{1}{2}$ to achieve $\frac{3}{5} - \epsilon$ erasure resilience. However, combining our second observation with the protocol from [9] would be enough to give an iECC with $\frac{3}{5} - \epsilon$ erasure resilience with communication $O(n \log n)$.

4.2 Positive Rate iECC Protocol (Informal)

Let $\mathsf{ECC}' : \{0,1\}^n \to \{0,1\}^m$ be an error correcting code satisfying the statement of Theorem 10 with $\alpha = \Theta(\log n)$, and let $\mathsf{ECC} : \{0,1\}^\alpha \times \{0,1\}^{\le \log n} \to \{0,1\}^p$ be an error correcting code with distance $\frac{1}{2}$ that is also relative distance $\frac{1}{2}$ from $0^p, 1^p$.

Our iECC consists of $O_\epsilon\left(\frac{m}{\alpha}\right)$ chunks, each consisting of Alice sending a p-bit message followed by Bob sending a $\frac{3p}{8}$-bit message. Bob's messages are always one of four words $\bar{0}, \bar{1}, \bar{2}, \bar{3} \in \{0,1\}^{3p/8}$ with relative distance $\frac{2}{3}$. We outline our protocol below. In what follows, we assume that all messages Bob receives are consistent with the same two values of x, otherwise Bob can rule out one of the values of x and determine Alice's true input.

1. Alice initially holds a string $\mathsf{ind} \in \{0,1\}^{\le \log n}$ initially set to the empty string $\mathsf{ind} = \emptyset$. Alice begins the protocol by sending $\mathsf{ECC}(\mathsf{ECC}'(x)[j\alpha, (j+1)\alpha - 1], \mathsf{ind})$ to Bob in every chunk.
2. Bob begins the protocol sending $\bar{0}$ every message. Every $\frac{m}{\alpha}$ chunks, he attempts to list-decode Alice's previous $\frac{m}{\alpha}$ messages to find consistent values of x. Note that by Lemma 11, if there are at most $\frac{3}{4} - \frac{3}{2}\epsilon$ erasures in Alice's message in those $\frac{m}{\alpha}$ chunks, then Bob is guaranteed to find at most two possible values of x.
3. When Bob has found two consistent values of x, say $\hat{x}_0$ and $\hat{x}_1$, he determines an index $i \in [n] = \{0,1\}^{\log n}$ such that $\hat{x}_0[i] \ne \hat{x}_1[i]$. His goal is now to communicate i to Alice, bit by bit. He does this by sending either $\bar{0}$, $\bar{1}$, or $\bar{2}$ every chunk.
   To communicate the next'th bit of i, Bob adds $i[\mathsf{next}] + 1$ to mes modulo 3, where $\overline{\mathsf{mes}}$ was the last message he sent, to get his new message mes', and begins sending $\overline{\mathsf{mes}'}$ every chunk. (When Alice receives a message from Bob that is different from the last message she received, she can calculate the difference in the two messages to determine the bit.) He does this until he list-decodes Alice's message to two possibilities $\mathsf{ECC}(\mathsf{ECC}'(\hat{x}_0)[j\alpha, (j+1)\alpha - 1, \mathsf{ind}_0)$ and $\mathsf{ECC}(\mathsf{ECC}'(\hat{x}_1)[j\alpha, (j+1)\alpha - 1], \mathsf{ind}_1)$ such that $\mathsf{ECC}'(\hat{x}_0)[j\alpha, (j+1)\alpha - 1] \ne \mathsf{ECC}'(\hat{x}_1)[j\alpha, (j+1)\alpha - 1]$, where at least one of ind_0, ind_1 has length next. If both have length next, he proceeds to communicate the $(\mathsf{next} + 1)$'th bit of i in the same way. If only one of the two Alice's has $|\mathsf{ind}_b| = \mathsf{next}$, Bob switches to sending $\bar{3}$ for the rest of the protocol, signaling to Alice to send him the parity of $|\mathsf{ind}|$ so that he can distinguish between whether Alice has $(\hat{x}_0, \mathsf{ind}_0)$ or $(\hat{x}_1, \mathsf{ind}_1)$.
4. Whenever Alice unambiguously sees a *change* in Bob's message from a $\bar{0}$ to a $\bar{1}$ or $\bar{2}$ (or cyclic), she calculates $b = \mathsf{mes}' - \mathsf{mes} - 1 \mod 3$ and appends b to ind. If she ever receives a $\bar{3}$, she switches to sending $(|\mathsf{ind}| \mod 2)^p$ for the rest of the protocol. Otherwise, at some point she has $|\mathsf{ind}| = \log n$, so she can convert ind into an index $i \in [n]$ and send $(x[i])^p$ for the rest of the protocol. Note that Bob can distinguish between $\hat{x}_0$ and $\hat{x}_1$ using the value of x at the index i.

In the above outline, one has to be careful around $|\mathsf{ind}| = \log n - 1$. In particular, if one Alice has $|\mathsf{ind}| = \log n - 1$ and the other has $|\mathsf{ind}| = \log n$, the second will be sending $x[i]$ for the rest of the protocol and it is thus incorrect for Bob to send $\bar{3}$ to signal the first Alice to send the parity of the length of ind. Instead, once Bob has list-decoded Alice's message such that $|\mathsf{ind}_0| = |\mathsf{ind}_1| = \log n - 1$, Bob commits to sending the next message $\in \{\bar{0}, \bar{1}, \bar{2}\}$ that conveys to Alice the final bit of i for the rest of the protocol.

▶ **Theorem 12.** *The above* iECC *is resilient to a $\frac{6}{11} - O(\epsilon)$ fraction of erasures. For an input of size n, the total communication is $O_\epsilon(n)$. Alice and Bob run in $poly_\epsilon(n)$ time.*

The formal version of this protocol is excluded from this conference version of the paper. The full protocol can be found in [10].

5 Impossibility Bound on Maximal Noise Resilience of iECC

In this section, we present our main upper bound, that for any non-adaptive iECC, there is some attack consisting of at most $\frac{13}{47}$ corruptions such that Bob cannot guess Alice's input x correctly with probability better than $\frac{1}{2}$.

▶ **Theorem 13.** *main For sufficiently small $\epsilon > 0$, then for all $k > 100\epsilon^{-4}$, no* iECC *for $x \in \{0,1\}^k$ is resilient to more than $\frac{13}{47} + 2\epsilon$ fraction of errors with probability greater than $\frac{1}{2}$.*

The rest of this section will be devoted to the proof of this theorem. Throughout this section, Alice's input will always be denoted $x \in \{0,1\}^k$. The length of the iECC will be denoted by n.

At a high level, our proof will proceed as follows. We will split any candidate protocol into two sections, the first consisting of the first $\frac{21}{47}n$ rounds of the protocol, and the second consisting of the remaining $\frac{26}{47}n$ rounds of the protocol. In the first section, we denote the number of bits that Alice sends by A_1, and the number that Bob sends by B_1. Likewise, in the second section, we denote the number of bits that Alice and Bob send by A_2 and B_2 respectively. We will present three attacks in Sections 5.1, 5.2, and 5.3 such that depending on the values of A_1, B_1, A_2, B_2, at least one attack is guaranteed to succeed while using at most $\frac{13}{47}$ corruptions.

Throughout this section, a *transcript* is the sequence of bits that is received by either of the parties. Note that since the adversary may corrupt messages, the transcript may be different than what was sent by Alice and Bob. We say that an attack *succeeds* with α corruption if there exist two inputs $x_1, x_2 \in \{0,1\}^k$ along with respective strategies corrupting at most αn bits such that Bob's view of the transcript in both cases is identical. Then, Bob cannot guess Alice's true value of $x \in \{x_1, x_2\}$ with probability better than $\frac{1}{2}$.

5.1 Attack 1

In the first attack, the adversary behaves the same on both sections of the protocol. She corrupts Alice's bits while leaving Bob's untouched, such that there exist two inputs for which at most of $\frac{1}{3}$ of Alice's communication is corrupted. We remark that this attack has been known since [2].

▶ **Lemma 14.** *For any protocol consisting of A bits from Alice and B bits from Bob, and for any three possible inputs x_1, x_2, x_3, there exists two of the three inputs $y_1, y_2 \in \{x_1, x_2, x_3\}$ and a transcript $T \in \{0,1\}^{A+B}$ such that the adversary can corrupt at most $\frac{1}{3}A + 1$ bits so that the protocol transcript is T in both the case Alice has y_1 or y_2.*

The proof is omitted in this conference version.

The attack is stated below.

Attack 1

Let $x_1, x_2, x_3 \in \{0,1\}^k$ be three of Alice's possible inputs. By Lemma 14, there exist $y_1, y_2 \in \{x_1, x_2, x_3\}$ and transcript $T \in \{0,1\}^n$ such that the adversary can corrupt at most $\frac{1}{3}(A_1 + A_2) + 1$ bits to obtain transcript T in both the case Alice has y_1 and if she has y_2. The adversary simply corrupts the protocol so that the resulting transcript is T.

▶ **Lemma 15.** *Attack 1 succeeds with corrupting* $\frac{1}{3}A_1 + \frac{1}{3}A_2 + 1$ *bits.*

Proof. This follows immediately from Lemma 14: regardless of whether Alice has y_1 or y_2, the adversary is able to have Bob receive the same transcript, using $\frac{1}{3}(A_1 + A_2) + 1$ bits of corruption. ◀

5.2 Attack 2

In our second attack, the adversary behaves differently in the two sections of the protocol. In the first section, the adversary essentially causes Bob's feedback to look random, so that Alice can do no better than to send a distance $\frac{1}{2}$ error-correcting code. This allows the adversary to corrupt $\frac{1}{4}$ of Alice's bits during this first section so that Bob cannot distinguish between three inputs. Then, in the second section, we use Lemma 14 from the previous section to show that the adversary has a strategy corrupting only $\frac{1}{3}A_2$ bits to confuse Bob between two of the remaining three inputs.

To argue that the adversary can perform her attack in the first section, we need the following lemma.

▶ **Lemma 16.** *For any* $0 < \epsilon < 0.1$*, suppose Alice has* K *possible inputs where* $K > (4/\epsilon)^{1/3}$*. Then for any protocol consisting of* A *bits from Alice and* B *bits from Bob where* $A + B \geq \frac{3\log(1/\epsilon)}{\epsilon^3}$*, there exist three inputs* x_1, x_2, x_3 *and a transcript* T *such that regardless of which of* x_1, x_2, x_3 *Alice has as input, the adversary can corrupt at most* $\left(\frac{1}{4} + \frac{3\epsilon}{2}\right) \cdot A + \left(\frac{1}{2} + \epsilon\right) \cdot B + 1$ *bits so that the protocol transcript is* T*.*

The proof is omitted in this conference version.

We now state our second attack.

Attack 2

Let inputs $x_1, x_2, x_3 \in \{0,1\}^k$ and transcript $T_1 \in \{0,1\}^{21n/47}$ be such that they satisfy Lemma 16 for the first section of the protocol. Then, for the first section of the protocol, the adversary corrupts the transcript to look like T_1, using at most $\left(\frac{1}{4} + \frac{3\epsilon}{2}\right) A_1 + \left(\frac{1}{2} + \epsilon\right) B_1 + 1$ bits of corruption in the cases where Alice had x_1, x_2, x_3.

For the second section of the protocol, the adversary corrupts the communication to the transcript $T_2 \in \{0,1\}^{26n/47}$ as found in Lemma 14 such that there exist two of x_1, x_2, x_3, denoted y_1, y_2, for which the adversary can corrupt at most $\frac{1}{3}A_2 + 1$ of the communication so that the transcript of received bits is T_2 if Alice has y_1 or y_2.

▶ **Lemma 17.** *Suppose that* $k \geq \frac{141\log(1/\epsilon)}{21\epsilon^3}$*. Attack 2 succeeds with corrupting* $\left(\frac{1}{4} + \frac{3\epsilon}{2}\right) \cdot A_1 + \left(\frac{1}{2} + \epsilon\right) \cdot B_1 + \frac{1}{3}A_2 + 2$ *bits.*

Proof. Regardless of whether Alice has y_1 or y_2, the transcript from Bob's perspective when the adversary employs this attack looks like T_1 followed by T_2 (restricted to Bob's viewpoint). Since $A_1 + B_1 \geq \max\{\frac{21}{26}A_2, k - A_2\}$ (where $A_1 + A_2 \geq k$ holds since Alice needs to send k

bits to communicate x, even noiselessly), it follows that $A_1+B_1 \geq \frac{21}{47}k \geq 3\log(1/\epsilon)/\epsilon^3$, so the condition of Lemma 16 is satisfied. Then, by Lemma 16, the number of corruptions used in the first section of the protocol when Alice has y_1 or y_2 is at most $\left(\frac{1}{4}+\frac{3\epsilon}{2}\right)\cdot A_1+\left(\frac{1}{2}+\epsilon\right)\cdot B_1$, and by Lemma 14, the number of corrupted bits in the second section whether Alice has y_1 or y_2 is at most $\frac{1}{3}A_2+1$. ◀

5.3 Attack 3

In our third attack, we employ the following strategy. At a high level, we choose two inputs x_1 and x_2. In the first section of the protocol, Bob's view is as if Alice had x_1, while Bob's bits are corrupted so that Alice thinks that he has been receiving and responding correctly. In the second section of the protocol, Bob's bits are flipped randomly, and Alice's communication is corrupted to look like she has x_2.

The first lemma we will need is to show that for the first section of the protocol, there are many inputs for which the uncorrupted transcripts have pairwise small Hamming distance.

▶ **Lemma 18.** *Let $\epsilon > 0$ and suppose Alice has K possible inputs. Then for any protocol consisting of A bits from Alice and B bits from Bob, there exists a set Γ of size $K'_\epsilon(K) = K^\epsilon - \frac{1}{\epsilon}$ inputs such that for any two $x_1, x_2 \in \Gamma$, the relative distance of the (uncorrupted) transcripts in the case where Alice has x_1 or x_2 is $\leq \left(\frac{1}{2}+\epsilon\right)\cdot(A+B)$.*

The proof is omitted in this conference version.

▶ **Lemma 19.** *Let $\epsilon > 0$, and suppose Alice has $K' > \sqrt{2/\epsilon}$ possible inputs. For any protocol consisting of A bits from Alice and B bits from Bob such that $A+B > \frac{3\log(1/\epsilon)}{\epsilon^3}$, there exist two inputs x_1, x_2 such that for any advice α that Bob receives at the beginning of the protocol (after both Alice and Bob have fixed their strategies), there exist two transcripts T_1, T_2 such that the Bob's view of the two transcripts is the same, and that in the case of Alice having x_1, the adversary needs only corrupt $\left(\frac{1}{2}+2\epsilon\right)A+\left(\frac{1}{2}+\epsilon\right)B$ bits to get transcript T_1, and in the case of Alice having x_2, the adversary needs only corrupt $\left(\frac{1}{2}+\epsilon\right)B$ bits so that the transcript is T_2.*

The proof is omitted in this conference version.

Attack 3

Denote by $T_1(y)$ the uncorrupted transcript corresponding to Alice having input y in the first section of the protocol. By Lemma 18, there exists a set M of $2^{\epsilon k}-\frac{1}{\epsilon}$ inputs such that for every $y_1, y_2 \in M$, it holds that $\Delta(T_1(y_1), T_1(y_2)) \leq \left(\frac{1}{2}+\epsilon\right)\cdot n$. Next, consider the second section of the protocol, conditioned on Alice having seen $T_1(x)$ (restricted to her view) in the first section of the protocol. By Lemma 19 there exist $x_1, x_2 \in M$ such that no matter what advice α Bob receives at the beginning of this second section, there exist transcripts $T_{2,1}(\alpha)$ and $T_{2,2}(\alpha)$ such that Bob's view of the two transcripts are the same, and these $T_{2,1}(\alpha), T_{2,2}(\alpha)$ satisfy the properties listed in Lemma 19.

In the first section of the protocol, the adversary corrupts the communication so that Bob always receives $T_1(x_1)$ (restricted to the bits that Bob sees), and so that Alice receives $T(x)$ (restricted to the bits that she sees), where x denotes Alice's input.

In the second section of the protocol, the adversary corrupts the communication so that the transcript is $T_{2,1}(\alpha = T_1(x_1))$ in the case of Alice having x_1, and $T_{2,2}(\alpha = T_1(x_1))$ otherwise.

▶ **Lemma 20.** *Suppose that $k > \frac{141\log(1/\epsilon)}{26\epsilon^3}$. Attack 3 succeeds with corrupting*

$$\max\left\{\left(\frac{1}{2}+2\epsilon\right)\cdot A_2+\left(\frac{1}{2}+\epsilon\right)\cdot B_2\ ,\quad \left(\frac{1}{2}+\epsilon\right)\cdot A_1+\left(\frac{1}{2}+\epsilon\right)\cdot B_1+\left(\frac{1}{2}+\epsilon\right)\cdot B_2\right\}$$

bits.

Proof. Regardless of whether Alice has x_1 or x_2, Bob receives the same transcript (restricted to his view). Since $A_2 + B_2 \geq \max\{\frac{26}{21}A_1, k - A_1\}$ (where $A_1 + A_2 \geq k$ holds since Alice needs to send k bits to communicate x, even noiselessly), it follows that $A_2 + B_2 \geq \frac{26}{47}k > \frac{3\log(1/\epsilon)}{\epsilon^3}$, so the condition of Lemma 19 is satisfied. If Alice has x_1, the amount of corruption in the first section is 0, while in the second section the adversary corrupted at most $\left(\frac{1}{2}+2\epsilon\right)A_2 + \left(\frac{1}{2}+\epsilon\right)\cdot B_2$ bits. If Alice has x_2, the amount of corruption in the first section is $\Delta(T_1(x_1), T_1(x_2)) \leq \left(\frac{1}{2}+\epsilon\right)\cdot(A_1+B_1)$, and in the second section the adversary corrupts at most $\left(\frac{1}{2}+\epsilon\right)\cdot B_2$ bits. ◀

5.4 Proof of Theorem 13

In this section, we prove our main result, restated below.

▶ **Theorem 21.** *For sufficiently small $\epsilon > 0$, there exists $k_0 = k_0(\epsilon) \in \mathbb{N}$ such that for any $k > k_0$, no* iECC *over the binary bit flip channel where Alice is trying to communicate $x \in \{0,1\}^k$ is resilient to $\frac{13}{47}+\epsilon$ fraction of adversarial bit flips.*

We begin with the following lemma.

▶ **Lemma 22.** *For any nonnegative $a_1, b_1, a_2, b_2 \in \mathbb{R}$ where $a_1+b_1 = \frac{21}{47}$ and $a_1+b_1+a_2+b_2 = 1$, define*

$$\begin{aligned}\delta_1 &= \frac{1}{3}a_1+\frac{1}{3}a_2,\\ \delta_2 &= \frac{1}{4}a_1+\frac{1}{2}b_1+\frac{1}{3}a_2,\\ \delta_3 &= \max\left\{\frac{1}{2}a_2+\frac{1}{2}b_2, \frac{1}{2}a_1+\frac{1}{2}b_1+\frac{1}{2}b_2\right\}.\end{aligned}$$

It holds that

$$\min\{\delta_1,\delta_2,\delta_3\} \leq \frac{13}{47}.$$

Proof. Using that $b_1 = \frac{21}{47} - a_1$ and $b_2 = \frac{26}{47} - a_2$, we can substitute:

$$\begin{aligned}\delta_1 &= \frac{1}{3}a_1+\frac{1}{3}a_2,\\ \delta_2 &= \frac{21}{94}-\frac{1}{4}a_1+\frac{1}{3}a_2,\\ \delta_3 &= \max\left\{\frac{13}{37}, \frac{1}{2}-\frac{1}{2}a_2\right\}.\end{aligned}$$

Then,

$$\min\{\delta_1,\delta_2,\delta_3\} \leq \frac{13}{47} \iff \min\{\delta_1,\delta_2,\delta_3'\} \leq \frac{13}{47},$$

where $\delta_3' = \frac{1}{2} - \frac{1}{2}a_2$. But note that

$$\frac{9}{35}\delta_1+\frac{12}{35}\delta_2+\frac{2}{5}\delta_3' = \frac{13}{47}$$

where the weights $\frac{9}{35}, \frac{12}{35}, \frac{2}{5}$ sum to 1, so at least one of $\delta_1, \delta_2, \delta_3'$ must be at most $\frac{13}{47}$. ◀

Proof of Theorem 13. Recall that the length of the iECC is $n := A_1 + B_1 + A_2 + B_2 \geq A_1 + A_2 \geq k > \epsilon^{-3}$ (since Alice needs to send at least k bits to communicate x, even in the noiseless setting). Our goal is to show that regardless of the values of A_1, B_1, A_2, B_2, at least one of Attacks 1, 2, and 3 will require at most $\left(\frac{13}{47} + 2\epsilon\right) \cdot n$ corruptions.

By Lemma 15, Attack 1 succeeds using

$$\frac{1}{3}A_1 + \frac{1}{3}A_2 + 1 \leq (\delta_1 + 2\epsilon)\, n$$

bits of corruption, where $\delta_1 := (\frac{1}{3}A_1 + \frac{1}{3}A_2)/n$.

By Lemma 17, Attack 2 succeeds using

$$\left(\frac{1}{4} + \frac{3\epsilon}{2}\right) \cdot A_1 + \left(\frac{1}{2} + \epsilon\right) \cdot B_1 + \frac{1}{3}A_2 + 2 \leq \frac{1}{4}A_1 + \frac{1}{2}B_1 + \frac{1}{3}A_2 + 2\epsilon n = (\delta_2 + 2\epsilon)n$$

bits of corruption, where we define $\delta_2 = (\frac{1}{4}A_1 + \frac{1}{2}B_1 + \frac{1}{3}A_2)/n$.

By Lemma 20, Attack 3 succeeds using

$$\begin{aligned}
&\max\left\{\left(\frac{1}{2} + 2\epsilon\right) \cdot A_2 + \left(\frac{1}{2} + \epsilon\right) \cdot B_2\ ,\ \ \left(\frac{1}{2} + \epsilon\right) \cdot A_1 + \left(\frac{1}{2} + \epsilon\right) \cdot B_1 + \left(\frac{1}{2} + \epsilon\right) \cdot B_2\right\} \\
&\quad \leq \max\left\{\frac{1}{2}A_2 + \frac{1}{2}B_2 + 2\epsilon n, \frac{1}{2}A_1 + \frac{1}{2}B_1 + \frac{1}{2}B_2 + 2\epsilon n\right\} \\
&\quad = (\delta_3 + 2\epsilon)n
\end{aligned}$$

bits of corruption, where we define $\delta_3 = \max\left\{\frac{1}{2}A_2 + \frac{1}{2}B_2, \frac{1}{2}A_1 + \frac{1}{2}B_1 + \frac{1}{2}B_2\right\}/n$.

By Lemma 22, we have that $\min\{\delta_1, \delta_2, \delta_3\} \leq \frac{13}{47}$, so at least one of the three attacks succeeds with $\left(\frac{13}{47} + 2\epsilon\right) \cdot n$ corruption, regardless of the relative ratios of A_1, B_1, A_2, B_2. ◀

References

1 Rudolf Ahlswede, Christian Deppe, and Vladimir Lebedev. Non-binary error correcting codes with noiseless feedback, localized errors, or both. In *2006 IEEE International Symposium on Information Theory*, pages 2486–2487, 2006. doi:10.1109/ISIT.2006.262057.

2 Elwyn R. Berlekamp. Block coding with noiseless feedback. *Massachusetts Institute of Technology, USA*, 1964.

3 Elwyn R. Berlekamp. Block coding for the binary symmetric channel with noiseless, delayless feedback. *Error-correcting Codes*, pages 61–88, 1968.

4 Mark Braverman, Klim Efremenko, Gillat Kol, Raghuvansh Saxena, and Zhijun Zhang. Round-vs-resilience tradeoffs for binary feedback channels. *Electronic Colloquium on Computational Complexity*, TR22-179, December 2022.

5 Marat Burnashev and Hirosuke Yamamoto. On the zero-rate error exponent for a bsc with noisy feedback. *Problems of Information Transmission*, 44, September 2008. doi:10.1134/S0032946008030034.

6 Marat V. Burnashev and Hirosuke Yamamoto. On bsc, noisy feedback and three messages. In *2008 IEEE International Symposium on Information Theory*, pages 886–889, 2008. doi:10.1109/ISIT.2008.4595114.

7 Klim Efremenko, Gillat Kol, Raghuvansh Saxena, and Zhijun Zhang. Binary codes with resilience beyond 1/4 via interaction. *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 2022. arXiv:TR22-129.

8 Meghal Gupta, Venkatesan Guruswami, and Rachel Yun Zhang. Binary error-correcting codes with minimal noiseless feedback. *To appear in STOC 2023*, 2022.

9 Meghal Gupta, Yael Tauman Kalai, and Rachel Yun Zhang. Interactive error correcting codes over binary erasure channels resilient to > 1/2 adversarial corruption. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 609–622, 2022.

10 Meghal Gupta and Rachel Yun Zhang. Positive rate binary interactive error correcting codes resilient to $> 1/2$ adversarial erasures. *arXiv preprint*, 2022. arXiv:2201.11929.

11 Meghal Gupta and Rachel Yun Zhang. A new upper bound on the maximal error resilience of interactive error-correcting codes. *arXiv preprint*, 2023. arXiv:2305.04376.

12 V. Guruswami. List decoding from erasures: bounds and code constructions. *IEEE Transactions on Information Theory*, 49(11):2826–2833, 2003. doi:10.1109/TIT.2003.815776.

13 Venkatesan Guruswami and Madhu Sudan. List Decoding Algorithms for Certain Concatenated Codes. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, STOC '00, pages 181–190, New York, NY, USA, 2000. Association for Computing Machinery. doi:10.1145/335305.335327.

14 Bernhard Haeupler, Pritish Kamath, and Ameya Velingker. Communication with Partial Noiseless Feedback. In *APPROX-RANDOM*, 2015.

15 R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, 1950. doi:10.1002/j.1538-7305.1950.tb00463.x.

16 Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. doi:10.1002/j.1538-7305.1948.tb01338.x.

17 Joel Spencer and Peter Winkler. Three Thresholds for a Liar. *Combinatorics, Probability and Computing*, 1(1):81–93, 1992. doi:10.1017/S0963548300000080.

18 Gang Wang, Yanyuan Qin, and Chengjuan Chang. Communication with partial noisy feedback. In *2017 IEEE Symposium on Computers and Communications (ISCC)*, pages 602–607, 2017. doi:10.1109/ISCC.2017.8024594.

19 K.Sh. Zigangirov. Number of correctable errors for transmission over a binary symmetrical channel with feedback. *Problems Inform. Transmission*, 12:85–97, 1976.

Optimal Mixing via Tensorization for Random Independent Sets on Arbitrary Trees

Charilaos Efthymiou ✉
Computer Science, University of Warwick, Coventry, UK

Thomas P. Hayes ✉
Dept. of Computer Science and Engineering, University at Buffalo, NY, USA

Daniel Štefankovič ✉
Department of Computer Science, University of Rochester, NY, USA

Eric Vigoda ✉
Department of Computer Science, University of California, Santa Barbara, CA, USA

Abstract

We study the mixing time of the single-site update Markov chain, known as the Glauber dynamics, for generating a random independent set of a tree. Our focus is obtaining optimal convergence results for arbitrary trees. We consider the more general problem of sampling from the Gibbs distribution in the hard-core model where independent sets are weighted by a parameter $\lambda > 0$; the special case $\lambda = 1$ corresponds to the uniform distribution over all independent sets. Previous work of Martinelli, Sinclair and Weitz (2004) obtained optimal mixing time bounds for the complete Δ-regular tree for all λ. However, Restrepo et al. (2014) showed that for sufficiently large λ there are bounded-degree trees where optimal mixing does not hold. Recent work of Eppstein and Frishberg (2022) proved a polynomial mixing time bound for the Glauber dynamics for arbitrary trees, and more generally for graphs of bounded tree-width.

We establish an optimal bound on the relaxation time (i.e., inverse spectral gap) of $O(n)$ for the Glauber dynamics for unweighted independent sets on arbitrary trees. Moreover, for $\lambda \leq .44$ we prove an optimal mixing time bound of $O(n \log n)$. We stress that our results hold for arbitrary trees and there is no dependence on the maximum degree Δ. Interestingly, our results extend (far) beyond the uniqueness threshold which is on the order $\lambda = O(1/\Delta)$. Our proof approach is inspired by recent work on spectral independence. In fact, we prove that spectral independence holds with a constant independent of the maximum degree for any tree, but this does not imply mixing for general trees as the optimal mixing results of Chen, Liu, and Vigoda (2021) only apply for bounded degree graphs. We instead utilize the combinatorial nature of independent sets to directly prove approximate tensorization of variance/entropy via a non-trivial inductive proof.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures; Mathematics of computing → Discrete mathematics; Theory of computation → Design and analysis of algorithms

Keywords and phrases MCMC, Mixing Time, Independent Sets, Hard-Core Model, Approximate Counting Algorithms, Sampling Algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.33

Category RANDOM

Related Version *Full Version*: https://arxiv.org/pdf/2307.07727.pdf

Funding *Charilaos Efthymiou*: EPSRC New Investigator Award (grant no. EP/V050842/1) and Centre of Discrete Mathematics and Applications (DIMAP), The University of Warwick.
Daniel Štefankovič: Supported by NSF grant CCF-1563757.
Eric Vigoda: Supported by NSF grant CCF-2147094.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 33; pp. 33:1–33:16
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

This paper studies the mixing time of the Glauber dynamics for the hard-core model assuming that the underlying graph is an *arbitrary* tree. In the hard-core model, we are given a graph $G = (V, E)$ and an activity $\lambda > 0$. The model is defined on the collection of all independent sets of G (regardless of size), which we denote as Ω.

Each independent set $\sigma \in \Omega$ is assigned a weight $w(\sigma) = \lambda^{|\sigma|}$ where $|\sigma|$ is the number of vertices contained in the independent set σ. The Gibbs distribution μ is defined on Ω: for $\sigma \in \Omega$, let $\mu(\sigma) = w(\sigma)/Z$ where $Z = \sum_{\tau\in\Omega} w(\tau)$ is known as the partition function. When $\lambda = 1$ then every independent set has weight one and hence the Gibbs distribution μ is the uniform distribution over (unweighted) independent sets.

Our goal is to sample from μ (or estimate Z) in time polynomial in $n = |V|$. Our focus is on trees. These sampling and counting problems are computationally easy on trees using dynamic programming algorithms. Nevertheless, our interest is to understand the convergence properties of a simple Markov Chain Monte Carlo (MCMC) algorithm known as the Glauber dynamics for sampling from the Gibbs distribution.

The Glauber dynamics (also known as the Gibbs sampler) is the simple single-site update Markov chain for sampling from the Gibbs distribution of a graphical model. For the hard-core model with activity λ, the transitions $X_t \to X_{t+1}$ of the Glauber dynamics are defined as follows: first, choose a random vertex v. Then, with probability $\frac{\lambda}{1+\lambda}$ set $X' = X_t \cup \{v\}$ and with the complementary probability set $X' = X_t \setminus \{v\}$. If X' is an independent set, then set $X_{t+1} = X'$ and otherwise set $X_{t+1} = X_t$.

We consider two standard notions of convergence to stationarity. The *relaxation time* is the inverse spectral gap, i.e., $(1 - \lambda^*)^{-1}$ where $\lambda^* = \max\{\lambda_2, |\lambda_N|\}$ and $1 = \lambda_1 > \lambda_2 \geq \cdots \geq \lambda_N > -1$ are the eigenvalues of the transition matrix P for the Glauber dynamics. The relaxation time is a key quantity in the running time for approximate counting algorithms (see, e.g., [29]). The *mixing time* is the number of steps, from the worst initial state, to reach within total variation distance $\leq 1/2e$ of the stationary distribution, which in our case is the Gibbs distribution μ.

We say that $O(n)$ is the optimal relaxation time and that $O(n \log n)$ is the optimal mixing time (see Hayes and Sinclair [18] for a matching lower bound for any constant degree graph). Here, n denotes the size of the underlying graph. More generally, we say the Glauber dynamics is rapidly mixing when the mixing time is poly(n).

We establish bounds on the mixing time of the Glauber dynamics by means of *approximate tensorization inequalities* for the variance and the entropy of the hard-core model. Interestingly, our analysis utilizes nothing further than the inductive nature of the tree, e.g., we do not make any assumptions about spatial mixing properties of the Gibbs distribution. As a consequence, the bounds we obtain have no dependence on the maximum degree of the graph.

To be more specific we derive the following two group of results: We establish approximate tensorization of variance of the hard-core model on the tree for all $\lambda < 1.1$. This implies *optimal* $O(n)$ relaxation time for the Glauber dynamics. Notably this also includes the uniform distribution over independent sets, i.e., $\lambda = 1$. Furthermore, we establish approximate tensorization of entropy for the hard-core model on any tree for all $\lambda < 0.44$. In turn, this implies *optimal* mixing time $O(n \log n)$ for the Glauber dynamics.

We can now state our main results.

▶ **Theorem 1.** *For any n-vertex tree, for any $\lambda < 1.1$ the Glauber dynamics for sampling λ-weighted independent sets in the hard-core model has an optimal relaxation time of $O(n)$.*

Moreover, when $\lambda \leq 0.44$ then we can prove an optimal bound on the mixing time. (This extends to $\lambda \leq 1.05$ under an intriguing conjecture that we can numerically verify.)

▶ **Theorem 2.** *For any n-vertex tree, for any $\lambda \leq 0.44$ the Glauber dynamics for sampling λ-weighted independent sets in the hard-core model has an optimal mixing time of $O(n \log n)$.*

We believe the optimal mixing results of Theorems 1 and 2 are related to the reconstruction threshold, which we describe now. Consider the complete Δ-regular tree of height h; this is the rooted tree where all nodes at distance $\ell < h$ from the root have $\Delta - 1$ children and all nodes at distance h from the root are leaves. We are interested in how the configuration at the leaves affects the configuration at the root.

Consider fixing an assignment/configuration σ to the leaves (i.e., specifying which leaves are fixed to occupied and which are unoccupied), we refer to this fixed assignment to the leaves as a boundary condition σ. Let μ_σ denote the Gibbs distribution conditional on this fixed boundary condition σ, and let p_σ denote the marginal probability that the root is occupied in μ_σ.

The uniqueness threshold $\lambda_c(\Delta)$ measures the affect of the *worst-case* boundary condition on the root. For all $\lambda < \lambda_c(\Delta)$, all $\sigma \neq \sigma'$, in the limit $h \to \infty$, we have $p_\sigma = p'_\sigma$; this is known as the (tree) uniqueness region. In contrast, for $\lambda > \lambda_c(\Delta)$ there are pairs $\sigma \neq \sigma'$ (namely, all even occupied vs. odd occupied) for which the limits are different; this is the non-uniqueness region. The uniqueness threshold is at $\lambda_c(\Delta) = (\Delta - 1)^{\Delta-1}/(\Delta - 2)^{\Delta} = O(1/\Delta)$.

In contrast, the reconstruction threshold $\lambda_r(\Delta)$ measures the affect on the root of a random/typical boundary condition. In particular, we fix an assignment c at the root and then generate the Gibbs distribution via an appropriately defined broadcasting process. Finally, we fix the boundary configuration σ and ask whether, in the conditional Gibbs distribution μ_σ, the root has a bias towards the initial assignment c. The non-reconstruction region $\lambda < \lambda_r(\Delta)$ corresponds to when we cannot infer the root's initial value, in expectation over the choice of the boundary configuration σ and in the limit $h \to \infty$, see Mossel [24] for a more complete introduction to reconstruction.

The reconstruction threshold is not known exactly but close bounds were established by Bhatnagar, Sly, and Tetali [3] and Brightwell and Winkler [5] who showed that: $C_1 \log^2 \Delta / \log\log \Delta \leq \lambda_r(\Delta) \leq C_2 \log^2 \Delta$ for sufficiently large Δ, and hence $\lambda_r(\Delta)$ is "increasing asymptotically" with Δ whereas the uniqueness threshold is a decreasing function of Δ. Martin [21] showed that $\lambda_r(\Delta) > e - 1$ for all Δ. As a consequence, we conjecture that Theorems 1 and 2 holds for all trees for all $\lambda < e - 1$, which is close to the bound we obtain. A slowdown in the reconstruction region is known: as described below, Restrepo et al. [26] showed that there are trees for which there is a polynomial slow down for $\lambda > C$ for a constant $C > 0$; an explicit constant C is not stated in [26] but using the Kesten-Stigum bound one can show $C \approx 28$.

For general graphs the appropriate threshold is the uniqueness threshold, which is $\lambda_c(\Delta) = O(1/\Delta)$. In particular, for bipartite random Δ-regular graphs the Glauber dynamics has optimal mixing in the uniqueness region [13], and is exponentially slow in the non-uniqueness region [25, 17]. Moreover, for general graphs there is a computational phase transition at the uniqueness threshold: optimal mixing on all graphs of maximum degree Δ in the uniqueness region [13, 9, 10], and NP-hardness to approximately count/sample in the non-uniqueness region [27, 17, 28].

There are a variety of mixing results for the special case on trees, which is the focus of this paper. In terms of establishing optimal mixing time bounds for the Glauber dynamics, previous results only applied to complete, Δ-regular trees. Seminal work of Martinelli, Sinclair, and Weitz [22, 23] proved optimal mixing on complete Δ-regular trees for all λ. The intuitive reason this holds for all λ is that the complete tree corresponds to one of the two extremal phases (all even boundary or all odd boundary) and hence it does not exhibit the phase co-existence which causes mixing. As mentioned earlier, Restrepo et al. [26] shows that there is a there is a fixed assignment τ for the leaves of the complete, Δ-regular tree so that the mixing time slows down in the reconstruction region; intuitively, this boundary condition τ corresponds to the assignment obtained by the broadcasting process.

For more general trees the following results were known. A classical result of Berger et al. [2] proves polynomial mixing time for trees with constant maximum degree [2]. A very recent result of Eppstein and Frishberg [16] proved polynomial mixing time $n^{C(\lambda)}$ of the Glauber dynamics for graphs with bounded tree-width which includes arbitrary trees, however the polynomial exponent is roughly $C(\lambda) = 11 + 6\log(\lambda)$ for trees. For other combinatorial models, rapid mixing for the Glauber dynamics on trees with bounded maximum degree was established for k-colorings in [20] and edge-colorings in [15].

Spectral independence is a powerful notion in the analysis of the convergence rate of Markov Chain Monte Carlo (MCMC) algorithms. For independent sets on an n-vertex graph $G = (V, E)$, spectral independence considers the $n \times n$ pairwise influence matrix I_G where $I_G(v, w) = \mathrm{Prob}_{\sigma\sim\mu}(v \in \sigma \mid w \in \sigma) - \mathrm{Prob}_{\sigma\sim\mu}(v \in \sigma \mid w \notin \sigma)$; this matrix is closely related to the vertex covariance matrix. We say that spectral independence holds if the maximum eigenvalue of $I_{G'}$ for all vertex-induced subgraphs G' of G are bounded by a constant. Spectral independence was introduced by Anari, Liu, and Oveis Gharan [1]. Chen, Liu, and Vigoda [13] proved that spectral independence, together with a simple condition known as marginal boundedness which is a lower bound on the marginal probability of a valid vertex-spin pair, implies optimal mixing time of the Glauber dynamics for *constant-degree graphs.* This has led to a flurry of optimal mixing results, e.g., [14, 4, 19, 12, 11].

The limitation of the above spectral independence results is that they only hold for graphs with constant maximum degree Δ. There are several noteworthy results that achieve a stronger form of spectral independence which establishes optimal mixing for unbounded degree graphs [9, 10]; however all of these results for general graphs only achieve rapid mixing in the tree uniqueness region which corresponds to $\lambda = O(1/\Delta)$ whereas we are aiming for $\lambda = \Theta(1)$.

The inductive approach we use to establish approximate tensorization inequalities can also be utilized to establish spectral independence. In fact, we show that spectral independence holds for any tree when $\lambda < 1.3$, including the case where $\lambda = 1$, see Appendix A of the full version of this paper. Applying the results of Anari, Liu, and Oveis Gharan [1] we obtain a poly(n) bound on the mixing time, but with a large constant in the exponent of n. For constant degree trees we obtain the following optimal mixing result by applying the results of Chen, Liu, and Vigoda [13] (see also [4, 9, 10]).

▶ **Theorem 3.** *For all constant Δ, all $\lambda \le 1.3$, for any tree T with maximum degree Δ, the Glauber dynamics for sampling λ-weighted independent sets has an optimal mixing time of $O(n \log n)$.*

In the next section we recall the key functional definitions and basic properties of variance/entropy that will be useful later in the proofs. In Section 3 we prove approximate tensorization of variance which establishes Theorem 1. Then in Section 4 we prove Theorem 2. We establish spectral independence and prove Theorem 3 in Appendix A of the full version of this paper.

2 Preliminaries

2.1 Standard Definitions

Let P be the transition matrix of a Markov chain $\{X_t\}$ with a finite state space Ω and equilibrium distribution μ. For $t \geq 0$ and $\sigma \in \Omega$, let $P^t(\sigma, \cdot)$ denote the distribution of X_t when the initial state of the chain satisfies $X_0 = \sigma$. The *mixing time* of the Markov chain $\{X_t\}_{t \geq 0}$ is defined by

$$T_{\mathrm{mix}} = \max_{\sigma \in \Omega} \min \left\{ t > 0 \mid \|P^t(\sigma, \cdot) - \mu\|_{\mathrm{TV}} \leq \frac{1}{2\mathrm{e}} \right\} \ . \tag{1}$$

The transition matrix P with stationary distribution μ is called *time reversible* if it satisfies the so-called *detailed balance relation*, i.e., for any $x, y \in \Omega$ we have $\mu(x)P(x,y) = P(y,x)\mu(y)$. For P that is time reversible the set of eigenvalues are real numbers and we denote them as $1 = \lambda_1 \geq \lambda_2 \geq \ldots \lambda_{|\Omega|} \geq -1$. Let $\lambda^* = \max\{|\lambda_2|, |\lambda_{|\Omega|}|\}$, then we define the *relaxation time* T_{relax} by

$$T_{\mathrm{relax}}(P) = \frac{1}{1 - \lambda^*} \ . \tag{2}$$

The quantity $1 - \lambda^*$ is also known as the *spectral gap* of P. We use T_{relax} to bound T_{mix} by using the following inequality

$$T_{\mathrm{mix}}(P) \leq T_{\mathrm{relax}}(P) \cdot \log \left(\frac{2e}{\min_{x \in \Omega} \mu(x)} \right) \ . \tag{3}$$

2.2 Gibbs Distributions and Functional Analytic Definitions

For a graph $G = (V, E)$ and $\lambda > 0$, let $\mu = \mu_{G,\lambda}$ be the hard-core model on this graph with activity λ, while let $\Omega \subseteq 2^V$ be the support of μ. For any $\Lambda \subseteq V$ and any $\tau \subseteq \Lambda$, we let $\mu^{\Lambda,\tau}$ be the distribution μ conditional on that from Λ we choose exactly the vertices in τ. When there is no danger of confusion, we omit Λ. We let $\Omega^\tau \subseteq \Omega$ be the support of $\mu^{\Lambda,\tau}$, while we call τ *feasible* if Ω^τ is nonempty.

For any subset $S \subseteq V$, let μ_S denote the marginal of μ at S, while let $\Omega_S \subseteq 2^S$ denote the support of μ_S. That is, for any $\sigma \subseteq S$, we have that

$$\mu_S(\sigma) = \sum_{\eta \in 2^V} \mathbf{1}\{\eta \cap S = \sigma\} \mu(\eta) \ . \tag{4}$$

In a natural way, we define the conditional marginal. That is, for $\Lambda \subseteq V \setminus S$ and $\tau \subseteq \Lambda$, we let $\mu_S^{\Lambda,\tau}$ denote the marginal at S conditional on the configuration at Λ being τ. Similarly to what we had before, when there is no danger of confusion, we omit Λ. We let Ω_S^τ denote the support of μ_S^τ.

For any function $f : \Omega \to \mathbb{R}_{\geq 0}$, we let $\mu(f)$ is the expected value of f with respect to μ, i.e.,

$$\mu(f) = \sum_{\sigma \in \Omega} \mu(\sigma) f(\sigma) \ .$$

Analogously, we define variance of f with respect to μ by

$$\mathrm{Var}(f) = \mu(f^2) - (\mu(f))^2 \ . \tag{5}$$

We are also using the following equation for $\mathrm{Var}(f)$,

$$\mathrm{Var}(f) = \tfrac{1}{2}\textstyle\sum_{\sigma,\tau\in\Omega}\mu(\sigma)\mu(\tau)\left(f(\sigma)-f(\tau)\right)^2 \quad . \tag{6}$$

For any $S \subseteq V$, for any $\tau \in \Omega_{V\setminus S}$, we define the function $f_\tau : \Omega_S^\tau \to \mathbb{R}_{\geq 0}$ such that $f_\tau(\sigma) = f(\tau \cup \sigma)$ for all $\sigma \in \Omega_S^\tau$. Let $\mathrm{Var}_S^\tau(f_\tau)$ denote the variance of f_τ with respect to the conditional distribution μ_S^τ:

$$\begin{aligned}\mathrm{Var}_S^\tau(f_\tau) &= \mu_S^\tau(f_\tau^2) - \left(\mu_S^\tau(f_\tau)\right)^2 &(7)\\ &= \frac{1}{2}\sum_{\sigma,\eta\in\Omega}\frac{\mathbf{1}\{\sigma\setminus S=\tau,\ \eta\setminus S=\tau\}\mu(\sigma)\mu(\eta)}{\left(\sum_{\hat\sigma\in\Omega}\mathbf{1}\{\hat\sigma\setminus S=\tau\}\mu(\hat\sigma)\right)^2}\left(f(\sigma)-f(\eta)\right)^2 \quad . &(8)\end{aligned}$$

Furthermore, we let

$$\mu(\mathrm{Var}_S(f)) = \sum_{\tau\in\Omega_{V\setminus S}}\mu_{V\setminus S}(\tau)\cdot\mathrm{Var}_S^\tau(f_\tau) \quad , \tag{9}$$

i.e., $\mu(\mathrm{Var}_S(f))$ is the average of $\mathrm{Var}_S^\tau(f_\tau)$ with respect to the τ being distributed as in $\mu_{V\setminus S}(\cdot)$. For the sake of brevity, when $S=\{v\}$, i.e., the set S is a singleton, we use $\mu(\mathrm{Var}_v(f))$.

Similarly to $\mu(f)$ and $\mathrm{Var}(f)$ we define the entropy with respect to μ by

$$\mathrm{Ent}(f) = \mu\left(f\log\tfrac{f}{\mu(f)}\right) \quad , \tag{10}$$

where we use the convention that $0\log 0 = 0$. Analogously to $\mu(\mathrm{Var}_S(f))$, we let

$$\mu(\mathrm{Ent}_S(f)) = \sum_{\tau\in\Omega_{V\setminus S}}\mu_{V\setminus S}(\tau)\mathrm{Ent}_S^\tau(f_\tau) \quad . \tag{11}$$

That is, $\mu(\mathrm{Ent}_S(f))$ is the average of the entropy $\mathrm{Ent}_S^\tau(f_\tau)$ with respect to the measure $\mu_{V\setminus S}(\cdot)$.

When X is a Bernoulli random variable, i.e.,

$$X = \begin{cases} A & \text{with probability } p \\ B & \text{with probability } 1-p, \end{cases}$$

one formulation for the variance that will be convenient for us is

$$\mathrm{Var}(X) = p(1-p)(A-B)^2. \tag{12}$$

2.3 Approximate Tensorization of Variance/Entropy

To bound the convergence rate of the Glauber dynamics we consider the approximate tensorization of variance/entropy as introduced in [7].

We begin with the definition of approximate tensorization of variance.

▶ **Definition 4** (Variance Tensorization). *A distribution μ with support $\Omega \subseteq \{\pm 1\}^V$ satisfies the approximate tensorisation of Variance with constant $C > 0$, denoted using the predicate $VT(C)$, if for all $f : \Omega \to \mathbb{R}_{\geq 0}$ we have that*

$$\mathrm{Var}(f) \leq C \cdot \sum_{v\in V}\mu\left(\mathrm{Var}_v(f)\right) \quad .$$

For a vertex x, recall that $\mathrm{Var}_x[f] = \sum_\sigma \mu_{V\setminus\{x\}}(\sigma)\mathrm{Var}_x^\sigma[f_\sigma]$. Variance tensorization $VT(C)$ yields the following bound on the relaxation time of the Glauber dynamics [7, 6]:

$$T_{\text{relax}} \leq Cn. \tag{13}$$

We continue with the analog for entropy, which is the key step in our proofs establishing optimal mixing bounds of the Glauber dynamics.

▶ **Definition 5** (Entropy Tensorization)**.** *A distribution μ with support $\Omega \subseteq \{\pm 1\}^V$ satisfies the approximate tensorisation of Entropy with constant $C > 0$, denoted using the predicate $ET(C)$, if for all $f : \Omega \to \mathbb{R}_{\geq 0}$ we have that*

$$\mathrm{Ent}(f) \leq C \cdot \sum_{v\in V} \mu\left(\mathrm{Ent}_v(f)\right) \quad .$$

For a vertex x, recall that $\mathrm{Ent}_x[f] = \sum_\sigma \mu_{V\setminus\{x\}}(\sigma)\mathrm{Ent}_x^\sigma[f_\sigma]$. Entropy tensorization $ET(C)$ immediately yields the following mixing time bound for the Glauber dynamics [7, 6]:

$$T_{\text{mix}} \leq Cn\left(\log(\log(1/\mu^*)) + \log 2 + 2\right) \quad . \tag{14}$$

2.4 Decomposition of Variance/Entropy

We use the following basic decomposition properties for entropy and variance. The following lemma follows from a decomposition of relative entropy, see [8, Lemma 3.1] (see also [4, Lemma 2.3]).

▶ **Lemma 6.** *For any $S \subset V$, for any $f \geq 0$:*

$$\mathrm{Var}(f) = \mu[\mathrm{Var}_S(f)] + \mathrm{Var}(\mu_S(f)) \ , \tag{15}$$
$$\mathrm{Ent}(f) = \mu[\mathrm{Ent}_S(f)] + \mathrm{Ent}(\mu_S(f)) \ . \tag{16}$$

We utilize the following decomposition of a product measure, see [6, Eqn (4.7)].

▶ **Lemma 7.** *Consider $U, W \subset V$ where $\mathrm{dist}(U, W) \geq 2$. Then for all $f \geq 0$ we have:*

$$\mu[\mathrm{Var}_U(\mu_W f)] \leq \mu[\mathrm{Var}_U(f)] \ , \tag{17}$$
$$\mu[\mathrm{Ent}_U(\mu_W f)] \leq \mu[\mathrm{Ent}_U(f)] \ . \tag{18}$$

Proof. We apply [6, Eqn (4.7)], which reaches the same conclusion under the assumptions that $U \cap W = \emptyset$ and $\mu_U\mu_W = \mu_W\mu_U$. In the current context, the reason these conditional expectation operators commute here is that, because $\mathrm{dist}(U, W) \geq 2$, if we let S be an independent set sampled according to distribution μ, then the random variables $S \cap U$ and $S \cap W$ are conditionally independent given $S \setminus (U \cup W)$. ◀

3 Variance Factorization

In this section we prove Theorem 1, establishing an optimal bound on the relaxation time for the Glauber dynamics on any tree for $\lambda < 1.1$. We will prove this by establishing variance tensorization, see Definition 4.

Let $T' = (V', E')$ be a tree, let $\{\lambda'_w\}_{w\in V'}$ be a collection of fugacities and let μ' be the corresponding hard-core measure. We will establish the following variance tensorization inequality: for all $f' : 2^{V'} \to \mathbb{R}$

$$\mathrm{Var}(f') \le \sum_{x\in V'} F(\lambda'_x)\mu'(\mathrm{Var}_x(f')), \tag{19}$$

where $F : \mathbb{R}_{\ge 0} \to \mathbb{R}_{\ge 0}$ is a function to be determined later (in Lemma 8). We refer to $\mathrm{Var}(f')$ as the "global" variance and we refer to $\mu'(\mathrm{Var}_x(f'))$ as the "local" variance (of f' at x).

We will establish (19) using induction. Let v be a vertex of degree 1 in T' and let u be the unique neighbor of v. Let $T = (V, E)$ be the tree which is the induced subgraph of G' on $V = V' \setminus \{v\}$. Let $\{\lambda_w\}_{w\in V}$ be a collection of fugacities where $\lambda_w = \lambda'_w$ for $w \ne u$ and $\lambda_u = \lambda'_u/(1 + \lambda'_v)$. Let μ be the hard-core measure on T with fugacities $\{\lambda_w\}_{w\in V}$.

Note that for $S \subseteq V$ we have

$$\mu(S) = \mu'(S) + \mu'(S \cup \{v\}) = \mu'_V(S). \tag{20}$$

Fix a function $f' : 2^{V'} \to \mathbb{R}$. Let $f : 2^V \to \mathbb{R}$ be defined by

$$f(S) = \frac{\mu'(S)f'(S) + \mu'(S \cup \{v\})f'(S \cup \{v\})}{\mu'(S) + \mu'(S \cup \{v\})} = \mathbb{E}_{Z\sim\mu'}[f'(Z) \mid Z\cap V = S] = \mu'_v(f')(S). \tag{21}$$

Note that f' is defined on independent sets of the tree T' and f is the natural projection of f' to the tree T. Since $f = \mu'_v(f')$, then by Lemma 6 we have that:

$$\mathrm{Var}(f') = \mu'(\mathrm{Var}_v(f')) + \mathrm{Var}(f). \tag{22}$$

When we condition on the configuration at u then μ' becomes a product measure on $V \setminus \{u\}$ and $\{v\}$. Hence, from Equation (17) then for any $x \notin \{u, v\}$ (by setting $U = \{x\}$ and $W = \{v\}$) we have:

$$\mu'(\mathrm{Var}_x(f)) \le \mu'(\mathrm{Var}_x(f')).$$

Since by (20) we have $\mu(\mathrm{Var}_x(f)) = \mu'(\mathrm{Var}_x(f))$, the above implies that

$$\mu(\mathrm{Var}_x(f)) \le \mu'(\mathrm{Var}_x(f')). \tag{23}$$

The following lemma is the main technical ingredient. It bounds the local variance at u for the smaller graph T in terms of the local variance at u and v in the original graph T'.

▶ **Lemma 8.** *For $F(x) = 1000(1 + x)^2 \exp(1.3x)$ and any $\lambda_v, \lambda_u \in (0, 1.1]$ we have:*

$$F(\lambda_u)\mu(\mathrm{Var}_u(f)) \le (F(\lambda'_v) - 1)\mu'(\mathrm{Var}_v(f')) + F(\lambda'_u)\mu'(\mathrm{Var}_u(f')). \tag{24}$$

We now utilize the above lemma to prove the main theorem for the relaxation time. We then go back to prove Lemma 8.

Proof of Theorem 1. Note Equation (24) is equivalent to:

$$\mu'(\mathrm{Var}_v(f')) + F(\lambda_u)\mu(\mathrm{Var}_u(f)) \le F(\lambda'_v)\mu'(\mathrm{Var}_v(f')) + F(\lambda'_u)\mu'(\mathrm{Var}_u(f')). \tag{25}$$

We can prove variance tensorization by induction as follows:

$$\begin{aligned}
\mathrm{Var}(f') &= \mu'(\mathrm{Var}_v(f')) + \mathrm{Var}(f) \\
&\le \mu'(\mathrm{Var}_v(f')) + \sum_{x\in V} F(\lambda_x)\mu(\mathrm{Var}_x(f)) \\
&\le \mu'(\mathrm{Var}_v(f')) + F(\lambda_u)\mu(\mathrm{Var}_u(f)) + \sum_{x\in V\setminus\{u\}} F(\lambda'_x)\mu'(\mathrm{Var}_x(f')) \\
&\le F(\lambda'_v)\mu'(\mathrm{Var}_v(f')) + F(\lambda'_u)\mu'(\mathrm{Var}_u(f')) + \sum_{x\in V\setminus\{u\}} F(\lambda'_x)\mu'(\mathrm{Var}_x(f')) \\
&= \sum_{x\in V'} F(\lambda'_x)\mu'(\mathrm{Var}_x(f')),
\end{aligned}$$

where the first line follows by by Equation (22), the second line by induction, the third line by Equation (23), and the fourth line by Equation (25). ◀

Our task now is to prove Lemma 8. The following technical inequality will be useful.

▶ **Lemma 9.** *Let $p \in [0,1]$. Suppose $s_1, s_2 > 0$ satisfy $s_1 s_2 \ge 1$. Then for all $A, B, C \in \mathbb{R}$ we have*

$$(C - pA - (1-p)B)^2 \le (1+s_1)(C-A)^2 + (1-p)^2(1+s_2)(B-A)^2. \tag{26}$$

The proof of Lemma 9 is in Appendix B of the full version of this paper. We can now prove the main lemma.

Proof of Lemma 8. Our goal is to prove Equation (24), let us recall its statement:

$$F(\lambda_u)\mu(\mathrm{Var}_u(f)) \le (F(\lambda'_v) - 1)\mu'(\mathrm{Var}_v(f')) + F(\lambda'_u)\mu'(\mathrm{Var}_u(f')). \tag{24}$$

We will consider each of these local variances $\mu(\mathrm{Var}_u(f))$, $\mu'(\mathrm{Var}_v(f'))$, and $\mu'(\mathrm{Var}_u(f'))$. We will express each of them as a sum over independent sets S of V'. We can then establish Equation (24) in a pointwise manner by considering the corresponding inequality for each independent set S.

Let us begin by looking at the general definition of the expected local variance $\mu'(\mathrm{Var}_x(f'))$ for any $x \in V'$. Applying the definition in Equation (9) and then simplifying we obtain the following (a reader familar with the notation can apply Equation (12) to skip directly to the last line):

$$\begin{aligned}
&\mu'(\mathrm{Var}_x(f')) \\
&= \sum_{S\subseteq V'\setminus\{x\}} \mu'_{V'\setminus\{x\}}(S)\cdot \mathrm{Var}_x^S(f_S) \\
&= \sum_{S\subseteq V'\setminus\{x\}} \left(\sum_{T\subseteq\{x\}} \mu'(S\cup T)\right)\left(\frac{1}{2}\sum_{T,U\subseteq\{x\},T\ne U} \mu_x'^S(T)\mu_x'^S(U)(f'(S\cup T) - f'(S\cup U))^2\right) \\
&= \sum_{S\subseteq V'\setminus\{x\}} \left(\sum_{T\subseteq\{x\}} \mu'(S\cup T)\right)\left(\mu_x'^S(x)\mu_x'^S(\emptyset)(f'(S) - f'(S\cup\{x\}))^2\right) \\
&= \sum_{S\subseteq V'\setminus\{x\}} \Big(\mu'(S) + \mu'(S\cup\{x\})\Big)\frac{\mu'(S)\mu'(S\cup\{x\})}{(\mu'(S)+\mu'(S\cup\{x\}))^2}\Big(f'(S) - f'(S\cup\{x\})\Big)^2.
\end{aligned} \tag{27}$$

Notice in Equation (27) that the only $S \subset V' \setminus \{x\}$ which contribute are those where x is unblocked (i.e., no neighbor of x is included in the independent set S) because we need that S and $S \cup \{x\}$ are both independent sets and hence have positive measure in μ'.

Let us now consider each of the local variances appearing in Equation (24) (expressed using carefully chosen summations that will allow us to prove (24) term-by-term in terms of S).

Let $Q_1 := \mu(\mathrm{Var}_u(f))$ denote the expected local variance of f at u. We will use (27); note that only S where u is unblocked (that is, when no neighbor of u is occupied) contribute to the local variance. Such an S where u is unblocked and $u \notin S$ has the same contribution as $S \cup \{u\}$ times $1/\lambda_u$ (since $\mu(S \cup \{u\}) = \lambda_u \mu(S)$). Hence the expected local variance of f at u is given by

$$Q_1 := \mu(\mathrm{Var}_u(f)) = \sum_{S \subseteq V; u \in S} \mu(S) \left(1 + \frac{1}{\lambda_u}\right) \frac{\lambda_u}{1+\lambda_u} \frac{1}{1+\lambda_u} \left(f(S \setminus \{u\}) - f(S)\right)^2 .$$

We have $f(S) = f'(S)$ (since $u \in S$) and $f(S \setminus \{u\}) = \frac{1}{1+\lambda'_v} f'(S \setminus \{u\}) - \frac{\lambda'_v}{1+\lambda'_v} f'(S \setminus \{u\} \cup \{v\})$. Plugging these in and simplifying we obtain the following:

$$Q_1 = \frac{1+\lambda'_v}{1+\lambda'_u+\lambda'_v} \sum_{S \subseteq V; u \in S} \mu(S) \left(f'(S) - \frac{1}{1+\lambda'_v} f'(S-u) - \frac{\lambda'_v}{1+\lambda'_v} f'(S-u+v)\right)^2 . \quad (28)$$

We now consider $Q_2 := \mu'(\mathrm{Var}_u(f'))$. To compute the expected local variance of f' at u we need to generate Z from μ' but only Z where u is unblocked contribute to the local variance. We can generate Z by first generating S from μ and if $u \notin S$ adding v with probability $\lambda'_v/(1+\lambda'_v)$. The S where $u \notin S$ contribute only if u is unblocked; contributing the same amount as $S \cup \{u\}$ multiplied by $1/\lambda'_u$ (since $\mu'(S \cup \{u\}) = \lambda'_u \mu'(S)$) and by $1/(1+\lambda'_v)$ (since they only contribute if we do not add v). Hence, we have the following:

$$\begin{aligned} Q_2 := \mu(\mathrm{Var}_u(f')) &= \sum_{S \subseteq V; u \in S} \mu(S) \left(1 + \frac{1}{\lambda'_u} \frac{1}{1+\lambda'_v}\right) \frac{\lambda'_u}{1+\lambda'_u} \frac{1}{1+\lambda'_u} \left(f'(S-u) - f'(S)\right)^2 \\ &= \left(\lambda'_u + \frac{1}{1+\lambda'_v}\right) \frac{1}{(1+\lambda'_u)^2} \sum_{S \subseteq V; u \in S} \mu(S) \left(f'(S) - f'(S-u)\right)^2 . \quad (29) \end{aligned}$$

Finally, we consider $\mu'(\mathrm{Var}_v(f'))$, the expected local variance of f' at v. We will establish a lower bound which we will denote by Q_3 (note, Q_1 and Q_2 were identities but here we will have an inequality).

To compute $\mu'(\mathrm{Var}_v(f'))$, the expected local variance of f' at v, we need to generate an independent set Z from μ'. Only those Z where v is unblocked (that is where u is missing) contribute to the local variance. We can generate Z by generating S from μ (whether we add or do not add v does not change the contribution to the local variance). As in Equation (27), we obtain the following:

$$\begin{aligned} \mu'(\mathrm{Var}_v(f')) &= \sum_{S \subseteq V; u \notin S} \mu(S) \frac{1}{1+\lambda'_v} \frac{\lambda'_v}{1+\lambda'_v} \left(f'(S \cup \{v\}) - f'(S)\right)^2 \\ &\geq \sum_{S \subseteq V; u \in S} \mu(S) \frac{1}{\lambda'_u} \frac{1}{1+\lambda'_v} \frac{\lambda'_v}{1+\lambda'_v} \left(f'(S \cup \{v\} \setminus \{u\}) - f'(S \setminus \{u\})\right)^2 , \end{aligned}$$

where in the summation in the second line we effectively sum over a subset of sets not containing u (the ones that can be obtained by removing u from a set containing u).

Let Q_3 denote the lower bound we obtained above:

$$Q_3 := \frac{1}{\lambda'_u}\frac{1}{1+\lambda'_v}\frac{\lambda'_v}{1+\lambda'_v}\sum_{S\subseteq V;u\in S}\mu(S)\left(f'(S\cup\{v\}\setminus\{u\})-f'(S\setminus\{u\})\right)^2 \geq \mu'(\mathrm{Var}_v(f')). \tag{30}$$

Plugging in (28), (29), (30) we obtain that Equation (24) follows from the following inequality:

$$F(\lambda_u)Q_1 \leq (F(\lambda'_v)-1)Q_3 + F(\lambda'_u)Q_2. \tag{31}$$

We will establish (31) term-by-term, that is, for each S in the sums of (28), (29), (30). Fix $S \subseteq V$ such that $u \in S$ and let $A = f'(S-u)$, $B = f'(S-u+v)$, and $C = f'(S)$. We need to show

$$\begin{aligned}&\frac{1+\lambda'_v}{1+\lambda'_u+\lambda'_v}\left(C-\frac{1}{1+\lambda'_v}A-\frac{\lambda'_v}{1+\lambda'_v}B\right)^2 F\left(\frac{\lambda'_u}{1+\lambda'_v}\right)\\ &\leq \frac{1+\lambda'_v+\lambda'_u}{1+\lambda'_v}\frac{1}{(1+\lambda'_u)^2}(C-A)^2F(\lambda'_u)+\frac{1}{\lambda'_u(1+\lambda'_v)^2}(B-A)^2(F(\lambda'_v)-1).\end{aligned} \tag{32}$$

Let $p = 1/(1+\lambda'_v)$. We will match (26) to (32), by first dividing both sides of (32) by $\frac{1+\lambda'_v}{1+\lambda'_u+\lambda'_v}F\left(\frac{\lambda'_u}{1+\lambda'_v}\right)$ and then choosing

$$1+s_1 = \left(\frac{1+\lambda'_u+\lambda'_v}{(1+\lambda'_v)(1+\lambda'_u)}\right)^2\cdot\frac{F(\lambda'_u)}{F\left(\frac{\lambda'_u}{1+\lambda'_v}\right)} \quad\text{and}\quad 1+s_2 = \frac{1+\lambda'_u+\lambda'_v}{\lambda'_u(1+\lambda'_v){\lambda'_v}^2}\cdot\frac{F(\lambda'_v)-1}{F\left(\frac{\lambda'_u}{1+\lambda'_v}\right)}.$$

Note that with this choice of s_1 and s_2 equations (26) and (32) are equivalent, and hence to prove (32) it is enough to show $s_1s_2 \geq 1$.

▷ Claim 10. $s_1s_2 \geq 1$.

We defer the proof of this technical inequality to Appendix B of the full version of this paper. This completes the proof of the lemma. ◀

4 Entropy Factorization

Here we will prove Theorem 2 establishing $O(n\log n)$ mixing time for $\lambda \leq .44$. We will accomplish this task by proving the following approximate tensorization inequality:

$$\mathrm{Ent}(f') \leq \sum_{x\in V'} F(\lambda'_x)\mu'(\mathrm{Ent}_x(f')), \tag{33}$$

where $F : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a function to be determined later. By Equation (14) this implies a mixing time bound of $O(n\log n)$ and hence Theorem 2 follows from Equation (33).

We use the same notation as Section 3. Let T' be a tree and μ' be the hard-core measure on T' with fugacities $\{\lambda'_w\}_{w\in V'}$. Let v be a vertex of degree 1 in T' and u is the unique neighbor of v. Let v be a vertex of degree 1 in T' and let u be the unique neighbor of v. Let $T = (V, E)$ be the tree which is the induced subgraph of G' on $V = V' \setminus \{v\}$. Let $\{\lambda_w\}_{w\in V}$ be a collection of fugacities where $\lambda_w = \lambda'_w$ for $w \neq u$ and $\lambda_u = \lambda'_u/(1+\lambda'_v)$. Let μ be the hard-core measure on T with fugacities $\{\lambda_w\}_{w\in V}$.

Fix a function $f' : 2^{V'} \to \mathbb{R}_{\geq 0}$. Let $f : 2^V \to \mathbb{R}_{\geq 0}$ be defined by $f(S) = \mathbb{E}_{Z\sim\mu'}[f'(Z) \mid Z \cap V = S] = \mu'_v(f')(S)$, which is the same definition as in Equation (21).

Our main technical lemma is the following.

▶ **Lemma 11.** *For $F(x) = 1000(1+x)\exp(x)$ and $\lambda_u, \lambda'_v, \lambda'_u \in [0, 0.44]$ we have the following.*

$$F(\lambda_u)\mu(\mathrm{Ent}_u(f)) \le (F(\lambda'_v) - 1)\mu'(\mathrm{Ent}_v(f')) + F(\lambda'_u)\mu'(\mathrm{Ent}_u(f')). \tag{34}$$

Using the above lemma we will establish (33) using induction as done in Section 3 for variance.

Proof of Theorem 2.

$$\begin{aligned}
\mathrm{Ent}(f') &= \mu'(\mathrm{Ent}_v(f')) + \mathrm{Ent}(f) && \text{by Equation (16)}\\
&\le \mu'(\mathrm{Ent}_v(f')) + \sum_{x\in V} F(\lambda_x)\mu(\mathrm{Ent}_x(f)) && \text{by induction}\\
&\le \mu'(\mathrm{Ent}_v(f')) + F(\lambda_u)\mu(\mathrm{Ent}_u(f)) + \sum_{x\in V\setminus\{u\}} F(\lambda'_x)\mu'(\mathrm{Ent}_x(f')) && \text{by Equation (18)}\\
&= \sum_{x\in V'} F(\lambda'_x)\mu'(\mathrm{Var}_x(f')) && \text{by Equation (34).}
\end{aligned}$$

Hence, Theorem 2 follows from Lemma 11. ◀

Let

$$G(p, A, B) = pA\ln A + (1-p)B\ln B - (pA + (1-p)B)\ln(pA + (1-p)B).$$

Let $Q_1 := \mu(\mathrm{Ent}_u(f))$. Then we have the following analog of Equation (28):

$$Q_1 = \left(1 + \frac{1+\lambda'_v}{\lambda'_u}\right) \sum_{S\subseteq V; u\in S} \mu(S) G\left(\frac{1+\lambda'_v}{1+\lambda'_v+\lambda'_u}, \frac{f'(S\setminus\{u\})}{1+\lambda'_v} + \frac{\lambda'_v f'(S\cup\{v\}\setminus\{u\})}{1+\lambda'_v}, f'(S)\right). \tag{35}$$

Let $Q_2 := \mu'(\mathrm{Ent}_u(f'))$ denote the expected local entropy of f' at u. Then we have the analog of Equation (29):

$$Q_2 = \left(1 + \frac{1}{\lambda'_u(1+\lambda'_v)}\right) \sum_{S\subseteq V; u\in S} \mu(S) G\left(\frac{1}{1+\lambda'_u}, f'(S-u), f(S)\right). \tag{36}$$

Finally, as in Equation (30), we use Q_3 for a lower bound on the expected local entropy of f' at v. We prove $\mu'(\mathrm{Ent}_v(f')) \ge Q_3$ where:

$$Q_3 = \frac{1}{\lambda'_u} \sum_{S\subseteq V; u\in S} \mu(S) G\left(\frac{1}{1+\lambda'_v}, f'(S-u), f(S-u+v)\right). \tag{37}$$

Plugging in (35), (36), (37) we obtain that Equation (34) follows from the following inequality

$$F(\lambda_u)Q_1 \le F(\lambda'_u)Q_2 + (F(\lambda'_v) - 1)Q_3. \tag{38}$$

We will establish (34) term-by-term, that is, for each S in the sums of (35), (36), (37). Fix $S \subseteq V$ such that $u \in S$ and let $A = f'(S-u)$, $B = f'(S-u+v)$, and $C = f'(S)$. We need to show

$$\begin{aligned}
&\left(1 + \frac{1+\lambda'_v}{\lambda'_u}\right) G\left(\frac{1+\lambda'_v}{1+\lambda'_v+\lambda'_u}, \frac{1}{1+\lambda'_v}A + \frac{\lambda'_v}{1+\lambda'_v}B, C\right) F\left(\frac{\lambda'_u}{1+\lambda'_v}\right) \le\\
&\quad \left(1 + \frac{1}{\lambda'_u(1+\lambda'_v)}\right) G\left(\frac{1}{1+\lambda'_u}, A, C\right) F(\lambda'_u) + \frac{1}{\lambda'_u} G\left(\frac{1}{1+\lambda'_v}, A, B\right)(F(\lambda'_v) - 1).
\end{aligned} \tag{39}$$

We will show that the following lemma implies an optimal mixing time bound of $O(n \log n)$ for all $\lambda \leq .44$.

▶ **Lemma 12.** *Let $b, p \in (0,1)$. For any $A, B, C \geq 0$ we have*

$$G(b, A, C) + (1-b)G(p, A, B) - (b+p-bp)G\left(\frac{b}{b+p-bp}, pA + (1-p)B, C\right) \geq 0. \tag{40}$$

The proof of Lemma 12 appears in Appendix D of the full version of this paper.

Before finishing the proof of Lemma 11 using Lemma 12, we have the following conjecture which is a generalization of Lemma 12. This conjecture implies (39) for all $\lambda \leq 1.05$ and hence an optimal mixing time bound of $O(n \log n)$ for all $\lambda \leq 1.05$. Lemma 12 corresponds to Conjecture 13 "around" $W = 1$ (note that for $W = 1$ both sides of the below inequality are zero). We can numerical verify the conjecture.

▶ **Conjecture 13.** *Let $b, p \in (0,1)$. For any $W \in (1-b, 1/(1-p))$ and any $A, B, C \geq 0$ we have*

$$\Big|\ln W\Big|(b+p-bp)G\left(\frac{b}{b+p-bp}, pA + (1-p)B, C\right) \leq$$
$$p\Big|\ln \frac{pW}{pW - W + 1}\Big|G(b, A, C) + b\Big|\ln \frac{W+b-1}{bW}\Big|G(p, A, B).$$

In Appendix C of the full version of this paper we prove that Conjecture 13 implies a strengthening of Lemma 11 with the interval $[0, 1.05]$ and hence $O(n \log n)$ mixing time.

4.1 Proof of Lemma 11

Here we prove that Lemma 12 implies Equation (39), and hence Lemma 11. Recall, $F(x) = 1000(1+x)\exp(x)$.

Let

$$p = \frac{1}{1+\lambda'_v} \quad \text{and} \quad b = \frac{1}{1+\lambda'_u} \quad \text{and} \quad \alpha = \frac{1}{p(1-b)} F\left(\frac{p(1-b)}{b}\right).$$

Note that

$$1 + \frac{1+\lambda'_v}{\lambda'_u} = \frac{b+p-bp}{p(1-b)}, \qquad \frac{1+\lambda'_v}{1+\lambda'_v+\lambda'_u} = \frac{b}{b+p-bp}, \qquad \frac{\lambda'_u}{1+\lambda'_v} = \frac{p(1-b)}{b}.$$

We aim to prove (39) using the following sequence of inequalities

$$\left(1 + \frac{1+\lambda'_v}{\lambda'_u}\right) G\left(\frac{1+\lambda'_v}{1+\lambda'_v+\lambda'_u}, \frac{1}{1+\lambda'_v}A + \frac{\lambda'_v}{1+\lambda'_v}B, C\right) F\left(\frac{\lambda'_u}{1+\lambda'_v}\right)$$

$$= \alpha(b+p-bp)G\left(\frac{b}{b+p-bp}, pA + (1-p)B, C\right) \tag{41}$$

$$\leq \alpha p\Big(\frac{1}{p}\Big)G(b, A, C) + \alpha b\Big(\frac{1-b}{b}\Big)G(p, A, B) \tag{42}$$

$$\leq \left(1 + \frac{1}{\lambda'_u(1+\lambda'_v)}\right) G\left(\frac{1}{1+\lambda'_u}, A, C\right) F(\lambda'_u) + \frac{1}{\lambda'_u} G\left(\frac{1}{1+\lambda'_v}, A, B\right)(F(\lambda'_v) - 1), \tag{43}$$

Here, Eq. (41) follows from the definitions.

Inequality (42) follows from Lemma 12 (we avoided simplifying (42) in order to make the match to Lemma 12 easier).

Inequality (43) follows from the following two inequalities:

$$\alpha p\Big(\frac{1}{p}\Big) \le \frac{b+p-bp}{p(1-b)} F\Big(\frac{1-b}{b}\Big), \text{ and} \tag{44}$$

$$\alpha b\Big(\frac{1-b}{b}\Big) \le \frac{b}{1-b}\Big(F\Big(\frac{1-p}{p}\Big)-1\Big). \tag{45}$$

Replacing α by its definition, equations (44) and (45) become

$$F\left(\frac{p(1-b)}{b}\right) \le (b+p-bp)F\Big(\frac{1-b}{b}\Big), \quad \text{and} \tag{46}$$

$$F\left(\frac{p(1-b)}{b}\right)(1-b) \le bp\left(F\Big(\frac{1-p}{p}\Big)-1\right). \tag{47}$$

Let x and y be such that $b = 1/(1+x)$ and $p = 1/(1+y)$. Equations (46) and (47) simplify to the following

$$F\left(\frac{x}{1+y}\right) \le \frac{1+x+y}{(1+x)(1+y)}F(x) \quad \text{and} \quad F\left(\frac{x}{1+y}\right)x \le \frac{F(y)-1}{1+y}.$$

Note that $F(y) \ge 1000$ and hence it is enough to satisfy the following inequalities.

$$F\left(\frac{x}{1+y}\right) \le \frac{1+x+y}{(1+x)(1+y)}F(x) \quad \text{and} \quad F\left(\frac{x}{1+y}\right)x \le \frac{999}{1000}\frac{F(y)}{1+y}.$$

Recalling the definition of F, the constraints further simplify to:

$$\exp\left(\frac{x}{1+y}\right) \le \exp(x) \quad \text{and} \quad \exp\left(\frac{x}{1+y}\right)\frac{(1+x+y)x}{1+y} \le \frac{999}{1000}\exp(y). \tag{48}$$

Note that the first constraint follows from the fact that exp() is increasing and $x, y > 0$. The second constraint is addressed in the following lemma.

▶ **Lemma 14.** *For $x, y \in [0, u]$, where $u = 0.44$, we have*

$$\exp\left(\frac{x}{1+y}\right)\frac{(1+x+y)x}{1+y} \le \frac{999}{1000}\exp(y). \tag{49}$$

Proof. Let

$$Q = \frac{999}{1000}\exp(y) - \exp\left(\frac{x}{1+y}\right)\frac{(1+x+y)x}{1+y}.$$

We have

$$\frac{\partial}{\partial y}Q = \frac{999}{1000}\exp(y) + \frac{x^2(x+2y+2)\exp\left(\frac{x}{1+y}\right)}{(1+y)^3} > 0,$$

that is, Q is increasing in y and hence we only need to prove (49) for $y = 0$. We need to show

$$\frac{999}{1000} \ge \exp(x)(1+x)x. \tag{50}$$

Note that RHS of (50) is increasing in x and hence we need to check (50) for $x = u$. For $x = u = 0.44$ we have that (50) is satisfied (checked using interval arithmetic). ◀

▶ Remark 15. In order for (48) to hold for $x, y \in [0, u]$ we need it to hold for $x = y(y+1) \leq u$. Equation (48) then simplifies to

$$(1+y)^2 y \leq 1.$$

For (48) to hold (for all $x, y \in [0, u]$) we need $y \leq 0.47$ which in turn implies $u \leq 0.7$. This means that if we want to prove rapid mixing for unweighted independent sets ($\lambda = 1$) we have to go beyond Lemma 12, see Conjecture 13.

References

1 Nima Anari, Kuikui Liu, and Shayan Oveis Gharan. Spectral independence in high-dimensional expanders and applications to the hardcore model. In *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1319–1330, 2020.

2 Noam Berger, Claire Kenyon, Elchanan Mossel, and Yuval Peres. Glauber dynamics on trees and hyperbolic graphs. *Probability Theory and Related Fields*, 131(3):311–340, 2005.

3 Nayantara Bhatnagar, Allan Sly, and Prasad Tetali. Decay of correlations for the hardcore model on the d-regular random graph. *Electron. J. Probab.*, 21:1–42, 2016.

4 Antonio Blanca, Pietro Caputo, Zongchen Chen, Daniel Parisi, Daniel Štefankovič, and Eric Vigoda. On Mixing of Markov Chains: Coupling, Spectral Independence, and Entropy Factorization. In *Proceedings of the 33rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3670–3692, 2022.

5 Graham Brightwell and Peter Winkler. A second threshold for the hard-core model on a Bethe lattice. *Random structures and algorithms*, 24:303–314, 2004.

6 Pietro Caputo. Lecture notes on Entropy and Markov Chains. *Preprint, available from: http://www.mat.uniroma3.it/users/caputo/entropy.pdf*, 2023.

7 Pietro Caputo, Georg Menz, and Prasad Tetali. Approximate tensorization of entropy at high temperature. *Annales de la Faculté des sciences de Toulouse: Mathématiques*, 24(4):691–716, 2015.

8 Pietro Caputo and Daniel Parisi. Block factorization of the relative entropy via spatial mixing. *Communications in Mathematical Physics*, 388:793–818, 2021.

9 Xiaoyu Chen, Weiming Feng, Yitong Yin, and Xinyuan Zhang. Optimal mixing for two-state anti-ferromagnetic spin systems. In *Proceedings of the 63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2022.

10 Yuansi Chen and Ronen Eldan. Localization schemes: A framework for proving mixing bounds for Markov chains. In *Proceedings of the 63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2022.

11 Zongchen Chen and Yuzhou Gu. Fast sampling of b-matchings and b-edge covers. *arXiv preprint arXiv:2304.14289*, 2023.

12 Zongchen Chen, Kuikui Liu, Nitya Mani, and Ankur Moitra. Strong spatial mixing for colorings on trees and its algorithmic applications. *arXiv preprint arXiv:2304.01954*, 2023.

13 Zongchen Chen, Kuikui Liu, and Eric Vigoda. Optimal mixing of Glauber dynamics: Entropy factorization via high-dimensional expansion. In *Proceedings of the 53rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 1537–1550, 2021.

14 Zongchen Chen, Kuikui Liu, and Eric Vigoda. Spectral independence via stability and applications to holant-type problems. In *Proceedings of the 63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2022.

15 Michelle Delcourt, Marc Heinrich, and Guillem Perarnau. The Glauber dynamics for edge-colorings of trees. *Random Structures & Algorithms*, 57(4):1050–1076, 2020.

16 David Eppstein and Daniel Frishberg. Rapid mixing of the hardcore Glauber dynamics and other Markov chains in bounded-treewidth graphs. *arXiv preprint arXiv:2111.03898*, 2022.

17 Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Inapproximability of the partition function for the antiferromagnetic Ising and hard-core models. *Combinatorics, Probability and Computing*, 25(4):500–559, 2016.

18 Thomas P. Hayes and Alistair Sinclair. A general lower bound for mixing of single-site dynamics on graphs. *The Annals of Applied Probability*, 17(3):931–952, 2007.

19 Kuikui Liu. From coupling to spectral independence and blackbox comparison with the down-up walk. In *APPROX-RANDOM*, pages 32:1–32:21, 2021.

20 Brendan Lucier, Michael Molloy, and Yuval Peres. The Glauber dynamics for colourings of bounded degree trees. In *RANDOM*, pages 631–645, 2009.

21 James B. Martin. Reconstruction thresholds on regular trees. In *Discrete Random Walks*, DMTCS Proceedings, pages 191–204, 2003.

22 Fabio Martinelli, Alistair Sinclair, and Dror Weitz. The Ising model on trees: boundary conditions and mixing time. In *44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 628–639, 2003.

23 Fabio Martinelli, Alistair Sinclair, and Dror Weitz. Fast mixing for independent sets, colorings and other models on trees. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 456–465, 2004.

24 Elchanan Mossel. Survey: Information flow on trees. In *Graphs, Morphisms and Statistical Physics*, DIMACS series in discrete mathematics and theoretical computer science, pages 155–170, 2004.

25 Elchanan Mossel, Dror Weitz, and Nicholas C. Wormald. On the hardness of sampling independent sets beyond the tree threshold. *Probability Theory and Related Fields*, 143:401–439, 2007.

26 Ricardo Restrepo, Daniel Štefankovič, Juan C. Vera, Eric Vigoda, and Linji Yang. Phase transition for Glauber dynamics for independent sets on regular trees. *SIAM Journal on Discrete Mathematics*, 28, 2014.

27 Allan Sly. Computational transition at the uniqueness threshold. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 287–296, 2010.

28 Allan Sly and Nike Sun. The computational hardness of counting in two-spin models on d-regular graphs. *The Annals of Probability*, 42(6):2383–2416, 2014.

29 Daniel Štefankovič, Santosh Vempala, and Eric Vigoda. Adaptive simulated annealing: A near-optimal connection between sampling and counting. *Journal of the ACM*, 56(3):1–36, 2009.

Superpolynomial Lower Bounds for Learning Monotone Classes

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

Abstract

Koch, Strassle, and Tan [SODA 2023], show that, under the randomized exponential time hypothesis, there is no distribution-free PAC-learning algorithm that runs in time $n^{\tilde{O}(\log\log s)}$ for the classes of n-variable size-s DNF, size-s Decision Tree, and $\log s$-Junta by DNF (that returns a DNF hypothesis). Assuming a natural conjecture on the hardness of set cover, they give the lower bound $n^{\Omega(\log s)}$. This matches the best known upper bound for n-variable size-s Decision Tree, and $\log s$-Junta.

In this paper, we give the same lower bounds for PAC-learning of n-variable size-s Monotone DNF, size-s Monotone Decision Tree, and Monotone $\log s$-Junta by DNF. This solves the open problem proposed by Koch, Strassle, and Tan and subsumes the above results.

The lower bound holds, even if the learner knows the distribution, can draw a sample according to the distribution in polynomial time, and can compute the target function on all the points of the support of the distribution in polynomial time.

2012 ACM Subject Classification Theory of computation

Keywords and phrases PAC Learning, Monotone DNF, Monotone Decision Tree, Monotone Junta, Lower Bound

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.34

Category RANDOM

Acknowledgements I would like to express my sincere gratitude to the reviewers of RANDOM for their valuable comments on this research. Their feedback was greatly appreciated and helped improve the quality of this work.

1 Introduction

In the distribution-free PAC learning model [13], the learning algorithm of a class of functions C has access to an unknown target function $f \in C$ through labeled examples $(x, f(x))$ where x are drawn according to an unknown but fixed probability distribution $\mathcal{D}$. For a class of hypothesis $H \supseteq C$, we say that the learning algorithm $\mathcal{A}$ *PAC-learns* C *by* H in time T and error ϵ if for every target $f \in C$ and distribution $\mathcal{D}$, $\mathcal{A}$ runs in time T and outputs a hypothesis $h \in H$ which, with probability at least $2/3$, is ϵ-close to f with respect to $\mathcal{D}$. That is, satisfies $\Pr_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x}) \neq h(\boldsymbol{x})] \leq \epsilon$.

Koch et al., [10], show that, under the randomized exponential time hypothesis (ETH), there is no PAC-learning algorithm that runs in time $n^{\tilde{O}(\log\log s)}$ for the classes of n-variable size-s DNF, size-s Decision Tree and $\log s$-Junta by DNF. Assuming a natural conjecture on the hardness of set cover, they give the lower bound $n^{\Omega(\log s)}$. Their lower bound holds, even if the learner knows the distribution, can draw a sample according to the distribution in polynomial time and can compute the target function on all the points of the support of the distribution in polynomial time.

In this paper, we give the same lower bounds for PAC-learning of the classes n-variable size-s Monotone DNF, size-s Monotone Decision Tree and Monotone $\log s$-Junta by DNF. This solves the open problem proposed by Koch, Strassle, and Tan [10].

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 34; pp. 34:1–34:20
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In various learning models, it is widely recognized that the task of learning classes of monotone Boolean functions is significantly simpler compared to learning classes of non-monotone Boolean functions. An illustrative example is the distribution-free PAC learning of monotone DNF within a model that permits membership queries, which can be accomplished efficiently in polynomial time [2]. Conversely, the challenge of learning DNF in the same model remains an unsolved problem, emphasizing the inherent difficulty associated with non-monotone Boolean functions. In this paper, we demonstrate that for specific classes, the task of PAC-learning monotone Boolean functions under the uniform distribution is equally challenging as learning non-monotone Boolean functions.

1.1 Our Results

In this paper, we prove the following three Theorems.

▶ **Theorem 1.** *Assuming randomized ETH, there is a constant c such that any PAC learning algorithm for n-variable* MONOTONE $(\log s)$*-*JUNTA*, size-s* MONOTONE *DT and size-s* MONOTONE *DNF*[1] *by DNF with $\epsilon = 1/(16n)$ must take at least*

$$n^{c\frac{\log\log s}{\log\log\log s}}$$

time.

▶ **Theorem 2.** *Assuming a plausible conjecture on the hardness of* SET-COVER [2]*, there is a constant c such that any PAC learning algorithm for n-variable* MONOTONE $(\log s)$*-*JUNTA*, size-s* MONOTONE *DT and size-s* MONOTONE *DNF by DNF with $\epsilon = 1/(16n)$ must take at least*

$$n^{c\log s}$$

time.

▶ **Theorem 3.** *Assuming randomized ETH, there is a constant c such that any PAC learning algorithm for n-variable* MONOTONE $(\log s)$*-*JUNTA*, size-s* MONOTONE *DT and size-s* MONOTONE *DNF by size-s* DNF *with $\epsilon = 1/(16n)$ must take at least*

$$n^{c\log s}$$

time.

All the above lower bound holds, even if the learner knows the distribution, can draw a sample according to the distribution in polynomial time and can compute the target on all the points of the support of the distribution in polynomial time.

In the following two subsections, we give the technique used in [10] to prove Theorem 1 for $(\log s)$-JUNTA, and the technique we use here to extend the result to MONOTONE $(\log s)$-JUNTA.

[1] The results concerning *size-s* MONOTONE DT and *size-s* MONOTONE DNF stem from the result on MONOTONE $(\log s)$-JUNTA. This relationship also holds for the other theorems. Here, we present a comprehensive list of all these classes to highlight their significance.

[2] See Conjecture 1.

1.2 Previous Technique

In [10], Koch, Strassle, and Tan show that under the randomized exponential time hypothesis, there is no PAC-learning algorithm that runs in time $n^{\tilde{O}(\log\log n)}$ for the class of $\log n$-Junta[3] by DNF. The results for the other classes follow immediately from this result, since all other classes contain $\log n$-Junta. All prior works [1, 6] ruled out only $poly(n)$ time algorithms.

The result in [10] uses the hardness result of (k,k')-Set-Cover where one needs to distinguish between instances that have set cover of size at most k from instances that have minimum-size set cover greater than k':

1. For some parameters k and k' that depends on N, assuming randomized ETH, there is a constant $\lambda<1$ such that (k,k')-Set-Cover on N vertices cannot be solved in time $N^{\lambda k}$.

First, for each set cover instance $\mathcal{S}$, they identify each element in the universe with an assignment in $\{0,1\}^n$ and construct in polynomial time a target function $\Gamma^{\mathcal{S}}:\{0,1\}^n\to\{0,1\}$ and a distribution $\mathcal{D}^{\mathcal{S}}$ that satisfies:

2. The instance $\mathcal{S}$ has minimum-size set cover $\mathrm{opt}(\mathcal{S})$ if and only if the function $\Gamma^{\mathcal{S}}$ is a conjunction of $\mathrm{opt}(\mathcal{S})$ unnegated variables[4] over the distribution $\mathcal{D}^{\mathcal{S}}$.[5]

For a DNF F and $x\in\{0,1\}^n$, they define $\mathrm{width}_F(x)$ to be the size of the smallest term T in F that satisfies $T(x)=1$. They then show that

3. Any DNF F with expected width $\mathbf{E}_{\boldsymbol{x}\sim\mathcal{D}^{\mathcal{S}}}[\mathrm{width}_F(\boldsymbol{x})]\le \mathrm{opt}(\mathcal{S})/2$ is $(1/(2N))$-far from $\Gamma^{\mathcal{S}}$ with respect to $\mathcal{D}^{\mathcal{S}}$ where N is the size[6] of $\mathcal{S}$. That is, $\Pr_{\boldsymbol{x}\sim\mathcal{D}^{\mathcal{S}}}[F(\boldsymbol{x})\ne\Gamma^{\mathcal{S}}(\boldsymbol{x})]\ge 1/(2N)$.

They then use the following gap amplification technique. They define the function $\Gamma^{\mathcal{S}}_{\oplus\ell}:(\{0,1\}^\ell)^n\to\{0,1\}$ where for $y=(y_1,\ldots,y_n)$, $y_i=(y_{i,1},\ldots,y_{i,\ell})\in\{0,1\}^\ell$, $i\in[n]$, we have $\Gamma^{\mathcal{S}}_{\oplus\ell}(y)=\Gamma^{\mathcal{S}}(\oplus y_1,\ldots,\oplus y_n)$ and $\oplus y_i=y_{i,1}+\cdots+y_{i,\ell}$. They also extend the distribution $\mathcal{D}^{\mathcal{S}}$ to a distribution $\mathcal{D}^{\mathcal{S}}_{\oplus\ell}$ over domain $(\{0,1\}^\ell)^n$ and prove that

4. $\Gamma^{\mathcal{S}}_{\oplus\ell}(y)$ is a $(\mathrm{opt}(\mathcal{S})\ell)$-Junta over $\mathcal{D}^{\mathcal{S}}_{\oplus\ell}$.
5. Any DNF formula F with expected width $\mathbf{E}_{\boldsymbol{y}\sim\mathcal{D}^{\mathcal{S}}_{\oplus\ell}}[\mathrm{width}_F(\boldsymbol{y})]\le\mathrm{opt}(\mathcal{S})\ell/4$ is $(1/(4N))$-far from $\Gamma^{\mathcal{S}}_{\oplus\ell}$ with respect to $\mathcal{D}^{\mathcal{S}}_{\oplus\ell}$.

Item 4 follows from the definition of $\Gamma^{\mathcal{S}}_{\oplus\ell}$ and item 2. To prove Item 5, they show that if, to the contrary, there is a DNF F of expected width at most $\mathrm{opt}(\mathcal{S})\ell/4$ that is $1/(4N)$-close to $\Gamma^{\mathcal{S}}_{\oplus\ell}$ with respect to $\mathcal{D}^{\mathcal{S}}_{\oplus\ell}$, then there is $j\in[\ell]$ and a projection of all the variables that are not of the form $y_{i,j}$ that gives a DNF F^* of expected width at most $\mathrm{opt}(\mathcal{S})/2$ that is $1/(2N)$-close to $\Gamma^{\mathcal{S}}$ with respect to $\mathcal{D}^{\mathcal{S}}$. Then, by item 3, we get a contradiction.

They then show that

6. Any size-s DNF that is $(1/(4N))$-close to $\Gamma^{\mathcal{S}}_{\oplus\ell}$ with respect to $\mathcal{D}^{\mathcal{S}}_{\oplus\ell}$ has average width $\mathbf{E}_{\boldsymbol{y}\sim\mathcal{D}^{\mathcal{S}}_{\oplus\ell}}[\mathrm{width}_F(\boldsymbol{y})]\le 4\log s$.

If F is $(1/(4N))$-close to $\Gamma^{\mathcal{S}}_{\oplus\ell}$ with respect to $\mathcal{D}^{\mathcal{S}}_{\oplus\ell}$, then, by items 5 and 6, $4\log s\ge\mathbf{E}_{\boldsymbol{y}\sim\mathcal{D}^{\mathcal{S}}_{\oplus\ell}}[\mathrm{width}_F(\boldsymbol{y})]\ge\mathrm{opt}(\mathcal{S})\ell/4$ and then $s\ge 2^{\mathrm{opt}(\mathcal{S})\ell/16}$. Therefore,

7. Any DNF of size less than $2^{\mathrm{opt}(\mathcal{S})\ell/16}$ is $(1/(4N))$-far from $\Gamma^{\mathcal{S}}_{\oplus\ell}$ with respect to $\mathcal{D}^{\mathcal{S}}_{\oplus\ell}$.

Now, let $k=\tilde{O}(\log\log n)$. Suppose, to the contrary, that there is a PAC-learning algorithm for $\log n$-Junta by DNF with error $\epsilon=1/(8N)$ that runs in time $t=n^{\lambda k/2}=n^{\tilde{O}(\log\log n)}$, where λ is the constant in item 1. Given a (k,k')-Set-Cover instance, we run the learning

[3] k-Junta are Boolean functions that depend on at most k variables
[4] Their reduction gives a conjunction of negated variable. So here, we are referring to the dual function.
[5] That is, there is a term T with $\mathrm{opt}(\mathcal{S})$ variables such that for every x in the support of $\mathcal{D}^{\mathcal{S}}$, $\Gamma^{\mathcal{S}}(x)=T(x)$.
[6] N is the number of sets plus the size of the universe in $\mathcal{S}$.

algorithm for $\Gamma^{\mathcal{S}}_{\oplus\ell}$ for $\ell = \log n/k$. If the instance has set cover at most k, then by item 4, $\Gamma^{\mathcal{S}}_{\oplus\ell}$ is $\log n$-Junta. Then the algorithm learns the target and outputs a hypothesis that is $(1/(8N))$-close to $\Gamma^{\mathcal{S}}_{\oplus\ell}$ with respect to $\mathcal{D}^{\mathcal{S}}_{\oplus\ell}$.

On the other hand, if the instance has a minimum-size set cover of at least k', then any learning algorithm that runs in time $t = n^{\lambda k/2} = n^{\tilde{O}(\log\log n)}$ cannot output a DNF of size more than t terms. By item 7, any DNF of size less than $2^{k'\log n/(16k)} \le 2^{\mathrm{opt}(\mathcal{S})\ell/16}$ is $(1/(4N))$-far from $\Gamma^{\mathcal{S}}_{\oplus\ell}$ with respect to $\mathcal{D}^{\mathcal{S}}_{\oplus\ell}$. By choosing the right parameters k and k', we have $2^{k'\log n/(16k)} > t$, and therefore, any DNF that the algorithm outputs has error of at least $1/(4N)$.

Therefore, by estimating the distance of the output of the learning algorithm from $\Gamma^{\mathcal{S}}_{\oplus\ell}$ with respect to $\mathcal{D}^{\mathcal{S}}_{\oplus\ell}$, we can distinguish between instances that have set cover of size less than or equal to k from instances that have a minimum-size set cover greater than k' in time $t = n^{\lambda k/2}$. Thus, we got an algorithm for (k, k')-Set-Cover that runs in time $n^{\lambda k/2} < n^{\lambda k}$. This contradicts item 1 and finishes the proof of the first lower bound.

Assuming a natural conjecture on the hardness of set cover, they give the lower bound $n^{\Omega(\log s)}$. We will discuss this in Section 5.

1.3 Our Technique

In this paper, we also use the hardness result of (k, k')-Set-Cover . As in [10], we identify each element in the universe with an assignment in $\{0,1\}^n$ and use the function $\Gamma^{\mathcal{S}}$ and the distribution $\mathcal{D}^{\mathcal{S}}$ that satisfies:

1. The instance $\mathcal{S}$ has minimum-size set cover $\mathrm{opt}(\mathcal{S})$ if and only if the function $\Gamma^{\mathcal{S}}$ is a conjunction of $\mathrm{opt}(\mathcal{S})$ variables over the distribution $\mathcal{D}^{\mathcal{S}}$.

We then build a *monotone* target function $\Gamma^{\mathcal{S}}_{\ell}$ and a distribution $\mathcal{D}^{\mathcal{S}}_{\ell}$ and use a different approach to show that any DNF of size less than $2^{\mathrm{opt}(\mathcal{S})\ell/20}$ is $(1/(8N) - 2^{-\mathrm{opt}(\mathcal{S})\ell/20})$-far from $\Gamma^{\mathcal{S}}_{\ell}$ with respect to $\mathcal{D}^{\mathcal{S}}_{\ell}$.

We define, for any odd ℓ, the monotone function $\Gamma^{\mathcal{S}}_{\ell} : (\{0,1\}^{\ell})^n \to \{0,1\}$ where for $y = (y_1, \ldots, y_n)$, $y_i = (y_{i,1}, \ldots, y_{i,\ell})$, $i \in [n]$, we have $\Gamma^{\mathcal{S}}_{\ell}(y) = \Gamma^{\mathcal{S}}(\textsc{Majority}(y_1), \ldots, \textsc{Majority}(y_n))$ where Majority is the majority function. A distribution $\mathcal{D}^{\mathcal{S}}_{\ell}$ is also defined such that

2. $\Pr_{\boldsymbol{y}\sim\mathcal{D}^{\mathcal{S}}_{\ell}}[\Gamma^{\mathcal{S}}_{\ell}(\boldsymbol{y}) = 0] = \Pr_{\boldsymbol{y}\sim\mathcal{D}^{\mathcal{S}}_{\ell}}[\Gamma^{\mathcal{S}}_{\ell}(\boldsymbol{y}) = 1] = 1/2.$

In the paper, $\mathcal{D}^{\mathcal{S}}_{\ell}$ is denoted by $\mathcal{D}_{\ell}$. To see the definition, refer to Definitions 3 and 4. Roughly speaking, the distribution is defined in such a way that removing a few coordinates results in the distribution becoming almost uniform. It is clear from the definition of $\Gamma^{\mathcal{S}}_{\ell}$ and item 1 that

3. $\Gamma^{\mathcal{S}}_{\ell}(y)$ is a monotone $(\mathrm{opt}(\mathcal{S})\ell)$-Junta over $\mathcal{D}^{\mathcal{S}}_{\ell}$.

We then define the *monotone size* of a term T to be the number of unnegated variables that appear in T. The intuition for this definition is that negated variables do not contribute to reducing the size of the DNF of monotone functions, and therefore they may as well be ignored. We first show that

4. For every DNF $F : (\{0,1\}^{\ell})^n \to \{0,1\}$ of size $|F| \le 2^{\mathrm{opt}(\mathcal{S})\ell/5}$ that is ϵ-far from $\Gamma^{\mathcal{S}}_{\ell}$ with respect to $\mathcal{D}^{\mathcal{S}}_{\ell}$, there is another DNF F' of size $|F'| \le 2^{\mathrm{opt}(\mathcal{S})\ell/5}$ with terms of monotone size at most $\mathrm{opt}(\mathcal{S})\ell/5$ that is $(\epsilon - 2^{-\mathrm{opt}(\mathcal{S})\ell/20})$-far from $\Gamma^{\mathcal{S}}_{\ell}$ with respect to $\mathcal{D}^{\mathcal{S}}_{\ell}$.

This is done by simply showing that terms of large monotone size in the DNF F have a small weight according to the distribution $\mathcal{D}^{\mathcal{S}}_{\ell}$ and, therefore, can be removed from F with the cost of $-2^{-\mathrm{opt}(\mathcal{S})\ell/20}$ in the error.

We then, roughly speaking, show that

5. Let F' be a DNF of size $|F'| \le 2^{\mathrm{opt}(\mathcal{S})\ell/5}$ with terms of monotone size at most $\mathrm{opt}(\mathcal{S})\ell/5$. For every $y \in (\{0,1\}^\ell)^n$ in the support of $\mathcal{D}_\ell^{\mathcal{S}}$ that satisfies $\Gamma_\ell^{\mathcal{S}}(y) = 1$, either
   - $F'(y) = 0$ or
   - $F'(y) = 1$, and at least $1/(2N)$ fraction of the points z below y in the lattice $(\{0,1\}^\ell)^n$ that are in the support of $\mathcal{D}_\ell^{\mathcal{S}}$ satisfies $F'(z) = 1$ and $\Gamma_\ell^{\mathcal{S}}(z) = 0$.

Roughly speaking, the latter sub-item follows from the fact that when the monotone size of the terms of F is small and $F'(y) = 1$ then y satisfies a small term T in F'. Now, if z with $\Gamma_\ell^{\mathcal{S}}(z) = 0$ is a randomly chosen point that is almost uniformly distributed below y, but not significantly distant from y, then there exists a sufficiently large probability for $T(z) = 1$ and then $F'(z) = 1$.

By item 5, either $1/(4N)$ fraction of the vectors y that satisfy $\Gamma_\ell^{\mathcal{S}}(y) = 1$ satisfy $F'(y) = 0$ or $(1 - 1/(4N))/(2N) > 1/(4N)$ fraction of the points z that satisfy $\Gamma_\ell^{\mathcal{S}}(z) = 0$ satisfy $F'(z) = 1$. Therefore, with item 2, we get that F' is $1/(8N)$-far from $\Gamma_\ell^{\mathcal{S}}$ with respect to $\mathcal{D}_\ell^{\mathcal{S}}$. This, with item 4, implies that

6. If $F : (\{0,1\}^\ell)^n \to \{0,1\}$ is a DNF of size $|F| < 2^{\mathrm{opt}(\mathcal{S})\ell/20}$, then F is $(1/(8N) - 2^{-\mathrm{opt}(\mathcal{S})\ell/20})$-far from $\Gamma_\ell^{\mathcal{S}}$ with respect to $\mathcal{D}_\ell^{\mathcal{S}}$.

The rest of the proof is almost the same as in [10]. See the discussion in subsection 1.2 after item 7.

Assuming a natural conjecture on the hardness of set cover, we establish a lower bound of $n^{\Omega(\log s)}$. The details and proof of this result will be discussed in Section 5, where we utilize a stronger version of item 4.

1.4 Upper Bounds

The only known distribution-free algorithm for $\log s$-Junta is the trivial algorithm that, for every set of $m = \log s$ variables $S = \{x_{i_1}, \ldots, x_{i_m}\}$, checks if there is a function that depends on S and is consistent with the examples. This algorithm takes $n^{O(\log s)}$ time.

For size-s decision tree and monotone size-s decision tree, the classic result of Ehrenfeucht and Haussler [5] gives a distribution-free time algorithm that runs in time $n^{O(\log s)}$ and outputs a decision tree of size $n^{O(\log s)}$.

The learning algorithm is as follows: Let T be the target decision tree of size s. First, the algorithm guesses the variable at the root of the tree T and then guesses which subtree of the root has size at most $s/2$. Then, it recursively constructs the tree of size $s/2$. When it succeeds, it continues to construct the other subtree.

For size-s DNF and monotone size-s DNF, Hellerstein et al. [7] gave a distribution-free proper learning algorithm that runs in time $2^{\tilde{O}(\sqrt{n \log s})}$.

To the best of our knowledge, all the other results in the literature for learning the above classes are either restricted to the uniform distribution or, in addition, use black box queries or return hypotheses that are not DNF.

We recommend that readers who are not experts in the field refer to the first two sections in [10].

2 Definitions and Preliminaries

In this section, we give the definitions and preliminary results that are needed to prove our results.

2.1 Set Cover

Let $\mathcal{S} = (S, U, E)$ be a bipartite graph on $N = n + |U|$ vertices where $S = [n]$, and for every $u \in U$, $\deg(u) > 0$. We say that $C \subseteq S$ is a set cover of $\mathcal{S}$ if every vertex in U is adjacent to some vertex in C. The Set-Cover problem is to find a minimum-size set cover. We denote by $\mathrm{opt}(\mathcal{S})$ the size of a minimum-size set cover for $\mathcal{S}$.

We identify each element $u \in U$ with the vector $(u_1, \ldots, u_n) \in \{0,1\}^n$ where $u_i = 0$ if and only if $(i, u) \in E$. We will assume that those vectors are distinct. If there are two distinct elements $u, u' \in U$ that have the same vector, then you can remove one of them from the graph. This is because every set cover that covers one of them covers the other.

▶ **Definition 1.** *The (k, k')-Set-Cover problem is the following: Given as input a set cover instance $\mathcal{S} = (S, U, E)$, and parameters k and k'. Output Yes if* $\mathrm{opt}(\mathcal{S}) \le k$ *and No if* $\mathrm{opt}(\mathcal{S}) > k'$.

2.2 Hardness of Set-Cover

Our results are conditioned on the following randomized exponential time hypothesis (ETH) **Hypothesis:** [3, 4, 8, 9, 12]. There exists a constant $c \in (0, 1)$ such that 3-SAT on n variables cannot be solved by a randomized algorithm in $O(2^{cn})$ time with success probability at least 2/3.

The following is proved in [11]. See also Theorem 7 in [10]

▶ **Lemma 2** ([11]). *Let $k \le \frac{1}{2}\frac{\log\log N}{\log\log\log N}$ and $k' = \frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k}$ be two integers. Assuming randomized ETH, there is a constant $\lambda \in (0, 1)$ such that there is no randomized $N^{\lambda k}$ time algorithm that can solve (k, k')-Set-Cover on N vertices with high probability.*

2.3 Concept Classes

In Appendix A we give the definition of *monotone* function, *literal*, *term*,*clause*, *monotone term*, *DNF*, *monotone DNF*. Then the classes, size-s Monotone DNF, size s-Monotone DT and Monotone k-Junta.

The *size* of a term T, $|T|$, is the number of literals in the term T. The *size* $|F|$ of a DNF (resp. CNF) F is the number of terms (resp. clauses) in F.

It is well known that

$$\text{Monotone } (\log s)\text{-Junta} \subset \text{size-}s \text{ Monotone DT} \subset \text{size-}s \text{ Monotone DNF} . \qquad (1)$$

2.4 Functions and Distributions

For any set R, we define $\mathcal{U}(R)$ to be the uniform distribution over R. For a distribution $\mathcal{D}$ over $\{0,1\}^n$ and two Boolean functions f and g, we define $\mathrm{dist}_{\mathcal{D}}(f, g) = \Pr_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x}) \ne g(\boldsymbol{x})]$. Here, bold letters denote random variables. If $\mathrm{dist}_{\mathcal{D}}(f, g) = 0$, then we say that $f = g$ *over* $\mathcal{D}$. For a class of functions C, we say that f is in C over $\mathcal{D}$ (or just, is C over $\mathcal{D}$) if there is a function $g \in C$ such that $f = g$ over $\mathcal{D}$.

▶ **Definition 3** ($\Gamma^{\mathcal{S}}$ and $\mathcal{D}^{\mathcal{S}}$). *Let $\mathcal{S} = (S, U, E)$ be a set cover instance with $S = [n]$. Recall that we identify each element $u \in U$ with the vector $(u_1, \ldots, u_n) \in \{0,1\}^n$ where $u_i = 0$ if and only if $(i, u) \in E$. We define the partial function $\Gamma^{\mathcal{S}} : \{0,1\}^n \to \{0,1\}$ where $\Gamma^{\mathcal{S}}(x) = 0$ if $x \in U$ and $\Gamma^{\mathcal{S}}(1^n) = 1$. We define the distribution $\mathcal{D}^{\mathcal{S}}$ over $\{0,1\}^n$ where $\mathcal{D}^{\mathcal{S}}(x) = 1/2$ if $x = 1^n$, $\mathcal{D}^{\mathcal{S}}(x) = 1/(2|U|)$ if $x \in U$, and $\mathcal{D}^{\mathcal{S}}(x) = 0$ otherwise. We will remove the superscript $\mathcal{S}$ when it is clear from the context and write Γ and $\mathcal{D}$.*

▶ **Fact 1.** *We have*

1. *$C \subseteq S$ is a set cover of $\mathcal{S} = (S, U, E)$, if and only if $\Gamma(x) = \bigwedge_{i\in C} x_i$ over $\mathcal{D}$.*
2. *In particular, If T is a monotone term of size $|T| < \mathrm{opt}(\mathcal{S})$, then there is $u \in U$ such that $T(u) = 1$.*

Proof. Let C be a set cover of $\mathcal{S}$. First, we have $\Gamma(1^n) = 1$. Now, since C is a set cover, every vertex $u \in U$ is adjacent to some vertex in C. This is equivalent to: for every assignment $u \in U$, there is $i \in C$ such that $u_i = 0$. Therefore, $\wedge_{i\in C} u_i = 0$ for all $u \in U$. Thus, $\Gamma(x) = \bigwedge_{i\in C} x_i$ over $\mathcal{D}$.

The other direction can be easily seen by tracing backward in the above proof. ◀

For an odd ℓ, define $\Delta^0 = \{a \in \{0,1\}^\ell | \mathrm{wt}(a) = \lfloor \ell/2 \rfloor\}$ and $\Delta^1 = \{a \in \{0,1\}^\ell | \mathrm{wt}(a) = \lceil \ell/2 \rceil\}$, where $\mathrm{wt}(a)$ is the Hamming weight of a. Notice that $|\Delta^0| = |\Delta^1| = \binom{\ell}{\lfloor \ell/2 \rfloor}$.

▶ **Definition 4** **(Γ_ℓ, $\mathcal{D}_\ell$, Δ^0_n and Δ^1_n).** *For an odd ℓ, define $\Delta^1_n = (\Delta^1)^n$ and*[7] *$\Delta^0_n := \cup_{u\in U} \prod_{i=1}^n \Delta^{u_i} = \cup_{u\in U}(\Delta^{u_1} \times \Delta^{u_2} \times \cdots \times \Delta^{u_n})$. Define the distribution $\mathcal{D}_\ell : (\{0,1\}^\ell)^n \to [0,1]$ to be $\mathcal{D}_\ell(y) = 1/(2|\Delta^1_n|) = 1/(2|\Delta^1|^n)$ if $y \in \Delta^1_n$, $\mathcal{D}_\ell(y) = 1/(2|\Delta^0_n|) = 1/(2|U| \cdot |\Delta^0|^n)$ if $y \in \Delta^0_n$, and $\mathcal{D}_\ell(y) = 0$ otherwise. We define the partial function Γ_ℓ over the support $\Delta^0_n \cup \Delta^1_n$ of $\mathcal{D}_\ell$ to be 1 if $y \in \Delta^1_n$ and 0 if $y \in \Delta^0_n$.*

We note here that the distribution $\mathcal{D}_\ell$ is well-defined. This is because: First, the sum of the distribution of the points in Δ^1_n is $1/2$. Second, for two different $u, u' \in U$, we have that $\prod_{i=1}^n \Delta^{u_i}$ and $\prod_{i=1}^n \Delta^{u'_i}$ are disjoint sets. Therefore, $|\Delta^0_n| = |U| \cdot |\Delta^0|^n$, and therefore, the sum of the distribution of all the points in Δ^0_n is half. In particular,

▶ **Fact 2.** *We have* $\Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[\Gamma_\ell(\boldsymbol{y}) = 1] = \Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[\Gamma_\ell(\boldsymbol{y}) = 0] = \Pr_{\mathcal{D}_\ell}[\Delta^1_n] = \Pr_{\mathcal{D}_\ell}[\Delta^0_n] = \frac{1}{2}$.

For $y \in (\{0,1\}^\ell)^n$, we write $y = (y_1, \ldots, y_n)$, where $y_j = (y_{j,1}, y_{j,2}, \ldots, y_{j,\ell}) \in \{0,1\}^\ell$. Let $(\textsc{Majority}(y_i))_{i\in[n]} = (\textsc{Majority}(y_1), \ldots, \textsc{Majority}(y_n))$ where $\textsc{Majority}$ is the majority function.

▶ **Fact 3.** *If $C \subseteq S$ is a set cover of $\mathcal{S}$, then $\Gamma_\ell(y) = \Gamma((\textsc{Majority}(y_i))_{i\in[n]}) = \bigwedge_{i\in C}$ $\textsc{Majority}(y_i)$ over $\mathcal{D}$. In particular, Γ_ℓ is $\textsc{Monotone}$ $\mathrm{opt}(\mathcal{S})\ell$-$\textsc{Junta}$ over $\mathcal{D}$.*

Proof. First notice that $\textsc{Majority}(x) = 1$ if $x \in \Delta^1$ and $\textsc{Majority}(x) = 0$ if $x \in \Delta^0$. Therefore, for $x \in \Delta^\xi$, $\xi \in \{0,1\}$ we have $\textsc{Majority}(x) = \xi$.

For $y \in \Delta^1_n = (\Delta^1)^n$, $(\textsc{Majority}(y_i))_{i\in[n]} = 1^n$ and $\Gamma_\ell(y) = 1 = \Gamma(1^n)$.

For $y \in \Delta^0_n = \cup_{u\in U}(\Delta^{u_1} \times \Delta^{u_2} \times \cdots \times \Delta^{u_n})$, there is u such that $y \in \Delta^{u_1} \times \Delta^{u_2} \times \cdots \times \Delta^{u_n}$. Then, $(\textsc{Majority}(y_i))_{i\in[n]} = u$ and $\Gamma_\ell(y) = \Gamma((\textsc{Majority}(y_i))_{i\in[n]}) = \Gamma(u) = 0$. ◀

For $t \in [\ell], \xi \in \{0,1\}$ and $u \in \{0,1\}^\ell$, we define $u^{t\leftarrow\xi} \in \{0,1\}^\ell$ the vector that satisfies

$$u_i^{t\leftarrow\xi} = \begin{cases} u_i & i \neq t \\ \xi & i = t \end{cases}.$$

Let $z \in (\{0,1\}^\ell)^n$. For $j \in [\ell]^n$ and $a \in \{0,1\}^n$, define $z^{j\leftarrow a} = (z_1^{j_1\leftarrow a_1}, \ldots, z_n^{j_n\leftarrow a_n})$. For a set $V \subseteq \{0,1\}^n$, we define $z^{j\leftarrow V} = \{z^{j\leftarrow v} | v \in V\}$.

We define $\mathrm{one}(z) = \prod_{i=1}^n \{m_i | z_{i,m_i} = 1\} = \{m_1 | z_{1,m_1} = 1\} \times \cdots \times \{m_n | z_{n,m_n} = 1\}$.

[7] Here $\Delta^\xi = \Delta^0$ if $\xi = 0$ and Δ^1 if $\xi = 1$.

▶ **Fact 4.** *Let $w \in \Delta_n^1$, $j \in \mathrm{one}(w)$, and T be a term that satisfies $T(w) = 1$. Then*

1. $w^{j\leftarrow U} \subseteq \Delta_n^0$.
2. $|w^{j\leftarrow U}| = |U|$.
3. *If $T^j(y_{1,j_1}, \dots, y_{n,j_n})$ is the conjunction of all the variables that appear in T of the form y_{i,j_i}, then $T(w^{j\leftarrow a}) = T^j(a)$.*

Proof. We first prove item 1. Let $u \in U$ and i be any integer in $[n]$. Since $w \in \Delta_n^1$, we have $w_i \in \Delta^1$. Since $j \in one(w)$, we have $w_{i,j_i} = 1$. Therefore, $w_i^{j_i \leftarrow u_i} \in \Delta^{u_i}$ for all $i \in [n]$ and $w^{j\leftarrow u} \in \prod_{i=1}^n \Delta^{u_i}$. Thus, $w^{j\leftarrow u} \in \Delta_n^0$ for all $u \in U$.

To prove item 2, let u, u' be two distinct elements of U. There is i such that $u_i \neq u'_i$. Therefore $w_i^{j_i\leftarrow u_i} \neq w_i^{j_i \leftarrow u'_i}$ and $w^{j\leftarrow u} \neq w^{j\leftarrow u'}$.

We now prove item 3. Let T' be the conjunction of all the variables that appear in T that are not of the form y_{i,j_i}. Then $T = T' \wedge T^j$. Since $T(w) = 1$, we have $T'(w) = 1$. Since the entries of $w^{j\leftarrow a}$ are equal to those in w on all the variables that are not of the form y_{i,j_i}, we have $T'(w^{j\leftarrow a}) = 1$. Therefore, $T(w^{j\leftarrow a}) = T'(w^{j\leftarrow a}) \wedge T^j(w_{1,j_1}^{j\leftarrow a}, \dots, w_{n,j_n}^{j\leftarrow a}) = T^j(a)$. ◀

We now give a different way of sampling according to the distribution $\mathcal{D}_\ell$. The proof is in Appendix B.

▶ **Fact 5.** *Let $\mathcal{S}$ be a SET-COVER instance. The following is an equivalent way of sampling from $\mathcal{D}_\ell$.*

1. *Draw $\xi \in \{0,1\}$ u.a.r.*[8]
2. *Draw $w \in \Delta_n^1$ u.a.r.*
3. *If $\xi = 1$ then output $y = w$.*
4. *If $\xi = 0$ then*
   a. *draw $j \in \mathrm{one}(w)$ u.a.r.*
   b. *draw $v \in w^{j\leftarrow U}$ u.a.r.*
   c. *output $y = v$.*

In particular, for any event X,

$$\Pr_{\boldsymbol{y}\sim\mathcal{U}(\Delta_n^0)}[X] = \Pr_{\boldsymbol{w}\sim\mathcal{U}(\Delta_n^1), \boldsymbol{j}\sim\mathcal{U}(\mathrm{one}(\boldsymbol{w})), \boldsymbol{y}\sim\mathcal{U}(\boldsymbol{w}^{\boldsymbol{j}\leftarrow U})}[X].$$

3 Main Lemma

In this section, we prove

▶ **Lemma 5.** *Let $\mathcal{S} = (S, U, E)$ be a set cover instance. If $F : (\{0,1\}^\ell)^n \to \{0,1\}$ is a DNF of size $|F| < 2^{\mathrm{opt}(\mathcal{S})\ell/20}$, then $\mathrm{dist}_{\mathcal{D}_\ell}(F, \Gamma_\ell) \geq 1/(8|U|) - 2^{-\mathrm{opt}(\mathcal{S})\ell/20}$.*

Note that Lemma 5 is used to prove Theorem 1 and 2. To prove Theorem 3, we will need Lemma 11, a stronger version of Lemma 5.

To prove the lemma, we first establish some results.

For a term T, let $T_\mathcal{M}$ be the conjunction of all the unnegated variables in T. We define the *monotone size* of T to be $|T_\mathcal{M}|$. The proof of the following Claim is in Appendix B.

▷ Claim 6. Let $\mathcal{S} = (S, U, E)$ be a set cover instance and $\ell \geq 5$. If $F : (\{0,1\}^\ell)^n \to \{0,1\}$ is a DNF of size $|F| < 2^{\mathrm{opt}(\mathcal{S})\ell/20}$, then there is a DNF, F', of size $|F'| \leq 2^{\mathrm{opt}(\mathcal{S})\ell/20}$ with terms of monotone size at most $\mathrm{opt}(\mathcal{S})\ell/5$ such that $\mathrm{dist}_{\mathcal{D}_\ell}(\Gamma_\ell, F') \leq \mathrm{dist}_{\mathcal{D}_\ell}(\Gamma_\ell, F) + 2^{-\mathrm{opt}(\mathcal{S})\ell/20}$.

[8] Uniformly at random.

We now prove

▷ Claim 7. Let $z\in\Delta_n^1$. Let F be a DNF with terms of monotone size at most $\lceil\ell/2\rceil(\mathrm{opt}(\mathcal{S})-1)/2$ that satisfies $F(z)=1$. Then

$$\Pr_{\boldsymbol{j}\sim\mathcal{U}(\mathrm{one}(z)),\boldsymbol{y}\sim\mathcal{U}(z^{\boldsymbol{j}\leftarrow U})}[F(\boldsymbol{y})=1]\ge\frac{1}{2|U|}.$$

Proof. Since $F(z)=1$, there is a term T in F that satisfies $T(z)=1$. Let $Y_0=\{y_{i,m}|z_{i,m}=0\}$ and $Y_1=\{y_{i,m}|z_{i,m}=1\}$. Since $T(z)=1$, every variable in Y_0 that appears in T must be negated, and every variable in Y_1 that appears in T must be unnegated. For $j\in\mathrm{one}(z)$, define $q(j)$ to be the number of variables in $\{y_{1,j_1},\ldots,y_{n,j_n}\}$ that appear in $T(y)$. All those variables appear unnegated in T because $j\in\mathrm{one}(z)$. Recall that $T_{\mathcal{M}}$ is the conjunction of all unnegated variables in T. Then $|T_{\mathcal{M}}|\le\lceil\ell/2\rceil(\mathrm{opt}(\mathcal{S})-1)/2$. Each variable in $T_{\mathcal{M}}$ contributes $\lceil\ell/2\rceil^{n-1}$ to the sum $\sum_{j\in one(z)}q(j)$ and $|\mathrm{one}(z)|=\lceil\ell/2\rceil^n$. Therefore,

$$\mathop{\mathbb{E}}_{\boldsymbol{j}\sim\mathcal{U}(\mathrm{one}(z))}[q(\boldsymbol{j})]=\frac{|T_{\mathcal{M}}|}{\lceil\ell/2\rceil}\le\frac{\mathrm{opt}(\mathcal{S})-1}{2}.$$

By Markov's bound, at least half the elements $j\in\mathrm{one}(z)$ satisfies $q(j)\le\mathrm{opt}(\mathcal{S})-1$. Let $J=\{j\in\mathrm{one}(z)|q(j)<\mathrm{opt}(\mathcal{S})\}$. Then $\Pr_{\boldsymbol{j}\sim\mathcal{U}(\mathrm{one}(z))}[\boldsymbol{j}\in J]\ge 1/2$. Consider $j\in J$ and let T^j be the conjunction of all the variables that appear in T of the form y_{i,j_i}. Then $|T^j|=q(j)\le\mathrm{opt}(\mathcal{S})-1$. By Fact 1, there is $u\in U$ such that $T^j(u)=1$. By Fact 4, we have $T(z^{j\leftarrow u})=T^j(u)=1$. Then $F(z^{j\leftarrow u})=1$. Since by item 1 in Fact 4, $|z^{j\leftarrow U}|=|U|$, we have

$$\Pr_{\boldsymbol{j}\sim\mathcal{U}(one(z)),\boldsymbol{y}\sim\mathcal{U}(z^{\boldsymbol{j}\leftarrow U})}[F(\boldsymbol{y})=1|\boldsymbol{j}\in J]\ge\frac{1}{|U|}.$$

Therefore,

$$\Pr_{\boldsymbol{j}\sim\mathcal{U}(\mathrm{one}(z)),\boldsymbol{y}\sim\mathcal{U}(z^{\boldsymbol{j}\leftarrow U})}[F(\boldsymbol{y})=1]\ge\Pr_{\boldsymbol{j}\sim\mathcal{U}(\mathrm{one}(z))}[\boldsymbol{j}\in J]\times$$
$$\Pr_{\boldsymbol{j}\sim\mathcal{U}(\mathrm{one}(z)),\boldsymbol{y}\sim\mathcal{U}(z^{\boldsymbol{j}\leftarrow U})}[F(\boldsymbol{y})=1|\boldsymbol{j}\in J]\ge\frac{1}{2|U|}.$$ ◁

We are now ready to prove Lemma 5

Proof. By Claim 6, there is a DNF, F', of size $|F'|\le 2^{\mathrm{opt}(\mathcal{S})\ell/20}$ with terms of monotone size at most $\mathrm{opt}(\mathcal{S})\ell/5$ such that $\mathrm{dist}_{\mathcal{D}_\ell}(\Gamma_\ell,F')\le\mathrm{dist}_{\mathcal{D}_\ell}(\Gamma_\ell,F)+2^{-\mathrm{opt}(\mathcal{S})\ell/20}$. Therefore, it is enough to prove that $\mathrm{dist}_{\mathcal{D}_\ell}(\Gamma_\ell,F')\ge 1/(8|U|)$.

If $\Pr_{\boldsymbol{y}\sim\mathcal{U}(\Delta_n^1)}[F'(\boldsymbol{y})\ne 1]\ge 1/(4|U|)$, then by Fact 2, for the event $Y(\boldsymbol{y})=[\Gamma_\ell(\boldsymbol{y})=1]$, we have

$$\mathrm{dist}_{\mathcal{D}_\ell}(\Gamma_\ell,F')\ge\Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[\Gamma_\ell(\boldsymbol{y})\ne F'(\boldsymbol{y})|Y(\boldsymbol{y})]\Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[Y(\boldsymbol{y})]=\frac{1}{2}\Pr_{\boldsymbol{y}\sim\mathcal{U}(\Delta_n^1)}[F'(\boldsymbol{y})\ne 1]\ge\frac{1}{8|U|}.$$

If $\Pr_{\boldsymbol{y}\sim\mathcal{U}(\Delta_n^1)}[F'(\boldsymbol{y}) \neq 1] < 1/(4|U|)$, then by Fact 2 and 5, and Claim 7,

$$\begin{aligned}
\mathrm{dist}_{\mathcal{D}_\ell}(\Gamma_\ell, F') \geq & \Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[\Gamma_\ell(\boldsymbol{y}) \neq F'(\boldsymbol{y}) | \Gamma_\ell(\boldsymbol{y}) = 0] \cdot \Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[\Gamma_\ell(\boldsymbol{y}) = 0] \\
=& \frac{1}{2} \Pr_{\boldsymbol{y}\sim\mathcal{U}(\Delta_n^0)}[F'(\boldsymbol{y}) = 1] \\
=& \frac{1}{2} \Pr_{\boldsymbol{z}\sim\mathcal{U}(\Delta_n^1), \boldsymbol{j}\sim\mathcal{U}(\mathrm{one}(\boldsymbol{z})), \boldsymbol{y}\sim\mathcal{U}(\boldsymbol{z}^{\boldsymbol{j}\leftarrow U})}[F'(\boldsymbol{y}) = 1] \\
\geq & \frac{1}{2} \Pr_{\boldsymbol{z}\sim\mathcal{U}(\Delta_n^1), \boldsymbol{j}\sim\mathcal{U}(\mathrm{one}(\boldsymbol{z})), \boldsymbol{y}\sim\mathcal{U}(\boldsymbol{z}^{\boldsymbol{j}\leftarrow U})}[F'(\boldsymbol{y}) = 1 | F'(\boldsymbol{z}) = 1] \times \\
& \Pr_{\boldsymbol{z}\sim\mathcal{U}(\Delta_n^1)}[F'(\boldsymbol{z}) = 1] \\
\geq & \frac{1}{2}\frac{1}{2|U|}\left(1 - \frac{1}{4|U|}\right) \geq \frac{1}{8|U|}.
\end{aligned}$$

◀

4 Superpolynomial Lower Bound

In this section, we prove the first results of the paper. First, we prove the following result for Monotone $(\log n)$-Junta.

▶ **Lemma 8.** *Assuming randomized ETH, there is a constant c such that any PAC learning algorithm for n-variable* Monotone $(\log n)$-Junta *by DNF with $\epsilon = 1/(16n)$ must take at least $n^{c\frac{\log\log n}{\log\log\log n}}$ time.*

The lower bound holds, even if the learner knows the distribution, can draw a sample according to the distribution in polynomial time and can compute the target on all the points of the support of the distribution in polynomial time.

Proof. Consider the constant λ in Lemma 2. Let $c = \min(1/40, \lambda/4)$. Suppose there is a PAC learning algorithm $\mathcal{A}$ for Monotone $(\log n)$-Junta by DNF with $\epsilon = 1/(16n)$ that runs in time $n^{c\frac{\log\log n}{\log\log\log n}}$. We show that there is k such that for

$$k' = \frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k},$$

(k, k')-Set-Cover can be solved in time $N^{4ck} \leq N^{\lambda k}$. By Lemma 2, the result then follows.

Let $\mathcal{S} = (S, U, E)$ be an N-vertex (k, k')-Set-Cover instance where

$$k = \frac{1}{2}\frac{\log\log N}{\log\log\log N} \text{ and } k' = \frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k}.$$

Let $\ell = \frac{\log N}{k}$ and consider Γ_ℓ and $\mathcal{D}_\ell$.

Consider the following algorithm $\mathcal{B}$

1. Input $\mathcal{S} = (S, U, E)$ an instance for (k, k')-Set-Cover .
2. Construct Γ_ℓ and $\mathcal{D}_\ell$.
3. Run $\mathcal{A}$ using Γ_ℓ and $\mathcal{D}_\ell$. If it runs more than N^{4ck} steps, then output No .
4. Let F be the output DNF.
5. Estimate $\eta = \mathrm{dist}_{\mathcal{D}_\ell}(F, \Gamma_\ell)$.
6. If $\eta \leq \frac{1}{16N}$, output Yes , otherwise output No .

The running time of this algorithm is $N^{4ck} \leq N^{\lambda k}$. Therefore, it is enough to prove the following

▷ Claim 9. Algorithm $\mathcal{B}$ solves (k,k')-SET-COVER .

Proof. YES case: Let $\mathcal{S}=(S,U,E)$ be a (k,k')-SET-COVER instance and $\mathrm{opt}(\mathcal{S})\le k$. Then, $\mathrm{opt}(\mathcal{S})\cdot\ell\le k\ell=\log N$, and by Fact 3, Γ_ℓ is MONOTONE $\log N$-JUNTA. Therefore, w.h.p., algorithm $\mathcal{A}$ learns Γ_ℓ and outputs a DNF that is $\eta=1/(16N)$ close to the target with respect to $\mathcal{D}_\ell$. Since $\mathcal{B}$ terminates $\mathcal{A}$ after N^{4ck} time, we only need to prove that $\mathcal{A}$ runs at most N^{4ck} time.

The running time of $\mathcal{A}$ is $N^{c\frac{\log\log N}{\log\log\log N}}<N^{4ck}$.

NO Case: Let $\mathcal{S}=(S,U,E)$ be a (k,k')-SET-COVER instance and $\mathrm{opt}(\mathcal{S})>k'$. By Lemma 5, any DNF, F, of size $|F|<2^{k'\ell/20}$ satisfies $\mathrm{dist}_{\mathcal{D}_\ell}(F,\Gamma_\ell)\ge 1/(8|U|)-2^{-k'\ell/20}$. First, we have

$$(2k)^{2k}=\left(\frac{\log\log N}{\log\log\log N}\right)^{\frac{\log\log N}{\log\log\log N}}<\frac{\log N}{\log\log N}.$$

Therefore, since $c\le 1/40$,

$$k'=\frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k}>\frac{1}{2}(2k)^2>80ck^2.$$

So $k'\ell/20>(k\ell)(4ck)$ and $2^{k'\ell/20}>(2^{k\ell})^{4ck}=N^{4ck}$. Now since the algorithm runs in time N^{4ck}, it cannot output a DNF F of size more than $N^{4ck}<2^{k'\ell/20}$, and by Lemma 5,

$$\mathrm{dist}_{\mathcal{D}_\ell}(F,\Gamma_\ell)\ge\frac{1}{8|U|}-\frac{1}{N^{4ck}}\ge\frac{1}{9N}.$$

So it either runs more than N^{4ck} steps and then outputs NO in step 3 or outputs a DNF with an error greater than $1/(9N)>1/(16N)$ and outputs NO in step 6. ◁

Notice that the learning algorithm knows Γ_ℓ and $\mathcal{D}_\ell$. It is also clear from the definition of Γ_ℓ and $\mathcal{D}_\ell$ that the learning algorithm can draw a sample according to the distribution $\mathcal{D}_\ell$ in polynomial time and can compute the target Γ_ℓ on all the points of the support of the distribution in polynomial time. ◀

We now prove

▶ **Theorem 1.** *Assuming randomized ETH, there is a constant c such that any PAC learning algorithm for n-variable size-s MONOTONE DT and size-s MONOTONE DNF by DNF with $\epsilon=1/(16n)$ must take at least*

$$n^{c\frac{\log\log s}{\log\log\log s}}$$

time.

The lower bound holds, even if the learner knows the distribution, can draw a sample according to the distribution in polynomial time and can compute the target on all the points of the support of the distribution in polynomial time.

Proof. By Lemma 8, assuming randomized ETH, there is a constant c such that any PAC learning algorithm for n-variable MONOTONE $(\log n)$-JUNTA by DNF with $\epsilon=1/(16n)$ runs in time

$$n^{c\frac{\log\log n}{\log\log\log n}}.$$

Now by (1) and since $s=n$, the result follows. ◀

5 Tight Bound Assuming some Conjecture

A plausible conjecture on the hardness of SET-COVER is the following.

▶ **Conjecture 1** ([10]). *There are constants $\alpha, \beta, \lambda \in (0, 1)$ such that, for $k < N^\alpha$, there is no randomized $N^{\lambda k}$ time algorithm that can solve $(k, (1-\beta)\cdot k \ln N)$-SET-COVER on N vertices with high probability.*

We now prove

▶ **Theorem 2.** *Assuming Conjecture 1, there is a constant c such that any PAC learning algorithm for n-variable MONOTONE $(\log s)$-JUNTA, size-s MONOTONE DT and size-s MONOTONE DNF by DNF with $\epsilon = 1/(16n)$ must take at least*

$$n^{c \log s}$$

time.

The lower bound holds, even if the learner knows the distribution, can draw a sample according to the distribution in polynomial time and can compute the target on all the points of the support of the distribution in polynomial time.

Proof. We give the proof for MONOTONE $(\log s)$-JUNTA. As in the proof of Theorem 1, the result then follows for the other classes.

Consider the constants α, β and λ in Conjecture 1. Let $c = \min(\lambda/10, (1-\beta)/(20 \log e))$. Suppose there is a PAC learning algorithm $\mathcal{A}$ for MONOTONE $(\log s)$-JUNTA by DNF with $\epsilon = 1/(16n)$ that runs in time $n^{c \log s}$. We show that there is $k < N^\alpha$, $k = \omega(1)$, such that (k, k')-SET-COVER can be solved in time $N^{\lambda k}$ where $k' = (1-\beta)k \ln N$. By Conjecture 1, the result then follows.

Consider the following algorithm $\mathcal{B}$

1. Input $\mathcal{S} = (S, U, E)$ an instance for (k, k')-SET-COVER .
2. Construct Γ_5 and $\mathcal{D}_5$.
3. Run $\mathcal{A}$ using Γ_5 and $\mathcal{D}_5$ with $s = 2^{5k}$. If it runs more than N^{5ck} steps, then output NO .
4. Let F be the output DNF.
5. Estimate $\eta = \mathrm{dist}_{\mathcal{D}_5}(F, \Gamma_5)$.
6. If $\eta \le \frac{1}{16N}$, output YES , otherwise output NO .

Since $c < \lambda/10$, the running time of this algorithm is $N^{5ck} < N^{\lambda k}$. Therefore, it is enough to prove the following

▷ Claim 10. Algorithm $\mathcal{B}$ solves (k, k')-SET-COVER .

Proof. YES case: Let $\mathcal{S} = (S, U, E)$ be a (k, k')-SET-COVER instance and $\mathrm{opt}(\mathcal{S}) \le k$. Then, $5 \cdot \mathrm{opt}(\mathcal{S}) \le 5k = \log s$, and by Fact 3, Γ_5 is MONOTONE $\log s$-JUNTA. Therefore, w.h.p., algorithm $\mathcal{A}$ learns Γ_5 and outputs a DNF that is $\eta = 1/(16N)$ close to the target with respect to $\mathcal{D}_5$. Since $\mathcal{B}$ terminates $\mathcal{A}$ after N^{5ck} time, we only need to prove that $\mathcal{A}$ runs at most N^{5ck} time.

The running time of $\mathcal{A}$ is

$$n^{c \log s} \le N^{5ck}.$$

No Case: Let $\mathcal{S} = (S, U, E)$ be a (k, k')-SET-COVER instance and $\mathrm{opt}(\mathcal{S}) > k' = (1-\beta)k \ln N$. By Lemma 5, any DNF, F, of size $|F| < 2^{k'/4}$ satisfies $\mathrm{dist}_{\mathcal{D}_5}(F, \Gamma_5) \ge 1/(8|U|) - 2^{-k'/4}$. Since, $c < (1-\beta)/(20 \log e)$,

$$2^{k'/4} = 2^{\frac{(1-\beta)k \ln N}{4}} = N^{\frac{(1-\beta)k}{4 \log e}} > N^{5ck},$$

any DNF, F, that the learning outputs satisfies

$$\mathrm{dist}_{\mathcal{D}_5}(F,\Gamma_5) \geq \frac{1}{8|U|} - 2^{-k'/4} \geq \frac{1}{8N} - \frac{1}{N^{5ck}} \geq \frac{1}{9N}.$$

Therefore, with high probability the algorithm answer No . ◁

◀

6 Strictly Proper Learning

In this section, we prove

▶ **Theorem 3.** *Assuming randomized ETH, there is a constant c such that any PAC learning algorithm for n-variable* Monotone $(\log s)$-Junta*, size-s* Monotone *DT and size-s* Monotone *DNF by size-s DNF with $\epsilon = 1/(16n)$ must take at least*

$$n^{c\log s}$$

time.

The lower bound holds, even if the learner knows the distribution, can draw a sample according to the distribution in polynomial time and compute the target on all the points of the support of the distribution in polynomial time.

We first prove the following stronger version of Lemma 5

▶ **Lemma 11.** *Let $\mathcal{S} = (S, U, E)$ be a set cover instance, and let $\ell \geq 5$. If $F : (\{0,1\}^\ell)^n \to \{0,1\}$ is a DNF of size $|F| < 2^{\mathrm{opt}(\mathcal{S})\ell/16}$, then $\mathrm{dist}_{\mathcal{D}_\ell}(F,\Gamma_\ell) \geq 1/(8|U|)$.*

To prove this lemma, we will give some more results.

Recall that, for a term T, $T_{\mathcal{M}}$ is the conjunction of all the unnegated variables in T. We define the *monotone size* of T to be $|T_{\mathcal{M}}|$. For a DNF $F = T_1 \vee T_2 \vee \cdots \vee T_s$ and $z \in (\{0,1\}^\ell)^n$, we define the *monotone width* of z in F as

$$\mathrm{mwidth}_F(z) := \left\{ \begin{array}{ll} \min_{T_i(z)=1} |(T_i)_{\mathcal{M}}| & F(z)=1 \\ 0 & F(z)=0 \end{array} \right. .$$

We define $F^{-1}(1) = \{z | F(z) = 1\}$ and

$$\Omega = \Delta_n^1 \cap F^{-1}(1).$$

▷ Claim 12. Let F be a DNF with

$$\mathop{\mathbb{E}}_{\boldsymbol{z}\sim\mathcal{U}(\Omega)}[\mathrm{mwidth}_F(\boldsymbol{z})] \leq \mathrm{opt}(S)\cdot \ell/4.$$

Then

$$\Pr_{\boldsymbol{z}\sim\mathcal{U}(\Omega),\boldsymbol{j}\sim\mathcal{U}(\mathrm{one}(\boldsymbol{z})),\boldsymbol{y}\sim\mathcal{U}(\boldsymbol{z}^{\boldsymbol{j}\leftarrow U})}[F(\boldsymbol{y})=1] \geq \frac{1}{2|U|}.$$

Proof. Let $z \in \Omega$. Then $F(z) = 1$ and $z \in \Delta_n^1$. Let T^z be the term in F with $|T^z_{\mathcal{M}}| = \mathrm{mwidth}_F(z)$ that satisfies $T^z(z) = 1$. Let $Y_0 = \{y_{i,m} | z_{i,m} = 0\}$ and $Y_1 = \{y_{i,m} | z_{i,m} = 1\}$. Since $T^z(z) = 1$, every variable in Y_0 that appears in T^z must be negated, and every variable in Y_1 that appears in T^z must be unnegated. For $j \in \text{one}(z)$, define $q_z(j)$ to be the number of variables in $\{y_{1,j_1}, \ldots, y_{n,j_n}\}$ that appear in $T^z(y)$. All those variables appear unnegated in T because $j \in \text{one}(z)$. Each variable in $T^z_{\mathcal{M}}$ contributes $\lceil \ell/2 \rceil^{n-1}$ to the sum $\sum_{j \in one(z)} q_z(j)$ and $|\text{one}(z)| = \lceil \ell/2 \rceil^n$. Therefore,

$$\mathop{\mathbb{E}}_{\boldsymbol{j} \sim \mathcal{U}(\text{one}(z))}[q_z(\boldsymbol{j})] = \frac{|T^z_{\mathcal{M}}|}{\lceil \ell/2 \rceil} = \frac{\text{mwidth}_F(z)}{\lceil \ell/2 \rceil}.$$

Now,

$$\begin{aligned} \mathop{\mathbb{E}}_{\boldsymbol{z} \sim \mathcal{U}(\Omega), \boldsymbol{j} \sim \mathcal{U}(\text{one}(\boldsymbol{z}))}[q_{\boldsymbol{z}}(\boldsymbol{j})] &= \frac{\mathop{\mathbb{E}}_{\boldsymbol{z} \sim \mathcal{U}(\Omega)}[\text{mwidth}_F(\boldsymbol{z})]}{\lceil \ell/2 \rceil} \\ &\le \frac{\text{opt}(\mathcal{S})}{2}. \end{aligned}$$

By Markov's bound,

$$\Pr_{\boldsymbol{z} \sim \mathcal{U}(\Omega), \boldsymbol{j} \sim \mathcal{U}(\text{one}(\boldsymbol{z}))}[q_{\boldsymbol{z}}(\boldsymbol{j}) < \text{opt}(\mathcal{S})] \ge \frac{1}{2}.$$

Suppose for some $z \in \Omega$ and $j \in \text{one}(z)$, we have $q_z(j) < \text{opt}(\mathcal{S})$. Let T^j be the conjunction of all the variables that appear in $T^z_{\mathcal{M}}$ of the form y_{i,j_i}. Then $|T^j| = q_z(j) < \text{opt}(\mathcal{S})$. By Fact 1, there is $u \in U$ such that $T^j(u) = 1$. By Fact 4, we have $T^z(z^{j \leftarrow u}) = T^j(u) = 1$. Then $F(z^{j \leftarrow u}) = 1$. Since by item 1 in Fact 4, $|z^{j \leftarrow U}| = |U|$, we have

$$\Pr_{\boldsymbol{z} \sim \mathcal{U}(\Omega), \boldsymbol{j} \sim \mathcal{U}(one(\boldsymbol{z})), \boldsymbol{y} \sim \mathcal{U}(\boldsymbol{z}^{\boldsymbol{j} \leftarrow U})}[F(\boldsymbol{y}) = 1 | q_{\boldsymbol{z}}(\boldsymbol{j}) < \text{opt}(\mathcal{S})] \ge \frac{1}{|U|}.$$

Therefore, for $\mathcal{D}'(\boldsymbol{y}) = [\boldsymbol{z} \sim \mathcal{U}(\Omega), \boldsymbol{j} \sim \mathcal{U}(\text{one}(\boldsymbol{z})), \boldsymbol{y} \sim \mathcal{U}(\boldsymbol{z}^{\boldsymbol{j} \leftarrow U})]$

$$\begin{aligned} \Pr_{\mathcal{D}'}[F(\boldsymbol{y}) = 1] \ge & \Pr_{\boldsymbol{z} \sim \mathcal{U}(\Omega), \boldsymbol{j} \sim \mathcal{U}(\text{one}(\boldsymbol{z}))}[q_{\boldsymbol{z}}(\boldsymbol{j}) < \text{opt}(\mathcal{S})] \cdot \\ & \Pr_{\boldsymbol{z} \sim \mathcal{U}(\Omega), \boldsymbol{j} \sim \mathcal{U}(one(\boldsymbol{z})), \boldsymbol{y} \sim \mathcal{U}(\boldsymbol{z}^{\boldsymbol{j} \leftarrow U})}[F(\boldsymbol{y}) = 1 | q_{\boldsymbol{z}}(\boldsymbol{j}) < \text{opt}(\mathcal{S})] \\ \ge & \frac{1}{2|U|}. \end{aligned}$$

◁

▷ Claim 13. Let $\mathcal{S} = (S, U, E)$ be a set cover instance, and let $\ell \ge 5$. If $F : (\{0,1\}^\ell)^n \to \{0,1\}$ is a DNF and $\mathop{\mathbb{E}}_{\boldsymbol{z} \sim \mathcal{U}(\Omega)}[\text{mwidth}_F(\boldsymbol{z})] \le \text{opt}(\mathcal{S})\ell/4$, then $\text{dist}_{\mathcal{D}_\ell}(F, \Gamma_\ell) \ge 1/(8|U|)$.

Proof. If $\Pr_{\boldsymbol{y} \sim \mathcal{U}(\Delta^1_n)}[F(\boldsymbol{y}) \ne 1] \ge 1/(4|U|)$, then by Fact 2, for the event $Y(\boldsymbol{y}) = [\Gamma_\ell(\boldsymbol{y}) = 1]$, we have

$$\text{dist}_{\mathcal{D}_\ell}(\Gamma_\ell, F) \ge \Pr_{\boldsymbol{y} \sim \mathcal{D}_\ell}[\Gamma_\ell(\boldsymbol{y}) \ne F(\boldsymbol{y}) | Y(\boldsymbol{y})] \Pr_{\boldsymbol{y} \sim \mathcal{D}_\ell}[Y(\boldsymbol{y})] = \frac{1}{2} \Pr_{\boldsymbol{y} \sim \mathcal{U}(\Delta^1_n)}[F(\boldsymbol{y}) \ne 1] \ge \frac{1}{8|U|}.$$

If $\Pr_{\boldsymbol{y}\sim\mathcal{U}(\Delta_n^1)}[F(\boldsymbol{y})\neq 1] < 1/(4|U|)$, then by Fact 2 and 5, and Claim 12,

$$\begin{aligned}
\mathrm{dist}_{\mathcal{D}_\ell}(\Gamma_\ell, F) &\geq \Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[\Gamma_\ell(\boldsymbol{y})\neq F(\boldsymbol{y})|\Gamma_\ell(\boldsymbol{y})=0] \Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[\Gamma_\ell(\boldsymbol{y})=0]\\
&= \frac{1}{2}\Pr_{\boldsymbol{y}\sim\mathcal{U}(\Delta_n^0)}[F(\boldsymbol{y})=1]\\
&= \frac{1}{2}\Pr_{\boldsymbol{z}\sim\mathcal{U}(\Delta_n^1),\boldsymbol{j}\sim\mathcal{U}(\mathrm{one}(\boldsymbol{z})),\boldsymbol{y}\sim\mathcal{U}(\boldsymbol{z}^{\boldsymbol{j}\leftarrow U})}[F(\boldsymbol{y})=1]\\
&\geq \frac{1}{2}\Pr_{\boldsymbol{z}\sim\mathcal{U}(\Delta_n^1),\boldsymbol{j}\sim\mathcal{U}(\mathrm{one}(\boldsymbol{z})),\boldsymbol{y}\sim\mathcal{U}(\boldsymbol{z}^{\boldsymbol{j}\leftarrow U})}[F(\boldsymbol{y})=1|F(\boldsymbol{z})=1]\cdot \Pr_{\boldsymbol{z}\sim\mathcal{U}(\Delta_n^1)}[F(\boldsymbol{z})=1]\\
&= \frac{1}{2}\Pr_{\boldsymbol{z}\sim\mathcal{U}(\Omega),\boldsymbol{j}\sim\mathcal{U}(\mathrm{one}(\boldsymbol{z})),\boldsymbol{y}\sim\mathcal{U}(\boldsymbol{z}^{\boldsymbol{j}\leftarrow U})}[F(\boldsymbol{y})=1]\cdot \Pr_{\boldsymbol{z}\sim\mathcal{U}(\Delta_n^1)}[F(\boldsymbol{z})=1]\\
&\geq \frac{1}{2}\frac{1}{2|U|}\left(1-\frac{1}{4|U|}\right)\geq\frac{1}{8|U|}.
\end{aligned}$$

◁

▷ Claim 14. Let F be a size-s DNF formula for $s\geq 2$ such that $\mathrm{dist}_{\mathcal{D}_\ell}(F,\Gamma_\ell)\leq 1/4$, then

$$\mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{U}(\Omega)}[\mathrm{mwidth}_F(\boldsymbol{y})]\leq 4\log s.$$

Proof. First, we have

$$\begin{aligned}
\frac{3}{4} &\leq \Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[F(\boldsymbol{y})=\Gamma_\ell(\boldsymbol{y})]\\
&= \frac{1}{2}\Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[F(\boldsymbol{y})=\Gamma_\ell(\boldsymbol{y})|\Gamma_\ell(\boldsymbol{y})=1]+\frac{1}{2}\Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[F(\boldsymbol{y})=\Gamma_\ell(\boldsymbol{y})|\Gamma_\ell(\boldsymbol{y})=0]\\
&\leq \frac{1}{2}\Pr_{\boldsymbol{y}\sim\mathcal{U}(\Delta_n^1)}[F(\boldsymbol{y})=1]+\frac{1}{2}.
\end{aligned}$$

Therefore, $\Pr_{\boldsymbol{y}\sim\mathcal{U}(\Delta_n^1)}[F(\boldsymbol{y})=1]\geq 1/2$.

Let $F=T_1\vee T_2\vee\cdots\vee T_s$. For $y\in\Omega$, let $\omega(y)\in[s]$ be the minimum integer such that $\mathrm{mwidth}_F(y)=|(T_{\omega(y)})_{\mathcal{M}}|$ and $T_{\omega(y)}(y)=1$.

Then, by (6),

$$\Pr_{\boldsymbol{y}\sim\mathcal{U}(\Omega)}[T_i(\boldsymbol{y})=1]=\Pr_{\boldsymbol{y}\sim\mathcal{U}(\Delta_n^1)}[T_i(\boldsymbol{y})=1|F(\boldsymbol{y})=1]=\frac{\Pr_{\boldsymbol{y}\sim\mathcal{U}(\Delta_n^1)}[T_i(\boldsymbol{y})=1]}{\Pr_{\boldsymbol{y}\sim\mathcal{U}(\Delta_n^1)}[F(\boldsymbol{y})=1]}\leq 2^{-|(T_i)_{\mathcal{M}}|/2+1}.$$

Now, by the concavity of log,

$$\begin{aligned}
\frac{1}{2}\mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{U}(\Omega)}[\mathrm{mwidth}_F(\boldsymbol{y})]-1 &= \mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{U}(\Omega)}\left[\log\left(2^{\mathrm{mwidth}_F(\boldsymbol{y})/2-1}\right)\right]\\
&\leq \log\left(\mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{U}(\Omega)}\left[2^{\mathrm{mwidth}_F(\boldsymbol{y})/2-1}\right]\right)\\
&= \log\left(\sum_{i\in[s]}2^{|(T_i)_{\mathcal{M}}|/2-1}\Pr_{\boldsymbol{y}\sim\mathcal{U}(\Omega)}[\omega(\boldsymbol{y})=i]\right)\\
&\leq \log\left(\sum_{i\in[s]}2^{|(T_i)_{\mathcal{M}}|/2-1}\Pr_{\boldsymbol{y}\sim\mathcal{U}(\Omega)}[T_i(\boldsymbol{y})=1]\right)\\
&\leq \log\left(\sum_{i\in[s]}2^{|(T_i)_{\mathcal{M}}|/2-1}2^{-|(T_i)_{\mathcal{M}}|/2+1}\right)\\
&= \log s.
\end{aligned}$$

Therefore, $\mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{U}(\Omega)}[\mathrm{mwidth}_F(\boldsymbol{y})]\leq 4\log s$. ◁

We are now ready to prove Lemma 11

Proof. If $\mathrm{dist}_{\mathcal{D}_\ell}(F,\Gamma_\ell) > 1/4$, then the result follows. Now suppose $\mathrm{dist}_{\mathcal{D}_\ell}(F,\Gamma_\ell) \le 1/4$. If $s = |F| < 2^{\mathrm{opt}(\mathcal{S})\ell/16}$, then by Claim 14, $\mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{U}(\Omega)}[\mathrm{mwidth}_F(\boldsymbol{y})] \le 4\log s = \mathrm{opt}(\mathcal{S})\ell/4$. Then by Claim 13, $\mathrm{dist}_{\mathcal{D}_\ell}(F,\Gamma_\ell) \ge 1/(8|U|)$. ◀

The proof of Theorem 3 is the same as the proof of Theorem 14 in [10]. We give the proof in Appendix C for completeness.

References

1 Michael Alekhnovich, Mark Braverman, Vitaly Feldman, Adam R. Klivans, and Toniann Pitassi. The complexity of properly learning simple concept classes. *J. Comput. Syst. Sci.*, 74(1):16–34, 2008. doi:10.1016/j.jcss.2007.04.011.

2 Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987.

3 Chris Calabro, Russell Impagliazzo, Valentine Kabanets, and Ramamohan Paturi. The complexity of unique k-sat: An isolation lemma for k-cnfs. *J. Comput. Syst. Sci.*, 74(3):386–393, 2008. doi:10.1016/j.jcss.2007.06.015.

4 Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlen. Exponential time complexity of the permanent and the tutte polynomial. *ACM Trans. Algorithms*, 10(4):21:1–21:32, 2014. doi:10.1145/2635812.

5 Andrzej Ehrenfeucht and David Haussler. Learning decision trees from random examples. *Inf. Comput.*, 82(3):231–246, 1989. doi:10.1016/0890-5401(89)90001-1.

6 Thomas R. Hancock, Tao Jiang, Ming Li, and John Tromp. Lower bounds on learning decision lists and trees. *Inf. Comput.*, 126(2):114–122, 1996. doi:10.1006/inco.1996.0040.

7 Lisa Hellerstein, Devorah Kletenik, Linda Sellie, and Rocco A. Servedio. Tight bounds on proper equivalence query learning of DNF. In *COLT 2012 - The 25th Annual Conference on Learning Theory, June 25-27, 2012, Edinburgh, Scotland*, pages 31.1–31.18, 2012. URL: http://proceedings.mlr.press/v23/hellerstein12/hellerstein12.pdf.

8 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

9 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

10 Caleb Koch, Carmen Strassle, and Li-Yang Tan. Superpolynomial lower bounds for decision tree learning and testing. *CoRR*, abs/2210.06375, 2022. doi:10.48550/arXiv.2210.06375.

11 Bingkai Lin. A simple gap-producing reduction for the parameterized set cover problem. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 81:1–81:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.81.

12 Craig A. Tovey. A simplified np-complete satisfiability problem. *Discret. Appl. Math.*, 8(1):85–89, 1984. doi:10.1016/0166-218X(84)90081-7.

13 Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984. doi:10.1145/1968.1972.

A Definitions

A.1 Concept Classes

For the lattice $\{0,1\}^n$, and $x, y \in \{0,1\}^n$, we define the partial order $x \le y$ if $x_i \le y_i$ for every i. When $x \le y$ and $x \ne y$, we write $x < y$. If $x < y$, we say that x is *below* y, or y is *above* x. A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is *monotone* if, for every $x \le y$, we have $f(x) \le f(y)$. A *literal* is a variable or negated variable. A *term* is a conjunction ($\wedge$) of literals. A *clause* is a disjunction ($\vee$) of literals. A *monotone term* (resp. clause) is a conjunction (resp. disjunction) of unnegated variables. The *size* of a term T, $|T|$, is the number of literals in the term T. A DNF (resp. CNF) is a disjunction (resp. conjunction) of terms (resp. clauses). The *size* $|F|$ of a DNF (resp. CNF) F is the number of terms (resp. clauses) in F. A *monotone DNF* (resp. monotone CNF) is a DNF (resp. CNF) with monotone terms (resp. clauses).

We define the following classes

1. size-s DNF and size-s Monotone DNF are the classes of DNF and monotone DNF, respectively, of size at most s.
2. size s-DT and size-s Monotone DT are the classes of decision trees and monotone decision trees, respectively, with at most s leaves.
3. k-Junta and Monotone k-Junta are the classes of Boolean functions and monotone Boolean functions that depend on at most k variables.

It is well known that

$$\textsc{Monotone } (\log s)\text{-}\textsc{Junta} \subset \text{ size-}s \textsc{ Monotone DT} \subset \text{ size-}s \textsc{ Monotone DNF} .$$

B Proofs

B.1 Proof of Fact 5

Proof. Denote the above distribution by $\mathcal{D}'$. By Item 1 in Fact 4, if $w \in \Delta_n^1$ and $j \in \text{one}(w)$, then $w^{j\leftarrow U} \subseteq \Delta_n^0$. Therefore, for $z \in \Delta_n^1$, $\Pr_{\boldsymbol{y}\sim\mathcal{D}'}[\boldsymbol{y} = z|\boldsymbol{\xi} = 0] = 0$ and then

$$\Pr_{\boldsymbol{y}\sim\mathcal{D}'}[\boldsymbol{y} = z] = \Pr_{\boldsymbol{\xi}\sim\mathcal{U}(\{0,1\})}[\boldsymbol{\xi} = 1] \cdot \Pr_{\boldsymbol{y}\sim\mathcal{U}(\Delta_n^1)}[\boldsymbol{y} = z] = \frac{1}{2|\Delta_n^1|} = \frac{1}{2|\Delta^1|^n}.$$

For $z \in \Delta_n^0$, suppose $z \in \Delta^{u_1} \times \cdots \times \Delta^{u_n}$ where $u \in U$. In the sampling according to $\mathcal{D}'$ and when $\xi = 0$, since for $j \in \text{one}(w)$, the elements of $w^{j\leftarrow U}$ are below w, we have $\Pr_{\boldsymbol{y}\sim\mathcal{D}'}[\boldsymbol{y} = z|\boldsymbol{w} \not> z] = 0$. Therefore,

$$\begin{aligned}\Pr_{\boldsymbol{y}\sim\mathcal{D}'}[\boldsymbol{y} = z] \quad &= \quad \Pr_{\boldsymbol{\xi}\sim\mathcal{U}(\{0,1\})}[\boldsymbol{\xi} = 0] \cdot \Pr_{\boldsymbol{w}\sim\mathcal{U}(\Delta_n^1)}[\boldsymbol{w} > z] \times \\ &\qquad \Pr_{\boldsymbol{j}\sim\mathcal{U}(one(\boldsymbol{w}))}[z \in \boldsymbol{w}^{\boldsymbol{j}\leftarrow U}|\boldsymbol{w} > z, \boldsymbol{w} \in \Delta_n^1] \times \\ &\qquad \Pr_{\boldsymbol{v}\sim\mathcal{U}(\boldsymbol{w}^{\boldsymbol{j}\leftarrow U})}[\boldsymbol{v} = z|z \in \boldsymbol{w}^{\boldsymbol{j}\leftarrow U}]. \end{aligned} \tag{2}$$

Now, since, for $x \in \Delta^0$, the number of elements in Δ^1 that are above x is $\lceil \ell/2 \rceil$, we have that the number of $w \in \Delta_n^1 = (\Delta^1)^n$ that are above $z \in \Delta^{u_1} \times \cdots \times \Delta^{u_n}$ is $\lceil \ell/2 \rceil^{n-\text{wt}(u)}$. Therefore,

$$\Pr_{\boldsymbol{w}\sim\mathcal{U}(\Delta_n^1)}[\boldsymbol{w} > z] = \frac{\lceil \ell/2 \rceil^{n-\text{wt}(u)}}{|\Delta_n^1|}. \tag{3}$$

Now let $w > z$ and $w \in \Delta_n^1$. Since for two different $u, u' \in U$, we have $\prod_{i=1}^n \Delta^{u_i}$ and $\prod_{i=1}^n \Delta^{u'_i}$ are disjoint sets, and since $z \in \Delta^{u_1} \times \cdots \times \Delta^{u_n}$, we have $z \in w^{j\leftarrow U}$ if and only if $z = w^{j\leftarrow u}$. Therefore, the number of elements $j \in \text{one}(w)$ that satisfy $z \in w^{j\leftarrow U}$ is the number of elements $j \in \text{one}(w)$ that satisfy $z = w^{j\leftarrow u}$. This is the number of elements $j \in \text{one}(w)$ that satisfies for every $u_i = 0$, $z_{i,j_i} = 0$. For a j u.a.r. and a fixed i where $u_i = 0$, the probability that z_i and w_i differ only in entry j_i is $1/\lceil \ell/2 \rceil$. Therefore,

$$\Pr_{\boldsymbol{j}\sim\mathcal{U}(one(\boldsymbol{w}))}[z \in \boldsymbol{w}^{\boldsymbol{j}\leftarrow U}|\boldsymbol{w} > z, \boldsymbol{w} \in \Delta_n^1] = \frac{1}{\lceil \ell/2 \rceil^{n-\text{wt}(u)}}. \tag{4}$$

Finally, by item 2 in Fact 4, since $|w^{j\leftarrow U}| = |U|$, we have

$$\Pr_{\boldsymbol{v}\sim\mathcal{U}(\boldsymbol{w}^{j\leftarrow U})}[\boldsymbol{v} = z | z \in \boldsymbol{w}^{j\leftarrow U}] = \frac{1}{|\boldsymbol{w}^{j\leftarrow U}|} = \frac{1}{|U|}. \tag{5}$$

By (2), (3), (4), and (5), we have

$$\Pr_{\boldsymbol{y}\sim\mathcal{D}'}[\boldsymbol{y} = z] = \frac{1}{2}\cdot\frac{\lceil \ell/2\rceil^{n-\mathrm{wt}(u)}}{|\Delta^1_n|}\cdot\frac{1}{\lceil \ell/2\rceil^{n-\mathrm{wt}(u)}}\cdot\frac{1}{|U|} = \frac{1}{2|U|\cdot|\Delta^1_n|} = \frac{1}{2|U|\cdot|\Delta^0|^n}. \quad ◀$$

B.2 Proof of Claim 6

Proof. Let T be a term of monotone size at least $\mathrm{opt}(\mathcal{S})\ell/5$. Let b_i denote the number of unnegated variables of T of the form $y_{i,j}$ and let T_i be their conjunction. Then $T_{\mathcal{M}} = \wedge_{i=1}^n T_i$ and $\sum_{i=1}^n b_i = |T_{\mathcal{M}}| \geq \mathrm{opt}(\mathcal{S})\ell/5$. If, for some i, $b_i > \lceil \ell/2\rceil$, then the term T_i is zero on all $\Delta^0 \cup \Delta^1$, and therefore, T is zero on all $\Delta^0_n \cup \Delta^1_n$. Thus, it can be just removed from F. So, we may assume that $b_i \leq \lceil \ell/2\rceil$ for all i. First,

$$\begin{aligned}
\Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[T(\boldsymbol{y}) = 1 | \Gamma_\ell(\boldsymbol{y}) = 1] &= \Pr_{\boldsymbol{y}\sim\mathcal{U}(\Delta^1_n)}[T(\boldsymbol{y}) = 1] \leq \Pr_{\boldsymbol{y}\sim\mathcal{U}(\Delta^1_n)}[T_{\mathcal{M}}(\boldsymbol{y}) = 1] \\
&= \prod_{i=1}^n \Pr_{\boldsymbol{y}_i\sim\mathcal{U}(\Delta^1)}[T_i(\boldsymbol{y}_i) = 1] \\
&= \prod_{i=1}^n \frac{\binom{\ell-b_i}{\lceil \ell/2\rceil - b_i}}{\binom{\ell}{\lceil \ell/2\rceil}} \\
&= \prod_{i=1}^n \left(1 - \frac{b_i}{\ell}\right)\left(1 - \frac{b_i}{\ell-1}\right)\cdots\left(1 - \frac{b_i}{\lceil \ell/2\rceil + 1}\right) \\
&\leq \prod_{i=1}^n \prod_{j=\lceil \ell/2\rceil+1}^{\ell} \exp(-b_i/j) = \prod_{i=1}^n \exp\left(-b_i \sum_{j=\lceil \ell/2\rceil+1}^{\ell} 1/j\right) \\
&= \exp\left(-|T_{\mathcal{M}}| \sum_{j=\lceil \ell/2\rceil+1}^{\ell} 1/j\right) \\
&\leq 2^{-|T_{\mathcal{M}}|/2} \leq 2^{-\mathrm{opt}(\mathcal{S})\ell/10}.
\end{aligned} \tag{6}$$

Let F' be the disjunction of all the terms in F of monotone size at most $\mathrm{opt}(\mathcal{S})\ell/5$. Let $T^{(1)}, \ldots, T^{(m)}$ be all the terms of monotone size greater than $\mathrm{opt}(\mathcal{S})\ell/5$ in F. Then, by (6) and the union bound,

$$\begin{aligned}
\Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[F(\boldsymbol{y}) \neq F'(\boldsymbol{y}) | \Gamma_\ell(\boldsymbol{y}) = 1] &\leq \Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[\vee_{i=1}^m T^{(i)}(\boldsymbol{y}) = 1 | \Gamma_\ell(\boldsymbol{y}) = 1] \\
&\leq 2^{-\mathrm{opt}(\mathcal{S})\ell/10} m \leq 2^{-\mathrm{opt}(\mathcal{S})\ell/20}.
\end{aligned} \tag{7}$$

and (Here we abbreviate $F'(\boldsymbol{y}), F(\boldsymbol{y})$ and $\Gamma_\ell(\boldsymbol{y})$ by F', F and Γ_ℓ)

$$\begin{aligned}
\mathrm{dist}_{\mathcal{D}_\ell}(\Gamma_\ell, F') &= \Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[F' \neq \Gamma_\ell] \\
&= \frac{1}{2}\Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[F' \neq \Gamma_\ell | \Gamma_\ell = 1] + \frac{1}{2}\Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[F' \neq \Gamma_\ell | \Gamma_\ell = 0] \qquad (8)\\
&= \frac{1}{2}\Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[F' \neq F | \Gamma_\ell = 1] + \\
&\quad \frac{1}{2}\Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[F \neq \Gamma_\ell | \Gamma_\ell = 1] + \frac{1}{2}\Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[F' \neq \Gamma_\ell | \Gamma_\ell = 0] \qquad (9)\\
&\leq 2^{-\mathrm{opt}(\mathcal{S})\ell/20} + \frac{1}{2}\Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[F \neq \Gamma_\ell | \Gamma_\ell = 1] + \frac{1}{2}\Pr_{\boldsymbol{y}\sim\mathcal{D}_\ell}[F \neq \Gamma_\ell | \Gamma_\ell = 0] \quad (10)\\
&= 2^{-\mathrm{opt}(\mathcal{S})\ell/20} + \mathrm{dist}_{\mathcal{D}_\ell}(\Gamma_\ell, F).
\end{aligned}$$

In (8), we used Fact 2. In (9), we used the probability triangle inequality. In (10), we used (7) and the fact that if $F'(\boldsymbol{y}) \neq 0$, then $F(\boldsymbol{y}) \neq 0$. ◀

C The proof of Theorem 3

Proof. Consider the constant λ in Lemma 2. Let $c = \lambda/6$. Suppose there is a PAC learning algorithm $\mathcal{A}$ for Monotone $(\log s)$-Junta by size-s DNF with $\epsilon = 1/(16n)$ that runs in time $n^{c \log s}$. We show that there is k such that for

$$k' = \frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k},$$

(k,k')-Set-Cover can be solved in time $N^{5ck} \leq N^{\lambda k}$. By Lemma 2, the result then follows.

Let $\mathcal{S} = (S, U, E)$ be an N-vertex (k,k')-Set-Cover instance where

$$k = \frac{1}{2}\frac{\log\log N}{\log\log\log N} \text{ and } k' = \frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k}.$$

Consider the following algorithm $\mathcal{B}$

1. Input $\mathcal{S} = (S, U, E)$ an instance for (k,k')-Set-Cover .
2. Construct Γ_5 and $\mathcal{D}_5$.
3. Run $\mathcal{A}$ using Γ_5 and $\mathcal{D}_5$ with $s = 2^{5k}$ and $n = N$. If it runs more than N^{5ck} steps, then output No .
4. Let F be the output DNF.
5. If $|F| > s$ then output No .
6. Estimate $\eta = \mathrm{dist}_{\mathcal{D}_5}(F, \Gamma_5)$.
7. If $\eta \leq \frac{1}{16N}$, output Yes , otherwise output No .

The running time of this algorithm is $N^{5ck} \leq N^{\lambda k}$. Therefore, it is enough to prove the following

▷ Claim 15. Algorithm $\mathcal{B}$ solves (k,k')-Set-Cover .

Proof. Yes case: Let $\mathcal{S} = (S, U, E)$ be a (k,k')-Set-Cover instance and $\mathrm{opt}(\mathcal{S}) \leq k$. Then, $size(\Gamma_5) \leq 2^{5\cdot\mathrm{opt}(\mathcal{S})} \leq 2^{5k} = s$, and by Fact 3, Γ_5 is Monotone $\log s$-Junta. Therefore, w.h.p., algorithm $\mathcal{A}$ learns Γ_5 and outputs a DNF that is $\eta = 1/(16N)$ close to the target with respect to $\mathcal{D}_5$. Since $\mathcal{B}$ terminates $\mathcal{A}$ after N^{5ck} time, we only need to prove that $\mathcal{A}$ runs at most N^{5ck} time.

The running time of $\mathcal{A}$ is

$$n^{c\log s} = N^{c\log s} \leq N^{5ck}.$$

No Case: Let $\mathcal{S} = (S, U, E)$ be a (k, k')-Set-Cover instance and $\text{opt}(\mathcal{S}) > k'$. By Lemma 11, any DNF, F, of size $|F| < 2^{5k'/16}$ satisfies $\text{dist}_{\mathcal{D}_5}(F, \Gamma_5) \geq 1/(8|U|)$. First, we have, for large N

$$k' = \frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k} > 32k.$$

Therefore, any DNF, F, of size $|F| < 2^{10k}$ satisfies $\text{dist}_{\mathcal{D}_5}(F, \Gamma_5) \geq 1/(8|U|)$.

We have $2^{10k} > s$. So, $\mathcal{B}$ either runs more than N^{5ck} steps and then outputs No in step 3 or outputs a DNF of size more than s and then outputs No in step 4 or outputs a DNF of size at most s with $\text{dist}_{\mathcal{D}_5}(F, \Gamma_5) \geq 1/(8|U|) > 1/(8N) > 1/(16N)$ and outputs No in step 6. ◁

◀

An Embarrassingly Parallel Optimal-Space Cardinality Estimation Algorithm

Emin Karayel
Department of Computer Science, Technische Universität München, Germany

Abstract

In 2020 Błasiok (ACM Trans. Algorithms 16(2) 3:1-3:28) constructed an optimal space streaming algorithm for the cardinality estimation problem with the space complexity of $\mathcal{O}(\varepsilon^{-2}\ln(\delta^{-1}) + \ln n)$ where ε, δ and n denote the relative accuracy, failure probability and universe size, respectively. However, his solution requires the stream to be processed sequentially. On the other hand, there are algorithms that admit a merge operation; they can be used in a distributed setting, allowing parallel processing of sections of the stream, and are highly relevant for large-scale distributed applications. The best-known such algorithm, unfortunately, has a space complexity exceeding $\Omega(\ln(\delta^{-1})(\varepsilon^{-2}\ln\ln n + \ln n))$. This work presents a new algorithm that improves on the solution by Błasiok, preserving its space complexity, but with the benefit that it admits such a merge operation, thus providing an optimal solution for the problem for both sequential and parallel applications. Orthogonally, the new algorithm also improves algorithmically on Błasiok's solution (even in the sequential setting) by reducing its implementation complexity and requiring fewer distinct pseudo-random objects.

2012 ACM Subject Classification Mathematics of computing → Probabilistic algorithms; Theory of computation → Pseudorandomness and derandomization; Theory of computation → Distributed algorithms; Theory of computation → Sketching and sampling

Keywords and phrases Distinct Elements, Distributed Algorithms, Randomized Algorithms, Expander Graphs, Derandomization, Sketching

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.35

Category RANDOM

Related Version *Full Version*: https://arxiv.org/abs/2307.00985 [26]

Supplementary Material
Software: https://isa-afp.org/entries/Distributed_Distinct_Elements.html [25]
Software: https://isa-afp.org/entries/Expander_Graphs.html [27]

Acknowledgements Thanks to the anonymous reviewers for their careful review and helpful comments and to Tobias Nipkow for many helpful discussions.

1 Introduction

In 1985 Flajolet and Martin [14] introduced a space-efficient streaming algorithm for the estimation of the count of distinct elements in a stream $a_1, \ldots, a_m$ whose elements are from a finite universe U. Their algorithm does not modify the stream, observes each stream element exactly once and its internal state requires space logarithmic in $n = |U|$. However, their solution relies on the model assumption that a given hash function can be treated like a random function selected uniformly from the family of all functions with a fixed domain and range. Despite the ad-hoc assumption, their work spurred a large number of publications[1], improving the space efficiency and runtime of the algorithm. In 1999 Alon et al. [4] identified

[1] Pettie and Wang [33, Table 1] summarized a comprehensive list.

a solution that avoids the ad-hoc model assumption. They use 2-independent families of hash functions, which can be seeded by a logarithmic number of random bits in $|U|$ while retaining a restricted set of randomness properties. Their refined solution was the first rigorous Monte-Carlo algorithm for the problem. Building on their work, Bar-Yossef et al. in 2002 [6], then Kane et al. in 2010 [24] and lastly, Błasiok in 2020 [9][2] developed successively better algorithms achieving a space complexity of $\mathcal{O}(\varepsilon^{-2}\ln(\delta^{-1}) + \ln n)$, which is known to be optimal [23, Theorem 4.4].

Table 1 Important cardinality estimation algorithms.

Year, Author	Space Complexity	Merge
1981, Flajolet and Martin	$\mathcal{O}(\varepsilon^{-2}\ln n)$ for constant δ a)	Yes
1999, Alon et al.	$\mathcal{O}(\ln\ln n)$ for $\delta = 2(\varepsilon+1)^{-1}$	Yes
2002, Bar-Yossef et al.b)	$\mathcal{O}(\ln(\delta^{-1})(\varepsilon^{-2}\ln\ln n + \mathrm{poly}(\ln(\varepsilon^{-1}), \ln\ln n)\ln n))$ c)	Yes
2010, Kane et al.	$\mathcal{O}(\ln(\delta^{-1})(\varepsilon^{-2} + \ln n))$	No
2020, Błasiok	$\mathcal{O}(\ln(\delta^{-1})\varepsilon^{-2} + \ln n)$	No
This work	$\mathcal{O}(\ln(\delta^{-1})\varepsilon^{-2} + \ln n)$	Yes

a) Random oracle model.
b) Algorithm 2 from the publication.
c) The notation $\mathrm{poly}(a, b)$ stands for a term polynomial in a and b.

These algorithms return an approximation Y of the number of distinct elements $|A|$ (for $A := \{a_1, \ldots, a_m\}$) with relative error ε and success probability $1-\delta$, i.e.:

$$\mathcal{P}(|Y - |A|| \le \varepsilon\,|A|) \ge 1 - \delta$$

where the probability is only over the internal random coin flips of the algorithm but holds for all inputs.

Unmentioned in the source material is the fact that it is possible to run the older algorithms by Alon et al. and Bar-Yossef et al. in a parallel mode of operation. This is due to the fact that the algorithms make the random coin flips only in a first initialization step, proceeding deterministically afterwards and that the processing step for the stream elements is commutative. For example, if two runs for sequences a and b of the algorithm had been started with the same coin flips, then it is possible to introduce a new operation that merges the final states of the two runs and computes the state that the algorithm would have reached if it had processed the concatenation of the sequences a and b sequentially.

This enables processing a large stream using multiple processes in parallel. The processes have to communicate at the beginning and at the end to compute an estimate. The communication at the beginning is to share random bits, and the communication at the end is to merge the states. Because there is no need for communication in between, the speed-up is optimal with respect to the number of processes, such algorithms are also called embarrassingly parallel [15, Part 1]. This mode of operation has been called the distributed streams model by Gibbons and Tirthaputra [17]. Besides the distributed streams model, such a merge operation allows even more varied use cases, for example, during query processing in a Map-Reduce pipeline [11]. Figure 1 illustrates two possible modes of operation (among many) enabled by a merge function.

[2] An earlier version of Błasiok's work was presented in the ACM-SIAM Symposium on Discrete Algorithms in 2018. [8]

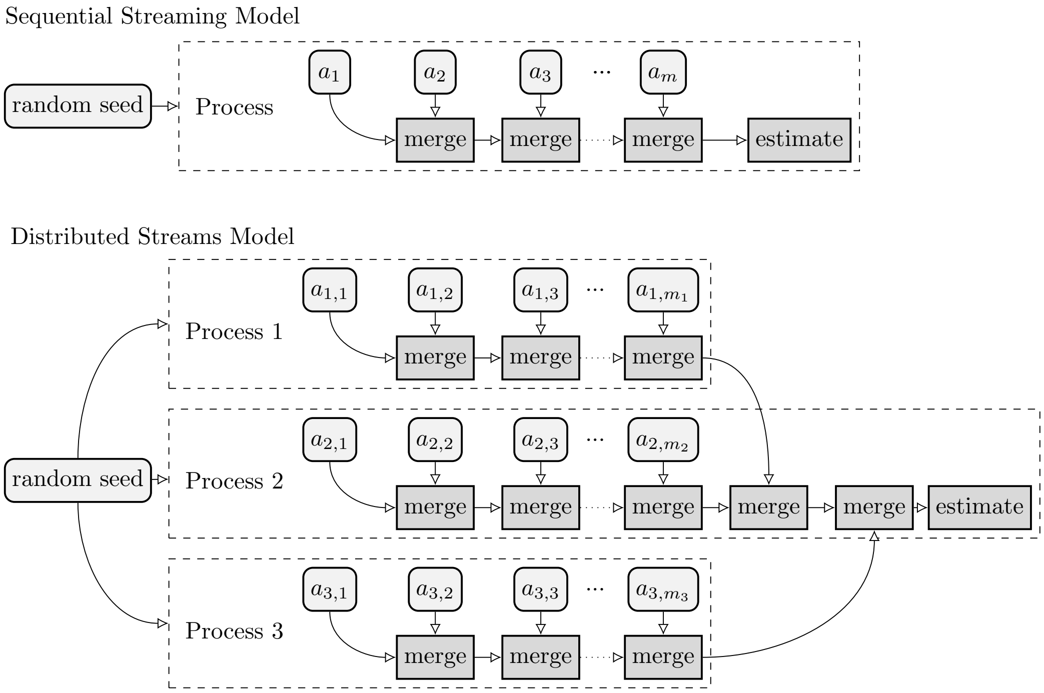


Figure 1 Example use cases for cardinality estimation algorithms that support merge.

However, an extension with such a merge operation is not possible for the improved algorithms by Kane et al. and Błasiok. This is because part of their correctness proof relies inherently on the assumption of sequential execution, in particular, that the sequence of states is monotonically increasing, which is only valid in the sequential case. This work introduces a new distributed cardinality estimation algorithm which supports a merge operation with the same per-process space usage as the optimal sequential algorithm by Błasiok: $\mathcal{O}(\varepsilon^{-2}\ln(\delta^{-1}) + \ln n)$. Thus the algorithm in this work has the best possible space complexity in *both* the sequential and distributed streaming model.[3] (Table 1 provides a summary of the algorithms mentioned here.)

The main idea was to modify the algorithm by Błasiok into a history-independent algorithm. This means that the algorithm will, given the same coin-flips, reach the same state independent of the order in which the stream elements arrive, or more precisely, independent of the execution tree as long as its nodes contain the same set of elements. This also means that the success event, i.e., whether an estimate computed from the state has the required accuracy, only depends on the set of distinct stream elements encountered (during the execution tree) and the initial random coin flips. As a consequence and in contrast to previous work, the correctness proof does not rely on bounds on the probability of certain events over the entire course of the algorithm, but can be established independent of past events.

Błasiok uses a pseudo-random construction based on hash families, expander walks, an extractor based on Parvaresh-Vardy codes [21] and a new sub-sampling strategy [9][Lem. 39]. I was able to build a simpler stack that only relies on hash families and a new two-stage expander graph construction, for which I believe there may be further applications. To summarize – the solution presented in this work has two key improvements:

[3] That the complexity is also optimal for the distributed setting is established in Section 7.

- Supports the sequential *and* distributed streaming model with optimal space.
- Requires fewer pseudo-random constructs, i.e., only hash families and expander walks.

The next section introduces notation. After that, I present new results on expander walks (Section 3) needed in the new pseudo-random construction and a self-contained presentation of the new algorithm and its correctness proof (Sections 5 and 6). Concluding with a discussion of its optimality in the distributed setting (Section 7) and a discussion of open research questions (Section 8). It is worthwhile noting that an extended version of this work [26] is available, which includes more background and more detailed proofs.

The results obtained in this work have also been formally verified [25] using the proof assistant Isabelle [31]. Isabelle has been used to verify many [1] advanced results from mathematics (e.g. the prime number theorem [12]) and computer science (e.g. the Cook-Levin theorem [5]). For readers mainly interested in the actual results, the formalization can be ignored as the theorems all contain traditional mathematical proofs. Nevertheless, Table 3 references the corresponding formalized fact for every theorem in this work.

2 Notation and Preliminaries

General constants are indicated as $C_1, C_2, \cdots$ etc. Their values are fixed throughout this work and are summarized in Table 2. For $n \in \mathbb{N}$, let us define $[n] := \{0, 1, \ldots, n-1\}$. The notation $[P]$ for a predicate P denotes the Iverson bracket, i.e., its value is 1 if the predicate is true and 0 otherwise. The notation $\operatorname{ld} x$ (resp. $\ln x$) stands for the logarithm to base 2 (resp. e) of $x \in \mathbb{R}_{>0}$. The notations $\lfloor x \rfloor$ and $\lceil x \rceil$ represent the floor and ceiling functions: $\mathbb{R} \to \mathbb{Z}$. For a probability space Ω, the notation $\mathcal{P}_{\omega\sim\Omega}(F(\omega))$ is the probability of the event: $\{\omega | F(\omega)\}$. And $\mathbb{E}_{\omega\sim\Omega}(f(\omega))$ is the expectation of f if ω is sampled from the distribution Ω, i.e., $\mathbb{E}_{\omega\sim\Omega}(f(\omega)) := \int_\Omega f(\omega)\, d\omega$. Similarly, $\mathbb{V} f = \mathbb{E}(f - \mathbb{E} f)^2$. For a finite non-empty set S, $U(S)$ is the uniform probability space over S, i.e., $\mathcal{P}(\{x\}) = |S|^{-1}$ for all $x \in S$. (Usually, we will abbreviate $U(S)$ with S when it is obvious from the context.) All probability spaces mentioned in this work will be discrete, i.e., measurability will be trivial.

All graphs in this work are finite and are allowed to contain parallel edges and self-loops. For an ordering of the vertices of such a graph, it is possible to associate an adjacency matrix $A = (a_{ij})$, where a_{ij} is the count of the edges between the i-th to the j-th vertex. We will say it is undirected d-regular if the adjacency matrix is symmetric and all its row (or equivalently) column sums are d. Such an undirected d-regular graph is called a λ-expander if the second largest absolute eigenvalue of its adjacency matrix is at most $d\lambda$.

Given an expander graph G, we denote by $\text{Walk}(G, l)$, the set of walks of length l. For a walk $w \in \text{Walk}(G, l)$ we write w_i for the i-th vertex and $w_{i,i+1}$ for the edge between the i-th and $(i+1)$-th vertex. Because of the presence of parallel edges, two distinct walks may have the same vertex sequence. As a probability space $U(\text{Walk(G,l)})$ corresponds to choosing a random starting vertex and performing an $(l-1)$-step random walk.

3 Chernoff-type estimates for Expander Walks

The following theorem has been shown implicitly by Impagliazzo and Kabanets [22, Th. 10]:

▶ **Theorem 1** (Impagliazzo and Kabanets). *Let $G = (V, E)$ be a λ-expander graph and f a boolean function on its vertices, i.e.: $f : V \to \{0, 1\}$ s.t. $\mu = \mathbb{E}_{v\sim U(V)} f(v)$, $6\lambda \leq \mu$ and $2\lambda < \varepsilon < 1$ then:*

$$\mathcal{P}_{w\sim\text{Walk}(G,l)}\left(\textstyle\sum_{i\in[l]} f(w_i) \geq (\mu + \varepsilon)l\right) \leq \exp(-lD(\mu + \varepsilon || \mu + 2\lambda))$$

Especially, the restriction $\mu \geq 6\lambda$ in the above result causes technical issues since usually one only has an upper bound for μ. The result follows in Impagliazzo and Kabanets work as a corollary from the application of their main theorem [22][Thm. 1] to the hitting property established by Alon et al. [3, Th. 4.2] in 1995. It is easy to improve Theorem 1 by using an improved hitting property:

▶ **Theorem 2** (Hitting Property for Expander Walks). *Let $G = (V, E)$ be a λ-expander graph and $W \subseteq V$, $I \subseteq [l]$ and let $\mu := \frac{|W|}{|V|}$ then:*

$$\mathcal{P}_{w \sim \text{Walk}(G,l)} \left(\bigwedge_{i \in I} w_i \in W \right) \leq (\mu(1-\lambda)+\lambda)^{|I|} \leq (\mu+\lambda)^{|I|}$$

The above theorem for the case where $I = [l]$ is shown by Vadhan [37, Theorem 4.17]. The extended, full version [26] of this manuscript describes how to extend Vadhan's proof to the case where $I \subset [l]$. With the previous result, it is possible to obtain a new, improved version of Theorem 1:

▶ **Theorem 3** (Improved version of Theorem 1). *Let $G = (V, E)$ be a λ-expander graph and f a boolean function on its vertices, i.e.: $f : V \to \{0, 1\}$ s.t. $\mu = \mathbb{E}_{v \sim U(V)} f(v)$ and $\mu + \lambda \leq \gamma \leq 1$ then:*

$$\mathcal{P}_{w \sim \text{Walk}(G,l)} \left(\sum_{i \in [l]} f(w_i) \geq \gamma l \right) \leq \exp\left(-lD\left(\gamma || \mu + \lambda\right)\right)$$

Proof. This follows from Theorem 2 and the generalized Chernoff bound [22][Thm. 1]. ◀

Impagliazzo and Kabanets approximate the divergence $D(\gamma||\mu + \lambda)$ by $2(\gamma - (\mu + \lambda))^2$. In this work, we are interested in the case where $\mu + \lambda \to 0$, where such an approximation is too weak, so we cannot follow that approach. (Note that $D(\gamma||\mu + \lambda)$ can be arbitrarily large, while $(\gamma - (\mu + \lambda))^2$ is at most 1.) Instead, we derive a bound of the following form:

▶ **Lemma 4.** *Let $G = (V, E)$ be a λ-expander graph and f a boolean function on its vertices, i.e.: $f : V \to \{0, 1\}$ s.t. $\mu = \mathbb{E}_{v \sim U(V)} f(v)$ and $\mu + \lambda \leq \gamma < 1$ then:*

$$\mathcal{P}_{w \sim \text{Walk}(G,l)} \left(\sum_{i \in [l]} f(w_i) \geq \gamma l \right) \leq \exp\left(-l(\gamma \ln((\mu+\lambda)^{-1}) - 2e^{-1})\right)$$

Proof. The result follows from Theorem 3 and the inequality: $D(\gamma||p) \geq \gamma \ln(p^{-1}) - 1$ for $0 < \gamma < 1$ and $0 < p < 1$. ◀

An application for the above inequality, where the classic Chernoff-bound by Gillman [18] would not be useful, is establishing a failure probability for the repetition of an algorithm that already has a small failure probability. For example, if an algorithm has a failure probability of δ^*, then it is possible to repeat it $\mathcal{O}\left(\frac{\ln(\delta^{-1})}{\ln((\delta^*)^{-1})}\right)$-times to achieve a failure probability of δ. (This is done in Section 6.) Another consequence of this is a deviation bound for unbounded functions with a sub-gaussian tail bound:

▶ **Lemma 5** (Deviation Bound). *Let $G = (V, E)$ be a λ-expander graph and $f : V \to \mathbb{R}_{\geq 0}$ s.t. $\mathcal{P}_{v \sim U(V)}(f(v) \geq x) \leq \exp(-x(\ln x)^3)$ for $x \geq 20$ and $\lambda \leq \exp(-l(\ln l)^3)$ then*

$$\mathcal{P}_{w \sim \text{Walk}(G,l)} \left(\sum_{i \in [l]} f(w_i) \geq C_1 l \right) \leq \exp(-l)$$

where $C_1 := e^2 + e^3 + (e - 1) \leq 30$.

Note that the class includes sub-gaussian random variables but is even larger. The complete proof is in Appendix A. The proof essentially works by approximating the function f using the Iverson bracket: $f(x) \leq \Sigma_k [e^k \leq f(x) \leq e^{k+1}] e^{k+1}$ and establishing bounds on the frequency of each bracket. For large k this is established using the Markov inequality, and for small k the previous lemma is used. The result is a stronger version of a lemma established by Błasiok [9][Lem. 36], and the proof in this work is heavily inspired by his.

4 Explicit Pseudo-random Constructions

4.1 Strongly explicit expander graphs

For the application in this work, it is necessary to use *strongly explicit expander graphs*. For such a graph, it is possible to sample a random walk without having to represent the graph in memory. Moreover, sampling a random walk from a d-regular graph G with n-vertices is possible using a random sample from $[nd^{l-1}]$, i.e., we can map such a number to a walk algorithmically, such that the resulting distribution corresponds to the distribution from $\text{Walk}(G, l)$ – this allows the previously mentioned two-stage construction.

A possible construction for strongly explicit expander graphs for every vertex count n and spectral bound λ is described by Murtagh et al. [29][Thm. 20, Apx. B]. Note that the degree d in their construction only grows polynomially with λ^{-1}, hence $\ln(d(\lambda)) \in \mathcal{O}(\ln(\lambda^{-1}))$. We will use the notation $\mathcal{E}([n], \lambda, l)$ for the sample space of random walks of length l in the described graph over the vertex set $[n]$. The same construction can also be used on arbitrary finite vertex sets S, if it is straightforward to map $[|S|]$ to S algorithmically. Thus we use the notation $\mathcal{E}(S, \lambda, l)$ for such S. Importantly $|\mathcal{E}(S, \lambda, l)| = |S|\, d(\lambda)^{l-1}$. Thus a walk in such a graph requires $\mathcal{O}(\text{ld}\, |S| + l\, \text{ld}(\lambda^{-1}))$ bits to represent.

4.2 Hash Families

Let us introduce the notation: $\mathcal{H}_k([n], [2^c])$ for the k-independent hash-family [39] from $[n]$ to $[2^c]$. Note that $\text{ld}\,(|\mathcal{H}_k([n], [2^c])|) \in \mathcal{O}(k(c + \ln n))$.

For our application, we will need a second family with a geometric distribution (as opposed to uniform) on the range, in particular such that $\mathcal{P}(f(a) \geq k) = 2^{-k}$. A straightforward method to achieve that is to compose the functions of the hash family $\mathcal{H}_k([2^d], [2^d])$ with the function that computes the number of trailing zeros of the binary representation of its input $[2^d] \to [d]$. We denote such a hash family with $\mathcal{G}_k([2^d])$ where the range is $[d+1]$. Such a hash family is also one for a domain $[n] \subseteq [2^d]$, and hence we can extend the notation: $\mathcal{G}_k([n])$. Note that: $\mathcal{P}_{f \sim G_k([n])}(f(a) \geq k) = 2^{-k}$ for all $k \leq \lceil \text{ld}\, n \rceil$ and also $\text{ld}\,(|\mathcal{G}_k([n])|) \in \mathcal{O}(k \ln n)$.

5 The Algorithm

Because of all the distinct possible execution models, it is best to present the algorithm as a purely functional data structure with four operations:

$$\begin{aligned} \text{init} &: () \to \text{seed} & \text{single} &: [n] \to \text{seed} \to \text{sketch} \\ \text{merge} &: \text{sketch} \to \text{sketch} \to \text{sketch} & \text{estimate} &: \text{sketch} \to \mathbb{R} \end{aligned}$$

The init step should be called only once globally – it is the only random operation – its result forms the seed and must be the same during the entire course of the algorithm. The operation single returns a sketch for a singleton set corresponding to its first argument. The operation merge computes a sketch representing the union of its input sketches and the operation estimate returns an estimate for the number of distinct elements for a given sketch.

The algorithm will be introduced in two successive steps. The first step is a solution that works for $(\ln n)^{-1} \leq \delta < 1$. The sketch requires only $\mathcal{O}(\ln(\delta^{-1})\varepsilon^{-2} + \ln\ln n)$, but the initial coin flips require $\mathcal{O}(\ln n + \ln(\varepsilon^{-1})^2 + \ln(\delta^{-1})^3)$ bits. For $\delta \geq (\ln n)^{-1}$ this is already optimal. In the second step (Section 6) a black-box vectorization of the previous algorithm will be needed to achieve the optimal $\mathcal{O}(\ln(\delta^{-1})\varepsilon^{-2} + \ln n)$ space usage for all $0 < \delta < 1$.

For this entire section let us fix a universe size $n > 0$, a relative accuracy $0 < \varepsilon < 1$, a failure probability $(\ln n)^{-1} \leq \delta < 1$ and define:

$$\begin{array}{ll}
l := \lceil C_6 \ln(2\delta^{-1}) \rceil & b := 2^{\lceil \operatorname{ld}(C_4 \varepsilon^{-2}) \rceil} \\
k := \lceil C_2 \ln b + C_3 \rceil & \lambda := \min\left(\dfrac{1}{16}, \exp(-l(\ln l)^3)\right) \\
\Psi := \mathcal{G}_2([n]) \times \mathcal{H}_2([n],[C_7 b^2]) \times \mathcal{H}_k([C_7 b^2],[b]) & \Omega := \mathcal{E}(\Psi, \lambda, l)
\end{array}$$

The implementation of the operations is presented in Algorithm 1. Note that these are functional programs and pass the state as arguments and results; there is no global (mutable) state. The sketch consists of two parts (B, q). The first part is a two-dimensional table of sizes b and l. The second part is a single natural number, the cut-off level. The function compress is an internal operation and is not part of the public API. It increases the cut-off level and decreases the table values if the space usage is too high.

Algorithm 1 Algorithm for $\delta > (\ln n)^{-1}$.

```
function init() : $\Omega$
    return random $U(\Omega)$

function compress($(B, q) : \mathcal{S}$) : $\mathcal{S}$
    while $\sum_{i \in [l], j \in [b]} \lfloor \operatorname{ld}(B[i,j] + 2) \rfloor > C_5 b l$
        $q \leftarrow q + 1$
        $B[i,j] \leftarrow \max(B[i,j] - 1, -1)$ for $i \in [l], j \in [b]$
    return $(B, q)$

function single($x : U, \omega : \Omega$) : $\mathcal{S}$
    $B[i,j] \leftarrow -1$
    for $i \in [l]$
        $B[i, h(g(x))] = f(x)$ where $(f, g, h) = \omega_i$
    return compress$(B, 0)$

function merge($(B_a, q_a) : \mathcal{S}, (B_b, q_b) : \mathcal{S}$) : $\mathcal{S}$
    $q \leftarrow \max(q_a, q_b)$
    $B[i,j] \leftarrow \max(B_a[i,j] + q_a - q, B_b[i,j] + q_b - q)$ for $i \in [l], j \in [b]$
    return compress$(B, q)$

function estimate($(B, q) : \mathcal{S}$) : $\mathbb{R}$
    for $i \in [l]$
        $s \leftarrow \max(0, \max\{B[i,j] + q \mid |j \in [b]\} - \operatorname{ld} b + 9)$
        $p \leftarrow |\{j \in [b] | B[i,j] + q \geq s\}|$
        $Y_i \leftarrow 2^s \ln(1 - pb^{-1})(\ln(1 - b^{-1}))^{-1}$
    return median$(Y_0, \ldots, Y_{l-1})$
```

5.1 History-Independence

As mentioned in the introduction, this algorithm is history-independent, meaning that given the initial coin flips, it will reach the same state no matter in which permutation or frequency the stream elements are encountered. More precisely, the final state only depends on the set of encountered distinct elements over the execution tree and the initial coin flips, but not the shape of the tree. Informally, this is easy to see because the chosen cut-off level is the smallest possible with respect to the size of the values in the bins, and that property is maintained because the values in the bins are monotonically increasing with respect to the set of elements in the execution tree. Nevertheless, let us prove the property more rigorously. Let $\omega \in \Omega$ be the initial coin flips. Then there is a function $\tau(\omega, A)$ fulfilling the equations:

$$\begin{aligned} \mathrm{single}(\omega, x) &= \tau(\omega, \{x\}) && (1) \\ \mathrm{merge}(\tau(\omega, A), \tau(\omega, B)) &= \tau(\omega, A \cup B) && (2) \end{aligned}$$

The function τ is defined as follows:

$$\begin{aligned} \tau_0((f,g,h),A) &:= j \to \max\{f(a) \mid a \in A \wedge h(g(a)) = j\} \cup \{-1\} \\ \tau_1(\psi, A, q) &:= j \to \max\{\tau_0(\psi, A) - q, -1\} \\ \tau_2(\omega, A, q) &:= (i,j) \to \tau_1(\omega_i, A, q)[j] \\ q(\omega, A) &:= \min\left\{q \geq 0 \,\middle|\, \textstyle\sum_{i \in [l], j \in [b]} \lfloor \mathrm{ld}(\tau_2(\omega, A, q)[i,j] + 2) \rfloor \leq C_5 bl \right\} \\ \tau_3(\omega, A, q) &:= (\tau_2(\omega, A, q), q) \\ \tau(\omega, A) &:= \tau_3(\omega, A, q(\omega, A)) \end{aligned}$$

The function τ_0 describes the values in the bins if there were no compression, i.e., when $q = 0$. The function τ_1 describes the same for the given cut-off level q. Both are with respect to the selected hash functions $\psi = (f, g, h)$. The function τ_2 represents the state of all tables based on a seed for the expander. The next function τ_3 represents the entire state, which consists of the tables and the cut-off level. The function q represents the actual cut-off level that the algorithm would choose based on the values in the bins. Finally, the full state is described by the function τ for a given seed ω and set of elements A.

▶ **Lemma 6.** *Equations 1 and 2 hold for all $\omega \in \Omega$ and $\emptyset \neq A \subset [n]$.*

Proof. Let us also introduce the algorithms merge_1 and single_1. These are the algorithms merge and single but without the final compression step.

The following properties follow elementarily[4] from the definition of τ, s and the algorithms:

(i) $\tau(\omega, A) = \mathrm{compress}(\tau_3(\omega, A, q))$ for all $0 \leq q \leq q(\omega, A)$
(ii) $\tau_3(\omega, A_1 \cup A_2, \max(q(\omega, A_1), q(\omega, A_2))) = \mathrm{merge}_1(\tau(\omega, A_1), \tau(\omega, A_2))$
(iii) $\tau_3(\{x\}, 0) = \mathrm{single}_1(\omega, x)$
(iv) $q(\omega, A_1) \leq q(\omega, A_2)$ if $A_1 \subseteq A_2$
(v) $q(A) \geq 0$

To verify Eq. 1 we can use i, iii and v and to verify Eq. 2 we use i, ii taking into account that $\max(q(\omega, A_1), q(\omega, A_2)) \leq q(\omega, A_1 \cup A_2)$ because of iv. ◀

[4] The verification relies on the semi-lattice properties of the max operator, as well as its translation invariance (i.e. $\max(a + c, b + c) = \max(a, b) + c$).

5.2 Overall Proof

Because of the argument in the previous section, $\tau(\omega, A)$ will be the state reached after any execution tree over the set A and the initial coin flips, i.e., $\omega \in \Omega$. Hence for the correctness of the algorithm, we only need to show that:

▶ **Theorem 7.** *Let $\emptyset \neq A \subseteq [n]$ then $\mathcal{P}_{\omega \in U(\Omega)} (\text{estimate}(\tau(\omega, A)) - |A| > \varepsilon\, |A|) \leq \delta$.*

Proof: Postponed. This will be shown in two steps: First, we want to establish that the cut-off threshold q will be equal to or smaller than $q_{\max} := \max(0, \lceil \text{ld}\, |A| \rceil - \text{ld}\, b)$ with high probability. And if the latter is true, then the estimate will be within the desired accuracy with high probability. For the second part, we verify that the estimation step will succeed with high probability for all $0 \leq q \leq q_{\max}$. (This will be because the sub-sampling threshold s in the estimation step will be $\geq q_{\max}$ with high probability.)

For the remainder of this section, let $\emptyset \neq A \subset [n]$ be fixed and we will usually omit the dependency on A. For example, we will write $\tau(\omega)$ instead of $\tau(\omega, A)$. Then, we can express the decomposition discussed above using the following chain:

$$\begin{aligned}&\mathcal{P}_{\omega \in \Omega} (|\text{estimate}(\tau(\omega)) - |A|| > \varepsilon\, |A|) \leq \\ &\mathcal{P}_{\omega \in \Omega} (\exists q \leq q_{\max}.\, |\text{estimate}(\tau_2(\omega, q)) - |A|| > \varepsilon\, |A| \vee q(\omega) > q_{\max}) \leq \frac{\delta}{2} + \frac{\delta}{2}\end{aligned} \tag{3}$$

Thus we only have to show the following two inequalities:

- $\mathcal{P}_{\omega \in \Omega} (q(\omega) > q_{\max}) \leq \frac{\delta}{2}$
- $\mathcal{P}_{\omega \in \Omega} (\exists q \leq q_{\max}.\, |\text{estimate}(\tau_2(\omega, q)) - |A|| > \varepsilon\, |A|) \leq \frac{\delta}{2}$

The first will be shown in the following subsection, and the next in the subsequent one.

5.3 Cut-off Level

This subsection proves that the cut-off level will be smaller than or equal to $q_{\max}$. This is the part where the tail estimate for sub-gaussian random variables over expander walks (Lemma 5) is applied:

▶ **Lemma 8.** $\mathcal{P}_{\omega \in \Omega} (q(\omega) > q_{\max}) \leq \frac{\delta}{2}$

Proof. Let us make a few preliminary observations:

$$\lfloor \text{ld}(x+2) \rfloor \leq \text{ld}(x+2) \leq (c+2) + \max(x - 2^c, 0) \text{ for } (-1) \leq x \in \mathbb{R} \text{ and } c \in \mathbb{N}. \tag{4}$$

This can be verified using case distinction over $x \geq 2^c + 2$.

$$\mathbb{E}_{f \sim \mathcal{G}_2([n])} \max(f(a) - q_{\max} - 2^c, 0) \leq 2^{-q_{\max}} 2^{-2^c} \text{ for all } a \in [n] \text{ and } c \in \mathbb{N} \tag{5}$$

Note that this relies on the fact f is geometrically distributed.

$$|A|\, b^{-1} 2^{-q_{\max}} \leq 1 \tag{6}$$

This follows from the definition of $q_{\max}$ via case distinction.

To establish the result, we should take into account that $q(\omega)$ is the smallest cut-off level q fulfilling the inequality: $\sum_{i \in [l], j \in [b]} \lfloor \text{ld}(\tau_2(\omega, q)[i,j] + 2) \rfloor \leq C_5 bl$. In particular, if the inequality is true for $q_{\max}$, then we can conclude that $q(\omega)$ is at most $q_{\max}$, i.e.:

$$\mathcal{P}_{\omega \in \Omega} (q(\omega) > q_{\max}) = \mathcal{P}_{\omega \in \Omega} \left(\sum_{i \in [l], j \in [b]} \lfloor \text{ld}(\tau_2(\omega, q_{\max})[i,j] + 2) \rfloor > C_5 bl \right) \tag{7}$$

Let us introduce the random variable X over the seed space Ψ. It describes the space usage of a single column of the table B:

$$X(\psi) := \sum_{j\in[b]} \lfloor \mathrm{ld}(\tau_1(\psi, q_{\max})[j] + 2) \rfloor$$

Which can be approximated using Eq. 4 as follows:

$$X(\psi) \le \sum_{j\in[b]} c + 2 + \max(\tau_1(\psi, q_{\max})[j] - 2^c, 0) = \sum_{j\in[b]} c + 2 + \max(\tau_0(\psi)[j] - q_{\max} - 2^c, 0)$$

for all $0 \le c \in \mathbb{N}$. Hence:

$$\begin{aligned}
&\mathcal{P}_{\psi\sim\Psi}\left(X(\psi) \ge (c+3)b\right) \le \mathcal{P}_{\psi\sim\Psi}\left(\textstyle\sum_{j\in[b]} \max(\tau_0(\psi)[j] - q_{\max} - 2^c, 0) \ge b\right) \le \\
&\mathcal{P}_{(f,g,h)\sim\Psi}\left(\textstyle\sum_{j\in[b]} \max\{f(a) - q_{\max} - 2^c \mid a \in A \wedge h(g(a)) = j\} \cup \{0\} \ge b\right) \le \\
&\mathcal{P}_{(f,g,h)\sim\Psi}\left(\textstyle\sum_{a\in A} \max(f(a) - q_{\max} - 2^c, 0) \ge b\right) \le \\
&b^{-1} \sum_{a\in A} \mathbb{E}_{(f,g,h)\sim\Psi} \max(f(a) - q_{\max} - 2^c, 0) \le b^{-1}\, |A|\, 2^{-q_{\max}} 2^{-2^c} \le 2^{-2^c}
\end{aligned}$$

where the third and second-last inequality follow from Eq. 5 and 6. It is straightforward to conclude from the latter that for *all* $20 \le x \in \mathbb{R}$:

$$\mathcal{P}_{\psi\sim\Psi}\left(\frac{X(\psi)}{b} - 3 \ge x\right) \le \mathcal{P}_{\psi\sim\Psi}\left(X(\psi) \ge b(\lfloor x\rfloor + 3)\right) \le \exp(-2^{\lfloor x\rfloor} \ln 2) \le e^{-x(\ln x)^3}$$

Hence, it is possible to apply Lemma 5 on the random variables $b^{-1}X(\psi) - 3$ obtaining:

$$\mathcal{P}_{\omega\in\Omega}\left(\textstyle\sum_{i\in[l]} b^{-1}X(h(\omega, i)) - 3 \ge C_1 l\right) \le \exp(-l) \le \frac{\delta}{2}$$

This lemma now follows using $C_5 \ge C_1 + 3$ and that $\sum_{i\in[l]} X(h(\omega, i)) \le C_5 bl$ implies $q(\omega) \le q_{\max}$ as discussed at the beginning of the proof (Eq. 7). ◀

5.4 Accuracy

Let us introduce the random variables:

$$\begin{aligned}
t(f) &:= \max\{f(a) \mid a \in A\} - \mathrm{ld}\, b + 9 & s(f) &:= \max(0, t(f)) \\
p(f, g, h) &:= |\{j \in [b] \mid \tau_1((f, g, h), 0)[j] \ge s(f)\}| & Y(f, g, h) &:= 2^{s(f)} \rho^{-1}(p(f, g, h))
\end{aligned}$$

where $\rho(x) := b(1 - (1 - b^{-1})^x)$ – the expected number of hit bins when x balls are thrown into b bins. Note that the definitions t, p and Y correspond to the terms within the loop in the estimate function under the condition that the approximation threshold q is 0. In particular: $\mathrm{estimate}(\tau_3(\omega, 0)) = \mathrm{median}_{i\in[l]} Y(\omega_i)$ for $\omega \in \Omega$. Moreover, we denote by $R(f)$ the set of elements in A whose level is above the sub-sampling threshold, i.e.: $R(f) := \{a \in A \mid f(a) \ge s(f)\}$. The objective is to show that the individual estimates obtained in the loop in the estimate function (assuming $q = 0$) have the right accuracy and that the threshold $s \ge q_{\max}$ with high probability, i.e.:

$$\mathcal{P}_{\psi\sim\Psi}\left(|Y(\psi) - |A|| > \varepsilon\, |A| \vee s(f) < q_{\max}\right) \le \frac{1}{16} \tag{8}$$

In Lemma 14 this will be generalized to $0 \le q \le q_{\max}$. To be able to establish a bound on the above event, we need to check the likelihood of the following 4 events:

- The computed sub-sampling threshold $s(f)$ is approximately $\operatorname{ld}(|A|)$.
- The size of the sub-sampled elements $R(f)$ is a good approximation of $2^{-s(f)}|A|$.
- There is no collision during the application of g on the sub-sampled elements $R(f)$.
- The count of elements above the sub-sampling threshold in the table is close to the expected number $\rho(R(f))$ (taking collisions due to the application of h into account).

Then it will be possible to conclude that one of the above must fail if the approximation is incorrect. More formally:

$$E_1(\psi) :\leftrightarrow 2^{-16}b \le 2^{-t(f)}|A| \le 2^{-1}b \qquad E_2(\psi) :\leftrightarrow \left||R(f)| - 2^{-s(f)}|A|\right| \le \tfrac{\varepsilon}{3}2^{-s(f)}|A|$$
$$E_3(\psi) :\leftrightarrow \forall a \ne b \in R(f). g(a) \ne g(b) \qquad E_4(\psi) :\leftrightarrow |p(\psi) - \rho(|R(f)|)| \le \tfrac{\varepsilon}{12}|R(f)|$$

for $\psi = (f, g, h) \in \Psi$. The goal is to show all four events happen simultaneously w.h.p.:

$$\mathcal{P}_{\psi\sim\Psi}(\neg E_1(\psi) \vee \neg E_2(\psi) \vee \neg E_3(\psi) \vee \neg E_4(\psi)) \le \frac{1}{16} \tag{9}$$

which can be shown by verifying: $\mathcal{P}_{\psi\sim\Psi}\left(\bigwedge_{j<i} E_j(\psi) \wedge \neg E_i(\psi)\right) \le 2^{-6}$ for each $i \in \{1, \ldots, 4\}$. Let us start with the $i = 1$ case:

▶ **Lemma 9.** $\mathcal{P}_{\psi\in\Psi}(\neg E_1(\psi)) \le 2^{-6}$

Proof. For $X(f) = \max\{f(a) \mid a \in A\}$ it is possible to show:

$$\mathcal{P}_{(f,g,h)\sim\Psi}(X(f) < \operatorname{ld}(|A|) - k - 1) \le 2^{-k} \qquad \mathcal{P}_{(f,g,h)\sim\Psi}(X(f) > \operatorname{ld}(|A|) + k) \le 2^{-k}$$

using the proof for the F_0 algorithm by Alon et al. [4][Proposition 2.3]. The desired result follows taking $k = 7$ and that $t(f) = X(f) - \operatorname{ld} b + 9$. ◀

The following lemma is the interesting part of the proof in this subsection. In previous work, the sub-sampling threshold is obtained using a separate parallel algorithm, which has the benefit that it is straightforward to verify that $|R(f)|$ approximates $2^{-s}|A|$. The drawback is, of course, additional algorithmic complexity and an additional independent hash function. However, in the solution presented here, the threshold is determined from the data to be sub-sampled itself, which means it is not possible to assume independence. The solution to the problem is to show that $|R(f)|$ approximates $2^{-s}|A|$ with high probability for *all* possible values $s(f)$ assuming E_1.

▶ **Lemma 10.** $L := \mathcal{P}_{\psi\sim\Psi}(E_1(\psi) \wedge \neg E_2(\psi)) \le 2^{-6}$

Proof. Let $r(f, t) := |\{a \in A \mid f(a) \ge t\}|$ and $t_{\max}$ be maximal, s.t. $2^{-16}b \le 2^{-t_{\max}}|A|$. Then $2^7 \le \frac{\varepsilon^2}{9}2^{-16}b \le \frac{\varepsilon^2}{9}2^{-t_{\max}}|A|$. Hence: $2^{7+t_{\max}-t} \le \frac{\varepsilon^2}{9}2^{-t}|A| = \frac{\varepsilon^2}{9}\mathbb{E}\, r(f, t)$. Thus:

$$2^{7+t_{\max}-t}\,\mathbb{V}\, r(f, t) \le 2^{7+t_{\max}-t}\,\mathbb{E}\, r(f, t) \le \frac{\varepsilon^2}{9}(\mathbb{E}\, r(f, t))^2$$

for all $0 < t \le t_{\max}$. (This may be a void statement if $t_{\max} \le 0$.) Hence:

$$\mathcal{P}_{(f,g,h)\in\Psi}\left(\exists t. 0 < t \le t_{\max} \wedge |r(f, t) - \mathbb{E}\, r(\cdot, t)| > \frac{\varepsilon}{3}\mathbb{E}\, r(\cdot, t)\right) \le$$
$$\sum_{t=1}^{t_{\max}} \mathcal{P}_{(f,g,h)\in\Psi}\left(|r(f, t) - \mathbb{E}\, r(\cdot, t)| > \sqrt{2^{7+t_{\max}-t}\,\mathbb{V}\, r(f, t)}\right) \le \sum_{t=1}^{t_{\max}} 2^{-7-t_{\max}+t} \le 2^{-6}$$

Note that the predicate $E_2(\psi)$ is always true if $s(f) = 0$ because, in that case, there is no sub-sampling, i.e., $|R(f)| = |A|$. On the other hand if $s(f) > 0$, then $s(f) = t(f) \leq t_{\max}$ assuming $E_1(\psi)$. Hence:

$$\begin{aligned} L &\leq \mathcal{P}_{(f,g,h)} \left(s(f) > 0 \wedge E_1(f,g,h) \wedge \neg E_2(f,g,h) \right) \\ &\leq \mathcal{P}_{(f,g,h)} \left(0 < t(f) \leq t_{\max} \wedge \left| |R(f)| - 2^{-t(f)} |A| \right| > \tfrac{\varepsilon}{3} 2^{-t(f)} |A| \right) \\ &\leq \mathcal{P}_{(f,g,h)} \left(0 < t(f) \leq t_{\max} \wedge \left| r(f,t(f)) - 2^{-t(f)} |A| \right| > \tfrac{\varepsilon}{3} 2^{-t(f)} |A| \right) \leq 2^{-6} \end{aligned}$$

where the last step follows from the previous equation. ◀

$$\text{Note that: } E_1(f,g,h) \wedge E_2(f,g,h) \rightarrow |R(f)| \leq \frac{2}{3} b \text{ for } (f,g,h) \in \Psi \qquad (10)$$

▶ **Lemma 11.** $L := \mathcal{P}_{\psi \sim \Psi}(E_1(\psi) \wedge E_2(\psi) \wedge \neg E_3(\psi)) \leq 2^{-6}$

Proof. Using Eq. 10 we can conclude:

$$\begin{aligned} L &\leq \mathcal{P}_{(f,g,h) \sim \Psi} \left(|R(f)| \leq b \wedge (\exists a < b \in R(f). g(a) = g(b)) \right) \\ &\leq \int_{\mathcal{G}_2([n])} [|R(f)| \leq b] \, \mathcal{P}_{g \sim \mathcal{H}_2([n],[C_7 b^2])} (\exists a < b \in R(f). g(a) = g(b)) \, df \\ &\leq \int_{\mathcal{G}_2([n])} [|R(f)| \leq b] \sum_{a < b \in R(f)} \mathcal{P}_{g \sim \mathcal{H}_2([n],[C_7 b^2])} (g(a) = g(b)) \, df \\ &\leq \int_{\mathcal{G}_2([n])} \frac{b(b-1)}{2 C_7 b^2} \, df \leq \frac{1}{2 C_7} = 2^{-6}. \end{aligned}$$

◀

▶ **Lemma 12.** $L := \mathcal{P}_{\psi \sim \Psi}(E_1(\psi) \wedge E_2(\psi) \wedge E_3(\psi) \wedge \neg E_4(\psi)) \leq 2^{-6}$

Proof. Let $\tilde{R}(f,g,h) = \{ i \in [C_7 b^2] \mid f(a) \geq t(f) \wedge g(a) = i \wedge a \in A \}$ denote the indices hit in the domain $[C_7 b^2]$ by the application of g on the elements above the sub-sampling threshold. If $E_3(f,g,h)$, then $\left| \tilde{R}(f,g,h) \right| = |R(f)|$ and if $E_1(f,g,h) \wedge E_2(f,g,h)$,then $|R(f)| \leq b$ (see Eq. 10). Recalling that $p(\psi)$ is the number of bins hit by the application of k-independent family from $\tilde{R}(\psi) \subseteq [C_7 b^2]$ to $[b]$ we can apply Lemma 20. This implies:

$$\begin{aligned} &\mathcal{P}_{(f,g,h) \sim \Psi} \left(\textstyle\bigwedge_{i \in \{1,2,3\}} E_i(f,g,h) \wedge |p(f,g,h) - \rho(|R(f)|)| \geq \frac{\varepsilon}{12} |R(f)| \right) \leq \\ &\mathcal{P}_{\psi \sim \Psi} \left(\left| \tilde{R}(\psi) \right| \leq b \wedge \left| p(\psi) - \rho\left(\left| \tilde{R}(\psi) \right| \right) \right| \geq \frac{\varepsilon}{12} \left| \tilde{R}(\psi) \right| \right) \leq \\ &\mathcal{P}_{\psi \sim \Psi} \left(\left| \tilde{R}(\psi) \right| \leq b \wedge \left| p(\psi) - \rho\left(\left| \tilde{R}(\psi) \right| \right) \right| \geq 9 b^{-1/2} \left| \tilde{R}(\psi) \right| \right) \leq 2^{-6} \end{aligned}$$

where we used, that $b \geq 9^2 12^2 \varepsilon^{-2}$ (i.e. $C_4 >= 9^2 12^2$). ◀

▶ **Lemma 13.** *Equation 8 is true.*

Proof. Let us start by observing that $E_1(\psi) \wedge E_2(\psi) \wedge E_4(\psi) \rightarrow |A^*(\psi) - |A|| \leq \varepsilon |A|$. This is basically an error propagation argument. First note that by using Eq. 10: $p(f,g,h) \leq \rho(R(f)) + \frac{\varepsilon}{12} |R(f)| \leq \rho(\frac{2}{3} b) + \frac{1}{12} |R(f)| \leq \frac{41}{60} b$. Moreover, using the mean value theorem:

$$\left| \rho^{-1}(p(f,g,h)) - |R(f)| \right| = (\rho^{-1})'(\xi) \, |p(f,g,h) - \rho(|R(f)|)| \leq \tfrac{\varepsilon}{3} |R(f)|$$

for some ξ between $\rho(|B(f)|)$ and $p(f,g,h)$ where we can approximate $(\rho^{-1})'(\xi) < 4$. Hence:

$$\begin{aligned}\left|\rho^{-1}(p(f,g,h)) - 2^{-s(f)}\,|A|\right| &\leq \left|\rho^{-1}(p(f,g,h)) - |R(f)|\right| + \left||R(f)| - 2^{-s(f)}\,|A|\right| \\ &\leq \frac{\varepsilon}{3}\,|R(f)| + \left||R(f)| - 2^{-s(f)}\,|A|\right| \\ &\leq \left(\frac{2\varepsilon}{3} + \frac{\varepsilon^2}{9}\right) 2^{-s(f)}\,|A| \leq \varepsilon 2^{-s(f)}\,|A|\end{aligned}$$

It is also possible to deduce that $E_1(f,g,h) \to t(f) \geq \lceil \mathrm{ld}(|A|)\rceil - \mathrm{ld}\, b \to s(f) \geq q_{\max}$. Using Lemma 9 to 12 we can conclude that Equation 9 is true. And the implications derived here show that then Equation 8 must be true as well. ◀

To extend the previous result to the case: $q \leq q_{\max}$, let us introduce the random variables:

$$\begin{aligned} t_c(\psi,q) &:= \max\{\tau_1(\psi,q)[j] + q \mid j \in [b]\} - \mathrm{ld}\, b + 9 & s_c(\psi,q) &:= \max(0, t_c(\psi,q)) \\ p_c(\psi,q) &:= |\{j \in [b] \mid \tau_1(\psi,q)[j] + q \geq s_c(\psi,q)\}| & Y_c(\psi,q) &:= 2^{s_c(\psi,q)}\rho^{-1}(p_c(\psi,q))\end{aligned}$$

These definitions t_c, p_c and Y_c correspond to the terms within the loop in the estimate function for arbitrary q.

▶ **Lemma 14.** $\mathcal{P}_{\psi\sim\Psi}\left(\exists q \leq q_{\max}.\, |Y_c(\psi,q) - |A|| > \varepsilon\,|A|\right) \leq \frac{1}{16}$

Proof. It is possible to see that $t_c(\psi,q) = t(\psi)$ if $q \leq t(\psi)$. This is because $\tau_1(\psi,q) + q$ and $\tau_1(\psi,0)$ are equal except for values strictly smaller than q. With a case distinction on $t(\psi) \geq 0$ it is also possible to deduce that $s(\psi,q) = s(\psi)$ if $q \leq s(\psi)$. Hence: $p_c(\psi,q) = p(\psi)$ and $Y_c(\psi,q) = Y(\psi)$ (for $q \leq s(\psi)$). Thus this lemma is a consequence of Lemma 13. ◀

▶ **Lemma 15.** $L := \mathcal{P}_{\omega\in\Omega}\left(\exists q \leq q_{\max}.\, |\mathrm{estimate}(\tau_2(\omega,q)) - |A|| > \varepsilon\,|A|\right) \leq \frac{\delta}{2}$

Proof. Because the median of a sequence will certainly be in an interval, if more than half of the elements are in it, we can approximate the left-hand side as:

$$\begin{aligned} L &\leq \mathcal{P}_{\omega\in\Omega}\left(\exists q \leq q_{\max}.\, \sum_{i\in[l]} [|Y(\omega_i,q) - |A|| > \varepsilon\,|A|] \geq \frac{l}{2}\right) \\ &\leq \exp\left(-l\left(\frac{1}{2}\ln\left(\left(\frac{1}{16}+\frac{1}{16}\right)^{-1}\right) - 2e^{-1}\right)\right) \leq \exp\left(-\frac{l}{4}\right) \leq \frac{\delta}{2}\end{aligned}$$

The second inequality follows from Lemma 4 and 14 as well as $\lambda \leq \frac{1}{16}$. ◀

Proof of Theorem 7. Follows from Lemma 8 and the previous lemma, as well as the reasoning established in Equation 3. ◀

5.5 Space Usage

It should be noted that the data structure requires an efficient storage mechanism for the levels in the bins. We need to store the table values in a manner in which the number of bits required for a value x is proportional to $\ln x$. A simple strategy would be to store each value using a prefix-free universal code and concatenating the encoded variable-length bit strings.[5] A well-known universal code for positive integers is the Elias-gamma code, which

[5] Note that a vector of prefix-free values can be decoded even if they are just concatenated.

requires $2\lfloor \operatorname{ld} x \rfloor + 1$ bits for $x \geq 1$ [13]. Since, in our case, the values are integers larger or equal to (-1), they can be encoded using $2\lfloor \operatorname{ld}(x+2) \rfloor + 1$ bits.[6] In combination with the condition established in the compress function of Algorithm 1 the space usage for the table is thus $(2C_5 + 1)bl \in \mathcal{O}(bl) \subseteq \mathcal{O}(\ln(\delta^{-1})\varepsilon^2)$. Additionally, the approximation threshold needs to be stored. This threshold is a non-negative integer between 0 and $\operatorname{ld} n$ requiring $\mathcal{O}(\ln \ln n)$ bits to store. In summary, the space required for the sketch is $\mathcal{O}(\ln(\delta^{-1})\varepsilon^2 + \ln \ln n)$. For the coin flips, we need to store a random choice from Ω, i.e., we need to store $\ln(|\Omega|)$ bits. The latter is in $\mathcal{O}(\ln(|\Omega|)) \subseteq \mathcal{O}(\ln(|\Psi|) + l \ln(\lambda^{-1})) \subseteq \mathcal{O}(\ln n + \ln(\varepsilon^{-1})^2 + \ln(\delta^{-1})^3)$. Overall the total space for the coin flips and the sketch is $\mathcal{O}(\ln(\delta^{-1})\varepsilon^{-2} + \ln n + \ln(\delta^{-1})^3)$.

6 Extension to small failure probabilities

The data structure described in the previous section has a space complexity that is close but exceeds the optimal $\mathcal{O}(\ln(\delta^{-1})\varepsilon^{-2} + \ln n)$. The main reason this happens is that, with increasing length of the random walk, the spectral gap of the expander is increasing as well – motivated by the application of Lemma 5 in Subsection 5.3, with which we could establish that the cut-level could be shared between all tables. A natural idea is to restrict that.

If δ^{-1} is smaller than $\ln n$ the term $(\ln(\delta^{-1}))^3$ in the complexity of the algorithm is not a problem because it is dominated by the $\ln n$ term. If it is larger, we can split the table into sub-groups and introduce multiple cut-levels. Hence a single cut-level would be responsible for a smaller count of tables, and thus the spectral gap would be lower. (See also Figure 2).

A succinct way to precisely prove the correctness of the proposal is to repeat the previous algorithm, which has only a single shared cut-level, in a black-box manner for the same universe size and accuracy but for a higher failure probability. The seeds of each repetition are selected again using an expander walk. Here the advantage of Lemma 4 is welcome, as the inner algorithm needs to have a failure probability depending on n – the natural choice is $(\ln n)^{-1}$. The length of the walk of the inner algorithm matches the number of bits of the cut-level $\mathcal{O}(\ln \ln n)$. The repetition count of the outer algorithm is then $\mathcal{O}\left(\frac{\ln(\delta^{-1})}{\ln \ln n}\right)$.

▶ **Theorem 16.** *Let $n > 0$, $0 < \varepsilon < 1$ and $0 < \delta < 1$. Then there exists a cardinality estimation data structure for the universe $[n]$ with relative accuracy ε and failure probability δ with space usage $\mathcal{O}(\ln(\delta^{-1})\varepsilon^{-2} + \ln n)$.*

Proof. If $\delta^{-1} < \ln n$, then the result follows from Theorem 7 and the calculation in Subsection 5.5. Moreover, if $n < \exp(e^5)$, then the theorem is trivially true, because there is an exact algorithm with space usage $\exp(e^5) \in \mathcal{O}(1)$. Hence we can assume $e^5 \leq \ln n \leq \delta^{-1}$. Let Ω^*, single*, merge* and estimate* denote the seed space and the API of Algorithm 1 for the universe $[n]$, relative accuracy ε and failure probability $\delta^* := (\ln n)^{-1}$. Moreover, let $m := \left\lceil 4\frac{\ln(\delta^{-1})}{\ln \ln n} \right\rceil$ – the plan is to show that with these definitions Algorithm 2 fulfills the conditions of this theorem. Let $\nu(\theta, A)[i] := \tau^*(\theta_i, A)$ for $i \in [m]$ and $\theta \in \Theta := U(\mathcal{E}(\Omega^*, \delta^*, m))$. Then it is straightforward to check that:

$$\operatorname{single}(\theta, x) = \nu(\theta, \{x\}) \qquad \operatorname{merge}(\nu(\theta, A), \nu(\theta, B)) = \nu(\theta, A \cup B)$$

for $x \in [n]$ and $\emptyset \neq A, B \subseteq [n]$ taking into account Lemma 6. Hence the correctness follows if: $\mathcal{P}_{\theta \in \Theta}(|\operatorname{estimate}(\nu(\theta, A)) - |A|| > \varepsilon |A|) \leq \delta$. Because the estimate is the median of the individual estimates, this is true if at least half of the individual estimates are in the desired range. Similar to the proof of Lemma 15 we can apply Lemma 4. This works if

[6] There are more sophisticated strategies for representing a sequence of variable-length strings that allow random access. [7]

$$\exp\left(-m\left(\frac{1}{2}\ln\left((\delta^* + \delta^*)^{-1}\right) - 2e^{-1}\right)\right) \leq \delta$$

which follows from $m \geq 4\ln(\delta^{-1})(\ln\ln n)^{-1}$ and $\ln\ln n \geq 5$. The space usage for the seed is: $\ln|\Theta| \in \mathcal{O}(\ln n + \ln(\varepsilon^{-1})^2 + (\ln((\delta^*)^{-1}))^3 + m\ln((\delta^*)^{-1})) \subseteq \mathcal{O}(\ln n + \ln(\varepsilon^{-1})^2 + \ln(\delta^{-1}))$. The space usage for the sketch is: $\mathcal{O}(m\ln((\delta^*)^{-1})\varepsilon^{-2} + m\ln\ln n) \subseteq \mathcal{O}(\ln(\delta^{-1})\varepsilon^{-2} + \ln\ln n)$. ◀

Algorithm 2 Algorithm for $0 < \delta < (\ln n)^{-1}$.

```
function init() : $\Theta$
    return random $U(\Theta)$

function single($x : U, \theta : \Theta$) : $\mathcal{S}$
    $D[i] = \text{single}^*(x, \theta_i)$ for $i \in [m]$
    return $D$

function merge($D_a : \mathcal{S}, D_b : \mathcal{S}$) : $\mathcal{S}$
    $D[i] \leftarrow \text{merge}^*(D_a[i], D_b[i])$ for $i \in [m]$
    return $D$

function estimate($D : \mathcal{S}$) : $\mathbb{R}$
    $Y_i \leftarrow \text{estimate}^*(D[i])$ for $i \in [m]$
    return median($Y_0, \ldots, Y_{m-1}$)
```

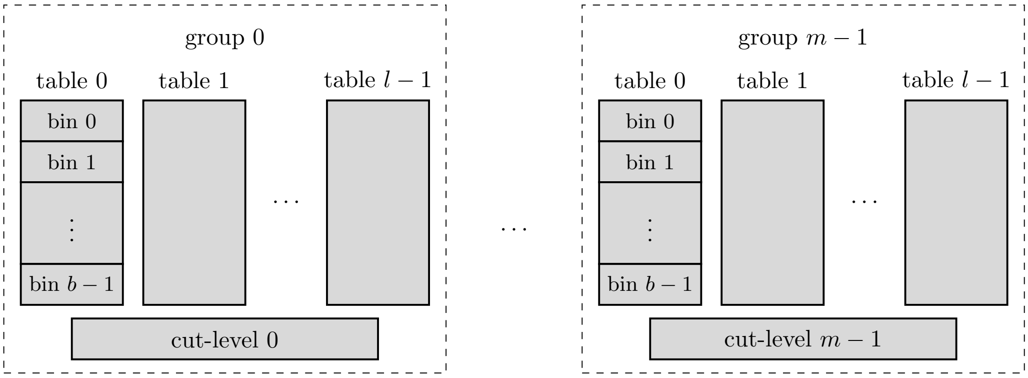


Figure 2 Schematic representation of the states of Algorithm 2 with $m \in \mathcal{O}\left(\frac{\ln(\delta^{-1})}{\ln\ln n}\right)$ repetitions of the inner algorithm. The inner algorithm uses $b \in \mathcal{O}(\varepsilon^{-2})$ bins and $l \in \mathcal{O}(\ln\ln n)$ tables.

7 Optimality

The optimality of the algorithm introduced by Błasiok [9] follows from the lower bound established by Jayram and Woodruff [23, Theorem 4.4]. The result (as well as its predecessors [4, 40]) follows from a reduction to a communication problem. This also means that their theorem is a lower bound on the information the algorithm needs to retain between processing successive stream elements.

An immediate follow-up question to Theorem 16 is whether the space usage is also optimal in the distributed setting. Let us assume there are p processes, each retaining m stream elements, and they are allowed to communicate at the beginning, before observing the stream elements, and after observing all stream elements. Even with these relaxed constraints, the number of bits that each process will need to maintain will be the same as the minimum number of bits of a sequential streaming solution. This follows by considering a specific subset of the input set where except for process 0, the stream elements on all the other processes are equal to the last stream element of process 0. In particular, the information the processes $1, 2, \ldots, p-1$ have is 0 bits from the perspective of process 0. If our distributed hypothetical algorithm is correct, it can only be so if the worst-case space usage per process is $\Omega(\ln(\delta^{-1})\varepsilon^{-2} + \ln n)$.

8 Conclusion

A summary of this work would be that for the space complexity of cardinality estimation algorithms, there is no gap between the distributed and sequential streaming models. Moreover, it is possible to solve the problem optimally (in either model) with expander graphs and hash families without using code-based extractors (as they were used in previous work). The main algorithmic idea is to avoid using a separate rough estimation data structure for quantization (cut-off); instead, the cut-off is guided by the space usage. During the estimation step at the end, an independent rough estimate is still derived, but it may be distinct from the cut-off reached at that point. This is the main difference between this solution and the approach by Kane et al. [24]. The main mathematical idea is to take the tail estimate based on the Kullback-Leibler divergence for random walks on expander graphs, first noted by Impagliazzo and Kabanets [22, Th. 10] seriously. With which, it is possible to achieve a failure probability of δ using $\mathcal{O}\left(\frac{\ln(\delta^{-1})}{\ln((\delta^*)^{-1})}\right)$ repetitions of an inner algorithm with a failure probability $\delta^* > \delta$. Note that the same cannot be done with the standard Gillman-type Chernoff [18] bounds. This allows the two-stage expander construction that we needed. As far as I can tell, this strategy is new and has not been used before.

An interesting question is whether the two-stage expander construction can somehow be collapsed into a single stage. For that, it is best to consider the following non-symmetric aggregate:

$$\mathcal{P}_{\omega \in \mathcal{E}(\mathcal{E}(S, \exp(-l(\ln l)^3), l), \exp(-l/m), m)} \left(\sum_{i \in [m]} \left[\sum_{j \in [l]} X(\omega_{ij}) \geq C_1 \right] \geq \frac{m}{2} \right) \leq \exp(-\mathcal{O}(lm))$$

where X may be an unbounded random variable with, e.g., sub-gaussian distribution. Indeed, the bound on the count of too-large cut-off values from Algorithm 2 turns out to be a tail estimate of the above form. I tried to obtain such a bound using only a single-stage expander walk but did not succeed without requiring too large spectral gaps, i.e., with $\lambda^{-1} \in \mathcal{O}(1)$ for $m \ll l$. There is a long list of results on more advanced Chernoff bounds for expander walks [2, 28, 30, 34, 35, 38] and investigations into more general aggregation (instead of summation) functions [10, 16, 19, 20, 32, 36], but I could not use any of these results/approaches to avoid the two-stage construction. This suggests that either there are more advanced results to be found or multi-stage expander walks are inherently more powerful than single-stage walks.

References

1 Archive of Formal Proofs. https://isa-afp.org. Accessed: 2023-07-03.

2 Rohit Agrawal. Samplers and Extractors for Unbounded Functions. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*, volume 145 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 59:1–59:21, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.59.

3 Noga Alon, Uriel Feige, Avi Wigderson, and David Zuckerman. Derandomized graph products. *computational complexity*, 5:60–75, 1995. doi:10.1007/BF01277956.

4 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999. doi:10.1006/jcss.1997.1545.

5 Frank J. Balbach. The cook-levin theorem. *Archive of Formal Proofs*, January 2023. , Formal proof development. URL: https://isa-afp.org/entries/Cook_Levin.html.

6 Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *Randomization and Approximation Techniques in Computer Science*, pages 1–10. Springer Berlin Heidelberg, 2002. doi:10.1007/3-540-45726-7_1.

7 Daniel K. Blandford and Guy E. Blelloch. Compact dictionaries for variable-length keys and data with applications. *ACM Trans. Algorithms*, 4(2), May 2008. doi:10.1145/1361192.1361194.

8 Jarosław Błasiok. Optimal streaming and tracking distinct elements with high probability. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 2432–2448, 2018. doi:10.1137/1.9781611975031.156.

9 Jarosław Błasiok. Optimal streaming and tracking distinct elements with high probability. *ACM Trans. Algorithms*, 16(1):3:1–3:28, 2020. doi:10.1145/3309193.

10 Gil Cohen, Noam Peri, and Amnon Ta-Shma. Expander random walks: A fourier-analytic approach. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 1643–1655, New York, NY, USA, 2021. doi:10.1145/3406325.3451049.

11 Jeffrey Dean and Sanjay Ghemawat. Mapreduce: A flexible data processing tool. *Commun. ACM*, 53(1):72–77, January 2010. doi:10.1145/1629175.1629198.

12 Manuel Eberl and Lawrence C. Paulson. The prime number theorem. *Archive of Formal Proofs*, September 2018. , Formal proof development. URL: https://isa-afp.org/entries/Prime_Number_Theorem.html.

13 P. Elias. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, 21(2):194–203, 1975.

14 Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985. doi:10.1016/0022-0000(85)90041-8.

15 Ian Foster. *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison-Wesley Longman Publishing Co., Inc., USA, 1995.

16 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. doi:10.1145/3188745.3188890.

17 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*, SPAA '01, pages 281–291, 2001. doi:10.1145/378580.378687.

18 David Gillman. A chernoff bound for random walks on expander graphs. *SIAM Journal on Computing*, 27(4):1203–1220, 1998. doi:10.1137/S0097539794268765.

19 Louis Golowich. A new berry-esseen theorem for expander walks. *Electron. Colloquium Comput. Complex.*, TR22, 2022.

20 Louis Golowich and Salil Vadhan. Pseudorandomness of expander random walks for symmetric functions and permutation branching programs. In *Proceedings of the 37th Computational Complexity Conference*, CCC '22, Dagstuhl, Germany, 2022. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CCC.2022.27.

21 Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced expanders and randomness extractors from parvaresh–vardy codes. *J. ACM*, 56(4), July 2009. doi:10.1145/1538902.1538904.

22 Russell Impagliazzo and Valentine Kabanets. Constructive proofs of concentration bounds. In Maria Serna, Ronen Shaltiel, Klaus Jansen, and José Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 617–631, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. doi:10.1007/978-3-642-15369-3_46.

23 T. S. Jayram and David P. Woodruff. Optimal bounds for johnson-lindenstrauss transforms and streaming problems with subconstant error. *ACM Trans. Algorithms*, 9(3), June 2013. doi:10.1145/2483699.2483706.

24 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 '10, pages 41–52, New York, 2010. doi:10.1145/1807085.1807094.

25 Emin Karayel. Distributed distinct elements. *Archive of Formal Proofs*, April 2023. , Formal proof development. URL: https://isa-afp.org/entries/Distributed_Distinct_Elements.html.

26 Emin Karayel. An embarrassingly parallel optimal-space cardinality estimation algorithm, 2023. arXiv:2307.00985.

27 Emin Karayel. Expander graphs. *Archive of Formal Proofs*, March 2023. , Formal proof development. URL: https://isa-afp.org/entries/Expander_Graphs.html.

28 Pascal Lezaud. Chernoff-type bound for finite Markov chains. *The Annals of Applied Probability*, 8(3):849–867, 1998. doi:10.1214/aoap/1028903453.

29 Jack Murtagh, Omer Reingold, Aaron Sidford, and Salil Vadhan. Deterministic Approximation of Random Walks in Small Space. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*, volume 145 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 42:1–42:22, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.42.

30 Assaf Naor, Shravas Rao, and Oded Regev. Concentration of markov chains with bounded moments. *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, 56(3):2270–2280, 2020. doi:10.1214/19-AIHP1039.

31 Tobias Nipkow, Lawrence C Paulson, and Markus Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Heidelberg, first edition, 2002.

32 Daniel Paulin. Concentration inequalities for Markov chains by Marton couplings and spectral methods. *Electronic Journal of Probability*, 20:1–32, 2015. doi:10.1214/EJP.v20-4039.

33 Seth Pettie and Dingyu Wang. Information theoretic limits of cardinality estimation: Fisher meets shannon. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 556–569, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3406325.3451032.

34 Shravas Rao. A hoeffding inequality for markov chains. *Electronic Communications in Probability*, 24:1–11, 2019. doi:10.1214/19-ECP219.

35 Shravas Rao and Oded Regev. A sharp tail bound for the expander random sampler, 2017. arXiv:1703.10205.

36 Omer Reingold, Thomas Steinke, and Salil Vadhan. Pseudorandomness for regular branching programs via fourier analysis. In Prasad Raghavendra, Sofya Raskhodnikova, Klaus Jansen, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 655–670, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. doi:10.1007/978-3-642-40328-6_45.

37 Salil P. Vadhan. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1-3):1–336, 2012. doi:10.1561/0400000010.

38 Roy Wagner. Tail estimates for sums of variables sampled by a random walk. *Comb. Probab. Comput.*, 17(2):307–316, March 2008. doi:10.1017/S0963548307008772.

39 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. doi:10.1016/0022-0000(81)90033-7.

40 David Woodruff. Optimal space lower bounds for all frequency moments. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 167–175, USA, 2004. Society for Industrial and Applied Mathematics.

A Proof of Lemma 5

▶ **Lemma 5** (Deviation Bound). *Let* $G = (V, E)$ *be a* λ*-expander graph and* $f : V \to \mathbb{R}_{\geq 0}$ *s.t.* $\mathcal{P}_{v\sim U(V)}(f(v) \geq x) \leq \exp(-x(\ln x)^3)$ *for* $x \geq 20$ *and* $\lambda \leq \exp(-l(\ln l)^3)$ *then*

$$\mathcal{P}_{w\sim \mathrm{Walk}(G,l)}\left(\textstyle\sum_{i\in[l]} f(w_i) \geq C_1 l\right) \leq \exp(-l)$$

where $C_1 := e^2 + e^3 + (e-1) \leq 30$.

Proof. Let $\mu_k := \mathbb{E}_{v\sim V}[e^k \leq f(v)] \leq \exp(-e^k k^3)$ for $k \geq 3$. We will show

$$L_k := \mathcal{P}_{w\sim \mathrm{Walk}(G,l)}\left(\sum_{i\in[l]} [e^k \leq f(w_i)] \geq l e^{-k} k^{-2}\right) \leq \exp(-l-k+2) \text{ for all } k \geq 3 \quad (11)$$

by case distinction on the range of k:

Case $k \geq \max(\ln l, 3)$: In this case the result follows using Markov's inequality. Note that the random walk starts from and remains in the stationary distribution, and thus for any index $i \in [l]$ the distribution of the i-th walks step w_i will be uniformly distributed over V, hence:

$$\begin{aligned} L_k &\leq e^k k^2 l^{-1}\,\mathbb{E}_{w\sim \mathrm{Walk}(G,l)} \textstyle\sum_{i\in[l]}[e^k \leq f(w_i)] = e^k k^2\,\mathbb{E}_{v\sim V}[e^k \leq f(v)] \\ &\leq e^k k^2 \exp(-e^k k^3) = \exp(k + 2\ln k - e^k k^3) \leq \exp(2k - e^k(k^2+2)) \\ &\leq \exp(2k - e^k k^2 - e^k - e^k) \leq \exp(-l-k+2) \end{aligned}$$

Here we use that $k^3 \geq k^2 + 2$ and $e^k \geq k$ for $k \geq 3$ and $e^k \geq l$.

Case $3 \leq k < \ln l$: Then we have

$$\begin{aligned} L_k &\leq \exp\left(-l(e^{-k}k^{-2}\ln((\mu_k+\lambda)^{-1}) - 2e^{-1})\right) \text{ using Lemma 4} \\ &\leq \exp\left(-l(e^{-k}k^{-2}(e^k k^3 - \ln 2) - 2e^{-1})\right) \leq \exp\left(-l(k - e^{-k}k^{-2}\ln 2 - 2e^{-1})\right) \\ &\leq \exp\left(-l(k-1)\right) \leq \exp\left(-l-k+2\right) \end{aligned}$$

Concluding the proof of Eq. 11.

Note that:

$$\begin{aligned} \sum_{i\in[l]} f(w_i) &\leq e^2 l + \sum_{i\in[l]}\sum_{k\geq 2} e^{k+1}[e^k \leq f(w_i) < e^{k+1}] \\ &\leq e^2 l + \sum_{i\in[l]}\left(\sum_{k\geq 2} e^{k+1}[e^k \leq f(w_i)] - \sum_{k\geq 2} e^{k+1}[e^{k+1} \leq f(w_i)]\right) \\ &\leq (e^2+e^3)l + (e-1)\sum_{i\in[l]}\left(\sum_{k\geq 3} e^k [e^k \leq f(w_i)]\right) \end{aligned}$$

Hence:

$$\begin{aligned}\mathcal{P}_{w\sim\mathrm{Walk}(G,l)}\left(\sum_{i\in[l]} f(w_i)\geq C_1 l\right) &\leq \mathcal{P}_{w\sim\mathrm{Walk}(G,l)}\left(\sum_{k\geq 3, i\in[l]} e^k[e^k\leq f(w_i)]\geq l\right)\\ &\leq \mathcal{P}_{w\sim\mathrm{Walk}(G,l)}\left(\bigvee_{k\geq 3}\sum_{i\in[l]}[e^k\leq f(w_i)]\geq le^{-k}k^{-2}\right)\\ &\leq \sum_{k\geq 3} L_k \leq \sum_{k\geq 3}\exp\left(-l-k+2\right)\leq \exp(-l).\end{aligned}$$ ◀

B Balls and Bins

Let $\Omega = U([r]\to[b])$ be the uniform probability space over the functions from $[r]$ to $[b]$ for $b\geq 1$ and $0\leq r\leq b$ and let $X(\omega) = |\omega([r])|$ be the size of the image of such a function. This models throwing r balls into b bins independently, where X is the random variable counting the number of hit bins. Moreover, let $E_i(\omega) = \{\omega \mid i\in\omega([r])\}$ be the event that the bin i was hit. Note that $X(\omega) = \sum_{i\in[b]} E_i(\omega)$. And we want to show that

$$\mathbb{E}_{\omega\sim\Omega} X(\omega) = b\left(1-\left(1-\frac{1}{b}\right)^r\right) \qquad \mathbb{V}_{\omega\sim\Omega} X(\omega)\leq \frac{r(r-1)}{b}$$

▶ **Lemma 17.** $\mathbb{E}_{\omega\sim\Omega} X(\omega) = b\left(1-\left(1-\frac{1}{b}\right)^r\right)$

The proof is available in the full version [26].

▶ **Lemma 18.** $\mathbb{V}_{\omega\sim\Omega} X(\omega)\leq\frac{r(r-1)}{b}$

The proof is available in the full version [26]. The above is a stronger version of the result by Kane et al. [24][Lem. 1]. Their result has the restriction that $r\geq 100$ and a superfluous factor of 4.

Interestingly, it is possible to obtain a similar result for k-independent balls into bins. For that let Ω' be a probability space of functions from $[r]$ to $[b]$ where

$$\mathcal{P}_{\omega\sim\Omega'}\left(\bigwedge_{i\in I}\omega(i) = x(i)\right) = r^{-|I|}$$

for all $I\subset[r]$, $|I|\leq k$ and all $x: I\to[b]$. As before let us denote $X'(\omega) := |\omega([r])|$ the number of bins hit by the r balls. Then the expectation (resp. variance) of X' approximates that of X with increasing independence k, more precisely:

▶ **Lemma 19.** *If* $\varepsilon\leq e^{-2}$ *and* $k\geq 1+5\ln(b\varepsilon^{-1})(\ln(\ln(b\varepsilon^{-1})))^{-1}$ *then:*

$$|\mathbb{E}_{\omega'\in\Omega'} X'(\omega') - \mathbb{E}_{\omega\in\Omega} X(\omega)|\leq\varepsilon r \qquad |\mathbb{V}_{\omega'\in\Omega'} X'(\omega') - \mathbb{V}_{\omega\in\Omega} X(\omega)|\leq\varepsilon^2\,.$$

This has been shown[7] by Kane et al. [24][Lem. 2]. The proof relies on the fact that $X = \sum_{i\in[b]}\max(1, Y_i)$ where Y_i denotes the random variable that counts the number of balls in bin i. It is possible to show that $\mathbb{E}(Y_i)^j = \mathbb{E}(Y_i')^j$ for all $j\leq k$ (where Y_i' denotes the same notion over Ω'). Their approach is to approximate $\max(1,\cdot)$ with a polynomial g of

[7] Without the explicit constants mentioned in here.

degree k. Since $\mathbb{E}\, g(Y_i) = \mathbb{E}\, g(Y'_i)$ they can estimate the distance between $\mathbb{E}\, X$ and $\mathbb{E}\, X'$ by bounding the expectation of each approximation error: $g(Y_i) - \max(1, Y_i)$. Obviously, larger degree polynomials (and hence increased independence) allow better approximations. The reasoning for the variance is analogous.

▶ **Lemma 20.** *If $k \geq C_2 \ln b + C_3$ then:*

$$L := \mathcal{P}_{\omega' \in \Omega'} \left(|X'(\omega') - \rho(r)| > 9b^{-1/2} r \right) \leq 2^{-6}$$

Proof. This follows from Lemma 17, 18 and the previous lemma for $\varepsilon = \min(e^{-2}, b^{-1/2})$ in particular: $\mathbb{V}\, X' \leq \mathbb{V}\, X + \frac{1}{b} \leq \frac{r^2}{b}$ and hence:

$$\begin{aligned} L &\leq \mathcal{P}_{\omega' \in \Omega'} \left(|X'(\omega') - \mathbb{E}\, X'| + |\mathbb{E}\, X' - \rho(r)| \geq 9b^{-1/2} r \right) \\ &\leq \mathcal{P}_{\omega' \in \Omega'} \left(|X'(\omega') - \mathbb{E}\, X'| + b^{-1/2} r \geq 9b^{-1/2} r \right) \\ &\leq \mathcal{P}_{\omega' \in \Omega'} \left(|X'(\omega') - \mathbb{E}\, X'| \geq 8b^{-1/2} r \right) \\ &\leq \mathcal{P}_{\omega' \in \Omega'} \left(|X'(\omega') - \mathbb{E}\, X'| \geq 8\sqrt{\mathbb{V}\, X'} \right) \leq 2^{-6} \end{aligned}$$

where the last line follows from Chebychev's inequality. ◀

C Table of Constants

Table 2 Table of Constants.

Constant	References	Constant	References
$C_1 := e^2 + e^3 + (e-1)$	Lemma 5	$C_2 := \frac{15}{2}$	Lemma 19
$C_3 := 16$	Lemma 19	$C_4 := 3^2 2^{23}$	Lemma 9 and 12
$C_5 := \lceil C_1 + 3 \rceil = 33$	Lemma 8	$C_6 := 4$	Lemma 15
$C_7 := 2^5$	Lemma 11		

D Formalization

As mentioned in the introduction the proofs in this work have been machine-checked using Isabelle. They are available [25, 27] in the AFP (Archive of Formal Proofs) [1] – a site hosting formal proofs verified by Isabelle. Table 3 references the corresponding facts in the AFP entries. The first column refers to the lemma in this work. The second is the corresponding name of the fact in the formalization. The formalization can be accessed in two distinct forms: As a source repository with distinct theory files, as well as two "literate-programming-style" PDF documents with descriptive text alongside the Isabelle facts (optionally with the proofs). The latter is much more informative. The third column of the table refers to the file name [8] of the corresponding source file, while the last column contains the reference of the AFP entry, including the section in the PDF versions.

[8] `Distributed_Distinct_Elements` is abbreviated by `DDE` and `Without` with `WO`.

Table 3 Reference to the formal entities.

Lemma	Formalized Entity	Theory	Src.
Thm. 1	This theorem from Impagliazzo and Kabanets was stated for motivational reasons and is never used in any of the following results, hence it is not formalized.		
Thm. 2	**theorem** *hitting-property*	`Expander_Graphs_Walks`	[27, §9]
Thm. 3	**theorem** *kl-chernoff-property*	`Expander_Graphs_Walks`	[27, §9]
Lem. 4	**lemma** *walk-tail-bound*	`DDE_Tail_Bounds`	[25, §5]
Lem. 5	**lemma** *deviation-bound*	`DDE_Tail_Bounds`	[25, §5]
Lem. 6 (1)	**lemma** *single-result*	`DDE_Inner_Algorithm`	[25, §6]
Lem. 6 (2)	**lemma** *merge-result*	`DDE_Inner_Algorithm`	[25, §6]
Lem. 8	**lemma** *cutoff-level*	`DDE_Cutoff_Level`	[25, §8]
Lem. 9	**lemma** *e-1*	`DDE_Accuracy_WO_Cutoff`	[25, §7]
Lem. 10	**lemma** *e-2*	`DDE_Accuracy_WO_Cutoff`	[25, §7]
Lem. 11	**lemma** *e-3*	`DDE_Accuracy_WO_Cutoff`	[25, §7]
Lem. 12	**lemma** *e-4*	`DDE_Accuracy_WO_Cutoff`	[25, §7]
Lem. 13	**lemma** *accuracy-without-cutoff*	`DDE_Accuracy_WO_Cutoff`	[25, §7]
Lem. 14	**lemma** *accuracy-single*	`DDE_Accuracy`	[25, §9]
Lem. 15	**lemma** *estimate-result-1*	`DDE_Accuracy`	[25, §9]
Thm. 7	**lemma** *estimate-result*	`DDE_Accuracy`	[25, §9]
Thm. 16 (1)	**theorem** *correctness*	`DDE_Outer_Algorithm`	[25, §10]
Thm. 16 (2)	**theorem** *space-usage*	`DDE_Outer_Algorithm`	[25, §10]
Thm. 16 (3)	**theorem** *asymptotic-space-complexity*	`DDE_Outer_Algorithm`	[25, §10]
Lem. 17	**lemma** *exp-balls-and-bins*	`DDE_Balls_And_Bins`	[25, §4]
Lem. 18	**lemma** *var-balls-and-bins*	`DDE_Balls_And_Bins`	[25, §4]
Lem. 19 (1)	**lemma** *exp-approx*	`DDE_Balls_And_Bins`	[25, §4]
Lem. 19 (2)	**lemma** *var-approx*	`DDE_Balls_And_Bins`	[25, §4]
Lem. 20	**lemma** *deviation-bound*	`DDE_Balls_And_Bins`	[25, §4]

Sampling and Certifying Symmetric Functions

Yuval Filmus
Technion – Israel Institute of Technology, Haifa, Israel

Itai Leigh
Tel-Aviv University, Israel

Artur Riazanov
École Polytechnique Fédérale de Lausanne, Switzerland

Dmitry Sokolov
École Polytechnique Fédérale de Lausanne, Switzerland

Abstract

A circuit $\mathcal{C}$ samples a distribution $\boldsymbol{X}$ with an error ε if the statistical distance between the output of $\mathcal{C}$ on the uniform input and $\boldsymbol{X}$ is ε. We study the hardness of sampling a uniform distribution over the set of n-bit strings of Hamming weight k denoted by $\boldsymbol{U}^n_k$ for *decision forests*, i.e. every output bit is computed as a decision tree of the inputs. For every k there is an $O(\log n)$-depth decision forest sampling $\boldsymbol{U}^n_k$ with an inverse-polynomial error [26, 11]. We show that for every $\varepsilon > 0$ there exists τ such that for decision depth $\tau \log(n/k)/\log\log(n/k)$, the error for sampling $\boldsymbol{U}^n_k$ is at least $1-\varepsilon$. Our result is based on the recent robust sunflower lemma [1, 23].

Our second result is about matching a set of n-bit strings with the image of a d-*local* circuit, i.e. such that each output bit depends on at most d input bits. We study the set of all n-bit strings whose Hamming weight is at least $n/2$. We improve the previously known locality lower bound from $\Omega(\log^* n)$ [5] to $\Omega(\sqrt{\log n})$, leaving only a quartic gap from the best upper bound of $O(\log^2 n)$.

2012 ACM Subject Classification Theory of computation → Circuit complexity; Theory of computation → Generating random combinatorial structures

Keywords and phrases sampling, lower bounds, robust sunflowers, decision trees, switching networks

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.36

Category RANDOM

Related Version *Full Version*: `https://arxiv.org/abs/2305.04363`

Funding This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 802020-ERC-HARMONIC. This work was supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number MB22.00026.

Acknowledgements We thank Mika Göös, Aleksandr Smal, and Anastasia Sofronova for fruitful discussions and suggestions. We thank the RANDOM 2023 anonymous referees for their helpful comments.

1 Introduction

Studying the hardness of sampling has been proposed in the paper [26], which spurred an active line of research [27, 21, 3, 28, 29, 30, 14, 31, 6, 8]. The basic setting is the following: we are given an infinite supply of independent uniform random bits as input, and our goal is to design a circuit with multiple output bits whose output is close in statistical distance to a given target distribution. Sampling circuit constructions have been applied in

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 36; pp. 36:1–36:21
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

cryptography [18, 7] and algorithms [15]. Sampling hardness results[1] have been instrumental in inspiring and improving two-source extractor constructions [28, 9, 10] (see [30] for an extended discussion), and have yielded lower bounds on succinct data structures [26, 30, 31].

Sampling hardness results are more challenging than computational ones. For example, while it is known since Smolensky's classical work [25] that parity requires an exponential number of gates to compute by an $\mathbf{AC}^0[3]$ circuit, no hard distributions are known for the circuit class $\mathbf{AC}^0[p]$ for any p, and while $\mathbf{AC}^0$ requires exponentially many gates to compute parity [17], a random vector with parity 0 can be sampled by an $\mathbf{NC}^0$ circuit. Moreover, this distribution can be sampled by a 2-local circuit, in the sense that each output bit depends only on two input bits [2, 19]. A very simple mapping achieves this: $(x_1, \dots, x_n) \mapsto (x_1, x_1 \oplus x_2, x_2 \oplus x_3, \dots, x_{n-1} \oplus x_n, x_n)$. A more general and striking fact is that $\mathbf{AC}^0$ can sample random permutations and all distributions of form $(\boldsymbol{X}, f(\boldsymbol{X}))$ where $\boldsymbol{X}$ is uniform over $\{0,1\}^n$ and f is *symmetric* i.e. its value depends only on the Hamming weight of the input [26].

We conjecture that the power of $\mathbf{NC}^0$ in regard to sampling symmetric distributions is in essence limited to the parity example above. Observe that a function is computable by an $\mathbf{NC}^0$ circuit if and only if it is $O(1)$-*local* i.e. each of its output bits depends on at most a constant number of input bits. For simplicity, we focus only on uniform distributions with symmetric support: let $\boldsymbol{U}^n_S$ be the uniform distribution over strings in $\{0,1\}^n$ with Hamming weight in the set $S \subseteq \{0, \dots, n\}$.

▶ **Conjecture 1.** *For every $d \in \mathbb{N}, \varepsilon \in (0,1)$ for all large enough n, if $\boldsymbol{X}$ is samplable by a d-local function and is ε-close to $\boldsymbol{U}^n_S$ for some $S \subseteq \{0, \dots, n\}$, then $\boldsymbol{X}$ is $O(\varepsilon)$-close to $\boldsymbol{U}^n_T$, where T is one of the following: $\{0\}, \{n\}, \{0, n\}, \{0, 2, 4, \dots\}, \{1, 3, 5, \dots\}, [n]$.*

Our results on sampling slices (namely, Theorem 2) imply this conjecture for all sets S which only contain small values, in the sense that $\max_{x \in S} x = o(n)$.

Quantum separations

A stronger version of Conjecture 1 would identify the family of sets S such that every $\mathbf{NC}^0$-samplable distribution is $1 - o(1)$-far from $\boldsymbol{U}^n_S$. There exists a set S for which this implies a separation between $\mathbf{NC}^0$ and $\mathbf{QNC}^0$ for sampling. This is due to the recent partial separation in [32]: they show that there exists a symmetric function f such that $(\boldsymbol{X}, f(\boldsymbol{X}))$ for uniform $\boldsymbol{X} \sim \{0,1\}^n$ can be sampled by a $\mathbf{QNC}^0$ circuit. Observe, however, that if an $\mathbf{NC}^0$-samplable distribution $\boldsymbol{Y}$ is at distance η from $(\boldsymbol{X}, f(\boldsymbol{X}))$, then the first n bits of $\boldsymbol{Y}$ are $(1/2 + \eta + o(1))$-close to the uniform distribution over $f^{-1}(0)$, due to the fact that the function f used in [32] is almost balanced (in the sense that $|f^{-1}(0)| = (1 + o(1)) \cdot |f^{-1}(1)|$). Now, if the uniform distribution over $f^{-1}(0)$ is not $\mathbf{NC}^0$-samplable within the distance $1 - \Omega(1)$, we get the separation. Conjecture 1 implies a weaker lower bound for the distance: a constant instead of a function approaching 1, but most of the known lower bounds for distribution sampling have very strong distance guarantees.

Local certificates

One interesting class of sets which is also not covered by our and prior results is $M_a = \{x \in \{0, \dots, n\} \mid x \bmod a = 0\}$, where a is a constant. However, the simple construction for sampling parity-0 vectors can be adapted to *match the support* of any $\boldsymbol{U}^n_{M_a}$, i.e. generate

[1] That is, showing that any circuit from a certain class produces a distribution that is far from the target.

a (not necessarily uniform) distribution whose support is exactly the set of strings with Hamming weight in M_a. This construction was given in [5, Proposition 3.1], where it was presented as a *proof system*. The idea is to interpret the input bits as a *certificate* that the output is in the target language (in this case, all n-bits strings whose Hamming weight is divisible by a). This connection motivates our study of locality in the context of proof systems. We drastically simplify and improve the locality lower bound of [20] for the language of n-bit strings whose Hamming weight is at least $n/2$ (in other words, 1-inputs of the majority function).

1.1 Notation

We use boldface letters for random variables, e.g. $\boldsymbol{a}, \boldsymbol{A}, \boldsymbol{b}, \boldsymbol{B}$. We write $\boldsymbol{a} \sim A$ to say that $\boldsymbol{a}$ is distributed according to a distribution A, or if A is a set, according to the uniform distribution over it. For $x \in \{0,1\}^n$, we denote its Hamming weight by $w(x) = |x| \coloneqq \{i \in [n] \mid x_i = 1\}$. We denote $e_i = 0^{i-1}10^{n-i} \in \{0,1\}^n$. For a string $x \in \{0,1\}^n$ and a set $T \subseteq [n]$, we write $x_T \coloneqq (x_i)_{i \in T} \in \{0,1\}^T$. We write $\boldsymbol{U}_k^n$ for the uniform distribution of Hamming weight k vectors.

For two distributions S and T over the same domain $\mathcal{X}$, the *statistical distance* is defined as

$$\Delta(S,T) \coloneqq \max_{A \subseteq \mathcal{X}} \left| \Pr_{\boldsymbol{x} \sim S}[\boldsymbol{x} \in A] - \Pr_{\boldsymbol{x} \sim T}[\boldsymbol{x} \in A] \right| = \frac{1}{2} \sum_{a \in \mathcal{X}} \left| \Pr_{\boldsymbol{x} \sim S}[\boldsymbol{x} = a] - \Pr_{\boldsymbol{x} \sim T}[\boldsymbol{x} = a] \right|.$$

The statistical distance between two random variables is the statistical distance between their distributions. We say that the distribution S is *ε-close* from the distribution T if $\Delta(S,T) < \varepsilon$. Otherwise, the distributions are *ε-far.*

We say that an input bit $i \in [m]$ *affects* an output bit $j \in [n]$, or equivalently that the output bit j *depends* on the input bit i, if there exist inputs $x, x' \in \{0,1\}^m$, differing only in the ith bit, such that $f(x)_j \neq f(x')_j$. A function $f\colon \{0,1\}^m \to \{0,1\}^n$ is *d-local* if each of its output bits depends only on d input bits. A function $f\colon \{0,1\}^m \to \{0,1\}^n$ has *decision depth d* if each of its output bits can be computed as a depth-d decision tree, i.e. decided with at most d adaptive input bit queries. If a function is d-local, then it has decision depth at most d.

1.2 Sampling Slices

The *k-slice* is the set of all n-bit strings of Hamming weight k. We denote the uniform distribution over the k-slice by $\boldsymbol{U}_k^n$. A simple computation (see Section 3.5) shows that $\boldsymbol{U}_S^n$ and $\boldsymbol{U}_{\max S}^n$ are close in statistical distance whenever $\max S = o(n)$. This means that in order to show the hardness of sampling from symmetric distributions over sublinear-Hamming-weight strings it is sufficient to study $\boldsymbol{U}_k^n$ for $k = o(n)$.

Although in the context of Conjecture 1 it is sufficient to lower bound the locality of a sampler, in the context of slices the more natural complexity measure is *decision depth.* A function $f\colon \{0,1\}^m \to \{0,1\}^n$ is computable by a *decision forest* of depth d if every output bit of f can be computed by a decision tree of depth d, i.e. with at most d *adaptive* queries to the input (in contrast, a d-local function is computed by d *non-adaptive* queries). Viola [26, Lemma 6.4] shows that a decision depth d sampler for $\boldsymbol{U}_k^n$ can be obtained from a depth-d *switching network.* Czumaj [11, Theorem 3.7] proves the existence of such switching networks with $d = O(\log n)$. The following theorem shows that for $k = o(n)$, this construction is almost tight.

▶ **Theorem 2.** *Suppose that $\boldsymbol{U}_k^n$ can be sampled with decision depth d and error η in variation distance.*

1. *For every $\varepsilon > 0$ there exists a constant τ such that $d \leq \tau \log(n/k)/\log\log(n/k)$ implies $\eta \geq 1 - \varepsilon$.*
2. *There exists a constant τ for which the following holds. For every $\varepsilon \in (0, 1)$, if $k \in [\log_2 n, 2^{\log^{1-\varepsilon} n}]$ and $d \leq \tau \log^{\varepsilon} n$, then $\eta = 1 - n^{-\Omega(k)}$. The same bound holds for $k \in [1, \log_2 n)$ and $d \leq \tau \log^{\varepsilon} n/\log\log n$.*

Moreover, Item 1 holds for any $\boldsymbol{U}_S^n$ with $\max_{x \in S} x = k$ or $\min_{x \in S} x = n - k$.

The first key observation in our proof of Theorem 2 is that it is sufficient to prove it for $k = 1$, namely:

▶ **Theorem 3.** *There exists a constant $\tau > 0$ such that any distribution sampled with decision depth $\tau \log n/\log\log n$ is $(1 - n^{-\Omega(1)})$-far from $\boldsymbol{U}_1^n$.*

To see why this implies Item 1, observe that the marginal distribution of the first n/k bits of $\boldsymbol{U}_k^n$ is $(1 - 1/e + o(1))$-close to $\boldsymbol{U}_1^{n/k}$. Theorem 3 then implies the distance lower bound $1/e - o(1)$ for sampling $\boldsymbol{U}_k^n$ with depth $\tau \log(n/k)/\log\log(n/k)$. The $1 - \varepsilon$ distance lower bound for every ε is achieved by generalizing Theorem 3 so it applies to distributions of the form "first $\Theta(n/k)$ bits of $\boldsymbol{U}_k^n$" directly.

Item 2 is implied by another reduction from Theorem 3. Suppose we have a depth-d sampler for $\boldsymbol{U}_k^n$. Using this sampler, we can construct a depth-kd sampler for $\boldsymbol{U}_1^{\binom{n}{k}}$ as follows. Identify each output bit with a unique subset of $[n]$ of size k, and assign to it the conjunction of the corresponding k bits in the output of the sampler for $\boldsymbol{U}_k^n$. It is easy to see that the resulting distribution has the same distance to $\boldsymbol{U}_1^{\binom{n}{k}}$ as the initial sampler has to $\boldsymbol{U}_k^n$.

Technique for Proving Theorem 3

The locality of a sampler is the maximum number of input bits that an output bit depends on. The locality is always bounded from above by the decision depth. As a warm-up, let us discuss an $\Omega(\log\log n)$ locality lower bound for sampling $\boldsymbol{U}_1^n$ which is close in spirit to [30, Theorem 3].

The main idea in the locality lower bound is a *hitting set* versus *independent set* dichotomy: for a d-local source, it is easy to see that either there are $\tau 2^d$ independent output bits, or there is a hitting set of input bits of size $\tau d 2^d$ such that every output bit depends on one of the bits in this set. In the former case, we can show that it is very likely that at least two of the independent bits evaluate to 1, since for each output bit its probability to be 1 is at least 2^{-d}.[2] In the latter case, by fixing the hitting set bits in every possible way, we observe that our source is a mixture (convex combination) of $2^{\tau d 2^d}$ many $(d-1)$-local sources. If we show that $(d-1)$-local sources must be $(1-\varepsilon)$-far from the target distribution, then our source is $(1 - \varepsilon 2^{\tau d 2^d})$-far by [30, Corollary 18]. Picking $d = \delta \log\log n$ for small enough δ yields a $1 - o(1)$ lower bound on the distance to the target distribution.

In order to improve this lower bound from $\log\log n$ to $\log n/\log\log n$, we introduce several new ideas.

[2] There is a caveat that some bits might be identically zero, but it is not a real issue, since there cannot be too many of them.

Monotonization. We observe that it is in some sense sufficient to deal with sources where each bit is a *monotone term* in the input bits. The intuitive reason is that the expected number of output bits evaluating to 1 is $2^{-d} \cdot n$ (again, this is not always true, since there are identically zero outputs), so there exists an assignment where that many bits evaluate to 1. By focusing on those bits and replacing them with terms corresponding to the satisfying assignments, we show that it is likely that at least two of these terms evaluate to 1.

Sunflowers. Let us pretend that all of the output bits are monotone terms. For $i \in [n]$, let N_i be the set of inputs mentioned by the term of the ith output bit. We find a sunflower $\mathcal{S} \subseteq \{N_1, \dots, N_n\}$, i.e. there exists a kernel K such that the intersection of any pair of sets in $\mathcal{S}$ is K. If the kernel is fixed to 1, the output bits in the sunflower become independent, so if the sunflower is large enough, it is likely that at least two of them evaluate to 1. For some small enough $d = \Omega(\sqrt{\log n})$, a large sunflower always exists among any n sets. Moreover, we can cover all but $o(n)$ output bits with sunflowers. Then we have the following dichotomy: either all kernels evaluate to 0, so the source is likely to be identically zero, or at least one kernel evaluates to 1, which makes it very likely that at least two output bits from the corresponding sunflower evaluate to 1.

Using *robust sunflowers* instead of classical sunflowers, we obtain a lower bound of $\Omega(\log n/\log\log n)$ on the decision depth.

1.3 Local Proof Systems

Local proof systems, introduced in [5] and further studied in [20], are defined as follows: a local proof system for a language L is an $\mathbf{NC}^0$-circuit family C_n such that $L \cap \{0,1\}^n$ is exactly the set of all possible outputs of C_n. A language L has a *d-local* proof system if for each n there exists a d-local function whose image is $L \cap \{0,1\}^n$. In relation to the sampling, it can be viewed as follows: the "sampler" needs to match the support of the given distribution exactly, but we do not care about matching the actual probabilities.

The hardness landscape in this setting is different from the setting of sampling distributions. The most notable difference is that the language of strings with Hamming weight divisible by p always has an $O(1)$-local proof system, while we conjecture that sampling from the uniform distribution over this language requires a super-constant locality, unless $p = 1$ or $p = 2$.

Our main contribution is in improving the locality lower bound for another symmetric language: $\textsc{Maj}^{-1}(1) := \{x \in \{0,1\}^n \mid |x| \geq n/2\}$. The previous best locality lower bound for proof systems for this language was $\Omega(\log^* n)$ [20], with a very complex proof which we expose in Appendix B. We simplify this proof and improve the locality lower bound to $\Omega(\sqrt{\log n})$, which is only polynomially smaller than the current best upper bound of $O(\log^2 n)$. The key idea is to consider proof systems with bounded locality of both input and output bits. For such proof systems it is easy to derive very strong requirements on locality. These can then be used to count the number of input-output bit pairs where the input affects the output, which yields the output locality lower bounds.

1.4 Switching Networks

The technique behind Theorem 2 breaks down for linear slices $\boldsymbol{U}^n_{\alpha n}$. Does it mean there is a low-depth sampler for such distributions? The only construction of a sampler we have is based on switching networks [11, Theorem 3.7].

A switching network of depth d is a layered graph with d layers and n nodes in each layer. The edges of the switching network do not cross between layers, and in each layer, the edges form a partial matching. A switching network defines a process over permutations of $[n]$: we start with the identity permutation, and then, for each layer, we toss a coin for each edge in the matching of this layer, and if the coin comes up heads, we transpose the endpoints of the edge.

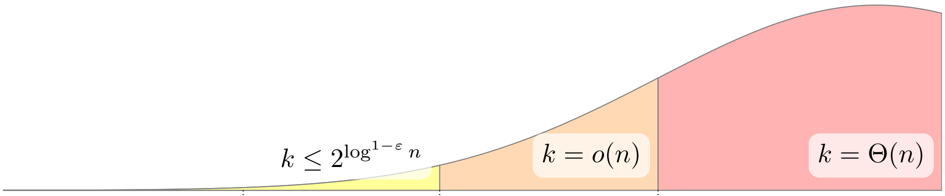
Currently, the best upper bound on the depth of a switching network which produces a distribution close to the uniform distribution over all permutations is $O(\log^2 n)$ [11]. The situation is better if the switching network only needs to shuffle sequences of zeroes and ones: the input is now $1^k 0^{n-k}$. [11, Theorem 3.7] gives a $O(\log n)$ depth upper bound for this case (the construction is randomized), [13] gives an *explicit* lower bound for generating $\boldsymbol{U}_k^n$ for $k \leq \sqrt{n}$.

It is almost immediate that a switching network that samples $\boldsymbol{U}_k^n$ within a non-trivial distance must have depth $\Omega(\log(n/k))$: each input bit of a network of depth d has at most 2^d potential positions that it can take in the output, so if $d = o(\log(n/k))$, the switching network produces a distribution where only $o(n)$ bits have a non-zero probability to have value 1, which is $(1 - o(1))$-far from $\boldsymbol{U}_k^n$.

In Appendix A we show that switching networks that produce a distribution close to $\boldsymbol{U}_{\alpha n}^n$ must have depth $\Omega(\log \log n)$. We use the following properties of samplers that are constructed from switching networks of depth d: the first is that each input bit of such a sampler affects at most 2^d output bits, the second is that the error is one-sided, i.e. such a sampler never outputs a string outside the domain of $\boldsymbol{U}_{\alpha n}^n$. The second property highlights the similarity with local certificates, and indeed our lower bound proof uses similar ideas.

1.5 Further Research

Our results on sampling slices with decision forests taken together with results of Viola [26] are summarized in Figure 1.



Theorem 2	$d = \tilde{\Omega}(\log^{\varepsilon} n)$ $\Delta = 1 - n^{-\Omega(k)}$	$d = \tilde{\Omega}(\log(n/k))$ $\Delta = 1 - \delta$	
[26, Thm 1.6]			$\Delta = 2^{-O(d)} - O(1/n)$

Figure 1 The table above depicts the implications of Theorem 2 and [26, Theorem 1.6] for sampling $\boldsymbol{U}_k^n$ for different k. The plot above it illustrates the size of the corresponding set of bitstrings in the boolean cube.

Here are some important challenges that are left open:

- Give any non-trivial decision depth (or even locality!) lower bound for linear Hamming weight in the constant-error regime. For a non-dyadic α[3] an $\omega(1)$ decision depth lower bound for sampling $\boldsymbol{U}_{\alpha n}^n$ follows (in a not completely straightforward way) from the new separator theorem in [31]: the key idea is to use the fact that biases of all bits of a distribution generated by a decision forest all have form $a/2^d$, so they are $\Theta(2^{-d})$-off from any non-dyadic number. The challenge is to show any non-trivial lower bound for, say $\boldsymbol{U}_{n/2}^n$ or $\boldsymbol{U}_{n/4}^n$, where the bit biases can be matched exactly by a decision forest.

[3] That is, not representable in the form $a/2^t$ for integers a and t, e.g. $1/3$.

- Tighten up the decision depth lower bound for $\boldsymbol{U}_1^n$ to $\Omega(\log n)$. This would immediately yield tight (up to a constant) decision depth lower bounds for all polynomial k.
- Give any locality lower bound for $(\boldsymbol{X}, f(\boldsymbol{X}))$, where $\boldsymbol{X} \sim \{0,1\}^n$ and $f = [(x_1 + \cdots + x_n) \bmod p \geq p/2] \oplus x_1 \oplus \cdots \oplus x_n$, as this would separate quantum and classical $\mathbf{NC}^0$ circuits for sampling, improving on the partial separation of [32].
- Find any non-trivial lower bounds for sampling a uniform vector with Hamming weight divisible by k, for any $k > 2$. This is likely to give insights on the $\mathbf{QNC}^0$ versus $\mathbf{NC}^0$ separation for sampling.
- Determine the optimal depth of a switching network that samples a uniform permutation or a uniform vector in a slice.

2 Tools

In this section, we describe two off-the-shelf tools we use in our proof.

2.1 FKG inequality

▶ **Theorem 4** ([16, 12]). *Suppose that X is a product distribution over $\{0,1\}^n$ (that is, $\Pr[X = x] = \prod_{i \in [n]} \Pr[X_i = x_i]$). Let $A, B \subseteq \{0,1\}^n$ be two monotone events (if $x \in A$ and $x_i \leq y_i$ for all $i \in [n]$ then $y \in A$, and similarly for B). Then*

$$\Pr_{\boldsymbol{x} \sim X}[\boldsymbol{x} \in A \cap B] \geq \Pr_{\boldsymbol{x} \sim X}[\boldsymbol{x} \in A] \cdot \Pr_{\boldsymbol{x} \sim X}[\boldsymbol{x} \in B].$$

2.2 Robust Sunflowers

In this section, we discuss robust sunflowers.

▶ **Definition 5** (Robust sunflower). *Let $0 < \alpha, \beta < 1$ be parameters, let $\mathcal{F}$ be a set system over a finite universe, and let $K := \bigcap_{S \in \mathcal{F}} S$ be the intersection of all sets in $\mathcal{F}$, which we refer to as the* kernel. *The family $\mathcal{F}$ is an (α, β)-*robust sunflower *if*

1. *$K \notin \mathcal{F}$;*
2. *$\Pr_{\boldsymbol{R}}[\exists S \in \mathcal{F} \colon S \subseteq \boldsymbol{R} \cup K] \geq 1 - \beta$, where each element of the universe appears in $\boldsymbol{R}$ with probability α independently.*

   We can write this condition in the equivalent form $\Pr_{\mathbf{R}}[\exists S \in \mathcal{F} \colon \mathbf{R} \supseteq S \mid \mathbf{R} \supseteq K] \geq 1 - \beta$.

*A set system is called (α, β)-*satisfying *if it is an (α, β)-robust sunflower with an empty kernel.*

Large enough set systems always contain a robust sunflower, as proved by Rossman [24] and improved by later authors.

▶ **Theorem 6** ([1, 4, 23]). *There exists a constant $B > 0$ such that the following holds for all $p, \varepsilon \in (0, 1/2]$. Let $\mathcal{F}$ be a family of sets of size exactly d such that $|\mathcal{F}| \geq (B \log(d/\varepsilon)/p)^d$. Then $\mathcal{F}$ contains a (p, ε)-robust sunflower.*

▶ **Corollary 7.** *There exists a constant $B > 0$ such that the following holds for all $p, \varepsilon \in (0, 1/2]$. Let $\mathcal{F}$ be a family of non-empty sets of size* at most *d such that $|\mathcal{F}| \geq d \cdot (B \log(d/\varepsilon)/p)^d$. Then $\mathcal{F}$ contains a (p, ε)-robust sunflower.*

Proof. Let $d_0 \in [d]$ be the most common size of sets in $\mathcal{F}$. Then the number of sets of size d_0 is at least $(B \log(d/\varepsilon)/p)^d$, which allows us to apply Theorem 6 to these sets. ◀

If we remove a single petal from a robust sunflower, then it remains a robust sunflower (with slightly worse parameters).

▶ **Lemma 8.** *Suppose that $N_1, \dots, N_k \subseteq X$ is a (p, ε)-robust sunflower with kernel K. Then for every i, the sets $N_1, \dots, N_{i-1}, N_{i+1}, \dots, N_k$ form a $(2p, 2\varepsilon)$-robust sunflower with kernel K.*

Proof. Let $\boldsymbol{\tau}$ be distributed over $[3]^{X \setminus K}$ such that $\Pr[\boldsymbol{\tau}_i = 1] = \Pr[\boldsymbol{\tau}_i = 2] = p$, $\Pr[\boldsymbol{\tau}_i = 3] = 1 - 2p$, and the coordinates of $\boldsymbol{\tau}$ are independent. For $\ell \in [3]$, let $\boldsymbol{\tau}^\ell = \{j \in X \mid \boldsymbol{\tau}_j = \ell\} \cup K$. The definition of (p, ε)-robust sunflower implies that for every $\ell \in [2]$ we have $\Pr[\exists j \in [k] \colon \boldsymbol{\tau}^\ell \supseteq N_j] \geq 1 - \varepsilon$. An application of the union bound implies that

$$\Pr\left[\exists j \in [k] \colon \boldsymbol{\tau}^1 \supseteq N_j \wedge \exists j' \in [k] \colon \boldsymbol{\tau}^2 \supseteq N_{j'}\right] \geq 1 - 2\varepsilon.$$

If $j = j'$, then since $\boldsymbol{\tau}^1 \cap \boldsymbol{\tau}^2 = K$, we have $N_j = K$, which is impossible by the definition of a sunflower. Thus $j \neq j'$ whenever the event happens. Let $\boldsymbol{R}$ be a distribution of subsets of X where each element appears in $\boldsymbol{R}$ independently with probability $2p$. Then since $(\boldsymbol{R} | \boldsymbol{R} \supseteq K)$ has the same distribution as $\boldsymbol{\tau}^1 \cup \boldsymbol{\tau}^2$, we have

$$\Pr_{\boldsymbol{R}}\left[\exists j \neq j' \in [k] \colon \boldsymbol{R} \supseteq N_j \wedge \boldsymbol{R} \supseteq N_{j'} \mid \boldsymbol{R} \supseteq K\right] \geq 1 - 2\varepsilon.$$

In particular, for every $i \in [k]$ we have

$$\Pr_{\boldsymbol{R}}\left[\exists j \neq i \in [k] \colon \boldsymbol{R} \supseteq N_j \mid \boldsymbol{R} \supseteq K\right] \geq 1 - 2\varepsilon,$$

and so the sets $N_1, \dots, N_{i-1}, N_{i+1}, N_k$ form a $(2p, 2\varepsilon)$-robust sunflower with kernel K. ◀

Another lemma we use is very similar to the standard connection between robust sunflowers and the classical ones (see e.g. Lemma 1.6 in [1]):

▶ **Lemma 9.** *Suppose that $N_1, \dots, N_m \subseteq X$ is a $(1/(2k), \varepsilon)$-robust sunflower with a kernel K. Then*

$$\Pr_{\boldsymbol{R} \sim 2^X}\left[\exists I \in \binom{[m]}{k} \forall i \in I \colon \boldsymbol{R} \supseteq N_i \middle| \boldsymbol{R} \supseteq K\right] \geq 1 - \varepsilon k.$$

Proof. Let $\tau \sim [2k]^{X \setminus K}$, and $\tau^i := \{j \in X \setminus K \mid \tau_j = i\} \cup K$ for $i \in [2k]$. Then $\boldsymbol{\tau}^1 \cup \dots \cup \boldsymbol{\tau}^k$ is distributed equivalently to $(\boldsymbol{R} \mid \boldsymbol{R} \supseteq K)$ where $\boldsymbol{R} \sim 2^X$. On the other hand, by the definition of the $(1/(2k), \varepsilon)$-robust sunflower, for each $i \in [2k]$ we get

$$\Pr[\exists j \in [m] \colon \boldsymbol{\tau}^i \supseteq N_j] \geq 1 - \varepsilon.$$

Since $\boldsymbol{\tau}^i \cap \boldsymbol{\tau}^{i'} = K$ for any $i \neq i' \in [2k]$, the lemma follows by the union bound over $i \in [k]$. ◀

3 Sampling Uniform Hamming Weight k Distributions

In this section we prove the following results mentioned in the introduction, which we restate here for convenience.

▶ **Theorem 2.** *Suppose that $\boldsymbol{U}_k^n$ can be sampled with decision depth d and error η in variation distance.*

1. *For every $\varepsilon > 0$ there exists a constant τ such that $d \leq \tau \log(n/k)/\log\log(n/k)$ implies $\eta \geq 1 - \varepsilon$.*

2. *There exists a constant τ for which the following holds. For every $\varepsilon \in (0,1)$, if $k \in [\log_2 n, 2^{\log^{1-\varepsilon} n}]$ and $d \leq \tau \log^{\varepsilon} n$, then $\eta = 1 - n^{-\Omega(k)}$. The same bound holds for $k \in [1, \log_2 n)$ and $d \leq \tau \log^{\varepsilon} n / \log\log n$.*

Moreover, Item 1 holds for any $\boldsymbol{U}_S^n$ with $\max_{x \in S} x = k$ or $\min_{x \in S} x = n - k$.

▶ **Theorem 3.** *There exists a constant $\tau > 0$ such that any distribution sampled with decision depth $\tau \log n / \log\log n$ is $(1 - n^{-\Omega(1)})$-far from $\boldsymbol{U}_1^n$.*

We first prove Theorem 3, in Section 3.1. We then prove Item 2 of Theorem 2 in Section 3.3, and Item 1 of the Theorem in Section 3.4. We prove the "moreover" part in Section 3.5.

3.1 Proof of Theorem 3

We prove a more general result which immediately yields Theorem 3.

First let us sketch a proof of Theorem 3 for d-local functions. Suppose that $\Delta(\boldsymbol{U}_1^n, \boldsymbol{X}) \leq 1 - \eta$. Call a coordinate i *good* if $\Pr[\boldsymbol{X} = e_i] \geq 1/n^2$. Since $\Pr[U_1 = e_i] = 1/n$, many coordinates are good: at least $\Omega(\eta n)$.

Let $\boldsymbol{Y} \sim \{0,1\}^m$ denote the random input bits. Each X_i depends on some subset $N_i \subseteq [m]$ of coordinates of size at most $d = \tau \log n / \log\log n$, say $\boldsymbol{X}_i = f_i(\boldsymbol{Y}_{N_i})$.

For each good coordinate i, we choose an assignment $\alpha_i \in f_i^{-1}(1)$ which maximizes the conditional probability $\Pr[\boldsymbol{X} = e_i \mid \boldsymbol{Y}_{N_i} = \alpha_i]$, that is, the probability that if $\boldsymbol{Y}_{N_i} = \alpha_i$ then all other output bits are 0. This probability is at least $1/(2^d n^2) = \Omega(1/n^3)$.

The assignments α_i do not necessarily agree with each other. However, a random assignment ρ to $\boldsymbol{Y}$ agrees with at least $\Omega(\eta n / 2^d) = \Omega(\eta n^{1-o(1)})$ of them. Let T consists of the domains of the assignments α_i which agree with ρ. These domains are distinct since $N_i = N_j$ implies $\alpha_i = \alpha_j$ and hence that $\Pr[\boldsymbol{X} = e_i \mid \boldsymbol{Y}_{N_i} = \alpha_i] = 0$. The choice of d guarantees that T supports a $(1/4, \varepsilon)$-robust sunflower $\mathcal{S}$, for any ε which is inverse-polynomial in n. Let K be the kernel of $\mathcal{S}$.

If we remove any single petal i from $\mathcal{S}$ then by Lemma 8 the result is a $(1/2, 2\varepsilon)$-robust sunflower, and so given that $\boldsymbol{Y}_K$ agrees with ρ, the probability that $\boldsymbol{X}_j = 1$ for some $j \neq i$ is at least $1 - 2\varepsilon$. If we replace the condition with "$\boldsymbol{Y}_{N_i}$ agrees with ρ" (and so with α_i), then intuitively, the probability can only increase, and this can be formalized using the FKG inequality (Theorem 4). By definition of α_i, this means that $\Pr[\boldsymbol{X} = e_i] \leq |f_i^{-1}(1)| 2\varepsilon \leq 2^{d+1}\varepsilon$. Choosing $\varepsilon = 1/2^{d+1} n^2$ shows that i is not good, and we reach a contradiction.

We move on to prove the generalization of Theorem 3.

▶ **Theorem 10.** *Let $\boldsymbol{Y} \sim \{0,1\}^m$ be the input bits of the n-bit source $\boldsymbol{X}$. Suppose that every bit of $\boldsymbol{X}$ is computed as a DNF of bits of $\boldsymbol{Y}$ of size at most s and width at most d. For every $\kappa \in \mathbb{R}$ there exists a constant τ such that for $d = \tau \log n / \log\log n$ and $s \leq \kappa n^{\kappa}$, we have $\Delta(\boldsymbol{X}, \boldsymbol{U}_1^n) = 1 - \eta = 1 - n^{-\Omega(1)}$.*

This implies Theorem 3 since the output of a decision tree of depth d can be represented as a DNF of size at most 2^d and width at most d. In our case $d = o(\log n)$ and so $2^d \leq n$ (for large enough n).

Proof. We say that an output bit $i \in [n]$ is *good* if $\Pr[\boldsymbol{X} = e_i] \geq 1/n^2$. Let $G \subseteq [n]$ be the set of all good bits, and let $\overline{G} = [n] \setminus G$. Let us estimate the size of G: $\Pr[\boldsymbol{X} \in \{e_i \mid i \notin G\}] \leq |\overline{G}|/n^2$, but $\Pr[\boldsymbol{U}_1^n \in \{e_i \mid i \notin G\}] = |\overline{G}|/n)$, so $|\overline{G}| \cdot (1/n - 1/n^2) \leq 1 - \eta$, which yields $|G| = \Omega(\eta n)$.

For each $i \in G$, since each bit of $\boldsymbol{X}$ is represented as a DNF we have $\boldsymbol{X}_i = \bigvee_{j\in[s_i]}[\boldsymbol{Y}_{N_i^j} = \alpha_i^j]$, where $N_i^1,\ldots,N_i^{s_i} \subseteq [m]$ are sets, $\alpha_i^j \in \{0,1\}^{N_i^j}$ are truth assignments, and $s_i \le s$. By the law of total probability we have

$$\Pr[\boldsymbol{X}=e_i] \le \sum_{j\in[s_i]} \Pr[\boldsymbol{X}=e_i \wedge \boldsymbol{Y}_{N_i^j}=\alpha_i^j] \le s\Pr[\boldsymbol{X}=e_i\wedge \boldsymbol{Y}_{N_i^{\max}}=\alpha_i^{\max}],$$

where $N_i^{\max}$ and $\alpha_i^{\max}$ correspond to the term in the DNF maximizing the probability $\Pr[\boldsymbol{X}=e_i\wedge\boldsymbol{Y}_{N_i^j}=\alpha_i^j]$.

Consider the expected number of good output bits such that $\boldsymbol{Y}_{N_i^{\max}}=\alpha_i^{\max}$:

$$\mathbb{E}\left[\sum_{i\in G}[\boldsymbol{Y}_{N_i^{\max}}=\alpha_i^{\max}]\right] \ge |G|2^{-d}.$$

Hence there exists an assignment ρ to the input bits such that for at least $|G|2^{-d}$ good output bits, we have $\rho_{N_i^{\max}}=\alpha_i^{\max}$. Let $T\subseteq G$ be the set of those output bits. If $i,j\in T$ then $N_i^{\max}\neq N_j^{\max}$, since otherwise $\boldsymbol{Y}_{N_i^{\max}}=\alpha_i^{\max}$ implies that also $\boldsymbol{Y}_{N_j^{\max}}=\alpha_j^{\max}$ and so $\boldsymbol{X}\neq e_i$, and so $\Pr[\boldsymbol{X}=e_i]=0$, contradicting $i\in G$. Observe moreover that none of the sets N_i for $i\in T$ is empty, since otherwise $|T|=1$ and we get an immediate contradiction with the size of G for any $d=o(\log n)$.

Case 1. $|T|<d(4B\log(d/\varepsilon))^d$. In this case, we immediately get the lower bound on δ. Indeed, the inequality $|G|2^{-d}\le|T|<d(4B\log(d/\varepsilon))^d$ implies $|G|\le d(8B\log(d/\varepsilon))^d$, which together with $|G|=\Omega(\eta n)$ yields $\eta\le d(8B\log(d/\varepsilon))^d/n$. If ε is inverse polynomial in n, then for small enough τ we get $\eta=n^{-\Omega(1)}$ with $d=\tau\log n/\log\log n$.

Case 2. $|T|\ge d(4B\log(d/\varepsilon))^d$. Then by Corollary 7 there exists a $(1/4,\varepsilon)$-robust sunflower formed by the sets $N_{t_1}^{\max},\ldots,N_{t_k}^{\max}$ for $\{t_1,\ldots,t_k\}\subseteq T$ (recall the sets $N_i^{\max}$ for $i\in T$ are all distinct, and none of them is empty). Let K denote the kernel of this sunflower. Consider an arbitrary petal t_i of this sunflower. By Lemma 8 we have that $\{N_i^{\max}\}_{i\in T\smallsetminus\{t_i\}}$ is a $(1/2,2\varepsilon)$-robust sunflower. Let $\boldsymbol{U}$ be the set of indices such that $\boldsymbol{Y}_k=\rho_k$. Then

$$\begin{aligned}
\Pr[\boldsymbol{X}_j=1\text{ for some }j\in T\smallsetminus\{t_i\}\mid \boldsymbol{Y}_{N_{t_i}^{\max}}=\rho_{N_{t_i}^{\max}}]\ge{}&\\
\Pr[\boldsymbol{U}\supseteq N_j^{\max}\text{ for some }j\in T\smallsetminus\{t_i\}\mid \boldsymbol{U}\supseteq N_{t_i}^{\max}]={}&\\
\frac{\Pr[\boldsymbol{U}\supseteq N_{t_i}^{\max}\text{ and }\boldsymbol{U}\supseteq N_j^{\max}\text{ for some }j\in T\smallsetminus\{t_i\}\mid \boldsymbol{U}\supseteq K]}{\Pr[\boldsymbol{U}\supseteq N_{t_i}^{\max}\mid\boldsymbol{U}\supseteq K]}\ge{}& \qquad\text{Theorem 4}\\
\Pr[\boldsymbol{U}\supseteq N_j^{\max}\text{ for some }j\in T\smallsetminus\{t_i\}\mid \boldsymbol{U}\supseteq K]\ge{}&\\
1-2\varepsilon,&
\end{aligned}$$

where the last inequality is due to the definition of a $(1/2,2\varepsilon)$-robust sunflower. Recall that by the choice of T, we have $\rho_{N_{t_i}^{\max}}=\alpha_{t_i}^{\max}$. Therefore

$$\Pr[\boldsymbol{X}\neq e_{t_i}\mid \boldsymbol{Y}_{N_{t_i}^{\max}}=\alpha_{t_i}^{\max}]=\Pr[\boldsymbol{X}_j=1\text{ for some }j\in T\smallsetminus\{t_i\}\mid \boldsymbol{Y}_{N_{t_i}^{\max}}=\rho_{N_{t_i}^{\max}}]\ge1-2\varepsilon.$$

Thus $\Pr[\boldsymbol{X}=e_{t_i}\wedge\boldsymbol{Y}^{N_{t_i}^{\max}}=\alpha_{t_i}^{\max}]\le2\varepsilon$. By the choice of $\alpha_{t_i}^{\max}$, $\Pr[\boldsymbol{X}=e_{t_i}]\le s\cdot2\varepsilon$. Picking $\varepsilon<1/(2sn^2)$, which is inverse polynomial in n as required in Case 1, we get that $\Pr[\boldsymbol{X}=e_{t_i}]<1/n^2$, so t_i is bad, which contradicts the choice of T. ◀

3.2 A Generalized Version of Theorem 10

In this section, we generalize Theorem 10 so it can be used to prove Item 1 of Theorem 2. The proof follows the same path as the proof of Theorem 10, we decided to include both proofs for simplicity.

▶ **Theorem 11.** *Let $\boldsymbol{Y} \sim \{0,1\}^m$ be the input bits of the n-bit source $\boldsymbol{X}$. Suppose that every bit of $\boldsymbol{X}$ is computed as a DNF of bits of $\boldsymbol{Y}$ of size at most s and width at most d. Let $t \in [n]$ be a parameter, $\alpha(n)$ be a function and let $\boldsymbol{F}$ be distributed over $\{0,1\}^n$ and have the following properties:*

- *For every set $T \subseteq [n]$ such that $T \geq n/2$ we have $\Pr[\boldsymbol{F}_T = 0^T] \leq \alpha(n)$;*
- $\Pr[|\boldsymbol{F}| > t] \leq \alpha(n)$.

If $2d \cdot t \cdot (40Bt \log n)^d \leq n$ then $\Delta(\boldsymbol{X}, \boldsymbol{F}) \geq 1 - 2\alpha(n) - 1/2n$. Here B is the constant from Corollary 7.

Proof. We say that an output bit $i \in [n]$ is *good* if $\Pr[\boldsymbol{X}_i = 1 \wedge |\boldsymbol{X}| \leq t] \geq 1/n^2$. Let G be the set of good bits and let $\overline{G} \coloneqq [n] \setminus G$. Suppose that $|G| \leq n/2$. Then by the conditions on $\boldsymbol{F}$ we have $\Pr[\boldsymbol{F}_{\overline{G}} = 0^{\overline{G}}] \leq \alpha(n)$. On the other hand $\Pr[\boldsymbol{X}_{\overline{G}} \neq 0^{\overline{G}} \wedge |\boldsymbol{X}| \leq t] < |\overline{G}|/n^2 < 1/2n$. Then $\Delta(\boldsymbol{X}, \boldsymbol{F}) \geq 1 - 2\alpha(n) - 1/2n$, as required. In the rest of the proof, we derive a contradiction with $|G| \geq n/2$.

As in the proof of Theorem 10, we pick the likeliest term in the DNF representation of each of the output bits. For each $i \in G$, since each bit of $\boldsymbol{X}$ is represented as a DNF, we have $\boldsymbol{X}_i = \bigvee_{j \in [s_i]} [\boldsymbol{Y}_{N_i^j} = \alpha_i^j]$, where $N_i^1, \dots, N_i^{s_i} \subseteq [m]$ are sets, $\alpha_i^j \in \{0,1\}^{N_i^j}$ are truth assignments, and $s_i \leq s$. By the law of total probability, we have

$$\Pr[\boldsymbol{X}_i = 1 \wedge |\boldsymbol{X}| \leq t] \leq \sum_{j \in [s_i]} \Pr[|\boldsymbol{X}| \leq t \wedge \boldsymbol{Y}_{N_i^j} = \alpha_i^j] \leq s \Pr[|\boldsymbol{X}| \leq t \wedge \boldsymbol{Y}_{N_i^{\max}} = \alpha_i^{\max}],$$

where $N_i^{\max}$ and $\alpha_i^{\max}$ correspond to the term in the DNF maximizing the probability $\Pr[|\boldsymbol{X}| \leq t \wedge \boldsymbol{Y}_{N_i^j} = \alpha_i^j]$. The expected number of good output bits with $\boldsymbol{Y}_{N_i^{\max}} = \alpha_i^{\max}$ is at least $2^{-d}|G|$, so there exists an assignment ρ to the input bits such that $\rho_{N_i^{\max}} = \alpha_i^{\max}$ for at least $|G|2^{-d}$ good output bits. Let $T \subseteq G$ be the set of these output bits.

Let us estimate how many distinct elements are in the set $\mathcal{N} \coloneqq \{N_i^{\max} \mid i \in T\}$. Suppose there exist $i_1, \dots, i_{t+1} \in T$ such that $N_{i_1}^{\max} = \cdots = N_{i_{t+1}}^{\max}$. Then, by the definition of ρ, we have $\alpha_{i_1}^{\max} = \cdots = \alpha_{i_{t+1}}^{\max}$ as well. Thus $\boldsymbol{Y}_{N_{i_1}^{\max}} = \alpha_{i_1}^{\max}$ implies that for every $j \in [t+1]$ we have $\boldsymbol{Y}_{N_{i_j}^{\max}} = \alpha_{i_j}^{\max}$, which in turn implies that $|\boldsymbol{X}| \geq t+1$, and so $\Pr[\boldsymbol{Y}_{N_{i_1}^{\max}} = \alpha_{i_1}^{\max} \wedge |\boldsymbol{X}| \leq t] = 0$, which contradicts that i_1 is good. Hence $|\mathcal{N}| \geq |T|/t \geq |G|2^{-d}/t \geq 2^{-d}n/2t$.

Let $\varepsilon > n^{-5}$ be a parameter to be chosen later. By the condition on n we have $|\mathcal{N}| \geq 2^{-d}n/2t \geq d \cdot (4Bt \log(d/\varepsilon))^d$, and so $\mathcal{N}$ contains a $(1/(4t), \varepsilon)$-robust sunflower $\mathcal{S}$. Let K denote the kernel of this sunflower.

Fix an arbitrary petal $p \in \mathcal{N}$. Then $\mathcal{N} \setminus \{p\}$ is a $(1/(2t), 2\varepsilon)$-robust sunflower by Lemma 8. Now by Lemma 9 we have

$$\Pr_{\boldsymbol{R} \sim 2^{[m]}}[\text{There are } t \text{ distinct petals of } \mathcal{N} \setminus \{p\} \text{ contained in } \boldsymbol{R} \mid \boldsymbol{R} \supseteq K] \geq 1 - 2t\varepsilon.$$

Let $P \subseteq T$ be the indices of the output bits corresponding to the elements of $\mathcal{N} \setminus \{p\}$, let i be the index of the output bit corresponding to the petal p, and let $\boldsymbol{U}$ be the set of indices of input bits such that $\boldsymbol{Y}_t = \rho_t$. Then

$$\begin{aligned}
&\Pr[|\boldsymbol{X}| > t \mid \boldsymbol{Y}_{N_i^{\max}} = \rho_{N_i^{\max}}] = \\
&\Pr[|\boldsymbol{X}_{[n]\smallsetminus\{i\}}| \geq t \mid \boldsymbol{Y}_{N_i^{\max}} = \rho_{N_i^{\max}}] \geq \\
&\Pr\left[\sum_{j\in P}[\boldsymbol{U} \supseteq N_j^{\max}] \geq t \,\middle|\, \boldsymbol{U} \supseteq N_i^{\max}\right] = \\
&\frac{\Pr[\sum_{j\in P\cup\{i\}}[\boldsymbol{U} \supseteq N_j^{\max}] \geq t \mid \boldsymbol{U} \supseteq K]}{\Pr[\boldsymbol{U} \supseteq N_i^{\max} \mid \boldsymbol{U} \supseteq K]} \geq \ \text{Theorem 4} \\
&\Pr\left[\sum_{j\in P}[\boldsymbol{U} \supseteq N_j^{\max}] \geq t \,\middle|\, \boldsymbol{U} \supseteq K\right] \geq \\
&1 - 2t\varepsilon.
\end{aligned}$$

Recall that by the choice of T we have $\rho_{N_i^{\max}} = \alpha_i^{\max}$, hence $\Pr[|\boldsymbol{X}| > t \mid \boldsymbol{Y}_{N_i^{\max}} = \alpha_i^{\max}] \geq 1 - 2t\varepsilon$. Thus

$$\Pr[|\boldsymbol{X}| \leq t \wedge \boldsymbol{X}_i = 1] \leq s \cdot \Pr[|\boldsymbol{X}| \leq t \wedge \boldsymbol{Y}_{N_i^{\max}} = \alpha_i^{\max}] \leq 2st \cdot \varepsilon.$$

Picking $\varepsilon = 1/(4stn^2) \geq n^{-5}$, we get a contradiction with i being good. ◀

3.3 Subpolynomial Weights

Although Theorem 11 implies Item 2 of Theorem 2, we give a simpler proof via a reduction from $\boldsymbol{U}_1^n$.

▶ **Lemma 12.** *Let $S \subseteq \{0,1\}^n$ and let $\boldsymbol{S} \sim S$. Suppose that $\boldsymbol{S}$ can be sampled with a depth-d decision forest with error η. Assume furthermore that for each $s \in S$ there exists a decision tree T_s of depth k that accepts s and does not accept any of $S \smallsetminus \{s\}$. Then there exists a decision depth-kd sampler for $\boldsymbol{U}_1^{|S|}$ with error η.*

Proof. Let $\boldsymbol{Y}$ be the distribution sampled by the sampler for $\boldsymbol{S}$. For each output bit of our sampler for $\boldsymbol{U}_1^{|S|}$ we take a unique element $s \in S$ and implement each of the queries of T_s via the query to the bits of $\boldsymbol{Y}$ (which makes at most d queries to the input bits). This results in a kd-deep decision tree T'_s. Let $\boldsymbol{X}$ be the sampled distribution. Then

$$\begin{aligned}
\Delta(\boldsymbol{X}, \boldsymbol{U}_1^{|S|}) &= \frac{1}{2}\left(\Pr[w(\boldsymbol{X}) \neq 1] + \sum_{s\in S}\bigl|\Pr[\boldsymbol{X} = e_s] - 1/|S|\bigr|\right) \\
&= \frac{1}{2}\left(\Pr[\boldsymbol{Y} \notin S] + \sum_{s\in S}\bigl|\Pr[\boldsymbol{Y} = s] - 1/|S|\bigr|\right) = \Delta(\boldsymbol{Y}, \boldsymbol{S}) = \eta.
\end{aligned}$$

◀

▶ **Corollary 13.** *For some constant $\tau' > 0$ and every $\varepsilon \in (0,1)$ every $(\tau' \log^{\varepsilon} n)$-decision depth sampler outputs a distribution $(1 - n^{-\Omega(k)})$-far from $\boldsymbol{U}_k^n$ for $k \in [\log n, 2^{\log^{1-\varepsilon} n}]$. If $k < \log_2 n$, this holds for every $(\tau' \log^{\varepsilon} n/\log\log n)$-local sampler.*

Proof. The decision tree that queries all the elements of a k-size set and accepts iff all of them are 1 satisfies the condition in Lemma 12. If we had a $(1-\delta)$-error sampler for $\boldsymbol{U}_k^n$, we would get a $(1-\delta)$-error sampler for $\boldsymbol{U}_1^{\binom{n}{k}}$ with decision depth kd, which by Theorem 3 yields that $\delta = \binom{n}{k}^{-\Omega(1)} = n^{-\Omega(k)}$ whenever $kd \leq \tau \log\binom{n}{k}/\log\log\binom{n}{k}$. Since $\log\binom{n}{k} = \Theta(k\log(n/k))$, we get $d = \Omega(\log(n/k)/(\log k + \log\log n))$. ◀

3.4 Sublinear Weights

In this section, we prove Item 1 of Theorem 2.

▶ **Lemma 14.** *Suppose $\boldsymbol{X}$ is sampled with decision depth d. Then for every small enough $\varepsilon > 0$ there exists a constant τ such that if $d \le \tau \log(n/k)/\log\log(n/k)$ then $\Delta(\boldsymbol{X}, \boldsymbol{U}^n_k) \ge 1 - 2\varepsilon - \frac{1}{2n}$.*

Proof. Consider the first $\ell := \varepsilon^{-1} \cdot n/k$ bits of the sampler: $\boldsymbol{X}_{\le \ell} := \boldsymbol{X}_1, \dots, \boldsymbol{X}_\ell$. Let $\boldsymbol{Y}$ be the first ℓ bits of the distribution $\boldsymbol{U}^n_k$. We show that $\boldsymbol{Y}$ satisfies the conditions of Theorem 11 for $t = \varepsilon^{-2}$ and $\alpha(n) = \varepsilon$. First, we have

$$\Pr[|\boldsymbol{Y}| > t] = \Pr\left[|\boldsymbol{Y}| > \varepsilon^{-1}\mathbf{E}[|\boldsymbol{Y}|]\right] < \varepsilon.$$

Now let T be any subset of $[\ell]$ of size at least $\ell/2$. Then

$$\Pr[\boldsymbol{Y}_T = 0^T] = \binom{n-\ell/2}{k}\binom{n}{k}^{-1} = \prod_{i=0}^{k-1} \frac{n-i-\ell/2}{n-i} \le \left(1 - \frac{\ell}{2n}\right)^k = \left(1 - \frac{1}{2\varepsilon k}\right)^k \le e^{-2\varepsilon^{-1}} < \varepsilon.$$

Here we assumed $k < n/2$, since otherwise the lemma is trivially true.

Applying Theorem 11, for small enough τ (which depends on ε) we have $\Delta(\boldsymbol{X}_{\le \ell}, \boldsymbol{Y}) = 1 - 2\varepsilon - 1/2n$. To finish the proof, observe that $\Delta(\boldsymbol{X}, \boldsymbol{U}^n_k) \ge \Delta(\boldsymbol{X}_{\le \ell}, \boldsymbol{Y})$, since the random variables on the RHS are the marginals of the variables on the LHS. ◀

3.5 Unions of Slices

In this section, we prove the "moreover" part of Theorem 2 by observing that the distribution $\boldsymbol{U}^n_S$ is close to $\boldsymbol{U}^n_{\max_{x \in S} x}$ as long as $\max_{x \in S} x = o(n)$.

▶ **Proposition 15.** *Let $k = o(n)$ and suppose that $S \subseteq \{0, 1, \dots, k\}$ with $k \in S$. Then we have $\Delta(\boldsymbol{U}^n_k, \boldsymbol{U}^n_S) = o(1)$.*

Proof. We use the notation $\binom{n}{S} := \sum_{i \in S} \binom{n}{i}$.

$$\begin{aligned}\Delta(\boldsymbol{U}^n_k, \boldsymbol{U}^n_S) &= \frac{1}{2}\binom{n}{S \setminus \{k\}}\binom{n}{S}^{-1} + \frac{1}{2}\left(1 - \binom{n}{k}\binom{n}{S}^{-1}\right) \\ &= \binom{n}{S \setminus \{k\}}\binom{n}{S}^{-1} \\ &\le \binom{n}{k}^{-1} \sum_{i \in S \setminus \{k\}} \binom{n}{i} \\ &\le \sum_{i=0}^{k-1} \left(\frac{k}{n-i}\right)^{i-k} = \Theta(k/n).\end{aligned}$$ ◀

The case of S with $n - \min_{x \in S} x = o(n)$ reduced to the case where $\max_{x \in S} x = o(n)$ by observing that flipping all output bits can be done with no increase in the decision depth of the sampler.

4 Local Certificates

In this section, we explore the power of local proof systems. Section 4.1 gives an example of a language that requires locality $\Omega(n)$, which is inspired by a similar lower bound in the context of sampling [22]. Section 4.2 then gives our main result, a lower bound on the locality of proof systems for $\textsc{Maj}^{-1}(1)$.

4.1 Error-correcting codes

In this section, we show that a good error-correcting code requires a proof system of a linear locality. This showcases the simple counting technique that we also use for our majority lower bound.

▶ **Proposition 16.** *Let $C \subseteq \{0,1\}^n$ be a good code, that is, $|C| \geq 2^{\alpha n}$ and for every $x \neq y \in C$, the Hamming distance between x and y is at least βn, where α and β are constants in $(0,1)$.*

If $f\colon \{0,1\}^m \to \{0,1\}^n$ is a d-local function and $f(\{0,1\}^m) = C$ then $d \geq \alpha\beta n$

Proof. We may assume w.l.o.g. that all input bits of f affect some output bits. Take an arbitrary input bit $i \in [m]$ and an output bit $j \in [n]$ that depends on i. Then take $x, x' \in \{0,1\}^m$ that differ only in the ith coordinate and such that $f(x)_j \neq f(x')_j$. Since $f(x)$ and $f(x')$ are two distinct codewords of C, they must be at Hamming distance at least βn, hence i must affect at least βn output bits, as x and x' only differ in i. Thus every bit affects at least βn output bits.

There are at least αn input bits since $|C| = 2^{\alpha n}$. Therefore there are $\alpha\beta n^2$ input-output pairs in which the input bit affects the output bit. On the other hand, there are at most dn such pairs, hence $d \geq \alpha\beta n$. ◀

4.2 Majority

Let $\textsc{Maj}_n$ be the the set $\{x \in \{0,1\}^n \mid |x| \geq n/2\}$. First, let us give a simple upper bound on locality which is implicit in [20].

▶ **Proposition 17** (essentially Theorem 3.9 and Corollary 3.10 in [20]). *There exists an $O(\log^2 n)$-local function $f\colon \{0,1\}^* \to \{0,1\}^n$ such that $f(\{0,1\}^*) = \textsc{Maj}_n^{-1}(1)$.*

Proof. For simplicity, suppose that n is odd. Construct a binary tree whose root is the interval $[1,n]$, whose leaves are the singletons $\{1\}, \dots, \{n\}$, and in which each internal node $[\ell, r]$ has two children $[\ell, c], [c+1, r]$, where $c = \lfloor (\ell + r)/2 \rfloor$. We can construct such a tree whose depth is $O(\log n)$. For each interval $[\ell, r]$ in the tree we will have a label $w(\ell, r)$ whose value ranges from 0 to $r - \ell + 1$, which is supposed to indicate $x_\ell + \cdots + x_r$ (where $x_1, \dots, x_n$ is the output). We implement the variables using $O(\log n)$ input bits.

An internal node $[\ell, r]$ with children $[\ell, c], [c+1, r]$ is *consistent* if $w(\ell, r) = w(\ell, c) + w(c+1, r)$. In addition, if the internal node is the root $[1,n]$, we require $w(1,n) \geq (n+1)/2$. Each position $i \in \{1, \dots, n\}$ corresponds to the leaf $w(i,i)$, which has $O(\log n)$ ancestors. We say that position i is *good* if all its non-leaf ancestors are consistent. The i'th output is $w(i,i)$ if i is good, and 1 otherwise. Since each $w(\ell, r)$ is encoded using $O(\log n)$ bits, this system has locality $O(\log^2 n)$.

Every vector $x_1, \dots, x_n$ of weight at least $(n+1)/2$ can be generated using this system by taking $w(\ell, r) = x_\ell + \cdots + x_r$. In the other direction, consider any assignment of weights to the tree. If the root is inconsistent, then the output is $1, \dots, 1$, so we can assume that the root is consistent. Prune the tree by removing all children of inconsistent nodes. If $[\ell, r]$ is any node in the pruned tree then either $\ell = r$ and $x_\ell = w(\ell, r)$, or $\ell < r$ and $x_\ell + \cdots + x_r = r - \ell + 1 \geq w(\ell, r)$. It follows that $x_1 + \cdots + x_n \geq w(1,n) \geq (n+1)/2$. ◀

The $\Omega(\log^* n)$ locality lower bound in [5] is inspired by the following observation:

▶ **Proposition 18.** *Let $f\colon \{0,1\}^m \to \{0,1\}^n$ be such that every output bit is a function of at most c input bits, and every input bit affects at most d output bits. Suppose that $cd \leq (n+1)/2$. Then $f(\{0,1\}^m) \neq \textsc{Maj}^{-1}(1)$.*

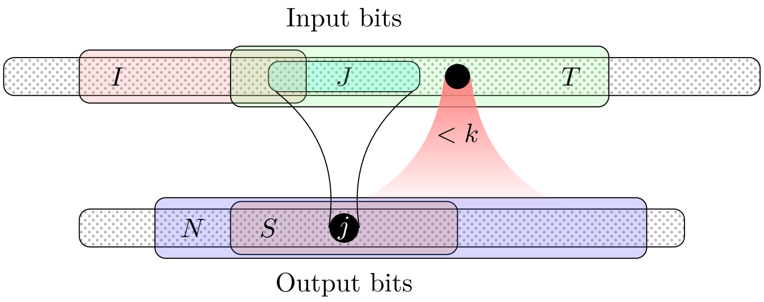


Figure 2 I is the set of k-influential inputs bits, and S is the given set. The set T consists of all inputs bits affecting S, and the set N consists of all bits influenced by $T \setminus I$.

Proof. Let $i \in [n]$ be an arbitrary output bit. There are at most cd many output bits in the "neighborhood" $N(i)$ of i, which is the set of outputs that share an input bit with i. If $|N(i)| \le (n+1)/2$ then we can find an output y of weight $(n+1)/2$ such that $y_{N(i)} = 1^{N(i)}$. Suppose that y is generated by the input x. There must be some setting to the inputs of i which sets it to zero (since there is a valid output z with $z_i = 0$). If we modify x using this setting then the new output z agrees with y outside of $N(i)$, and furthermore $z_i = 0$. Since $y_{N(i)} = 1^{N(i)}$, it follows that $|z| < |y|$, which is impossible, since y had the smallest possible weight. ◀

One of the steps in our proof (namely, Lemma 20) is essentially an adaptation of the proof of Proposition 18 for the sources with influential input bits.

We give a simplified exposition of their proof in Appendix B. Our own lower bound is contained in the following theorem.

▶ **Theorem 19.** *Let $f\colon \{0,1\}^m \to \{0,1\}^n$ be a d-local function such that $f(\{0,1\}^m) = \textsc{Maj}_n^{-1}(1)$, where n is odd. Then $d = \Omega(\sqrt{\log n})$.*

Proof. We say that an input bit $i \in [m]$ is *k-influential* if at least k output bits of f depend on it. We are going to show that there are $\Omega(n/(dk))$ many k-influential bits for every k. Then the number of input-output bit pairs where the output depends on the input is at least $\sum_{k\in[n]} cn/(dk) = \Omega(n \log n/d)$. On the other hand, there are at most nd such pairs since f is d-local. Therefore $d = \Omega(\sqrt{\log n})$.

It remains to show the lower bound on the number of k-influential bits. This is done by combining the following two lemmas.

▶ **Lemma 20.** *Let I be the set of all k-influential input bits. Then for every set of output bits S of size at most $n/(4kd)$ there exists an assignment ρ to I such that*

1. *All bits in S are fixed to 1 by ρ, i.e. for every total extension ρ' of ρ we have $f(\rho')_S = 1^S$.*
2. *ρ fixes to 1 at most $(n+1)/2$ output bits.*

Proof. Fix a set S of size at most $n/(4kd)$.

Let T be the set of input bits that affect S, so $|T| \le d|S|$. Let N be the set of all bits influenced by $T \setminus I$. See Figure 2 for a pictorial representation of these definitions.

Since I contains all k-influential input bits, $|N| \le kd|S|$. Let ρ' be a total assignment such that $f(\rho')_N = 1^N$ and $|f(\rho')| = (n+1)/2$. This is possible since $|N| \le n/4$ by the statement of the lemma. Let $\rho := \rho'_I$.

Since $|f(\rho')| = (n+1)/2$, in particular ρ fixes to 1 at most $(n+1)/2$ bits. We claim that ρ fixes all bits in S to 1. Suppose for the sake of contradiction that it doesn't fix to 1 the bit $j \in S$. Let J be the set of input bits affecting j. Let ρ'' be a total assignment consistent with ρ' everywhere except $J \setminus I$ such that $f(\rho'')_j = 0$. Observe that $f(\rho'')_{[n]\setminus N} = f(\rho')_{[n]\setminus N}$, hence $f(\rho'') \leq f(\rho')$ coordinate-wise. Since $f(\rho'')_j = 0$ and $f(\rho')_j = 1$, we have $|f(\rho'')| < n/2$, which contradicts the fact that the image of f is $\textsc{Maj}_n^{-1}(1)$. ◀

▶ **Lemma 21.** *Let I be an arbitrary set of input bits of size at most $n/20$. Then there exists a set S of output bits such that $|S| = O(|I|)$ and for every assignment ρ to I that fixes to 1 at most $(n+1)/2$ output bits, there exists a bit in S that is not fixed to 1 by ρ.*

Proof. Let $\rho_1, \dots, \rho_K$ be all assignments to I that fix at most $(n+1)/2$ output bits. Denote by $U_1, \dots, U_K \subseteq [n]$ the sets of bits that are not fixed by $\rho_1, \dots, \rho_K$, respectively, so that $|U_1|, \dots, |U_K| \geq (n-1)/2 \geq n/3$. Then

$$K\frac{n}{3} \leq \sum_{i=1}^{K} |U_i| = \sum_{j\in[n]} |\{i \in [K] \mid U_i \ni j\}|.$$

Hence there exists j such that $|\{i \in [K] \mid U_i \ni j\}| \geq K/3$. Let $S_1 \coloneqq \{j\}$, and continue this process for the set of bits $[n] \setminus \{j\}$ and the set of assignments $\{\rho_i \colon U_i \not\ni j\}$. Suppose the previous iteration yields a set $S_k \subseteq [n]$ of size k and a set of indices $T_k \subseteq [K]$. Then let $U'_i \coloneqq U_i \setminus S_k$ for $i \in T_k$. Then $|U'_1|, \dots, |U'_K| \geq (n-1)/2 - k \geq n/3$, where the last inequality is true if $k < n/6$. As before, there exists $j \notin S_k$ such that $|\{i \in T_k \mid U'_i \ni j\}| \geq |T_k|/3$. We then let $S_{k+1} \coloneqq S_k \cup \{j\}$ and $T_{k+1} \coloneqq \{i \in T_k \mid U'_i \not\ni j\}$.

Clearly $|T_k| \leq K \cdot (2/3)^{k-1}$, hence in $\tau = \lceil \log_{3/2} K \rceil \leq 2\log_2 K \leq 2|I| \leq n/10$ steps we eliminate all assignments from the set, i.e. S_τ satisfies that for every assignment ρ to I that fixes at most $(n+1)/2$ output bits, there exists $j \in S_\tau$ that is not fixed by ρ. (The bound $2|I| \leq n/10$ guarantees that the condition $k < n/6$ holds). ◀

Let I be the set of all k-influential bits. If $|I| \geq n/20$ we get the desired lower bound immediately, so assume otherwise. Then let S be the set given by Lemma 21, $|S| = O(|I|)$. By Lemma 20 we get that $|S| = \Omega(n/(dk))$, so $|I| = \Omega(n/(dk))$ as well. ◀

References

1 Ryan Alweiss, Shachar Lovett, Kewen Wu, and Jiapeng Zhang. Improved bounds for the sunflower lemma. *Annals of Mathematics*, 194(3):795–815, 2021. doi:10.4007/annals.2021.194.3.5.

2 Lászió Babai. Random oracles separate pspace from the polynomial-time hierarchy. *Information Processing Letters*, 26(1):51–53, 1987. doi:10.1016/0020-0190(87)90036-6.

3 Chris Beck, Russell Impagliazzo, and Shachar Lovett. Large deviation bounds for decision trees and sampling lower bounds for ac0-circuits. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 101–110, 2012. doi:10.1109/FOCS.2012.82.

4 Tolson Bell, Suchakree Chueluecha, and Lutz Warnke. Note on sunflowers. *Discrete Mathematics*, 344(7):112367, 2021. doi:10.1016/j.disc.2021.112367.

5 Olaf Beyersdorff, Samir Datta, Andreas Krebs, Meena Mahajan, Gido Scharfenberger-Fabian, Karteek Sreenivasaiah, Michael Thomas, and Heribert Vollmer. Verifying proofs in constant depth. *ACM Trans. Comput. Theory*, 5(1):Art. 2, 23, 2013. doi:10.1145/2462896.2462898.

6 Andrej Bogdanov, Krishnamoorthy Dinesh, Yuval Filmus, Yuval Ishai, Avi Kaplan, and Akshayaram Srinivasan. Bounded indistinguishability for simple sources. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 – February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 26:1–26:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.26.

7 Andrej Bogdanov, Yuval Ishai, Emanuele Viola, and Christopher Williamson. Bounded indistinguishability and the complexity of recovering secrets. In *Advances in Cryptology – CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, pages 593–618. Springer, 2016.

8 Eshan Chattopadhyay, Jesse Goodman, and David Zuckerman. The Space Complexity of Sampling. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, volume 215 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 40:1–40:23, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITCS.2022.40.

9 Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functions. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 670–683, 2016.

10 Gil Cohen and Leonard J Schulman. Extractors for near logarithmic min-entropy. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 178–187. IEEE, 2016.

11 Artur Czumaj. Random permutations using switching networks. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 703–712, New York, NY, USA, 2015. Association for Computing Machinery. doi:10.1145/2746539.2746629.

12 C. M. Fortuin, P. W. Kasteleyn, and J. Ginibre. Correlation inequalities on some partially ordered sets. *Communications in Mathematical Physics*, 22:89–103, 1971. doi:10.1007/bf01651330.

13 Efraim Gelman and Amnon Ta-Shma. The Benes Network is $q(q-1)/2n$-Almost q-set-wise Independent. In Venkatesh Raman and S. P. Suresh, editors, *34th International Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS 2014)*, volume 29 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 327–338, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.FSTTCS.2014.327.

14 Mika Göös and Thomas Watson. A lower bound for sampling disjoint sets. *ACM Trans. Comput. Theory*, 12(3):Art. 20, 13, 2020. doi:10.1145/3404858.

15 Torben Hagerup. Fast parallel generation of random permutations. In *Automata, Languages and Programming: 18th International Colloquium Madrid, Spain, July 8–12, 1991 Proceedings 18*, pages 405–416. Springer, 1991.

16 T. E. Harris. A lower bound for the critical probability in a certain percolation process. *Proc. Cambridge Philos. Soc.*, 56:13–20, 1960.

17 Johan Håstad. Computational limitations of small-depth circuits. *MIT Press*, 1987.

18 Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *Journal of cryptology*, 9(4):199–216, 1996.

19 Joe Kilian. Founding crytpography on oblivious transfer. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 20–31, New York, NY, USA, 1988. Association for Computing Machinery. doi:10.1145/62212.62215.

20 Andreas Krebs, Nutan Limaye, Meena Mahajan, and Karteek Sreenivasaiah. Small depth proof systems. *ACM Trans. Comput. Theory*, 9(1):Art. 2, 26, 2016. doi:10.1145/2956229.

21 Shachar Lovett and Emanuele Viola. Bounded-depth circuits cannot sample good codes. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, USA, June 8–10, 2011*, pages 243–251. IEEE Computer Society, 2011. doi:10.1109/CCC.2011.11.

22 Shachar Lovett and Emanuele Viola. Bounded-depth circuits cannot sample good codes. *Comput. Complexity*, 21(2):245–266, 2012. doi:10.1007/s00037-012-0039-3.

23 Anup Rao. Coding for sunflowers, 2019. doi:10.48550/ARXIV.1909.04774.

24 Benjamin Rossman. The monotone complexity of k-clique on random graphs. *SIAM J. Comput.*, 43(1):256–279, 2014. doi:10.1137/110839059.

25 R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 77–82, New York, NY, USA, 1987. Association for Computing Machinery. doi:10.1145/28395.28404.

26 Emanuele Viola. The complexity of distributions. *SIAM J. Comput.*, 41(1):191–218, 2012. doi:10.1137/100814998.

27 Emanuele Viola. Extractors for Turing-machine sources. In Anupam Gupta, Klaus Jansen, José D. P. Rolim, and Rocco A. Servedio, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques – 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15–17, 2012. Proceedings*, volume 7408 of *Lecture Notes in Computer Science*, pages 663–671. Springer, 2012. doi:10.1007/978-3-642-32512-0_56.

28 Emanuele Viola. Extractors for circuit sources. *SIAM J. Comput.*, 43(2):655–672, 2014. doi:10.1137/11085983X.

29 Emanuele Viola. Quadratic maps are hard to sample. *ACM Trans. Comput. Theory*, 8(4):18:1–18:4, 2016. doi:10.1145/2934308.

30 Emanuele Viola. Sampling lower bounds: Boolean average-case and permutations. *SIAM J. Comput.*, 49(1):119–137, 2020. doi:10.1137/18M1198405.

31 Emanuele Viola. New sampling lower bounds via the separator, 2021.

32 Adam Bene Watts and Natalie Parham. Unconditional quantum advantage for sampling with shallow circuits, 2023. arXiv:2301.00995.

A Switching Networks for Sampling Linear Slices

In this section, we discuss the limitations of switching networks for constructing decision tree samplers.

▶ **Definition 22.** *A switching network of depth d is a sequence of d matchings $M_1, \dots, M_d$. Each M_i is a set of $n/2$ disjoint pairs of elements from $[n]$. The distribution generated by a switching network over a slice $\binom{[n]}{\ell}$ is defined as follows:*

- *Initialize the string as $1^\ell 0^{n-\ell}$;*
- *For each $i \in [d]$: for every pair in $(a, b) \in M_i$, we toss a fair coin, and if it comes up heads, we swap the ath and the bth bits of the current sequence.*

▶ **Lemma 23** (A variation of [26]). *Suppose there exists a switching network of depth d that generates the variable $\boldsymbol{X}$ over $\binom{[n]}{\ell}$. Then there exists a decision depth d sampler for $\boldsymbol{X}$ such that each input bit affects at most 2^d output bits. Moreover, the support of $\boldsymbol{X}$ is a subset of $\binom{[n]}{\ell}$.*

Proof. The input bits correspond to the coin tosses in the switching network. Each output bit is computed by tracing back its initial position: first, we query the coin corresponding to the pair in M_d containing the bit, then we query the coin corresponding to the pair in M_{d-1} and so on until we compute the location of the bit in the initial sequence. Then if the location is in $[\ell]$ we output 1 and otherwise output 0. It is easy to see that the described sampler has the required properties. ◀

▶ **Lemma 24.** *Let $\alpha \in (0, 1)$ be a constant. Suppose $\boldsymbol{X}$ is samplable with a d-local sampler such that each input bit affects at most c output bits, the support of $\boldsymbol{X}$ is within $\binom{[n]}{\alpha n}$, and $n/(cd)^2 = \omega(2^{cd})$. Then $\Delta(\boldsymbol{X}, \boldsymbol{U}^n_{\alpha n}) = 1 - o(1)$.*

Proof. For each output bit $i \in [n]$ of $\boldsymbol{X}$, let $N(i) \subseteq [n]$, the *neighborhood* of i, be the set of output bits that share an affecting input bit with i. By assumption, $|N(i)| \le cd$. Let us greedily choose a set of output bits with disjoint neighborhoods: $t_1 = 1$, and for $j > 1$,

$t_j \in [n]$ is a bit such that $N(t_j) \cap (N(t_1) \cup \cdots \cup N(t_{j-1})) = \emptyset$. For each output bit i there are at most $(cd)^2$ output bits j for which $N(i) \cap N(j) \neq \emptyset$, so the greedy process yields $\ell \geq n/(cd)^2$ bits.

Suppose that there is a bit $i \in \{t_1, \ldots, t_\ell\}$ such that $\Pr[\boldsymbol{X}_i = 0] > 0$ and $\Pr[\boldsymbol{X}_{N(i)} = 1^{N(i)}] > 0$. Then we use the approach from Proposition 18: let $x \in \binom{[n]}{\alpha n}$ be a string in the support of $\boldsymbol{X}$ such that $x_{N(i)} = 1^{N(i)}$, and let ρ be the input bits that yield x. Since $\Pr[\boldsymbol{X}_i = 0] > 0$, we can change the bits of ρ affecting the ith output bit such that its value switches to 0. Denote the resulting input by x'. Observe then that $x_{[n] \setminus N(i)} = x'_{[n] \setminus N(i)}$, since $N(i)$ is the set of output bits that are affected by the input bits affecting the ith bit. Then $|x'| < |x|$, hence $x' \notin \binom{[n]}{\alpha n}$, so it does not lie in the support of $\boldsymbol{X}$, which is a contradiction.

Now let us analyze the case when there are no bits satisfying the condition. Let $I \subseteq \{t_1, \ldots, t_\ell\}$ consist of those output bits for which $\Pr[\boldsymbol{X}_i = 0] = 0$. If $|I| \geq \ell/2$ then

$$\begin{aligned}\Delta(\boldsymbol{X}, \boldsymbol{U}^n_{\alpha n}) &\geq \left|\Pr[\boldsymbol{X}_I = 1^I] - \Pr[(\boldsymbol{U}^n_{\alpha n})_I = 1^I]\right| = 1 - \binom{n-|I|}{\alpha n}\binom{n}{\alpha n}^{-1} \\ &= 1 - \prod_{j=0}^{\alpha n - 1} \frac{n - |I| - j}{n - j} \geq 1 - \left(1 - \frac{|I|}{n - \alpha n + 1}\right)^{\alpha n} \geq 1 - 2^{-\Omega(\frac{\alpha}{1-\alpha}|I|)} = 1 - 2^{-\Omega(\ell)}.\end{aligned}$$

Since $\ell = \Omega(n/(cd)^2) = \omega(1)$, in this case $\Delta(\boldsymbol{X}, \boldsymbol{U}^n_{\alpha n}) = 1 - o(1)$.

Now suppose that $|I| \leq \ell/2$, and let $J = \{t_1, \ldots, t_\ell\} \setminus I$. Assume that all bits in J satisfy $\Pr[\boldsymbol{X}_{N(i)} \neq 1^{N(i)}] = 1$. Let us compute the probability of this event for $\boldsymbol{U}^n_{\alpha n}$:

$$\begin{aligned}\Pr[(\boldsymbol{U}^n_{\alpha n})_{N(i)} \neq 1^{N(i)}] &= 1 - \binom{n - |N(i)|}{\alpha n}\binom{n}{\alpha n}^{-1} = 1 - \prod_{j=0}^{\alpha n - 1} \frac{n - |N(i)| - j}{n - j} \\ &\leq 1 - \left(1 - \frac{|N(i)|}{n}\right)^{\alpha n} \leq 1 - 2^{-\Omega(\alpha |N(i)|)} = 1 - 2^{-\Omega(cd)}.\end{aligned}$$

Therefore

$$\Pr[(\boldsymbol{U}^n_{\alpha n})_{N(i)} \neq 1^{N(i)} \text{ for all } i \in J] = \prod_{i \in J}(1 - 2^{-\Omega(cd)}) \leq (1 - 2^{-\Omega(cd)})^{\ell/2}.$$

Since $\ell/2 = n/(2(cd)^2) = \omega(2^{cd})$, we get the desired lower bound on the statistical distance in this case as well. ◀

▶ **Corollary 25.** *Let $\alpha \in (0,1)$ be a constant. Any switching network that generates a distribution that is $1 - \Omega(1)$ close to $\boldsymbol{U}^n_{\alpha n}$ has depth $\Omega(\log \log n)$.*

Proof. Consider a switching network of depth d that generates a distribution that is $1 - \Omega(1)$ close to $\boldsymbol{U}^n_{\alpha n}$. Lemma 23 translates it to a decision depth d sampler for $\boldsymbol{U}^n_{\alpha n}$ such that each input bit affects at most $c = 2^d$ output bits. The sampler is $1 - \Omega(1)$ close to $\boldsymbol{U}^n_{\alpha n}$, and supported within $\binom{[n]}{\alpha n}$. The result now follows from Lemma 24. ◀

B Exposition of the $\Omega(\log^* n)$ lower bound for Majority

The following is an exposition of the proof of [5, Theorem 5.1].

Suppose that n is odd, and consider a locality c proof system for the vectors containing more 1s than 0s, that is, having Hamming weight at least $(n+1)/2$. We can assume that $c \geq 2$.[4]

The proof system has *inputs* and *outputs*. The number of outputs is n, and each one depends on at most c input bits. An input bit is *d-influential* if at least d output bits depend on it. There are at most cn/d many d-influential input bits.

Let $1 = B(0) < B(1) < \cdots < B(c+1)$ be a sequence of constants (depending on c but not on n), and let $d(\ell) = cn/B(\ell)$, so that $cn = d(0) > d(1) > \cdots > d(c+1) = \Omega(n)$. Thus there are at most $B(\ell)$ many input bits which are $d(\ell)$-influential.

We will construct a sequence of sets $\emptyset = R_0 \subseteq R_1 \subseteq \cdots \subseteq R_c \subseteq [n]$ with the following property: *If ρ is a truth assignment to the $d(\ell)$-influential variables which extends to a complete truth assignment setting all coordinates in R_ℓ to zero, then for each coordinate $i \notin R_\ell$, ρ also extends to a complete truth assignment setting coordinate i to zero.*[5]

Given $R_{\ell-1}$, here is how we construct R_ℓ. We start with $R \coloneqq R_{\ell-1}$, and will potentially add more output coordinates to R. At any point, suppose that there is a truth assignment ρ to the $d(\ell)$-influential variables which (i) extends to a complete truth assignment setting all coordinates in R to zero, and (ii) for some coordinate $i \notin R$, any complete truth assignment extending ρ sets coordinate i to one. If that happens, then we add i to R. Henceforth, ρ will not come up again, since no complete truth assignment extending ρ sets i to one, and i belongs to R. Eventually, there is no such "bad" truth assignment, and we set $R_\ell \coloneqq R$.

If $i \in R_\ell$ then there exists some $r \leq \ell$, some $R \subseteq R_r$, and some truth assignment ρ_i to the $d(r)$-influential variables, such that ρ_i extends to a complete truth assignment setting all coordinates in R to zero, and any complete truth assignment extending ρ_i sets i to one. If $j \in R_\ell$ was added after i then the truth assignment ρ_j extends to a complete truth assignment which sets all coordinates in $R \cup \{i\}$ to zero. In particular, ρ_j doesn't extend ρ_i (as a special case, $\rho_j \neq \rho_i$). It follows that if we extend each ρ_i arbitrarily to a truth assignment to the $d(\ell)$-influential variables, then the resulting assignments will all be different. Consequently,

$$|R_\ell| \leq 2^{B(\ell)}.$$

For large enough n, this will be at most $\frac{n-1}{2}$.

We show below that for a proper choice of parameters, there is an output coordinate i which satisfies the following, for all $\ell \in \{0, \ldots, c\}$: *$i \notin R_\ell$, and all inputs to i which are also inputs to R_ℓ are $d(\ell+1)$-influential.* We will show that for each $\ell \in \{0, \ldots, c\}$, i has an input which is $d(\ell+1)$-influential but not $d(\ell)$-influential. For different ℓ these inputs are different (since an input which is not $d(\ell)$-influential is also not $d(r)$-influential for all $r < \ell$), and so i depends on $c+1$ inputs, which is impossible.

Let $\ell \in \{0, \ldots, c\}$. Let us show that i has an input which is $d(\ell+1)$-influential but not $d(\ell)$-influential. We do this by contradiction: suppose that all $d(\ell+1)$-influential inputs of i are $d(\ell)$-influential. By assumption, $i \notin R_\ell$ and all inputs to i which are also inputs to R_ℓ are $d(\ell)$-influential. Let $N(i)$ consist of i together with all other output bits which share some non-$d(\ell)$-influential bit with i. All of these shared input bits are in fact non-$d(\ell+1)$-influential, and so

$$|N(i)| \leq 1 + cd(\ell+1) \leq 1 + cd(1) = 1 + \frac{c^2 n}{B(1)}.$$

[4] Alternatively, change $B(0)$ or redefine d-influential as having *more than* d output bits depending on it.
[5] In the paper, ρ extends to a complete truth assignment setting both R_ℓ and i to zero.

If $B(1) > 2c^2$ then $|N(i)| < 1 + \frac{n}{2}$ and so $|N(i)| \leq (n+1)/2$. Since $|R_c| \leq \frac{n-1}{2}$, the proof system generates some vector v of weight $(n+1)/2$ in which all coordinates of R_c are zero and all coordinates of $N(i)$ are one. Consider an arbitrary complete truth assignment α which generates v, and let ρ be its restriction to the $d(\ell)$-influential coordinates. Since $i \notin R_\ell$, by construction, we know that ρ extends to some complete truth assignment β which sets i to zero. Now consider the following complete truth assignment:

$$\gamma(j) = \begin{cases} \rho(j) & \text{if } j \text{ is } d(\ell)\text{-influential,} \\ \beta(j) & \text{if } j \text{ is not } d(\ell)\text{-influential and influences } N(i), \\ \alpha(j) & \text{if } j \text{ is not } d(\ell)\text{-influential and doesn't influence } N(i). \end{cases}$$

Since α, β both extend ρ, γ agrees with them on the $d(\ell)$-influential variables. Therefore the output generated by γ agrees with that generated by α except for the coordinates in $N(i)$, which could change from one to zero. Moreover, the i'th output of γ agrees with the i'th output of β, namely, it is zero. Therefore the output generated by γ has Hamming weight strictly less than $(n+1)/2$, which is impossible.

It remains to show that there exists an output bit i such that $i \notin R_c$, and for all $\ell \in \{0, \ldots, c\}$, all inputs to i which are also inputs to R_ℓ are $d(\ell+1)$-influential. We do this by giving an upper bound on the number of bad output bits. An output bit is bad if it either belongs to R_c, or for some $\ell \in \{0, \ldots, c\}$, there is a joint input of i and R_ℓ which is not $d(\ell+1)$-influential. If i is bad due to some ℓ, then there must be some non-$d(\ell+1)$-influential input of R_ℓ which is an input of i. Therefore the number of bad inputs is at most

$$|R_c| + \sum_{\ell=0}^{c} |R_\ell| \cdot c \cdot d(\ell+1) \leq 2^{B(c)} + n \cdot c^2 \sum_{\ell=0}^{c} \frac{2^{B(\ell)}}{B(\ell+1)}.$$

For a judicious choice of the sequence $B(\ell)$, the coefficient of n will be strictly less than 1, and so for large enough n, there are fewer than n bad inputs.

One choice for the sequence $B(\ell)$ is

$$B(\ell+1) = 2^{B(\ell)} \cdot 2c^2(c+1).$$

In particular, $B(1) = 4c^2(c+1) > 2c^2$, which was needed above. For this choice of $B(0), \ldots, B(c)$, the number of bad inputs is at most

$$2^{B(c)} + \frac{n}{2},$$

which less than n if $2^{B(c)} \leq \frac{n-1}{2}$, a condition which was required at a different step of this proof.

Roughly speaking, $B(\ell+1) \approx 2c^3 \cdot 2^{B(\ell)}$, and so $B(c) \approx 2 \uparrow\uparrow c$. Therefore $2^{B(c)} \leq \frac{n-1}{2}$, and so the argument works, for $c \leq \kappa \log^* n$, for an appropriate constant κ.

Hardness of the (Approximate) Shortest Vector Problem: A Simple Proof via Reed-Solomon Codes*

Huck Bennett ✉
Oregon State University, Corvallis, OR, USA

Chris Peikert ✉
University of Michigan, Ann Arbor, MI, USA
Algorand, Inc., Boston, MA, USA

Abstract

We give a *simple* proof that the (approximate, decisional) Shortest Vector Problem is NP-hard under a randomized reduction. Specifically, we show that for any $p \geq 1$ and any constant $\gamma < 2^{1/p}$, the γ-approximate problem in the ℓ_p norm (γ-GapSVP$_p$) is not in RP unless NP $\subseteq$ RP. Our proof follows an approach pioneered by Ajtai (STOC 1998), and strengthened by Micciancio (FOCS 1998 and SICOMP 2000), for showing hardness of γ-GapSVP$_p$ using *locally dense lattices.* We construct such lattices simply by applying "Construction A" to Reed-Solomon codes with suitable parameters, and prove their local density via an elementary argument originally used in the context of Craig lattices.

As in all known NP-hardness results for GapSVP$_p$ with $p < \infty$, our reduction uses randomness. Indeed, it is a notorious open problem to prove NP-hardness via a deterministic reduction. To this end, we additionally discuss potential directions and associated challenges for derandomizing our reduction. In particular, we show that a close deterministic analogue of our local density construction would improve on the state-of-the-art explicit Reed-Solomon list-decoding lower bounds of Guruswami and Rudra (STOC 2005 and IEEE Transactions on Information Theory 2006).

As a related contribution of independent interest, we also give a polynomial-time algorithm for decoding n-dimensional "Construction A Reed-Solomon lattices" (with different parameters than those used in our hardness proof) to a distance within an $O(\sqrt{\log n})$ factor of Minkowski's bound. This asymptotically matches the best known distance for decoding near Minkowski's bound, due to Mook and Peikert (IEEE Transactions on Information Theory 2022), whose work we build on with a somewhat simpler construction and analysis.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Error-correcting codes; Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Lattices, Shortest Vector Problem, Reed-Solomon codes, NP-hardness, derandomization

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.37

Category RANDOM

Related Version *Full Version*: https://arxiv.org/abs/2202.07736 [6]

Funding *Huck Bennett*: Part of this work was completed while the author was at the University of Michigan and supported by NSF Grant No. CCF-2006857. The views expressed are those of the authors and do not necessarily reflect the official policy or position of the NSF.
Chris Peikert: This author's work was supported by NSF Grant No. CCF-2006857.

Acknowledgements We thank Swastik Kopparty [29] for very helpful answers to several of our questions.

* Due to space constraints, this version of the paper omits some material. We strongly encourage the reader to view the full version [6].

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 37; pp. 37:1–37:20
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

> [I]t may easily happen that other, perhaps in some sense simpler, lattices also have the properties that are required from L to complete the proof... There are different reasons which may motivate the search for such a lattice: to make the proof deterministic; to improve the factor in the approximation result; to make the proof simpler.
>
> Miklós Ajtai, [3, Remark 2]

A *lattice* $\mathcal{L}$ is the set of all integer linear combinations of some n linearly independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{R}^m$. The matrix $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ whose columns are these vectors is called a *basis* of $\mathcal{L}$, and n is called its *rank*. Formally, the lattice $\mathcal{L}$ generated by B is defined as

$$\mathcal{L} = \mathcal{L}(B) := \left\{ \sum_{i=1}^{n} a_i \boldsymbol{b}_i : a_1, \ldots, a_n \in \mathbb{Z} \right\} .$$

Lattices are classically studied mathematical objects, and have proved invaluable in many computer science applications, especially the design and analysis of cryptosystems. Indeed, the area of lattice-based cryptography, which designs cryptosystems whose security is based on the apparent intractability of certain computational problems on lattices, has flourished over the past quarter century. (See [36] and its bibliography for a comprehensive summary and list of references.)

The central computational problem on lattices is the Shortest Vector Problem (SVP): given a lattice basis B as input, the goal is to find a shortest non-zero vector in $\mathcal{L}(B)$. This paper is concerned with its γ-approximate decision version in the ℓ_p norm (γ-GapSVP$_p$), where $p \geq 1$ is fixed and the approximation factor $\gamma = \gamma(n) \geq 1$ is some function of the lattice rank n (often a constant). Here the input additionally includes a distance threshold $s > 0$, and the goal is to determine whether the length (in the ℓ_p norm) $\lambda_1^{(p)}(\mathcal{L}) := \min_{\boldsymbol{v} \in \mathcal{L} \setminus \{\boldsymbol{0}\}} \|\boldsymbol{v}\|_p$ of the shortest non-zero vector in $\mathcal{L}$ is at most s, or is strictly greater than γs, when one of the two cases is promised to hold. For the exact problem, where $\gamma = 1$, we often simply write GapSVP$_p$.

Motivated especially by its central role in the security of lattice-based cryptography, understanding the complexity of γ-GapSVP has been the subject of a long line of work. In an early technical report, van Emde Boas [38] initiated the study of the hardness of lattice problems more generally, and in particular showed that GapSVP$_\infty$ is NP-hard. Seventeen years later, Ajtai [3] finally showed similar hardness for the important Euclidean case of $p = 2$, i.e., he showed that exact GapSVP$_2$ is NP-hard, though under a *randomized* reduction. Subsequent work [10, 31, 27, 26, 23, 32] improved this by showing that γ-GapSVP$_p$ in any ℓ_p norm is NP-hard to approximate for any constant $\gamma \geq 1$, and hard for nearly polynomial factors $\gamma = n^{\Omega(1/\log\log n)}$ assuming stronger complexity assumptions, also using randomized reductions. Recent work [1, 7] has also shown the *fine-grained* hardness of γ-GapSVP$_p$ for small constants γ (again under randomized reductions). On the other hand, γ-GapSVP$_p$ for finite $p \geq 2$ is unlikely to be NP-hard for approximation factors $\gamma \geq C_p\sqrt{n}$ (where C_p is a constant depending only on p) [18, 2, 35], and the security of lattice-based cryptography relies on the conjectured hardness of GapSVP or other problems for even larger (but typically polynomial) factors.

While this line of work has been very successful in showing progressively stronger hardness of approximation and fine-grained hardness for γ-GapSVP$_p$, it leaves some other important issues unresolved. First, for $p \neq \infty$ the hardness reductions and their analysis are rather complicated, and second, they are randomized. Indeed, it is a notorious, long-standing open problem to prove that GapSVP_p is NP-hard, even in its exact form, under a deterministic reduction for some finite p. While there have been some potential steps in this direction [31, 32], e.g., using plausible number-theoretic conjectures that appear very hard to prove, there has been no new progress on this front for a decade.

1.1 Our Contributions

The primary contribution of this work is to give a substantially simpler proof that γ-GapSVP_p is NP-hard under a randomized reduction, for any $p \geq 1$ and constant $\gamma < 2^{1/p}$. The heart of our reduction is a family of "gadget" lattices $\mathcal{L}$ derived from Reed-Solomon codes $\mathcal{C} \subseteq \mathbb{F}_q^n$ (for prime q) via the very natural "Construction A" [13], which simply sets $\mathcal{L} = \mathcal{C} + q\mathbb{Z}^n$. These lattices and their properties were previously studied in work of Karabed, Roth, and Siegel [25, 37]. Additionally, they are closely related to a family of algebraic lattices studied by Craig [14]. We take advantage of this prior work in our analysis (see Section 1.3 for details).

▶ **Theorem 1** (Hardness of γ-GapSVP_p)**.** *For any $p \geq 1$ and constant γ satisfying $1 \leq \gamma < 2^{1/p}$, γ-GapSVP_p is not in* RP *unless* NP $\subseteq$ RP*.*

We note that Theorem 1 is actually identical to the main result in [31]. As such, it matches the best known NP-hardness of approximation for γ-GapSVP_p (i.e., largest γ) achieved by a "one-shot" reduction for all sufficiently small p, including $p = 2$. By "one-shot," we mean that the reduction does not amplify the approximation factor from an initial fixed constant to an arbitrary constant (or more) via tensoring, as is done in [26, 23, 32]. (It is an interesting question whether our hard γ-GapSVP_p instances are amenable to tensoring; see Section 1.4.)

Although our reduction still uses randomness, we believe that it may be easier to derandomize than previous reductions, both due to its simplicity, and because of its close connection to prior work showing hardness of minimum distance problem on codes via a deterministic reduction [12]. To that end, in the full version of our paper [6] we also describe two approaches to potentially derandomizing our reduction, both of which aim to deterministically construct a particular lattice coset and lower bound the number of short vectors in it (see Section 1.2 for the motivation for this). The first approach is based on Fourier analysis, using similar techniques to those in [12], and the second is based on "smooth" proxies for point-counting functions.

In the full version of our paper [6] we also show that a close deterministic analog of our randomized local-density construction would imply improved explicit Reed-Solomon list-decoding lower bounds, going beyond the current state of the art from [20]. One may interpret this implication either pessimistically, as a barrier to a very strong derandomization of our reduction, or optimistically, as a potential route to improve Reed-Solomon list-decoding lower bounds. Here there is a further connection between the two problems, in that [20] obtains its list-decoding lower bounds by using the same Fourier-analytic tool underlying one of our derandomization attempts – specifically, the Weil bound for character sums . Unfortunately, the Weil bound falls just short of what we need in our context. (The Weil bound and related techniques were first used for counting Reed-Solomon code words in [11], and were also used in the deterministic hardness reduction for the minimum distance problem on codes in [12].)

Efficient decoding near Minkowski's bound. As a separate contribution of independent interest, in Appendix A we give a polynomial-time algorithm for decoding "Construction A Reed-Solomon lattices" of rank n – the same family of lattices as in our hardness reduction, but instantiated with different parameters – to a distance within a $O(\sqrt{\log n})$ factor of

Minkowski's bound.[1] The $O(\sqrt{\log n})$ factor in our result asymptotically matches the best factor known from prior work [34], which is for a different family of lattices. In fact, we rely on one of the main underlying theorems from that work, but give a simpler construction and analysis based on individual Reed-Solomon codes instead of towers of BCH codes.

Let $\mathrm{RS}_q[k, S]$ denote the dimension-k Reed-Solomon code over $\mathbb{F}_q$ with evaluation set S (defined below in Equation (3)). Note that $n = q$ is the rank of the lattice $\mathcal{L}$ in the following theorem.

▶ **Theorem 2** (Decoding near Minkowski's bound, informal). *Let q be prime and let $k := \lfloor q/(2\log_2 q)\rfloor \leq q/2$. Then for the "Construction A Reed-Solomon" lattice $\mathcal{L} := \mathrm{RS}_q[q-k, \mathbb{F}_q] + q\mathbb{Z}^q \subseteq \mathbb{Z}^q$:*

1. *We have $\Omega(\sqrt{q/\log q}) \leq \lambda_1(\mathcal{L}) \leq \sqrt{q} \cdot \det(\mathcal{L})^{1/q} \leq O(\sqrt{q})$, i.e., the minimum distance is within a $O(\sqrt{\log q})$ factor of Minkowski's bound.*
2. *There is an algorithm that, on input q and a vector $\boldsymbol{y} \in \mathbb{R}^q$, outputs all lattice vectors $\boldsymbol{v} \in \mathcal{L}$ satisfying $\|\boldsymbol{y} - \boldsymbol{v}\| \leq C\sqrt{k} \approx C\sqrt{q/(2\log_2 q)}$ in time polynomial in q, for some universal constant $C > 0$.*

This result adds to a separate line of work on efficient (list) decoding for various families of lattices [33, 19, 15, 34]. Recently, Ducas and van Woerden [16] further motivated this study by showing cryptographic applications of lattices that can be efficiently decoded near Minkowski's bound. (However, their application is most compelling when the minimum distances of both the lattice and its dual are close to Minkowski's bound, which is not the case in the present setting.)

1.2 Technical Overview

Here we give an overview of the key new elements in the proof of our main hardness theorem (Theorem 1), which are the focus of Section 3. For concision, we defer the technical aspects of our efficient decoding algorithm to Appendix A and of our derandomization attempts to the full version of our paper [6], respectively.

Besides using randomness, another common feature in nearly all prior hardness results for GapSVP_p is the use of *locally dense lattices* as advice (the only exception being [27]). Roughly speaking, a locally dense lattice for relative distance $\alpha \in (0, 1)$ in the ℓ_p norm is a lattice $\mathcal{L}$ and a coset $\boldsymbol{x} + \mathcal{L}$ (i.e., the lattice "shifted by" some vector $\boldsymbol{x}$) such that there are at least subexponentially many (in the lattice rank) vectors $\boldsymbol{v} \in \boldsymbol{x} + \mathcal{L}$ satisfying $\|\boldsymbol{v}\|_p \leq \alpha \cdot \lambda_1^{(p)}(\mathcal{L})$. One may view such a coset $\boldsymbol{x} + \mathcal{L}$ as a "bad" configuration for list-decoding $\mathcal{L}$ to within relative distance α in the ℓ_p norm, because there are many lattice vectors relatively close to $-\boldsymbol{x}$.[2]

Prior works have obtained locally density from a variety of lattice families: the Schnorr-Adelman prime number lattices [3, 10, 31]; a variant of Construction A [26] and Construction D [32] applied to (towers of) BCH codes; and random sublattices of $\mathbb{Z}^n$ and lattices with exponential kissing number [1, 7]. In this work, we give a simple construction of locally dense lattices from Reed-Solomon codes, as described below.

[1] Minkowski's bound gives an upper bound on the "normalized density" of a lattice $\mathcal{L}$. Specifically, it asserts that $\lambda_1^{(2)}(\mathcal{L}) \leq \sqrt{n} \cdot \det(\mathcal{L})^{1/n}$ for all rank-n lattices $\mathcal{L}$, where $\det(\mathcal{L}) = \sqrt{\det(B^T B)}$ for any basis B of $\mathcal{L}$.

[2] For technical reasons, the formal definition of local density, given in Definition 8, also requires a linear transform that maps the short vectors in $\boldsymbol{x} + \mathcal{L}$ onto the set of all binary vectors of a given dimension. Such a transform can be obtained by random sampling using a probabilistic version of Sauer's Lemma (see Theorem 9) that is now standard in this context [3, 31]. Therefore, we defer further discussion of this issue to the main body.

Our main reduction (Theorem 12) shows how to use a locally dense lattice for relative distance α in the ℓ_p norm to prove NP-hardness (via a randomized reduction) of γ-GapSVP$_p$ for any constant $\gamma > 1/\alpha$. This reduction is very similar to those from prior works, so for the remainder of this section we focus on summarizing our new construction of locally dense lattices.

Locally dense lattices from Reed-Solomon codes. We start with some basic definitions and facts used in our construction. Recall that the Construction A lattice obtained from a linear code $\mathcal{C} \subseteq \mathbb{F}_q^n$ for some prime q is defined as $\mathcal{L} := \mathcal{C} + q\mathbb{Z}^n$, i.e., an integer vector $\boldsymbol{z} \in \mathbb{Z}^n$ is in the lattice if and only if $\boldsymbol{z} \bmod q$ is a code word. In fact, it will often be convenient to work with an equivalent "dual view" of Construction A lattices. Namely, if $H \in \mathbb{F}_q^{k \times n}$ is a parity-check matrix of a linear code $\mathcal{C} := \ker(H) \subseteq \mathbb{F}_q^n$ for prime q, then the *parity-check lattice* $\mathcal{L}^\perp(H)$ obtained from H is defined as

$$\mathcal{L}^\perp(H) := \{\boldsymbol{z} \in \mathbb{Z}^n : H\boldsymbol{z} = \boldsymbol{0} \in \mathbb{F}_q^k\} = \ker(H) + q\mathbb{Z}^n = \mathcal{C} + q\mathbb{Z}^n \ . \tag{1}$$

Such lattices have determinant $\det(\mathcal{L}^\perp(H)) = |\mathbb{Z}^n/\mathcal{L}^\perp(H)| \leq q^k$, with equality exactly when H has full row rank (see Lemma 4).

We next define the family of parity-check matrices $H = H_q(k, S)$ that we use to construct our family of locally dense lattices. Such a matrix is parameterized by a prime q, a positive integer k, and a set $S \subseteq \mathbb{F}_q$. Letting $s_0, \ldots, s_{n-1}$ be the elements of S in some arbitrary order, we define

$$H = H_q(k, S) := \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ s_0 & s_1 & s_2 & \cdots & s_{n-1} \\ s_0^2 & s_1^2 & s_2^2 & \cdots & s_{n-1}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_0^{k-1} & s_1^{k-1} & s_2^{k-1} & \cdots & s_{n-1}^{k-1} \end{pmatrix} \in \mathbb{F}_q^{k \times n} \ . \tag{2}$$

That is, $H_q(k, S)$ is the transposed Vandermonde matrix whose (i, s)th entry is s^i, where for convenience we index the rows and columns of $H_q(k, S)$ by $i \in \{0, \ldots, k-1\}$ and $s \in S$, respectively, and define $0^0 := 1$.

The matrix $H = H_q(k, S)$ defined in Equation (2) is a *generator matrix* of the dimension-k Reed-Solomon code

$$\mathrm{RS}_q[k, S] := \{(p(s))_{s \in S} : p \in \mathbb{F}_q[x], \deg(p) < k\} \tag{3}$$

over $\mathbb{F}_q$ with evaluation set S, and hence is a parity-check matrix of its dual code, which is a so-called *generalized* Reed-Solomon (GRS) code (see [22, Theorem 5.1.6]). Moreover, in the special case where $S = \mathbb{F}_q$, it turns out that $H_q(k, S)$ is a parity-check matrix for the (ordinary) Reed-Solomon code $\mathrm{RS}_q[q-k, \mathbb{F}_q]$ of dimension $q - k$ with evaluation set $S = \mathbb{F}_q$. So, $\mathcal{L}^\perp(H_q(k, \mathbb{F}_q)) = \mathrm{RS}_q[q-k, \mathbb{F}_q] + q\mathbb{Z}^q$ is the Construction A lattice corresponding to the Reed-Solomon code $\mathrm{RS}_q[q-k, \mathbb{F}_q]$. For simplicity, in this overview we restrict to these "Construction A Reed-Solomon" lattices by taking $S = \mathbb{F}_q$, but note that our results generalize to any sufficiently large $S \subseteq \mathbb{F}_q$.

It is easy to see that for $k < q$, the GRS code having parity-check matrix H has minimum distance (in the Hamming metric) $k + 1$: any k columns of H are linearly independent, because they form a transposed Vandermonde matrix, while any $k + 1$ obviously are not. Therefore, the corresponding Construction A lattice $\mathcal{L} := \ker(H) + q\mathbb{Z}^q$ has minimum distance $\lambda_1^{(1)}(\mathcal{L}) \geq k + 1$ in the ℓ_1 norm. The key to our local density construction and its $\alpha \approx 1/2^{1/p}$

relative distance is that the ℓ_1 minimum distance is in fact almost *twice* this large (at least): in Theorem 14 we show that $\lambda_1^{(1)}(\mathcal{L}) \geq 2k$ when $k \leq q/2$. The proof is short and elementary, proceeding via Newton's identities. Essentially the same result and proof originally appeared in work by Roth and Siegel [37], and closely related analysis also appeared in work on Craig lattices [14] (see Section 1.3).

Obtaining a dense coset. Because the determinant of $\mathcal{L}$ (i.e., the number of its integer cosets) is q^k, the pigeonhole principle immediately implies that there exists an integer coset of $\mathcal{L}$ containing at least $\binom{q}{h}/q^k$ *binary* vectors in $\{0,1\}^q$ with Hamming weight h, which have ℓ_1 norm h. By setting parameters appropriately, this yields a coset with subexponentially many vectors of ℓ_1 norm at most $\alpha \cdot \lambda_1^{(1)}(\mathcal{L})$, for any constant $\alpha > 1/2$.

More specifically, set $h := \alpha \cdot (2k) \leq \alpha \cdot \lambda_1^{(1)}(\mathcal{L})$ and $q \approx k^{1/\varepsilon}$ for some positive constant $\varepsilon < 1 - 1/(2\alpha)$. (For simplicity, assume that h is an integer.) Then there must exist an integer coset of $\mathcal{L}$ containing at least

$$\frac{\binom{q}{h}}{q^k} \geq \left(\frac{q}{h}\right)^h \cdot q^{-k} = \frac{q^{(2\alpha-1)k}}{(2\alpha k)^{2\alpha k}} \approx \frac{q^{(2\alpha-1)k}}{q^{2\varepsilon\alpha k}} = q^{(2(1-\varepsilon)\alpha-1)k} = q^{\Omega(k)} = q^{\Omega(q^{\varepsilon})} \tag{4}$$

weight-h binary vectors, which is subexponentially large in q.

The above shows the *existence* of a suitable coset, but following previous works, it is straightforward to show that a *randomly sampled* coset from a suitable distribution is likely to have enough short vectors (see Lemma 16). Indeed, the difference between showing that such a coset exists, versus sampling one efficiently, versus deterministically computing one efficiently, is the main technical difference between getting a non-uniform, versus randomized, versus deterministic hardness reduction (respectively) for GapSVP_p using these techniques.

The above argument generalizes to arbitrary ℓ_p norms for finite p, albeit for larger relative distances $\alpha > 1/2^{1/p}$. Because $\mathcal{L}$ is integral, $\lambda_1^{(1)}(\mathcal{L}) \geq 2k$ implies that $\lambda_1^{(p)}(\mathcal{L}) \geq (2k)^{1/p}$ for any finite $p \geq 1$. Moreover, reparameterizing the calculation in Equation (4) by choosing $\alpha > 1/2^{1/p}$ and setting $h := \alpha^p \cdot (2k)$ shows that some coset of $\mathcal{L}$ contains subexponentially many binary vectors of Hamming weight h, and hence of ℓ_p norm $h^{1/p} = \alpha \cdot (2k)^{1/p} \leq \alpha \cdot \lambda_1^{(p)}(\mathcal{L})$. Therefore, this construction yields locally dense lattices in the ℓ_p norm for any constant relative distance $\alpha > 1/2^{1/p}$, which by our main reduction implies Theorem 1, i.e., randomized NP-hardness of γ-GapSVP_p for any constant $\gamma < 2^{1/p}$.

1.3 Additional Related Work

First, we note that after this work was first published [5] used the properties of the gadget lattices $\mathcal{L} = \mathcal{L}^{\perp}(H_q(k, S))$ we construct to show improved results and answer an open question about the *parameterized* complexity of GapSVP. Somewhat more specifically, [5] gave a parameterized analog of Theorem 1 by showing $\mathsf{W}[1]$-hardness (under randomized reductions) of γ-GapSVP_p for any $p \geq 1$ and any γ satisfying $1 \leq \gamma < 2^{1/p}$. This in particular answered a question from [17, 9], which asked whether γ-GapSVP was $\mathsf{W}[1]$-hard in the ℓ_1 norm. Prior to [5] (which crucially relied on this work), such hardness was not known even in the exact case of $\gamma = 1$.

The key lower bound of $\lambda_1^{(1)}(\mathcal{L}) \geq 2k$ for $\mathcal{L} = \mathcal{L}^{\perp}(H_q(k, S))$ follows immediately from works by Karabed, Roth, and Siegel [25, 37], which adapted an argument by Immink and Beenker [24]. In fact, [25, 37] showed their lower bounds for the *Lee* minimum distance of a family of BCH codes, including ones that are Reed-Solomon codes with parity check matrices of the form $H_q(k, S)$. However, their results immediately apply to the ℓ_1 minimum distance of $\mathcal{L} = \mathcal{L}^{\perp}(H_q(k, S))$ as well, because Lee distance is essentially "ℓ_1 distance with wrap-around."[3] Even earlier, Berlekamp [8, Chapter 9] gave an argument using Newton's identities to prove a similar lower bound on the Lee minimum distance of negacyclic codes, which are codes whose parity check matrices are of the form in Equation (2) but with its even-power rows deleted (i.e., with only its odd-power rows). Additionally, our Construction A Reed-Solomon lattices $\mathcal{L}$ are closely related to a family of algebraic lattices studied by Craig [14]; see also [13, Chapter 8, Section 6].[4] Indeed, [13, Chapter 8, Section 6, Theorem 7] again gives a very similar argument for lower bounding the minimum distance of Craig lattices using Newton's identities.Our proof of Theorem 14 is inspired by and similar to all of these "minimum distance lower bounds via Newton's identities" arguments, and in particular is very similar to the ones in [37] and [13, Chapter 8, Section 6, Theorem 7].

We omit further discussion of related work due to space constraints and encourage the reader to view the full version of this work [6].

1.4 Open Questions

The obvious question left open by our work is whether our reduction can be derandomized, using the same family of lattices. Addressing this is the main focus of Section 5 in the full version of this paper [6]. The full version also discusses several other open questions. We omit this material due to space constraints, but strongly encourage the reader to view [6].

2 Preliminaries

Throughout this work we adopt the convention that $0^0 := 1$ in any ring. For a positive integer k, define $[k] := \{0, 1, \ldots, k-1\}$.

In general, every vector or matrix is indexed by some specified set S. For example, $\boldsymbol{x} \in \mathbb{Z}^S$ is an integer vector indexed by S, having an entry $x_s \in \mathbb{Z}$ for each $s \in S$ (and no other entries). When the index set is $[n]$ for some non-negative integer n, we usually omit the brackets in the exponent and just write, e.g., $\mathbb{Z}^n$. We emphasize that in this case the indices start from zero. An object indexed by a finite set S of size $n = |S|$ can be reindexed by $[n]$, simply by enumerating $S = \{s_0, \ldots, s_{n-1}\}$ under some arbitrary order, and identifying index s_i with index i.

For a finite set S and a positive integer $h \leq |S|$, let $B_{S,h} := \{\boldsymbol{v} \in \{0,1\}^S : \|\boldsymbol{v}\|_1 = h\}$ be the set of binary vectors indexed by S of Hamming weight h. As above, when $S = [n]$ we often write $B_{n,h}$. Finally, let $\mathcal{B}_p^n(r) := \{\boldsymbol{x} : \|\boldsymbol{x}\|_p \leq r\} \subset \mathbb{R}^n$ denote the real n-dimensional ℓ_p ball of radius r centered at the origin.

2.1 Basic Lattice Definitions

Given a lattice $\mathcal{L} = \mathcal{L}(B)$ with basis $B \in \mathbb{R}^{m \times n}$, we define the *rank* of $\mathcal{L}$ to be n and the (ambient) *dimension* of $\mathcal{L}$ to be m. We denote the *minimum distance* of $\mathcal{L}$ in the ℓ_p norm, which is the length of a shortest non-zero vector in $\mathcal{L}$, by

[3] More precisely, the Lee distance of $\boldsymbol{x} \in \mathbb{F}_q^n$ is $\sum_{i=1}^n \min\{x_i, q - x_i\}$, where elements of the prime-order field $\mathbb{F}_q$ are identified in the natural way with the integers $\{0, 1, \ldots, q-1\}$. This is the natural analog of ℓ_1 distance on $\mathbb{F}_q$ (or $\mathbb{Z}_q$). Moreover, the ℓ_1 minimum distance of the Construction A lattice $\mathcal{C} + q\mathbb{Z}^n$ is equal to the minimum of q and the Lee minimum distance of $\mathcal{C}$.

[4] Specifically, [13] considers Craig lattices obtained from the coefficient vectors of polynomials in principal ideals of the form $(x-1)^m R$ in rings of the form $R = \mathbb{Z}[x]/(x^p - 1)$, for some prime p and integer $m \geq 1$. The original definition of [14] is slightly different, and uses the "canonical" (Minkowski) embedding of such ideals in the ring of integers $R = \mathbb{Z}[x]/(\Phi_p(x))$ of the pth cyclotomic number field.

$$\lambda_1^{(p)}(\mathcal{L}) := \min_{\boldsymbol{x}\in\mathcal{L}\setminus\{0\}} \|\boldsymbol{x}\|_p \ .$$

The central problem that we study in this work asks about the value of $\lambda_1^{(p)}(\mathcal{L})$ for a given input lattice $\mathcal{L}$.

▶ **Definition 3.** *For $p \geq 1$ and $\gamma = \gamma(n) \geq 1$, the decisional, γ-approximate Shortest Vector Problem in the ℓ_p norm (γ-GapSVP_p) is the promise problem defined as follows. The input consists of a basis $B \in \mathbb{Z}^{m\times n}$ of an integer lattice $\mathcal{L}$ and a distance threshold $s > 0$, and the goal is to determine whether the input is a YES instance or a NO instance, where these are defined as follows:*

- *YES instance: $\lambda_1^{(p)}(\mathcal{L}) \leq s$.*
- *NO instance: $\lambda_1^{(p)} > \gamma s$.*

We define the *determinant* of $\mathcal{L}$ to be $\det(\mathcal{L}) := \sqrt{\det(B^T B)}$, which is equal to $|\det(B)|$ when $m = n$ (i.e., when $\mathcal{L}$ is full-rank). We note that determinant is well defined because, although lattice bases are not unique, they are equivalent up to multiplication on the right by unimodular matrices.

The density of a rank-n lattice $\mathcal{L}$ is captured by its so-called *root Hermite factor* $\lambda_1(\mathcal{L})/\det(\mathcal{L})^{1/n}$.[5] The density of a lattice corresponds to its quality in various applications, including as the set of centers of a sphere packing and as an error-correcting code. Minkowski's bound asserts that the root Hermite factor of such a rank-n lattice is at most $\sqrt{n}$, which is convenient to write in expanded form as

$$\lambda_1(\mathcal{L}) \leq \sqrt{n} \cdot \det(\mathcal{L})^{1/n} \ . \tag{5}$$

2.2 Parity-Check Matrices and Lattices

For a prime q and a matrix $H \in \mathbb{F}_q^{k\times n}$, we define the *parity-check lattice* $\mathcal{L}^\perp(H)$ obtained from H as

$$\mathcal{L}^\perp(H) := \{\boldsymbol{z} \in \mathbb{Z}^n : H\boldsymbol{z} = \boldsymbol{0}\} = \ker(H) + q\mathbb{Z}^n \ . \tag{6}$$

Note that $\mathcal{L}^\perp(H)$ is simply the "Construction A" lattice [13, Chapter 5, Section 2] of the linear error-correcting code $\mathcal{C}$ having H as a parity-check matrix, i.e., $\mathcal{C} = \{\boldsymbol{c} \in \mathbb{F}_q^n : H\boldsymbol{c} = \boldsymbol{0}\}$. More generally, for any "syndrome" $\boldsymbol{u} \in \mathbb{F}_q^k$ we define

$$\mathcal{L}_{\boldsymbol{u}}^\perp(H) := \{\boldsymbol{x} \in \mathbb{Z}^n : H\boldsymbol{x} = \boldsymbol{u}\} \ .$$

If there exists some $\boldsymbol{x} \in \mathbb{Z}^n$ such that $H\boldsymbol{x} = \boldsymbol{u}$, then it follows immediately that $\mathcal{L}_{\boldsymbol{u}}^\perp(H)$ is simply the lattice coset $\boldsymbol{x}+\mathcal{L}^\perp(H)$. So, we can identify cosets of $\mathcal{L}^\perp(H)$ by their corresponding syndromes. We recall some standard properties of parity-check lattices, and give a proof for self-containment.

▶ **Lemma 4.** *Let q be a prime, let k and n be positive integers, and let $H \in \mathbb{F}_q^{k\times n}$ be a parity-check matrix. Then the parity-check lattice $\mathcal{L} = \mathcal{L}^\perp(H)$ has rank n and determinant $\det(\mathcal{L}) \leq q^k$, with equality if and only if the rows of H are linearly independent.*

[5] This ratio is the square root of the Hermite factor $\gamma(\mathcal{L}) := (\lambda_1(\mathcal{L})/\det(\mathcal{L})^{1/n})^2$, which is defined in this way for historical reasons.

Proof. The first claim follows simply by noting that $q\mathbb{Z}^n \subseteq \mathcal{L}^\perp(H) \subseteq \mathbb{Z}^n$. For the determinant, observe that the map $\boldsymbol{x} \mapsto H\boldsymbol{x}$ is an additive-group homomorphism from $\mathbb{Z}^n$ to $\mathbb{F}_q^k$, and that $\mathcal{L}^\perp(H)$ is its kernel by definition. So, by the first isomorphism theorem, the map induces an isomorphism from the quotient group $\mathbb{Z}^n/\mathcal{L}^\perp(H)$ to the image $\mathrm{Im}(H) = \{H\boldsymbol{x} : \boldsymbol{x} \in \mathbb{Z}^n\} \subseteq \mathbb{F}_q^k$, where the subset relation is an equality if and only if the rows of H are linearly independent. The claim then follows from the fact that $\det(\mathcal{L}^\perp(H)) = \left|\mathbb{Z}^n/\mathcal{L}^\perp(H)\right| = |\mathrm{Im}(H)|$. ◀

We next formally define the family of parity-check lattices that are at the heart of our construction of locally dense lattices.

▶ **Definition 5.** *For a prime q, positive integer k, and set $S \subseteq \mathbb{F}_q$, define $H_q(k, S) \in \mathbb{F}_q^{k\times S}$ to be the matrix H whose rows and columns are respectively indexed by $[k] = \{0, 1, \dots, k-1\}$ and S, and whose (i, s)th entry is*

$$H_{i,s} := s^i \in \mathbb{F}_q \ .$$

(Recall that $0^0 := 1$.) Equivalently, if we enumerate $S = \{s_0, \dots, s_{n-1}\}$ in some arbitrary order, we have

$$H = H_q(k, S) := \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ s_0 & s_1 & s_2 & \cdots & s_{n-1} \\ s_0^2 & s_1^2 & s_2^2 & \cdots & s_{n-1}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_0^{k-1} & s_1^{k-1} & s_2^{k-1} & \cdots & s_{n-1}^{k-1} \end{pmatrix} \in \mathbb{F}_q^{k\times n} \ . \tag{7}$$

Notice that H is a transposed Vandermonde matrix. In particular, if $k \le n$ then its rows are linearly independent, and so $\det(\mathcal{L}^\perp(H)) = q^k$ by Lemma 4.

We recall from the introduction that $H = H_q(k, S)$ is a generator matrix (of row vectors) of the dimension-k Reed-Solomon code over $\mathbb{F}_q$ with evaluation set S, and hence H is a parity-check matrix of its dimension-$(n-k)$ dual code. So, $\mathcal{L}^\perp(H)$ is the Construction A lattice of this dual code. These dual codes are in fact *generalized Reed-Solomon codes*, a family of codes that include Reed-Solomon codes as a special case and that are closed under taking duals (see [22, Theorem 5.1.6]). Moreover, in the special case where $S = \mathbb{F}_q$, the matrix $H = H_q(k, \mathbb{F}_q)$ is in fact a parity-check matrix of an (ordinary) Reed-Solomon code. For our hardness proof it suffices to use this special case; i.e., we show NP-hardness (under a randomized reduction) of GapSVP by using Construction A lattices of (ordinary) Reed-Solomon codes as gadgets. See Appendix A and the section on derandomization in the full version of this paper [6] for other connections between these lattices and Reed-Solomon codes.

2.3 Symmetric Polynomials

A *symmetric polynomial* $P(x_1, x_2, \dots, x_m)$ is a polynomial that is invariant under any permutation of its variables, i.e., $P(x_1, \dots, x_m) = P(x_{\pi(1)}, \dots, x_{\pi(m)})$ as formal polynomials for all permutations π of $\{1, 2, \dots, m\}$. Because the order of the variables is immaterial, we usually just write a symmetric polynomial as $P(X)$, where $X = \{x_1, \dots, x_m\}$ is the set of variables, and we write $P(T)$ for its evaluation on a multiset T of values.

We next recall two important symmetric polynomials and the relationship between them. For a non-negative integer i, the ith *power sum* of a set X of variables is defined as

$$p_i(X) := \sum_{x\in X} x^i \ . \tag{8}$$

(Recall that $0^0 := 1$.) For $1 \le i \le |X|$, the ith *elementary symmetric polynomial* of X is defined as

$$e_i(X) := \sum_{\substack{Z \subseteq X, \\ |Z|=i}} \prod_{z \in Z} z \ . \tag{9}$$

That is, $e_i(X)$ is the multilinear polynomial whose monomials consist of all products of i distinct variables from X. We extend this definition to $i = 0$ by setting $e_0(X) := 1$ and to integers $i > |X|$ by setting $e_i(X) := 0$.

Power sums and elementary symmetric polynomials are related by *Newton's identities* (see, e.g., [30]), which assert that for $1 \le i \le |X|$,

$$i \cdot e_i(X) = \sum_{j=1}^{i} (-1)^{j-1} \cdot e_{i-j}(X) \cdot p_j(X) \ . \tag{10}$$

The following standard claim uses Newton's identities to show that if the first k power sums of two *multisets* of field elements coincide, then so do the first k elementary symmetric polynomials of those multisets.

► **Lemma 6.** *Let T, U be multisets over a prime field $\mathbb{F}_q$, let $k \le q$ be a positive integer, and suppose that $p_i(T) = p_i(U)$ for all $i \in [k]$. Then $e_i(T) = e_i(U)$ for all $i \in [k]$.*

Proof. The proof is by (strong) induction. For the base case where $i = 0$, we have by definition that $e_0(T) = e_0(U) = 1$. For the inductive case where $1 \le i < k$, because $i \ne 0 \in \mathbb{F}_q$ we have that

$$e_i(T) = i^{-1} \cdot \sum_{j=1}^{i} (-1)^{j-1} \cdot e_{i-j}(T) \cdot p_j(T) = i^{-1} \cdot \sum_{j=1}^{i} (-1)^{j-1} \cdot e_{i-j}(U) \cdot p_j(U) = e_i(U) \ ,$$

where the first and third equalities follow from Newton's identities (Equation (10)), and the second equality follows from the claim's hypothesis and the inductive hypothesis (note that the sums involve elementary symmetric polynomials e_{i-j} only for $i - j < i$). ◄

We define the *root polynomial* $f_T(x) \in \mathbb{F}[x]$ of a multiset T over a field $\mathbb{F}$ to be

$$f_T(x) := \prod_{t \in T} (x - t) = \sum_{i=0}^{|T|} (-1)^i \cdot e_i(T) \cdot x^{|T|-i} \ . \tag{11}$$

We then get the following result, which uses Lemma 6 to show that if sufficiently many of the initial power sums of two multisets are equal, then the multisets themselves are equal.

► **Proposition 7.** *Let q be a prime, let $k \le q/2$ be a positive integer, let T, U be multisets over $\mathbb{F} = \mathbb{F}_q$ of total cardinality $|T| + |U| < 2k$, and suppose that $p_i(T) = p_i(U)$ for all $i \in [k]$. Then $T = U$.*

Proof. Because

$$|T| \equiv p_0(T) = p_0(U) \equiv |U| \pmod q$$

and $0 \le |T| + |U| < 2k \le q$, it follows that $|T| = |U|$ and hence both $f_T(x), f_U(x)$ have degree $|T| < k$.

Next, by the hypotheses and Lemma 6, we have that $e_i(T) = e_i(U)$ for all $i \le |T|$. Therefore, by the equality in Equation (11), $f_T(x)$ and $f_U(x)$ are identical as formal polynomials in $\mathbb{F}[x]$. Finally, because the polynomial ring $\mathbb{F}[x]$ is a unique factorization domain, and because $f_T(x)$ and $f_U(x)$ split over $\mathbb{F}$ by construction, it follows that $T = U$. ◄

2.4 Locally Dense Lattices

Roughly speaking, *locally dense lattices* are lattices that have one or more cosets with many relatively short vectors. Somewhat more precisely, a locally dense lattice consists of an integer lattice $\mathcal{L} \subset \mathbb{Z}^n$ and a shift $\boldsymbol{x} \in \mathbb{Z}^n$ such that for some $\alpha \in (0, 1)$, the number of points in the coset $\boldsymbol{x} + \mathcal{L}$ of norm at most $\alpha \cdot \lambda_1(\mathcal{L})$ is large (for our purposes, greater than 2^{n^ε} for some constant $\varepsilon > 0$). Therefore, locally dense lattices are not efficiently list decodable, even combinatorially, to within distance $\alpha \cdot \lambda_1(\mathcal{L})$ in the worst case (in particular, around center $-\boldsymbol{x}$). For the purposes of proving hardness, we also require a linear map T that projects the short vectors in $\boldsymbol{x} + \mathcal{L}$ onto a lower-dimensional hypercube $\{0, 1\}^r$.

▶ **Definition 8.** *For $p \in [1, \infty)$, real $\alpha > 0$, and positive integers r and R, a (p, α, r, R)*-locally dense lattice *consists of an integer lattice of rank R (and some dimension n) represented by a basis matrix $A \in \mathbb{Z}^{n \times R}$, a positive integer ℓ, a shift $\boldsymbol{x} \in \mathbb{Z}^n$, and a matrix $T \in \mathbb{Z}^{r \times n}$, where*

1. $\lambda_1^{(p)}(\mathcal{L}(A)) \geq \ell^{1/p}$ *and*
2. $\{0, 1\}^r \subseteq T(V) := \{T\boldsymbol{v} : \boldsymbol{v} \in V\}$*, where* $V := (\boldsymbol{x} + \mathcal{L}(A)) \cap \mathcal{B}_p^n(\alpha \cdot \ell^{1/p})$ *is the set of all vectors of ℓ_p norm at most $\alpha \cdot \ell^{1/p}$ in the lattice coset $\boldsymbol{x} + \mathcal{L}(A)$.*

A useful tool for satisfying Item 2 in the above definition is the following probabilistic version of Sauer's Lemma due to Micciancio [31]. It roughly says that for $n \gg r$, for any large enough collection of vectors $W \subseteq B_{n,h}$ (the weight-h slice of $\{0, 1\}^n$), and for a random matrix $T \in \{0, 1\}^{r \times n}$ whose coordinates are sampled independently with a suitable bias, $\{0, 1\}^r \subseteq T(W)$ with good probability. We emphasize that all the arithmetic in this theorem is done over the integers (not over $\mathbb{F}_2$).

▶ **Theorem 9** ([31, Theorem 4]). *Let r, n, h be positive integers, let $W \subseteq B_{n,h}$, and let $\varepsilon > 0$. If $|W| \geq h! \cdot n^{24r\sqrt{h}/\varepsilon}$ and $T \in \{0, 1\}^{r \times n}$ is sampled by setting each entry to 1 independently with probability $1/(4hr)$, then $\{0, 1\}^r \subseteq T(W)$ with probability at least $1 - \varepsilon$.*

2.5 Hardness of GapSVP via Locally Dense Lattices

We next recall a variant of (the decision version of) the Closest Vector Problem (CVP), which will be the hard problem that we reduce to GapSVP_p. In this variant, called GapCVP'_p, the target vector is either within a specified distance of a lattice vector given by a *binary* combination of basis vectors, or *all non-zero integer multiples* of the target vector are more than a γ multiple of this distance from the lattice (where distance is measured in the ℓ_p norm).

▶ **Definition 10.** *For $p \in [1, \infty]$, an instance of the γ-GapCVP'_p problem consists of a rank-r lattice basis $B \in \mathbb{Z}^{d \times r}$, a target vector $\boldsymbol{t} \in \mathbb{Z}^d$, and a distance threshold $s > 0$. The goal is to determine whether an input is a YES instance or a NO instance, where these are defined as follows:*

- *YES instance: there exists a* binary $\boldsymbol{c} \in \{0, 1\}^r$ *such that* $\|B\boldsymbol{c} - \boldsymbol{t}\|_p \leq s$.
- *NO instance:* $\mathrm{dist}_p(w\boldsymbol{t}, \mathcal{L}(B)) > \gamma s$ *for all* $w \in \mathbb{Z} \setminus \{0\}$.

The following hardness theorem follows via a reduction from Exact Set Cover to GapCVP'_p.

▶ **Theorem 11** ([4]). *For every $p \in [1, \infty)$ and every constant $\gamma \geq 1$, γ-GapCVP'_p is* NP*-hard.*

The following theorem gives a polynomial-time reduction from γ-GapCVP'_p to γ'-GapSVP_p for some approximation factors $\gamma > \gamma' \geq 1$, which uses a locally dense lattice as advice. In general, this advice makes the reduction non-uniform, but when the advice is efficiently computable by a (randomized) algorithm, as it is in this and prior works, the procedure is an efficient (randomized) reduction. The reduction below is very similar to the one in [32, Theorem 5.1], but written so as to allow for using an arbitrary locally dense lattice as advice. Due to this similarity, and for concision we defer its proof to the full version of our paper [6].

▶ **Theorem 12.** *Let $p \geq 1$, r and n be positive integers, $\alpha > 0$ be a constant, and γ, γ' be constants satisfying*

$$1/\alpha > \gamma' \geq 1 \text{ and } \gamma \geq \gamma' \cdot \left(\frac{1}{1-(\alpha\gamma')^p}\right)^{1/p} .$$

There is a deterministic polynomial-time algorithm that, given a γ-GapCVP'_p instance $(B, \boldsymbol{t}, s)$ of rank r and a (p, α, r, R)-locally dense lattice $(A, \ell, \boldsymbol{x}, T)$ as input, outputs a γ'-GapSVP_p instance (B', s') of rank $R+1$ which is a YES (respectively, NO) instance if $(B, \boldsymbol{t}, s)$ is a YES (resp., NO) instance.

From these two theorems we get the following hardness results for GapSVP.

▶ **Corollary 13.** *Let $p \geq 1$, let r be a positive integer, let $\alpha > 0$ be a constant, and suppose that there is an algorithm A that computes a $(p, \alpha, r, \mathrm{poly}(r))$-locally dense lattice in $\mathrm{poly}(r)$ time. Let γ be a constant satisfying $1 \leq \gamma < 1/\alpha$. Then:*

1. *If A is deterministic, then γ-GapSVP_p is* NP*-hard (and exact GapSVP_p is* NP*-complete).*
2. *If A is randomized and its output satisfies Item 1 of Definition 8 with probability 1 and Item 2 of Definition 8 with probability at least 2/3, then γ-GapSVP_p is not in* RP *unless* NP ⊆ RP.[6]
3. *If A is randomized, and its output satisfies Items 1 and 2 of Definition 8 with probability at least 2/3, then there is no* randomized *polynomial-time algorithm for γ-GapSVP_p unless* NP ⊆ BPP.

Proof. Items 1 and 3 follow immediately by combining Theorems 11 and 12. Inspection of the proof of Theorem 12 shows that for NO instances to be mapped to NO instances, only Item 1 of Definition 8 is needed, from which Item 2 of the claim follows. ◀

3 Local Density from Reed-Solomon Codes

In this section we show how to obtain locally dense lattices from Reed-Solomon codes with appropriate parameters. More specifically, we show to satisfy Definition 8 using a lattice $\mathcal{L} := \mathcal{L}^{\perp}(H)$ corresponding to a parity-check matrix $H = H_q(k, S)$ from Definition 5. (Recall that H is the parity-check matrix of a Reed-Solomon code when $S = \mathbb{F}_q$, and of a generalized Reed-Solomon code for any $S \subseteq \mathbb{F}_q$.)

The overall structure of the argument is as follows. First, in Section 3.1 we give a lower bound of $\lambda_1^{(p)}(\mathcal{L}) \geq (2k)^{1/p}$, which corresponds to Item 1 of Definition 8, by using the connection between power sums and symmetric polynomials (see Section 2.3). Then, in Section 3.2 we use the upper bound $\det(\mathcal{L}) \leq q^k$ from Lemma 4 and the pigeonhole principle to show that there exists a lattice coset with many short (binary) vectors, and in fact a suitably sampled random coset has this property with good probability. Finally, in Section 3.3 we set parameters and use Theorem 9 to satisfy Item 2 of Definition 8 with good probability.

[6] The condition "γ-GapSVP_p is not in RP" is a slight abuse of notation, since γ-GapSVP_p for $\gamma > 1$ is a promise problem rather than a language. However, the definition of RP can naturally be extended to encompass promise problems, which is the intended meaning here.

3.1 Minimum Distance

The following theorem says that for any $k \le |S|/2$, the ℓ_1 minimum distance of $\mathcal{L} = \mathcal{L}^{\perp}(H)$ for $H = H_q(k, S)$ is at least $2k$. Essentially the same result and proof appeared in works of Karabed, Roth, and Siegel [25, 37], and a very similar theorem and proof for lower bounding the minimum distance of Craig lattices appears in [14] and [13, Chapter 8, Theorem 7]. We reprove the result here in a slightly different form for completeness.

Note that a weaker bound of $\lambda_1^{(1)}(\mathcal{L}) \ge k + 1$ (for any $k < q$) follows trivially from the minimum Hamming distance $k + 1$ of the (generalized Reed-Solomon) code having parity-check matrix H. However, this bound is not strong enough for the rest of the local-density argument below, which requires $\lambda_1^{(1)}(\mathcal{L}) \ge (1 + \Omega(1))k$.

▶ **Theorem 14.** *Let q be a prime, let $S \subseteq \mathbb{F}_q$, let $k \le |S|/2$ be a positive integer, and let $H := H_q(k, S) \in \mathbb{F}_q^{k \times S}$ be the matrix from Definition 5. Then $\mathcal{L} = \mathcal{L}^{\perp}(H)$ has ℓ_1 minimum distance $\lambda_1^{(1)}(\mathcal{L}) \ge 2k$.*
As a consequence, for any $p \in [1, \infty)$ the ℓ_p minimum distance satisfies $\lambda_1^{(p)}(\mathcal{L}) \ge (2k)^{1/p}$.

We point out that the $2^{1/p}$ factor in Theorem 14 propagates to the relative-distance bound for local density in Theorem 17 below, and then to the GapSVP approximation factor in our main hardness theorem, Theorem 1.

Proof. The consequence follows immediately from the fact that $\mathcal{L} \subseteq \mathbb{Z}^S$ and $\|\boldsymbol{v}\|_p \ge \|\boldsymbol{v}\|_1^{1/p}$ for all $\boldsymbol{v} \in \mathbb{Z}^S$.

Now consider some arbitrary $\boldsymbol{x} \in \mathcal{L} \subseteq \mathbb{Z}^S$ for which $\|\boldsymbol{x}\|_1 < 2k$; we will show that $\boldsymbol{x} = \boldsymbol{0}$. Let $\boldsymbol{x}^+, \boldsymbol{x}^- \in \mathbb{Z}^S$ be the unique non-negative integer vectors satisfying $\boldsymbol{x} = \boldsymbol{x}^+ - \boldsymbol{x}^-$. Define multisets T^+ and T^- over S that respectively depend on $\boldsymbol{x}^+$ and $\boldsymbol{x}^-$ as follows. For each $s \in S$ with $x_s^+ > 0$ (respectively, $x_s^- > 0$), let T^+ (respectively, T^-) contain s with multiplicity x_s^+ (respectively, x_s^-).[7]

Note that $|T^+| + |T^-| = \|\boldsymbol{x}\|_1 < 2k$. Because $H\boldsymbol{x} = H(\boldsymbol{x}^+ - \boldsymbol{x}^-) = \boldsymbol{0} \in \mathbb{F}_q^k$, by definition of H we have that $p_i(T^+) = p_i(T^-)$ for all $i \in [k]$ (where recall that p_i denotes the ith power sum). Because $k \le |S|/2 \le q/2$, by Proposition 7 it follows that $T^+ = T^-$. Since $T^+ \cap T^- = \emptyset$ by construction, we must have $T^+ = T^- = \emptyset$, and hence $\boldsymbol{x} = \boldsymbol{0}$, as desired. ◀

The following lemma (which is well known in other forms) shows that the lower bound $\lambda_1^{(p)}(\mathcal{L}^{\perp}(H)) \ge (2k)^{1/p}$ from Theorem 14 is in fact an equality under mild conditions on the parameters, by giving an explicit lattice coset that has multiple short vectors. However, because it proves only that the number of such vectors is polynomial in the dimension, it is insufficient to establish local density.

▶ **Lemma 15.** *Let q be a prime, let k be a positive integer that divides $q - 1$, and let $H := H_q(k, S) \in \mathbb{F}_q^{k \times S}$ where $\mathbb{F}_q^* \subseteq S \subseteq \mathbb{F}_q$. Then for $\boldsymbol{u} := (k, 0, \ldots, 0) \in \mathbb{F}_q^k$, the lattice coset $\mathcal{L}_{\boldsymbol{u}}^{\perp}(H) = \{\boldsymbol{x} \in \mathbb{Z}^S : H\boldsymbol{x} = \boldsymbol{u}\}$ contains $(q-1)/k$ binary vectors of Hamming weight k and pairwise disjoint support. As a consequence, when $k < q - 1$, we have $\lambda_1^{(p)}(\mathcal{L}^{\perp}(H)) = (2k)^{1/p}$ for any $p \in [1, \infty)$.*

Proof. Let G be the order-k subgroup of the (cyclic, multiplicative) group $\mathbb{F}_q^*$, i.e., the subgroup of the kth roots of unity. Then the binary indicator vectors $\boldsymbol{x}_C \in \{0, 1\}^S$ of each of the $(q-1)/k$ pairwise disjoint cosets $C = cG$ of G all belong to the coset $\mathcal{L}_{\boldsymbol{u}}^{\perp}(H)$. This

[7] For example, if $S = \{0, 1, 2, 3, 4\} = \mathbb{F}_q$ and $\boldsymbol{x} = (x_0, x_1, x_2, x_3, x_4)^t = (1, -2, 0, 1, 0)^t$, then $x^+ = (1, 0, 0, 1, 0)$, $x^- = (0, 2, 0, 0, 0)$, and accordingly $T^+ = \{0, 3\}$, $T^- = \{1, 1\}$.

is simply because for any such coset, the 0th power sum of its elements is k, and the ith power sum for $0 < i < k$ is zero; this can be seen by Newton's identities and the fact that the root polynomial of C is $f_C(x) = \prod_{c \in C}(x - c) = x^k - r_C$, where $r_C = c^k$ for every $c \in C$. Finally, when $k < q - 1$, there is more than one such vector $\boldsymbol{x}_C$, and the differences between distinct pairs of them are lattice vectors in $\{0, \pm 1\}^S$ of Hamming weight $2k$, and hence ℓ_p norm $(2k)^{1/p}$. ◀

3.2 Dense Cosets

Following an approach previously used in [31, 26, 32] (and implicitly in [3]), we first show via a pigeonhole argument that a dense lattice coset must exist, and then show how to sample such a coset efficiently (with good probability).

For a prime q, a positive integer k, and a set $S \subseteq \mathbb{F}_q$ of size n (with some arbitrary ordering of its elements), let $H = H_q(k, S) \in \mathbb{F}_q^{k \times n}$ be the parity-check matrix from Definition 5. By Lemma 4, the lattice $\mathcal{L} = \mathcal{L}^{\perp}(H) \subseteq \mathbb{Z}^n$ has $\det(\mathcal{L}) \leq q^k$ integer cosets. Recall that $B_{n,h}$ is the set of n-dimensional binary vectors of Hamming weight h, which has cardinality $|B_{n,h}| = \binom{n}{h}$. Therefore, by the pigeonhole principle, there must exist some integer coset $\boldsymbol{x} + \mathcal{L}$ with $|(\boldsymbol{x} + \mathcal{L}) \cap B_{n,h}| \geq \binom{n}{h}/q^k$ weight-h binary vectors. In particular, taking $n \approx q$, $h \approx \alpha^p \cdot (2k)$ for some constant $\alpha > 1/2^{1/p}$, and $k = q^{\varepsilon}$ for a suitable small constant $\varepsilon > 0$ implies the existence of a coset with roughly $q^{(2\alpha^p - 1)k} = q^{\Omega(q^{\varepsilon})}$ such vectors. These vectors have ℓ_p norm $h^{1/p} \approx \alpha \cdot (2k)^{1/p}$, whereas by Theorem 14 the lattice minimum distance is at least $(2k)^{1/p}$, yielding a local-density relative distance of roughly α.

The following lemma extends the above existential result by showing that something very similar holds for a *uniformly random* shift $\boldsymbol{x} \in B_{n,h}$: for any $\delta > 0$, the coset $\boldsymbol{x} + \mathcal{L}$ contains at least $\delta \cdot \binom{n}{h}/q^k$ weight-h binary vectors with probability greater than $1 - \delta$. The proof given below closely follows the structure of the very similar one of [26, Lemma 4.3].

▶ **Lemma 16.** *For a prime q, positive integer k, and set $S \subseteq \mathbb{F}_q$ of size n, let $H = H_q(k, S) \in \mathbb{F}_q^{k \times n}$ be the parity-check matrix from Definition 5. There is an efficient randomized algorithm that, for any $\delta \geq 0$, and on input H and any $h \in [n]$, outputs a shift $\boldsymbol{x} \in B_{n,h}$ such that*

$$\Pr_{\boldsymbol{x}}\left[|(\boldsymbol{x} + \mathcal{L}) \cap B_{n,h}| \geq \delta \cdot \binom{n}{h}/q^k\right] > 1 - \delta \ .$$

Proof. The algorithm simply samples and outputs a uniformly random binary vector $\boldsymbol{x} \in B_{n,h}$. This is clearly efficient. To show correctness, we will use the syndromes of H. For each $\boldsymbol{u} \in \mathbb{F}_q^k$, define $K_{\boldsymbol{u}} := |\{\boldsymbol{z} \in B_{n,h} : H\boldsymbol{z} = \boldsymbol{u}\}|$, and define $\boldsymbol{s} := H\boldsymbol{x} \in \mathbb{F}_q^k$ to be the syndrome corresponding to $\boldsymbol{x}$. So, we need to prove that $K_{\boldsymbol{s}} \geq \delta \cdot \binom{n}{h}/q^k$ with probability greater than $1 - \delta$. Indeed, we have:

$$\begin{aligned}
\Pr_{\boldsymbol{x}}\left[|(\boldsymbol{x} + \mathcal{L}) \cap B_{n,h}| < \delta \cdot \binom{n}{h}/q^k\right] &= \Pr_{\boldsymbol{x}}\left[K_{\boldsymbol{s}} < \delta \cdot \binom{n}{h}/q^k\right] \\
&= \sum_{\boldsymbol{u} \in \mathbb{F}_q^k : K_{\boldsymbol{u}} < \delta \cdot \binom{n}{h}/q^k} \Pr_{\boldsymbol{x}}[H\boldsymbol{x} = \boldsymbol{u}] \\
&= \sum_{\boldsymbol{u} \in \mathbb{F}_q^k : K_{\boldsymbol{u}} < \delta \cdot \binom{n}{h}/q^k} \frac{K_{\boldsymbol{u}}}{\binom{n}{h}} \\
&< \sum_{\boldsymbol{u} \in \mathbb{F}_q^k : K_{\boldsymbol{u}} < \delta \cdot \binom{n}{h}/q^k} \frac{\delta}{q^k} \\
&\leq \delta \ ,
\end{aligned}$$

where the first inequality uses the fact that the sum is over syndromes $\boldsymbol{u}$ with $K_{\boldsymbol{u}} < \delta \cdot \binom{n}{h}/q^k$, and the second inequality uses the fact that there are at most q^k terms in the sum. ◀

3.3 The Main Argument

▶ **Theorem 17** (Locally dense lattices from Reed-Solomon codes). *For any $p \in [1, \infty)$ and constant $\alpha > 1/2^{1/p}$, there exists a randomized polynomial-time algorithm that, given any sufficiently large positive integer r in unary as input, outputs a $(p, \alpha, r, R = \mathrm{poly}(r))$-locally dense lattice (Definition 8) with probability at least $2/3$. Moreover, the algorithm's output satisfies Item 1 of Definition 8 with probability 1.*

Proof. The algorithm starts by setting its parameters as follows. It sets $\varepsilon := 2\alpha^p - 1 > 0$, and chooses:

- a poly(r)-bounded integer $k \geq r^{1/(1/2-\delta)}$ for some arbitrary constant $\delta \in (0, 1/2)$, and
- a poly(r)-bounded prime $q \geq k^{3(1+\varepsilon)/\varepsilon}$. (Such a prime q always exists by Bertrand's Postulate.)

The algorithm then computes the components of a $(p, \alpha, r, R = q)$-locally dense lattice $(A, \ell, \boldsymbol{x}, T)$ as follows. It lets:

- $A \in \mathbb{Z}^{q \times q}$ be a basis of $\mathcal{L} := \mathcal{L}^\perp(H)$, where $H = H(k, S) \in \mathbb{F}_q^{k \times q}$ for $S = \mathbb{F}_q$;[8]
- $\ell := 2k$;
- $\boldsymbol{x} \in B_{q,h}$ be a uniformly random q-dimensional binary vector of Hamming weight $h := \lfloor (1+\varepsilon)k \rfloor$;
- $T \in \{0,1\}^{r \times q}$ be chosen by independently setting each of its entries to be 1 with probability $1/(4hr)$, and to be 0 otherwise.

It then outputs $(A, \boldsymbol{x}, \ell, T)$.

We first analyze the algorithm's running time. A suitable prime q can be found in poly(r) time using, e.g., trial division (recall that r is given in unary). The basis A can be computed in deterministic polynomial time from the generating set of column vectors $(B \mid qI_q)$, where B is a basis of $\ker(H) \subseteq \mathbb{F}_q^q$ (lifted to the integers). It is clear that ℓ can be computed in deterministic polynomial time, and that $\boldsymbol{x}$ and T can be computed in randomized polynomial time. So, the algorithm runs in randomized polynomial time.

It remains to show correctness, i.e., that $(A, \boldsymbol{x}, \ell, T)$ satisfies the two conditions in Definition 8 with suitable probability over the random choices of $\boldsymbol{x}$ and T. First, Item 1 is always satisfied, because by Theorem 14 we have

$$\lambda_1(\mathcal{L}) \geq (2k)^{1/p} = \ell^{1/p} \ .$$

In the rest of the proof we consider Item 2 of Definition 8. Let $W := (\boldsymbol{x} + \mathcal{L}) \cap B_{q,h}$. Because

$$\|\boldsymbol{w}\|_p^p = h \leq (1+\varepsilon)k = \alpha^p \cdot \ell$$

for each $\boldsymbol{w} \in W$, we have $W \subseteq V := (\boldsymbol{x} + \mathcal{L}) \cap \mathcal{B}_p^q(\alpha \cdot \ell^{1/p})$.

By Lemma 16, $\Pr_{\boldsymbol{x}}[|W| \geq \binom{q}{h}/(10q^k)] > 1 - 1/10 = 9/10$. If this event holds, and

$$\frac{\binom{q}{h}}{10q^k} \geq h! \cdot q^{240r\sqrt{h}} \ , \tag{12}$$

[8] For appropriate parameters, our argument works more generally for any sufficiently large subset $S \subseteq \mathbb{F}_q$, with $R = |S|$; we use $S = \mathbb{F}_q$ for simplicity.

then by Theorem 9 we have $\{0,1\}^r \subseteq T(W) \subseteq T(V)$ with probability at least $1-1/10 = 9/10$ (over the choice of T). So, it suffices to show that the condition in Equation (12) holds for all sufficiently large k, and hence for all sufficiently large r. By taking a union bound over the $1/10$ failure probabilities from Lemma 16 and Theorem 9, we get that the algorithm's overall success probability is at least $1 - 2/10 > 2/3$ for all sufficiently large r, as needed.

Using the standard bound $\binom{q}{h} \geq (q/h)^h$ for binomial coefficients and that $h \geq (1+\varepsilon)k - 1$, we have that

$$\frac{\binom{q}{h}}{10q^k} \geq \frac{q^{h-k}}{10h^h} = \Omega\Big(\frac{q^{\varepsilon k - 1}}{h^h}\Big) \ . \tag{13}$$

Furthermore, by the choice of k relative to r and $h \leq (1+\varepsilon)k$, we have that

$$h! \cdot q^{240r\sqrt{h}} \leq h^h \cdot q^{240k^{1/2-\delta}\sqrt{(1+\varepsilon)k}} \leq h^h \cdot q^{o(k)} \ . \tag{14}$$

So, by combining Equations (13) and (14), in order to establish Equation (12) it suffices to show that $q^{(1-o(1))\varepsilon k} \geq h^{2h}$. By taking logs, this is equivalent to

$$(1 - o(1)) \cdot \varepsilon k \log q \geq 2h \log h \ . \tag{15}$$

Finally, using that $k^{3(1+\varepsilon)/\varepsilon} \leq q \leq \mathrm{poly}(k)$ and $h \leq (1+\varepsilon)k$, in order for Equation (15) to hold it suffices to have

$$(1 - o(1)) \cdot \varepsilon k \cdot \frac{3(1+\varepsilon)}{\varepsilon} \cdot \log k = (3 - o(1)) \cdot (1+\varepsilon) \cdot k \log k \geq 2(1+\varepsilon) \cdot k \log k + O(k) \ ,$$

which indeed holds for all sufficiently large k, as needed. ◀

We emphasize that Theorem 17 uses randomness only to sample $\boldsymbol{x}$ and T. As an immediate corollary, we obtain our main hardness result, Theorem 1 – which, to recall, asserts that for all constants $p \in [1, \infty)$ and $\gamma < 2^{1/p}$, there is no polynomial-time algorithm for γ-GapSVP$_p$ unless $\mathsf{NP} \subseteq \mathsf{RP}$.

Proof of Theorem 1. Combine Item 2 of Corollary 13 with Theorem 17. ◀

References

1 Divesh Aggarwal and Noah Stephens-Davidowitz. (Gap/S)ETH hardness of SVP. In *STOC*, 2018.

2 Dorit Aharonov and Oded Regev. Lattice problems in NP cap coNP. *J. ACM*, 52(5):749–765, 2005. Preliminary version in FOCS 2004.

3 Miklós Ajtai. The shortest vector problem in L_2 is NP-hard for randomized reductions (extended abstract). In *STOC*, pages 10–19, 1998.

4 Sanjeev Arora, László Babai, Jacques Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *J. Comput. Syst. Sci.*, 54(2):317–331, 1997. Preliminary version in FOCS 1993.

5 Huck Bennett, Mahdi Cheraghchi, Venkatesan Guruswami, and João Ribeiro. Parameterized inapproximability of the Minimum Distance Problem over all fields and the Shortest Vector Problem in all ℓ_p norms. In *STOC*, 2023.

6 Huck Bennett and Chris Peikert. Hardness of the (approximate) shortest vector problem: A simple proof via reed-solomon codes, 2023. `arXiv:2202.07736`.

7 Huck Bennett, Chris Peikert, and Yi Tang. Improved hardness of BDD and SVP under Gap-(S)ETH. In *ITCS*, 2022.

8 Elwyn Berlekamp. *Negacyclic Codes for the Lee Metric*, chapter 9, pages 207–217. World Scientific, 2015. Preliminary version in Symposium on Combinatorial Mathematics and its Applications, 1967. URL: `https://www.worldscientific.com/doi/abs/10.1142/9789814635905_0009`.

9 Arnab Bhattacharyya, Édouard Bonnet, László Egri, Suprovat Ghoshal, Karthik C.S., Bingkai Lin, Pasin Manurangsi, and Dániel Marx. Parameterized intractability of even set and shortest vector problem. *J. ACM*, 68(3), 2021. `doi:10.1145/3444942`.

10 Jin-yi Cai and Ajay Nerurkar. Approximating the SVP to within a factor $(1 + 1/\dim^\epsilon)$ is NP-hard under randomized reductions. In *CCC*, 1998.

11 Qi Cheng and Daqing Wan. On the list and bounded distance decodibility of the Reed-Solomon codes (extended abstract). In *FOCS*, 2004.

12 Qi Cheng and Daqing Wan. A deterministic reduction for the gap minimum distance problem. *IEEE Trans. Inf. Theory*, 58(11):6935–6941, 2012. Preliminary version in STOC 2009.

13 John Conway and Neil J. A. Sloane. *Sphere packings, lattices, and groups*. Springer, 1999.

14 Maurice Craig. Automorphisms of prime cyclotomic lattices. Preprint.

15 Léo Ducas and Cécile Pierrot. Polynomial time bounded distance decoding near Minkowski's bound in discrete logarithm lattices. *Des. Codes Cryptogr.*, 87(8):1737–1748, 2019.

16 Léo Ducas and Wessel P. J. van Woerden. On the lattice isomorphism problem, quadratic forms, remarkable lattices, and cryptography. In *EUROCRYPT*, 2022.

17 Andreas Emil Feldmann, Karthik C. S., Euiwoong Lee, and Pasin Manurangsi. A survey on approximation in parameterized complexity: Hardness and algorithms. *Algorithms*, 13(6), 2020. `doi:10.3390/a13060146`.

18 Oded Goldreich and Shafi Goldwasser. On the limits of nonapproximability of lattice problems. *J. Comput. Syst. Sci.*, 60(3):540–563, 2000. Preliminary version in STOC 1998.

19 Elena Grigorescu and Chris Peikert. List-decoding Barnes-Wall lattices. *Comput. Complex.*, 26(2):365–392, 2017. Preliminary version in CCC 2012.

20 Venkatesan Guruswami and Atri Rudra. Limits to list decoding Reed-Solomon codes. *IEEE Trans. Inf. Theory*, 52(8):3642–3649, 2006. Preliminary version in STOC 2005.

21 Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Trans. Inf. Theory*, 45(6):1757–1767, 1999. Preliminary version in FOCS 1998.

22 Jonathan I. Hall. Notes on coding theory. Available at `https://users.math.msu.edu/users/halljo/classes/CODENOTES/CODING-NOTES.HTML`.

23 Ishay Haviv and Oded Regev. Tensor-based hardness of the shortest vector problem to within almost polynomial factors. *Theory Comput.*, 8(1):513–531, 2012. Preliminary version in STOC 2007.

24 K. Immink and G. Beenker. Binary transmission codes with higher order spectral zeros at zero frequency (corresp.). *IEEE Transactions on Information Theory*, 33(3):452–454, 1987.

25 R. Karabed and P.H. Siegel. Matched spectral-null codes for partial-response channels. *IEEE Transactions on Information Theory*, 37(3):818–855, 1991.

26 Subhash Khot. Hardness of approximating the shortest vector problem in lattices. *J. ACM*, 52(5):789–808, 2005. Preliminary version in FOCS 2004.

27 Subhash Khot. Hardness of approximating the shortest vector problem in high ℓ_p norms. *J. Comput. Syst. Sci.*, 72(2):206–219, 2006. Preliminary version in FOCS 2003.

28 Ralf Koetter and Alexander Vardy. Algebraic soft-decision decoding of Reed-Solomon codes. *IEEE Trans. Inf. Theory*, 49(11):2809–2825, 2003.

29 Swastik Kopparty. Personal communication, 2020.

30 D. G. Mead. Newton's identities. *The American Mathematical Monthly*, 99(8):749, October 1992. `doi:10.2307/2324242`.

31 Daniele Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. *SIAM J. Comput.*, 30(6):2008–2035, 2000. Preliminary version in FOCS 1998.

32 Daniele Micciancio. Inapproximability of the shortest vector problem: Toward a deterministic reduction. *Theory Comput.*, 8(1):487–512, 2012.

33 Daniele Micciancio and Antonio Nicolosi. Efficient bounded distance decoders for Barnes-Wall lattices. In *ISIT*, pages 2484–2488. IEEE, 2008.

34 Ethan Mook and Chris Peikert. Lattice (list) decoding near Minkowski's inequality. *IEEE Trans. Inf. Theory*, 68(2):863–870, 2022.

35 Chris Peikert. Limits on the hardness of lattice problems in ℓ_p norms. *Computational Complexity*, 17(2):300–351, May 2008. Preliminary version in CCC 2007.

36 Chris Peikert. A decade of lattice cryptography. *Found. Trends Theor. Comput. Sci.*, 10(4):283–424, 2016.

37 Ron M. Roth and Paul H. Siegel. Lee-metric BCH codes and their application to constrained and partial-response channels. *IEEE Trans. Inf. Theory*, 40(4):1083–1096, 1994. `doi:10.1109/18.335966`.

38 Peter van Emde Boas. Another NP-complete partition problem and the complexity of computing short vectors in a lattice. Technical Report, 1981. Available at `https://staff.fnwi.uva.nl/p.vanemdeboas/vectors/mi8104c.html`. URL: `https://staff.fnwi.uva.nl/p.vanemdeboas/vectors/mi8104c.html`.

A Efficient Decoding Near Minkowski's Bound

In this appendix, we show that a recent result of Mook and Peikert [34], which builds on work of Guruswami and Sudan [21] and Koetter and Vardy [28] on list-decoding Reed-Solomon codes, yields a polynomial-time algorithm for decoding lattices $\mathcal{L} = \mathcal{L}^{\perp}(H)$ with $H = H_q(k, \mathbb{F}_q)$ up to distance $\Theta(\sqrt{k})$. We additionally observe that by choosing $k = \Theta(q/\log q)$, such lattices are asymptotically nearly tight with Minkowski's bound (Equation (5)). Putting these observations together, we obtain an efficient algorithm for decoding to a distance within a $O(\sqrt{\log q})$ factor of Minkowski's bound (here $q = n$ is the lattice rank and dimension).

A.1 Construction and Algorithm

Define the additive quotient group $\mathbb{R}_q := \mathbb{R}/(q\mathbb{Z})$ and the Euclidean norm of any $\hat{\boldsymbol{y}} \in \mathbb{R}_q^n$ as

$$\|\hat{\boldsymbol{y}}\| := \min\{\|\boldsymbol{y}\| : \boldsymbol{y} \in \hat{\boldsymbol{y}} + q\mathbb{Z}^n\} \ . \tag{16}$$

Equivalently, $\|\hat{\boldsymbol{y}}\|$ is the standard $\mathbb{R}^n$ Euclidean norm of the unique real vector $\boldsymbol{y} \equiv \hat{\boldsymbol{y}} \pmod{q\mathbb{Z}^n}$ having coordinates in $[-q/2, q/2)$. In additive arithmetic that mixes elements of $\mathbb{F}_q$ and $\mathbb{R}_q$, we implicitly "lift" the former to the latter in the natural way.

We again use the fact that for evaluation set $S = \mathbb{F}_q$, the matrix $H = H_q(k, \mathbb{F}_q)$ defined in Equation (7) is a parity-check matrix of the Reed-Solomon code $\mathrm{RS}_q[q-k, \mathbb{F}_q]$, and therefore $\mathcal{L}^{\perp}(H_q(k, \mathbb{F}_q)) = \mathrm{RS}_q[q-k, \mathbb{F}_q] + q\mathbb{Z}^q$. This view lets us take advantage of the decoding algorithm from the following theorem of [34], which gives an efficient (list) decoder in the ℓ_2 norm for Reed-Solomon codes.[9]

[9] In fact, the cited result from [34] is more general, giving a decoder for $\mathbb{F}_p$-*subfield subcodes* of Reed-Solomon codes over finite fields of order $q = p^r$, for a prime p. Here we need only the special case where the Reed-Solomon code is over a prime field (i.e., where $r = 1$). On the other hand, we note that if Proposition 18 were extended to handle *generalized* Reed-Solomon codes, then we would get a corresponding strengthening of Corollary 19 for decoding lattices $\mathcal{L}^{\perp}(H_q(k, S))$ with general S, not just $S = \mathbb{F}_q$.

▶ **Proposition 18** ([34, Algorithm 1 and Theorem 3.4]). *Let q be a prime, $S \subseteq \mathbb{F}_q$ be an evaluation set of size $n := |S|$, $k \leq n$ be a nonnegative integer, and $\varepsilon > 0$. There is a deterministic algorithm that, on input q, S, k, ε, and a vector $\hat{\boldsymbol{y}} \in \mathbb{R}_q^n$, outputs all codewords $\boldsymbol{c} \in \mathrm{RS}_q[n-k, S]$ such that $\|\hat{\boldsymbol{y}} - \boldsymbol{c}\|^2 \leq (1-\varepsilon)(k+1)/2$, in time polynomial in n, $\log q$, and $1/\varepsilon$.*[10]

The following corollary, which is the main result of this section, says that by taking $S = \mathbb{F}_q$ and $k = \Theta(q/\log q)$, (1) the root Hermite factor of $\mathcal{L}^\perp(H)$ is within an $O(\sqrt{\log q})$ factor of Minkowski's bound (Equation (5)), and (2) it is possible to efficiently decode this lattice to a distance of $\Omega(\sqrt{k}) = \Omega(\sqrt{q/\log q})$, which is again within an $O(\sqrt{\log q})$ factor of Minkowski's bound.

We remark that by setting $\varepsilon \leq 1/(k+1)$ in Corollary 19, we get efficient decoding to a distance at least $\sqrt{k/2}$ but less than $\sqrt{(k+1)/2}$, which is slightly more than half the lower bound of $\sqrt{2k}$ on the minimum Euclidean distance of the lattice (Theorem 14). Recall that this lower bound is tight when k is a proper divisor of $q-1$ (see Lemma 15), so with this parameterization we get efficient *list* decoding (i.e., the algorithm may return more than one lattice vector) slightly beyond the unique-decoding bound of half the minimum distance.

▶ **Corollary 19** (Efficient decoding near Minkowski's bound). *Let $H = H_q(k, \mathbb{F}_q)$ for a prime q and $k := \lfloor q/(2\log q) \rfloor \leq q/2$, where all logarithms are base two. Then for $\mathcal{L} := \mathcal{L}^\perp(H) \subseteq \mathbb{Z}^q$:*

1. $\sqrt{q/\log q - 2} \leq \sqrt{2k} \leq \lambda_1(\mathcal{L}) \leq \sqrt{q} \cdot \det(\mathcal{L})^{1/q} \leq \sqrt{2q}$.
2. *For any $\varepsilon > 1/\mathrm{poly}(q)$, there is an algorithm that, on input q and a vector $\boldsymbol{y} \in \mathbb{R}^q$, outputs all lattice vectors $\boldsymbol{v} \in \mathcal{L}$ satisfying $\|\boldsymbol{y} - \boldsymbol{v}\| \leq \sqrt{(1-\varepsilon)(k+1)/2}$ in time polynomial in q.*

Proof. For Item 1, we have

$$\sqrt{q/\log q - 2} \leq \sqrt{2k} \leq \lambda_1(\mathcal{L}) \leq \sqrt{q} \cdot \det(\mathcal{L})^{1/q} = \sqrt{q} \cdot q^{k/q} \leq \sqrt{2q} \ .$$

The first inequality follows from the choice of k, the second inequality is by Theorem 14, the third inequality is Minkowski's bound (Equation (5)), the equality follows from Lemma 4 (recall that the rows of H are linearly independent), and the final inequality again follows from the choice of k.[11]

The algorithm claimed in Item 2 works as follows. First, it computes k and $\hat{\boldsymbol{y}} = \boldsymbol{y} \bmod q\mathbb{Z}^q \in \mathbb{R}_q^q$ from the input q and $\boldsymbol{y}$. It then calls the algorithm from Proposition 18 on q, $S = \mathbb{F}_q$, k, ε, and $\hat{\boldsymbol{y}}$, and receives as output zero or more codewords $\boldsymbol{c} \in \mathrm{RS}_q[q-k, \mathbb{F}_q]$. For each such $\boldsymbol{c}$, it outputs the unique vector $\boldsymbol{v} := \arg\min_{\boldsymbol{v}' \in \boldsymbol{c} + q\mathbb{Z}^q} \|\boldsymbol{y} - \boldsymbol{v}'\| \in \mathcal{L}$.

The value k and vectors $\hat{\boldsymbol{y}}, \boldsymbol{v}$ can be computed efficiently (assuming that $\boldsymbol{v}$ is well defined), so it is clear from Proposition 18 that this algorithm runs in time polynomial in q (recall that the dimension $n = q$). It remains to show correctness. First, it is immediate from the definitions that for any $r < q/2$, the function $f(\boldsymbol{v}) = \boldsymbol{v} \bmod q\mathbb{Z}^q$ is a bijection from the set of lattice vectors

$$\{\boldsymbol{v} \in \mathcal{L} : \|\boldsymbol{y} - \boldsymbol{v}\| \leq r\} \ ,$$

[10] Formally, the runtimes of the decoding algorithms in Proposition 18 and Corollary 19 additionally depend on the lengths of the respective "received words" $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$ that they take as input, which must be specified to finite precision. However, for simplicity we describe the algorithms in the "Real RAM model," while noting that their runtime dependence on the encoding lengths of $\hat{\boldsymbol{y}}, \boldsymbol{y}$ is polynomial.

[11] Analyzing the derivative of $\log(\sqrt{2k}/q^{k/q})$ with respect to k shows that our choice of k is asymptotically optimal for maximizing the root Hermite factor of $\mathcal{L}^\perp(H_q(k, \mathbb{F}_q))$.

to the set of codewords

$$\{\boldsymbol{c} \in \mathrm{RS}_q[q-k, \mathbb{F}_q] : \|\hat{\boldsymbol{y}} - \boldsymbol{c}\| \leq r\} ,$$

and that $g(\boldsymbol{c}) := \arg\min_{\boldsymbol{v}' \in \boldsymbol{c} + q\mathbb{Z}^q} \|\boldsymbol{y} - \boldsymbol{v}'\|$ is the inverse function of f, i.e., $g = f^{-1}$. Moreover, because $q \geq 2$, we have that the decoding distance r satisfies

$$r := \sqrt{(1-\varepsilon)(k+1)/2} \leq \sqrt{(1-\varepsilon)(q/(2\log q)+1)/2} \leq \sqrt{1-\varepsilon} \cdot q/2 < q/2 .$$

Because the algorithm from Proposition 18 outputs (exactly) $\{\boldsymbol{c} \in \mathrm{RS}_q[q-k, \mathbb{F}_q] : \|\hat{\boldsymbol{y}} - \boldsymbol{c}\| \leq r\}$, it follows that the algorithm described above outputs (exactly) $\{\boldsymbol{v} \in \mathcal{L} : \|\boldsymbol{y} - \boldsymbol{v}\| \leq r\}$, as needed. ◀

▶ Remark 20. We remark that the main consequence of Item 1 of Corollary 19 – namely, an explicit construction of a family of lattices having root Hermite factors within a $O(\sqrt{\log n})$ factor of Minkowski's bound, obtained via Construction A (where n is the lattice dimension) – only needs a family of codes satisfying milder conditions than what (generalized) Reed-Solomon codes satisfy. Namely, achieving this result only requires a family of linear q-ary codes $\mathcal{C}$ for prime q with block length n, codimension $k = \Theta(n/\log n)$, and minimum distance (in the Hamming metric) $d = \Omega(k)$. The latter is a weaker condition than *maximum distance separability* (MDS), which requires that $d = k + 1$. Indeed, $d = \Omega(k)$ implies that the corresponding Construction-A lattice $\mathcal{C} + q\mathbb{Z}^n$ has an ℓ_2 minimum distance of $\Omega(\min\{\sqrt{k}, q\})$, which is $\Omega(\sqrt{k})$ when $k = O(q^2)$. So, unlike our main hardness result, Corollary 19 does not use Theorem 14 in any essential way.

Finally, we also note that obtaining a direct analog of Item 2 of Corollary 19 – i.e., efficiently decoding to within an $O(\sqrt{\log n})$ factor of Minkowski's bound on $\mathcal{C} + q\mathbb{Z}^n$ – additionally requires an efficient algorithm for decoding $\mathcal{C}$ to an ℓ_2 distance of $\Omega(\sqrt{k})$, but that this is in turn a weaker requirement than what Proposition 18 fulfills.

Perfect Sampling for Hard Spheres from Strong Spatial Mixing

Konrad Anand
Queen Mary, University of London, UK

Andreas Göbel
Hasso Plattner Institute, University of Potsdam, Germany

Marcus Pappik
Hasso Plattner Institute, University of Potsdam, Germany

Will Perkins
School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA

Abstract

We provide a perfect sampling algorithm for the hard-sphere model on subsets of $\mathbb{R}^d$ with expected running time linear in the volume under the assumption of strong spatial mixing. A large number of perfect and approximate sampling algorithms have been devised to sample from the hard-sphere model, and our perfect sampling algorithm is efficient for a range of parameters for which only efficient approximate samplers were previously known and is faster than these known approximate approaches. Our methods also extend to the more general setting of Gibbs point processes interacting via finite-range, repulsive potentials.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures

Keywords and phrases perfect sampling, hard-sphere model, Gibbs point processes

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.38

Category RANDOM

Related Version *Full Version*: `https://arxiv.org/abs/2305.02450` [3]

Funding *Konrad Anand*: funded by a studentship from Queen Mary, University of London.
Andreas Göbel: funded by the project PAGES (project No. 467516565) of the German Research Foundation (DFG).
Marcus Pappik: funded by the HPI Research School on Data Science and Engineering.
Will Perkins: supported in part by NSF grant DMS-2309958.

Acknowledgements We thank Mark Jerrum for very helpful discussions on this topic.

1 Introduction

Gibbs point processes, or classical gases, are mathematical models of interacting particles. In statistical physics they are used to model gases, fluids, and crystals, while in other fields they are used to model spatial phenomena such as the growth of trees in a forest, the distribution of stars in the universe, or the location of cities on a map (see e.g. [68, 59, 73, 12]).

Perhaps the longest and most intensively studied Gibbs point process is the hard-sphere model: a model of a gas in which the only interaction between particles is a hard-core exclusion in a given radius around each particle. That is, it is a model of a random packing of equal-sized spheres. Despite the simplicity of its definition, the hard-sphere model is expected to exhibit the qualitative behavior of a real gas [2], and in particular exhibits gas, liquid, and solid phases, thus giving evidence for the hypothesis, dating back to at

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 38; pp. 38:1–38:18
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

least Boltzmann, that the macroscopic properties of a gas or fluid are determined by its microscopic interactions. This rich behavior exhibited by the hard-sphere model is very difficult to analyze rigorously, and the most fundamental questions about phase transitions in this model are open mathematical problems [68, 50].

In studying the hard-sphere model (or Gibbs point processes more generally), a fundamental task is to sample from the model. Sampling is used to estimate statistics, observe evidence of phase transitions, and perform statistical tests on data. A wide variety of methods have been proposed to sample from these distributions; for instance, the Markov chain Monte Carlo (MCMC) method was first proposed by Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller [53] to sample from the two-dimensional hard-sphere model. Understanding sampling methods for point processes in theory and in practice is a major area of study [58, 59, 16, 39, 47], and advances in sampling techniques have led to advances in the understanding of the physics of these models [53, 2, 50, 7, 6, 16].

In this paper we will be concerned with provably efficient sampling from the hard-sphere model. Rigorous guarantees for sampling algorithms come in several different varieties. One question is what notion of "efficient" to use; another is what guarantee we insist on for the output. In this paper we will provide an efficient sampling algorithm under the strictest possible terms with respect to both running time and accuracy of the output: a linear-time, *perfect* sampling algorithm.

For simplicity we focus on sampling from the hard-sphere model defined on finite boxes in $\mathbb{R}^d$. For fixed parameter values of the model, the typical number of points appearing in such a region is linear in the volume, and so any sampling algorithm will require at least this much time.

As for guarantees on the output, there are two main types of guarantees. The first type is an *approximate sampler*: the output of such an algorithm must be distributed within ε total variation distance of the desired target distribution. Perhaps the main approach to efficient sampling from distributions normalized by intractable normalizing constants is the MCMC method. In this approach, one devises a Markov chain with the target distribution as the stationary distribution and runs a given number steps of the chain from a chosen starting configuration; if the number of steps is at least the ε-mixing time, then the final state has distribution within ε total variation distance of the target [42, 65, 13]. In general, however, computing or bounding the mixing time can be a very challenging problem.

The second type of guarantee is that of a *perfect sampler* [63]. Such an algorithm has a running time that is random, but the distribution of the output is guaranteed to be *exactly* that of the target distribution. The main advantage of perfect sampling algorithms – and the primary reason they are studied and used in practice – is that one need not prove a theorem or understand the mixing time of a Markov chain to run the algorithm and get an accurate sample; one can simply run the algorithm and know that the output has the correct distribution. The drawback is that the running time may be very large, depending on the specific algorithm and on the parameter regime. Some naive sampling methods such as rejection sampling return perfect samples but are inefficient on large instances (exponential expected running time in the volume). The breakthrough of Propp and Wilson in introducing "coupling from the past" [63, 64] was to devise a procedure for using a Markov chain transition matrix to design perfect sampling algorithms which, under some conditions, could run in time polylogarithmic in the size of a discrete state space (polynomial-time in the size of the graph of a spin system), matching the efficiency of fast mixing Markov chains which only return approximate samples (see also [5, 49] for precedents in perfect sampling). The work of Propp and Wilson led to numerous constructions of perfect sampling algorithms for problems with both discrete and continuous state spaces including [17, 27, 45, 60, 28, 21, 46, 58, 23]. Notably, many of the first applications of Propp and Wilson's technique were in designing perfect sampling algorithms for Gibbs point processes (though often without rigorous guarantees on the efficiency of the algorithms).

Perfect sampling continues to be a very active area of research today, with a special focus on improving the range of parameters for which perfect sampling algorithms can (provably) run in expected linear or polynomial time [9, 40, 30]

In this paper we design a perfect sampling algorithm for the hard-sphere model (and Gibbs point processes interacting with a finite-range, repulsive pair potential more generally) that is guaranteed to run in linear expected time for activity parameters up to the best known bound for efficient approximate sampling via MCMC.

What is this bound and how do we design the algorithm? One central theme in the analysis of discrete spin systems is the relationship between spatial mixing (correlation decay properties) and temporal mixing (mixing times of Markov chains) [35, 1, 72, 52, 15]. At a high level, these works show that for discrete lattice systems a strong correlation decay property (*strong spatial mixing*) implies a near-optimal convergence rate for local-update Markov chains like the Glauber dynamics. Recently it has been showed that strong spatial mixing in a discrete lattice model also implies the existence of efficient *perfect* sampling algorithms [18, 4]. In parallel, there has been work establishing the connection between strong spatial mixing and optimal temporal mixing for Markov chains in the setting of the hard-sphere model and Gibbs point processes [33, 55, 56]. At a high level, our aim is to combine these threads to show that strong spatial mixing for Gibbs point processes implies the existence of an efficient perfect sampler. One challenge is that the approaches of [18, 4] are inherently discrete in that key steps of the algorithms involve enumerating over all possible configurations in a subregion, something that is not possible in the continuum. To overcome this we make essential use of Bernoulli factories – a method for perfect simulation of a coin flip with a bias $f(p)$ given access to coin flips of bias p. Bernoulli factories have recently been used in perfect sampling algorithms for solutions to constraint satisfaction problems in [31, 32].

1.1 The hard-sphere model, strong spatial mixing, and perfect sampling

The hard-sphere model is defined on a bounded, measurable subset Λ of $\mathbb{R}^d$ with an activity parameter $\lambda \geq 0$ that governs the density of the model and a parameter $r > 0$ that governs the range of interaction (though by re-scaling there is really only one meaningful parameter, and we could take $r = 1$ without loss of generality). In words, the hard-sphere model is the distribution of finite point sets in Λ obtained by taking a Poisson point process of activity λ on Λ and conditioning on the event that all pairs of points are at distance at least r from each other; in other words, on the event that spheres of radius $r/2$ centered at the given points form a sphere packing.

We can equivalently define the model more explicitly, and in doing so, introduce objects and notation we work with throughout the paper. To begin, let $\mathcal{N}_\Lambda$ be the set of all finite point sets in Λ. Each point set $\eta \in \mathcal{N}_\Lambda$ represents a particle configuration in Λ. Write $\mathfrak{R}_\Lambda \subseteq 2^{\mathcal{N}_\Lambda}$ for the σ-field generated by the maps $\left\{\mathcal{N}_\Lambda \to \mathbb{N}_0, \eta \mapsto |\eta \cap B| \;\middle|\; B \subseteq \Lambda \text{ Borel-measurable}\right\}$. The hard-sphere model (or in fact any Gibbs point process) is a probability measure μ_λ on the space $(\mathcal{N}_\Lambda, \mathfrak{R}_\Lambda)$.

Define for every $x_1, \ldots, x_k \in \mathbb{R}^d$ the indicator that the points are centers of non-overlapping spheres of radius $r/2$; that is,

$$D(x_1,\dots,x_k) = \prod_{\{i,j\}\in\binom{[k]}{2}} \mathbb{1}_{\mathrm{dist}(x_i,x_j)\geq r} \,.$$

Then define the partition function

$$Z_\Lambda(\lambda) = \sum_{k\geq 0} \frac{\lambda^k}{k!} \int_{\Lambda^k} D(x_1,\dots,x_k)\,\mathrm{d}x_1 \dots \mathrm{d}x_k \,.$$

For an event $A \in \mathfrak{R}_\Lambda$, the hard-sphere model assigns the probability

$$\mu_\lambda(A) = \frac{1}{Z_\Lambda(\lambda)} \sum_{k\geq 0} \frac{\lambda^k}{k!} \int_{\Lambda^k} \mathbb{1}_{\{x_1,\dots,x_k\}\in A} D(x_1,\dots,x_k)\,\mathrm{d}x_1 \dots \mathrm{d}x_k \,. \tag{1}$$

A very useful generalization of this model is to allow for a non-constant (but measurable) activity function $\boldsymbol{\lambda} : \Lambda \to [0,\infty)$. Here the model is a Poisson process with inhomogenous activity $\boldsymbol{\lambda}$ conditioned on the points forming the centers of a sphere packing; the partition function is now

$$Z_\Lambda(\boldsymbol{\lambda}) = \sum_{k\geq 0} \frac{1}{k!} \int_{\Lambda^k} \prod_{i=1}^{k} \boldsymbol{\lambda}(x_i) D(x_1,\dots,x_k)\,\mathrm{d}x_1 \dots \mathrm{d}x_k$$

and the measure $\mu_{\boldsymbol{\lambda}}$ is defined analogously to (1). This generalization allows modeling of non-homogenous spaces and generalizes the concept of imposing boundary conditions on the model. To see this, suppose we fix a particle configuration $\eta \in \mathcal{N}_\Lambda$ as boundary conditions. Additional points are forbidden within the balls of radius r around each point $x \in \eta$; we can implement the distribution of additional points by considering the measure $\mu_{\boldsymbol{\lambda}}$ with $\boldsymbol{\lambda}(y) = 0$ if $\mathrm{dist}(y, x) < r$ for some $x \in \eta$; and $\boldsymbol{\lambda}(y) = \lambda$ otherwise. We denote the resulting activity function by $\boldsymbol{\lambda}$ by λ_η. Further, we can use this generalization to restrict a point process to only place points in a subregion $\Lambda' \subseteq \Lambda$ by considering the measure $\mu_{\lambda \mathbb{1}_{\Lambda'}}$ with activity function $\lambda \mathbb{1}_{\Lambda'} : x \mapsto \lambda \mathbb{1}_{x\in\Lambda'}$. Of course the generalization to measurable activity functions is much more general than this, and activity functions $\boldsymbol{\lambda}$ need not be realizable by boundary conditions or restriction to a subregion, nor take only two values.

This generalization to activity functions is crucial for defining *strong spatial mixing*, the condition under which we can guarantee the efficiency of our perfect sampling algorithm.

To define the concept of strong spatial mixing we consider projections of the measure $\mu_{\boldsymbol{\lambda}}$ to subregions $\Lambda' \subseteq \Lambda$. We write $\mu_{\boldsymbol{\lambda}}[\Lambda']$ for the probability measure on $(\mathcal{N}_{\Lambda'}, \mathfrak{R}_{\Lambda'})$ induced by $\mu_{\boldsymbol{\lambda}}$ (we make this definition formal in Section 3). We can impose two distinct boundary conditions on Λ' by choosing two different activity functions $\boldsymbol{\lambda}, \boldsymbol{\lambda}'$. Strong spatial mixing asserts that the distributions $\mu_{\boldsymbol{\lambda}}[\Lambda']$, $\mu_{\boldsymbol{\lambda}'}[\Lambda']$ are close in total variation when $\boldsymbol{\lambda}, \boldsymbol{\lambda}'$ differ only on points far from Λ'; i.e., when $\mathrm{dist}(\Lambda', \mathrm{supp}(\boldsymbol{\lambda} - \boldsymbol{\lambda}'))$ is large (as $\mathrm{supp}(\boldsymbol{\lambda} - \boldsymbol{\lambda}')$ is the set of points at which the two activity functions disagree).

Writing $|\Lambda'|$ for the volume of Λ', strong spatial mixing with exponential decay is defined as follows.

▶ **Definition 1.1.** *Given $a, b \in \mathbb{R}_{>0}$, the hard-sphere model on $\mathbb{R}^d$ exhibits (a, b)-**strong spatial mixing** up to $\lambda \in \mathbb{R}_{>0}$ if for all bounded measurable $\Lambda \subset \mathbb{R}^d$ the following holds: For all measurable $\Lambda' \subseteq \Lambda$ and all activity functions $\boldsymbol{\lambda}, \boldsymbol{\lambda}' \leq \lambda$ it holds that*

$$d_{TV}(\mu_{\boldsymbol{\lambda}}[\Lambda'], \mu_{\boldsymbol{\lambda}'}[\Lambda']) \leq a|\Lambda'|\mathrm{e}^{-b\cdot\mathrm{dist}\left(\Lambda',\mathrm{supp}\left(\boldsymbol{\lambda}-\boldsymbol{\lambda}'\right)\right)},$$

where $d_{TV}(\cdot,\cdot)$ denotes total variation distance.

This definition of strong spatial mixing comes from [56], which in turn adapted similar notions from discrete spin systems [15, 74]. Strong spatial mixing has proved to be an essential definition in the analysis, both probabilistic and algorithmic, of spin systems on graphs, and many recent works are focused on either proving strong spatial mixing for a particular model, range of parameters, and class of graphs (e.g. [74, 22, 51, 69, 66, 10]) or deriving consequences of strong spatial mixing (e.g. [70, 19, 48, 18, 4]).

Our main result is a linear expected-time perfect sampling algorithm for the hard-sphere model under the assumption of strong spatial mixing.

▶ **Theorem 1.2.** *There is a perfect sampling algorithm for the hard-sphere model on finite boxes $\Lambda \subset \mathbb{R}^d$ with the property that if the hard-sphere model exhibits (a, b)-strong spatial mixing up to λ, then the expected running time of the algorithm at activity λ is $O(|\Lambda|)$, where the implied constant is a function of a, b, and λ.*

In particular, one can run the algorithm for any value of λ (without knowing whether or not strong spatial mixing holds) and the algorithm will terminate in finite time with an output distributed exactly as μ_λ; under the assumption of strong spatial mixing the expected running time is guaranteed to be linear in the volume.

Using bounds from [56] on strong spatial mixing in the hard-sphere model, we obtain the following explicit bounds on the activities for which the algorithm is efficient.

▶ **Corollary 1.3.** *The above perfect sampling algorithm runs in expected time $O(|\Lambda|)$ when $\lambda < \frac{\mathrm{e}}{v_d(r)}$, where $v_d(r)$ is the volume of the ball of radius r in $\mathbb{R}^d$.*

In comparison, near-linear time MCMC-based approximate samplers were given in [56] for the same range of parameters (following results for more restricted ranges in [43, 33]). For perfect sampling from the hard-sphere model, linear expected time algorithms were given in [36, 25] for more restrictive ranges of parameters.

1.2 Gibbs point processes with finite-range repulsive potentials

We now give a closely related result in the more general setting of Gibbs point processes interacting via finite-range, repulsive pair potentials.

Gibbs point processes are defined via a density against an underlying Poisson point process. In general, this density is the exponential of (the negative of) an energy function on point sets that captures the interactions between points. In many of the most studied cases, this energy function takes a special form: it is the sum of potentials over pairs of points in a configuration.

A *pair potential* is a measurable symmetric function $\phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R} \cup \{\infty\}$. For a bounded, measurable activity function $\boldsymbol{\lambda}$ on Λ the Gibbs point process with pair potential ϕ on Λ is defined via the partition function

$$Z_\Lambda(\boldsymbol{\lambda}) = \sum_{k \geq 0} \frac{1}{k!} \int_{\Lambda^k} \left(\prod_{i \in [k]} \boldsymbol{\lambda}(x_i) \right) \mathrm{e}^{-H(x_1, \ldots, x_k)} \, \mathrm{d}x_1 \ldots \mathrm{d}x_k$$

where

$$H(x_1, \ldots, x_k) = \sum_{\{i,j\} \in \binom{[k]}{2}} \phi(x_i, x_j) \, .$$

Again the corresponding probability measure $\mu_{\boldsymbol{\lambda}}$ is obtained as in (1). A pair potential ϕ is *repulsive* if $\phi(x, y) \geq 0$ for all x, y. It is of *finite-range* if there exists $r \geq 0$ so that $\phi(x, y) = 0$ whenever $\mathrm{dist}(x, y) > r$. As with the hard-sphere model, we can use the activity function to encode the influence of boundary conditions by defining the activity function $\lambda_\eta : y \mapsto \lambda \mathrm{e}^{-\sum_{x\in\eta}\phi(x,y)}$ for any activity $\lambda \in [0,\infty)$ and particle configuration $\eta \in \mathcal{N}_\Lambda$. Moreover, strong spatial mixing for a Gibbs point process is defined exactly as in Definition 1.1.

The hard-sphere model is one example of a model interacting via a finite-range, repulsive pair potential; it is obtained by letting $\phi(x,y)$ take the value $+\infty$ if $\mathrm{dist}(x,y) \le r$ and 0 otherwise. The Strauss process [71, 44] is another such example.

Our next result is a near-linear expected time perfect sampling algorithm for Gibbs point processes interacting via finite-range, repulsive potentials under the assumption of strong spatial mixing.

▶ **Theorem 1.4.** *Suppose ϕ is a finite-range, repulsive potential on $\mathbb{R}^d$ and suppose ϕ exhibits (a,b)-strong spatial mixing up to λ for some constants $a, b > 0$. Then there is a perfect sampling algorithm for the Gibbs point process defined by ϕ and activity functions bounded by λ on boxes Λ in $\mathbb{R}^d$ with expected running time $O\Big(|\Lambda| \log^{O(1)}|\Lambda|\Big)$.*

One difference between this algorithm and the hard-sphere algorithm of Theorem 1.2 is that this algorithm needs knowledge of the constants a, b in the assumption of strong spatial mixing, whereas the hard-sphere algorithm does not.

Using the results of [56], we can get explicit bounds for the existence of efficient perfect sampling algorithms in terms of the *temperedness constant* of the potential defined by

$$C_\phi := \sup_{x\in\mathbb{R}^d} \int_{\mathbb{R}^d} |1 - e^{-\phi(x,y)}| \, dy \,. \tag{2}$$

Under the assumption that ϕ is repulsive and of finite range r, we have $0 \le C_\phi \le v_d(r)$.

▶ **Corollary 1.5.** *The above perfect sampling algorithm runs in expected time $O\Big(|\Lambda| \log^{O(1)}|\Lambda|\Big)$ when $\lambda < \frac{\mathrm{e}}{C_\phi}$.*

▶ Remark 1.6. In fact, using the results of Michelen and Perkins [54], one can push the bound for strong spatial mixing up to e/Δ_ϕ, where $\Delta_\phi \le C_\phi$ is the *potential-weighted connective constant* defined therein; our perfect sampling algorithm is efficient up to that point.

1.3 Related work and future directions

Related work

In recent years there has been a moderate flurry of activity around proving rigorous results for Gibbs point processes in both the setting of statistical physics and probability theory and in the setting of provably efficient sampling algorithms.

Work on provably efficient approximate sampling methods for the hard-sphere model begins with the seminal paper of Kannan, Mahoney, and Montenegro [43], who used techniques from the analysis of discrete spin systems to prove mixing time bounds for Markov chains for the hard-sphere model. Improvements to the range of parameters for which fast mixing holds came in [29, 33], before Michelen and Perkins proved the bound $\mathrm{e}/v_d(r)$ in [56], which we match with a perfect sampling algorithm in Corollary 1.3.

Perfect sampling algorithms for the hard sphere model have been considered in [27, 46, 21, 25, 38]. In terms of rigorous guarantees of efficiency, Huber proved a bound of $2/v_d(r)$ for a near-linear expected time perfect sampler in [36]. The perfect sampling algorithm of Guo and Jerrum in [25] does not match this bound, but the algorithm, based on "partial rejection sampling" [26] is novel and particularly simple. Several of these approaches also apply for finite-range, repulsive potentials or can be extended to that setting (e.g. [57]).

In parallel, there has been much work on proving bounds on the range of activities for which no phase transition can occur in the hard-sphere model; and, in recent years in particular, the techniques used have close connections to algorithms and the study of Markov chains. The classic approach to proving absence of phase transition is by proving convergence of the cluster expansion; the original bound here is $1/(\mathrm{e}v_d(r))$ due to Groeneveld [24]. In small dimensions (most significantly in dimension 2) improvements to the radius of convergence can be obtained [20]. On the other hand, this approach is inherently limited by the presence of non-physical singularities on the negative real axis. Alternative approaches avoiding this obstruction include using the equivalence of spatial and temporal mixing [33, 56]; or disagreement percolation [11, 34, 8]. The best current bound for absence of phase transition for the hard-sphere model and for repulsive pair potentials is the bound of e/C_ϕ (and e/Δ_ϕ) obtained by Michelen and Perkins [55, 56, 54]. Theorem 1.4 brings the bound for efficient perfect sampling up to this bound.

On a technical level, the most relevant past work is [18], in which the authors prove that for discrete spin systems, strong spatial mixing and subexponential volume growth of a sequence of graphs imply the existence of an efficient perfect sampling algorithm. We take their approach as a starting point but need new ideas to replace their exhaustive enumeration of configurations.

A key step in our algorithm is the use of a Bernoulli factory to implement a Bayes filter. Bernoulli factories are algorithms by which a Bernoulli random variable with success probability $f(p)$ can be simulated (perfectly) by an algorithm with access to independent Bernoulli p random variables, where the algorithm does not know the value p. Whether a Bernoulli factory exists (and how efficient it can be) depends on the function $f(\cdot)$ and a priori bounds on the possible values p. Bernoulli factories have been studied in [61, 37, 14] and recently used in the design of perfect sampling algorithms for CSP solutions in [31, 32].

Future directions

There are a number of extensions and improvements to these results one could pursue. Perhaps most straightforward would be to relax the notion of strong spatial mixing from exponential decay to decay faster than the volume growth of $\mathbb{R}^d$ and to extend the results to repulsive potentials of unbounded range but finite temperedness constant C_ϕ. Moreover, it would be nice to upgrade the guarantees of the algorithm in Theorem 1.4 to that of Theorem 1.2: that the algorithm does not need prior knowledge of the strong spatial mixing constants a, b to run correctly.

An ambitious and exciting direction would be to remove the assumption of a repulsive potential and find efficient perfect sampling algorithms for the class of *stable* potentials (see e.g. [62, 67, 68] for a definition). A stable potential is repulsive at short ranges but can include a weak attractive part; such potentials include the physically realistic Lenard-Jones potential among others [75]. This would require some very new ideas, as much of the recent probabilistic and algorithmic work on Gibbs point processes (e.g. [55, 56, 8, 54]) has used repulsiveness as an essential ingredient (for one, repulsiveness of the potential implies stochastic domination by the underlying Poisson point process). As a notable exception, a deterministic approximation algorithm for partition functions of finite-range stable potentials based on cluster expansion was recently proposed in [41].

1.4 Outline of the paper

In Section 2, we describe the high-level idea and intuition behind the algorithm. In Section 3 we introduce some notation and present some preliminary results that we will use throughout the paper. In Section 4 we present the algorithm that we will apply to both hard spheres and more general processes, and we state the main theorems and lemmas that we use for proving Theorem 1.2. The more general setting of bounded-range repulsive potentials (i.e., Theorem 1.4) can be found in the full version of the paper [3]. Intermediate steps and proves are omitted and can be found in the full version as well.

2 Intuitive idea behind the algorithm

Our algorithm is an adaptation to continuum models of the work by Feng, Guo, and Yin [18] on perfect sampling from discrete spin systems. We mimic their setting of a spin system on a graph $G = (V, E)$ by putting a graphical structure on sub-regions of our continuous space.

Let $\Lambda = [0, L)^d \subset \mathbb{R}^d$ be the region considered, $\lambda > 0$ the activity, and let ϕ be a repulsive potential of range $r > 0$. We subdivide Λ into $(\Lambda_{\boldsymbol{v}})_{\boldsymbol{v}\in\mathcal{V}}$, a set of smaller boxes of side length r indexed by vertices of a graph: each box corresponds to a vertex and boxes are connected if they are within r of each other, i.e., particles in the boxes can interact directly through the potential ϕ. We fix the index set for the boxes to be $\mathcal{V} \subset \mathbb{N}_0^d$, where each $\boldsymbol{v} \in \mathcal{V}$ corresponds to the box $\Lambda_{\boldsymbol{v}} = [v_1 r, (v_1 + 1)r) \times \cdots \times [v_d r, (v_d + 1)r)$. We extend this notation to sets of indices $S \subseteq \mathcal{V}$ by setting $\Lambda_S = \bigcup_{\boldsymbol{v}\in S} \Lambda_{\boldsymbol{v}}$. Further, we denote by $\mathbb{B}_k(\boldsymbol{v})$ the set of indices $\boldsymbol{w} \in \mathcal{V}$ with $\|\boldsymbol{v} - \boldsymbol{w}\|_\infty \le k$. To shorten notation, we write $\partial S = \left(\bigcup_{\boldsymbol{v}\in S} \mathbb{B}_1(\boldsymbol{v})\right) \setminus S$ for the outer boundary of a set of boxes indexed by $S \subseteq \mathcal{V}$.

Our algorithm runs iteratively, keeping track of two random variables: a point configuration $X_t \in \mathcal{N}_\Lambda$ with $X_0 = \emptyset$, and a set of "incorrect" boxes $\mathcal{U}_t \subseteq \mathcal{V}$ with $\mathcal{U}_0 = \mathcal{V}$. With each iteration t we maintain the following *invariant*: the partial configuration $X_t \cap (\Lambda_{\mathcal{U}_t})^{\mathrm{c}}$ is distributed according to the projection of μ_λ to $(\Lambda_{\mathcal{U}_t})^{\mathrm{c}}$ under the boundary condition $X_t \cap \Lambda_{\mathcal{U}_t}$. It follows that X_t is distributed according to μ_λ once we reach the state $\mathcal{U}_t = \emptyset$.

We proceed by sketching an iteration of the algorithm. An example for the involved subregions is given in Figure 1. Each iteration runs as follows:

1. We choose $\boldsymbol{u}_t \in \mathcal{U}_t$ uniformly at random and attempt to "repair" it by updating X_t on a neighborhood of boxes $B = \{\boldsymbol{u}_t\} \cup (\mathbb{B}_\ell(\boldsymbol{u}_t) \setminus \mathcal{U}_t)$ for some *update radius* $\ell \in \mathbb{N}$.
2. We sample a Bayes filter F_t (i.e., a Bernoulli random variable) with probability depending on the potential ϕ, the activity λ, and the current point configuration X_t on $\Lambda_{\boldsymbol{u}_t}$ and $\Lambda_{\partial B}$.
3. a) If $F_t = 1$, we set $\mathcal{U}_{t+1} = \mathcal{U}_t \setminus \{\boldsymbol{u}_t\}$ and we get X_{t+1} by updating X_t on Λ_B according to a projection of μ_λ conditioned on the boundary configuration $X_t \cap (\Lambda_B)^{\mathrm{c}}$.
   b) If $F_t = 0$, the configuration is unchanged and we add the boundary boxes to our "incorrect" list, i.e., $X_{t+1} = X_t$ and $\mathcal{U}_{t+1} = \mathcal{U}_t \cup \partial B$.

We use the Bayes filter, as in [18], to remove bias from the resulting distribution. To give some intuition for its role, suppose we run a naive version of the algorithm where we always update X_t on Λ_B as in step 3.a) above. Assuming the desired invariant holds after t iterations, this naive algorithm gives a bias to the distribution of X_{t+1} proportional to $\frac{Z_{\Lambda_{B\setminus\{\boldsymbol{u}_t\}}}\left(\lambda_{X_t\cap\Lambda_{\partial B\cup\{\boldsymbol{u}_t\}}}\right)}{Z_{\Lambda_B}\left(\lambda_{X_t\cap\Lambda_{\partial B}}\right)}$. We choose the Bayes filter such that, conditioned on $F_t = 1$, the bias term gets canceled. This suggests the choice

$$\mathbb{P}[F_t = 1 \mid X_t, \mathcal{U}_t, \boldsymbol{u}_t] = C(\mathcal{U}_t, \boldsymbol{u}_t, X_t) \cdot \frac{Z_{\Lambda_B}\left(\lambda_{X_t\cap\Lambda_{\partial B}}\right)}{Z_{\Lambda_{B\setminus\{\boldsymbol{u}_t\}}}\left(\lambda_{X_t\cap\Lambda_{\partial B\cup\{\boldsymbol{u}_t\}}}\right)}, \tag{3}$$

where the choice $C(\mathcal{U}_t, \boldsymbol{u}_t, X_t)$ serves three main purposes.

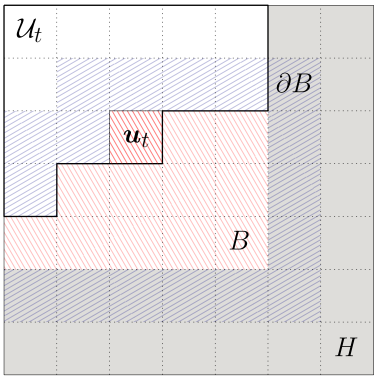


Figure 1 The box-shaped region $\Lambda \subset \mathbb{R}^2$ is divided into boxes of side length r (dotted lines). The boxes $\mathcal{U}_t$ are bordered by bold black lines. For $\boldsymbol{u}_t$ as given and update radius $\ell = 2$, the corresponding set B of boxes to be updated is indicated by the red hatched area (falling left to right). Its boundary boxes ∂B are shown as blue hatched area (rising left to right). The boxes in $H = (\mathcal{U}_t \cup B)^{\mathrm{c}}$ are shown with gray background.

First, it must guarantee that the right-hand side of (3) is a probability. To achieve this we need, for $H = (\mathcal{U}_t \cup B)^{\mathrm{c}}$ and almost all realizations of X_t, $\mathcal{U}_t$ and $\boldsymbol{u}_t$, that

$$C(\mathcal{U}_t, \boldsymbol{u}_t, X_t) \le \inf_{\xi \in \mathcal{N}_{\Lambda_H}} \frac{Z_{\Lambda_{B\setminus\{\boldsymbol{u}_t\}}}\left(\lambda_{\xi \cup (X_t \cap \Lambda_{\mathcal{U}_t})}\right)}{Z_{\Lambda_B}\left(\lambda_{\xi \cup (X_t \cap \Lambda_{\mathcal{U}_t \setminus \{\boldsymbol{u}_t\}})}\right)}. \tag{4}$$

Second, $C(\mathcal{U}_t, \boldsymbol{u}_t, X_t)$ must introduce no new bias. Carrying out the calculations, it can be seen that this is guaranteed if $C(\mathcal{U}_t, \boldsymbol{u}_t, X_t)$ only depends on $X_t \cap \Lambda_{\mathcal{U}_t}$. Finally, it must ensure that the algorithm terminates almost surely. It suffices to ensure $C(\mathcal{U}_t, \boldsymbol{u}_t, X_t)$ is uniformly bounded away from 0 for almost all realizations of X_t, implying that the same holds for the right-hand side of (3). We refer to a function $C(\cdot)$ satisfying these requirements as a *Bayes filter correction*.

If we use a Bayes filter as given in (3), keeping X_t and $\mathcal{U}_t$ unchanged whenever $F_t = 0$ introduces new bias. To prevent this, we set $\mathcal{U}_{t+1} = \mathcal{U}_t \cup \partial B$ in step 3.b), effectively deleting the part of the configuration that was revealed by the filter. Since the algorithm only terminates once $\mathcal{U}_t = \emptyset$, we further require the Bayes filter correction to ensure that the probability of $F_t = 0$ is small to guarantee efficiency.

Constructing a Bayes filter correction that satisfies the requirements above and allows for efficient sampling of F_t is a non-trivial task. In the next subsections, we present two approaches for this, the first specialized to the hard-sphere model without requirements, and the second one for more general potentials with strong spatial mixing of the point process. Crucially, assuming strong spatial mixing, both constructions allow us to control the success probability of the Bayes filter via the update radius ℓ in the construction of the updated set of boxes B (see step 1).

2.1 Bayes filter for the hard-sphere model

To construct a Bayes filter for the hard-sphere model, we efficiently approximate the right-hand side of (4). To approximate the infimum over the uncountable set of configurations $\xi \in \mathcal{N}_{\Lambda_H}$ we take the minimum over a finite, but sufficiently rich set of configurations, balancing the quality of approximation with the computation required. In fact the number of configurations needed will depend only on the volume of $\Lambda_{B \cup \partial B}$. We approximate the fraction of partition functions in (4) with running time only depending on the volume of $\Lambda_{B \cup \partial B}$. As a result, we efficiently compute a Bayes filter correction $C_\varepsilon(\cdot)$, with the parameter $\varepsilon > 0$ controlling how much $C_\varepsilon(\mathcal{U}_t, \boldsymbol{u}_t, X_t)$ deviates from the right-hand side of (4).

While our construction of $C_\varepsilon(\cdot)$ guarantees correctness of the sampling algorithm for any $\varepsilon > 0$, proving efficiency requires more. With strong spatial mixing, we choose ε so that the probability that $F_t = 0$ is uniformly bounded above, ensuring $\mathrm{O}(|\Lambda|)$ iterations in expectation.

It remains to argue that we can efficiently sample F_t, using the Bayes filter correction $C_\varepsilon(\cdot)$. Explicitly computing the success probability of F_t as in (3) would require computing the fraction of partition functions on the right-hand side exactly, while approximating these partition functions would require that the approximation error only depends on $X_t \cap \Lambda_{\mathcal{U}_t}$, to avoid new bias.

It is unclear how to implement these approaches, so instead we use Bernoulli factories to sample F_t without knowing the success probability. To do so, we observe that the fraction of partition functions can be written as a ratio of probabilities for drawing the empty set from a conditional hard-sphere model on Λ_B and $\Lambda_{B \setminus \{\boldsymbol{u}_t\}}$. Since both regions have constant volume, rejection sampling obtains Bernoulli random variables with these success probabilities in constant time. Hence, we obtain a Bernoulli factory for F_t with constant expected running time. Wald's identity yields a total expected running time $\mathrm{O}(|\Lambda|)$ for the algorithm.

2.2 Bayes filter for general potentials

We now consider the case of general bounded-range, repulsive potentials. Unlike the hard sphere model, it is not clear here how to approximate the infimum in (4) from a finite set of boundary configurations. However, given constants $a, b > 0$ such that ϕ satisfies (a, b)-strong spatial mixing, we can explicitly compute a function $\delta(a, b)$ so that

$$C_{a,b}(\mathcal{U}_t, \boldsymbol{u}_t, X_t) = \delta(a, b) \cdot \frac{Z_{\Lambda_{B \setminus \{\boldsymbol{u}_t\}}}\left(\lambda_{X_t \cap \Lambda_{\mathcal{U}_t}}\right)}{Z_{\Lambda_B}\left(\lambda_{X_t \cap \Lambda_{\mathcal{U}_t \setminus \{\boldsymbol{u}_t\}}}\right)}$$

is a Bayes filter correction. With strong spatial mixing, we use $C_{a,b}(\cdot)$ to construct a Bayes filter such that probability that $F_t = 0$ is bounded above, again implying a bound of $\mathrm{O}(|\Lambda|)$ on the expected number of iterations of the algorithm.

Note that in this setting, we require spatial mixing for both correctness and efficiency, while for the hard-sphere model we only need it for efficiency. Another crucial difference is that, while we can explicitly compute $\delta(a, b)$, the same does not hold for $C_{a,b}(\cdot)$ due to the fraction of partition functions involved. Again we circumvent this by rewriting the success probability of the Bayes filter in a suitable way and applying a Bernoulli factory for sampling F_t. Finally, we point out that we do not obtain a constant bound for the expected running time of each iteration, but instead the bound depends on the number of points in $X_t \cap \Lambda_{\partial B}$. Possible dependencies between the configuration X_t and the number of iterations prevent us from bounding the total expected running time using Wald's identity. Instead, we provide tail bounds on the number of iterations and the running time of each iteration, allowing us to derive an expected total running time that is linear in the volume of Λ up to polylogarithmic factors.

3 Preliminaries

Throughout the paper, we write $\mathbb{N}$ for the set of strictly positive integers, and we write $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. For any $k \in \mathbb{N}$, we denote by $[k]$ the set $[1, k] \cap \mathbb{N}$.

For a bounded measurable region $\Lambda \subset \mathbb{R}^d$ and any finite point configuration $\eta \in \mathcal{N}_\Lambda$, we write $|\eta|$ for the number of points in η. Note that this notation is the same that as the one we use for the volume of a region. The particular meaning will be clear from the context. Moreover, for $k \in \mathbb{N}$, we write $\binom{\eta}{k}$ for the set $\{\eta' \subseteq \eta \mid |\eta'| = k\}$.

3.1 Gibbs point processes

We introduce some additional notation for Gibbs point processes, used in the rest of the paper. Firstly, when dealing with a tuple $(x_1, \dots, x_k) \in (\mathbb{R}^d)^k$ we frequently denote it by the corresponding bold letter $\boldsymbol{x}$. Based on this, we write $\mathrm{d}\boldsymbol{x}$ for $\mathrm{d}x_1 \dots \mathrm{d}x_k$ and $H(\boldsymbol{x})$ for $H(x_1, \dots, x_k)$. Moreover, for any $k \in \mathbb{N}_0$ and $\boldsymbol{x} = (x_1, \dots, x_k) \in (\mathbb{R}^d)^k$ we write $\eta_{\boldsymbol{x}}$ for the set $\{x_1, \dots, x_k\}$, where the case $k = 0$ results in $\eta_{\boldsymbol{x}} = \emptyset$. Finally, for $\boldsymbol{x} \in \Lambda^k$ we write $\boldsymbol{\lambda}^{\boldsymbol{x}}$ for $\prod_{i\in[k]} \boldsymbol{\lambda}(x_i)$. This simplifies the definition of $\mu_{\boldsymbol{\lambda}}$ given in the introduction to

$$\mu_{\boldsymbol{\lambda}}(A) = \frac{1}{Z_\Lambda(\boldsymbol{\lambda})} \sum_{k\geq 0} \frac{1}{k!} \int_{\Lambda^k} \mathbb{1}_{\eta_{\boldsymbol{x}} \in A} \boldsymbol{\lambda}^{\boldsymbol{x}} \mathrm{e}^{-H(\boldsymbol{x})} \, \mathrm{d}\boldsymbol{x}.$$

Next, we formalize two different notions of restricting a Gibbs point process on Λ to a subregion $\Lambda' \subseteq \Lambda$ that are relevant throughout the paper.

The first is based on restricting the support of $\boldsymbol{\lambda}$ by defining a new activity function $\boldsymbol{\lambda}\mathbb{1}_{\Lambda'} : y \mapsto \boldsymbol{\lambda}(y) \cdot \mathbb{1}_{y\in\Lambda'}$ (for constant activity λ, we write $\lambda\mathbb{1}_{\Lambda'} : y \mapsto \lambda\mathbb{1}_{y\in\Lambda'}$). The resulting Gibbs point process is a probability measure on $(\mathcal{N}_\Lambda, \mathfrak{N}_\Lambda)$ with

$$\begin{aligned}\mu_{\boldsymbol{\lambda}\mathbb{1}_{\Lambda'}}(A) &= \frac{1}{Z_\Lambda(\boldsymbol{\lambda}\mathbb{1}_{\Lambda'})} \sum_{k\geq 0} \frac{1}{k!} \int_{\Lambda^k} \mathbb{1}_{\eta_{\boldsymbol{x}} \in A} (\boldsymbol{\lambda}\mathbb{1}_{\Lambda'})^{\boldsymbol{x}} \mathrm{e}^{-H(\boldsymbol{x})} \, \mathrm{d}\boldsymbol{x} \\ &= \frac{1}{Z_{\Lambda'}(\boldsymbol{\lambda})} \sum_{k\geq 0} \frac{1}{k!} \int_{\Lambda'^k} \mathbb{1}_{\eta_{\boldsymbol{x}} \in A} \boldsymbol{\lambda}^{\boldsymbol{x}} \mathrm{e}^{-H(\boldsymbol{x})} \, \mathrm{d}\boldsymbol{x}\end{aligned}$$

for all $A \in \mathfrak{N}_\Lambda$. In particular, for $A = \{\eta \in \mathcal{N}_\Lambda \mid \eta \cap (\Lambda')^{\mathrm{c}} > 0\}$, it holds that $\mu_{\boldsymbol{\lambda}\mathbb{1}_{\Lambda'}}(A) = 0$.

The second way of restricting a Gibbs point process $\mu_{\boldsymbol{\lambda}}$ is by projecting it to a measurable subregion $\Lambda' \subseteq \Lambda$. To this end, we write $\mu_{\boldsymbol{\lambda}}[\Lambda']$ for the image measure of $\mu_{\boldsymbol{\lambda}}$ under the map $\mathcal{N}_\Lambda \to \mathcal{N}_{\Lambda'}$, $\eta \mapsto \eta \cap \Lambda'$. By construction, $\mu_{\boldsymbol{\lambda}}[\Lambda']$ is a probability distribution on $(\mathcal{N}_{\Lambda'}, \mathfrak{N}_{\Lambda'})$ that, for every $A \in \mathfrak{N}_{\Lambda'}$, assigns a probability

$$\mu_{\boldsymbol{\lambda}}[\Lambda'](A) = \frac{1}{Z_\Lambda(\boldsymbol{\lambda})} \sum_{k\geq 0} \frac{1}{k!} \int_{\Lambda^k} \mathbb{1}_{\eta_{\boldsymbol{x}} \cap \Lambda' \in A} \boldsymbol{\lambda}^{\boldsymbol{x}} \mathrm{e}^{-H(\boldsymbol{x})} \, \mathrm{d}\boldsymbol{x}.$$

As discussed in Section 1, we frequently modify the activity function to encode the effect of fixing a certain point set (boundary condition). More precisely, for a fixed potential ϕ, an activity function $\boldsymbol{\lambda}$ and a point set $\eta \in \mathcal{N}_\Lambda$ we write $\boldsymbol{\lambda}_\eta$ for the function $y \mapsto \boldsymbol{\lambda}(y)\mathrm{e}^{-\sum_{x\in\eta} \phi(x,y)}$. Similarly, for $k \in \mathbb{N}$ and $\boldsymbol{x} \in \Lambda^k$ we write $\boldsymbol{\lambda}_{\boldsymbol{x}}$ for the activity function $y \mapsto \boldsymbol{\lambda}(y)\mathrm{e}^{-\sum_{i\in[k]} \phi(x_i,y)}$. We extend this notation to constant activity $\lambda \in \mathbb{R}_{\geq 0}$, writing $\lambda_\eta : y \mapsto \lambda\mathrm{e}^{-\sum_{x\in\eta} \phi(x,y)}$ and $\lambda_{\boldsymbol{x}} : y \mapsto \lambda\mathrm{e}^{-\sum_{i\in[k]} \phi(x_i,y)}$. Using this notation, a useful alternative definition of $\mu_{\boldsymbol{\lambda}}[\Lambda']$ is given by

$$\begin{aligned}\mu_{\boldsymbol{\lambda}}[\Lambda'](A) &= \frac{1}{Z_\Lambda(\boldsymbol{\lambda})} \sum_{k\geq 0} \frac{1}{k!} \int_{\Lambda'^k} \mathbb{1}_{\eta_{\boldsymbol{x}} \in A} \boldsymbol{\lambda}^{\boldsymbol{x}} \mathrm{e}^{-H(\boldsymbol{x})} Z_\Lambda\left(\boldsymbol{\lambda}_{\boldsymbol{x}}\mathbb{1}_{(\Lambda')^{\mathrm{c}}}\right) \mathrm{d}\boldsymbol{x} \\ &= \frac{1}{Z_\Lambda(\boldsymbol{\lambda})} \sum_{k\geq 0} \frac{1}{k!} \int_{\Lambda'^k} \mathbb{1}_{\eta_{\boldsymbol{x}} \in A} \boldsymbol{\lambda}^{\boldsymbol{x}} \mathrm{e}^{-H(\boldsymbol{x})} Z_{(\Lambda')^{\mathrm{c}}}(\boldsymbol{\lambda}_{\boldsymbol{x}}) \, \mathrm{d}\boldsymbol{x}\end{aligned}$$

for $A \in \mathfrak{N}_{\Lambda'}$. In particular, note that

$$\mu_{\boldsymbol{\lambda}\mathbb{1}_{\Lambda'}}[\Lambda'](A) = \frac{1}{Z_{\Lambda'}(\boldsymbol{\lambda})} \sum_{k \geq 0} \frac{1}{k!} \int_{\Lambda'^k} \mathbb{1}_{\eta_{\boldsymbol{x}} \in A} \boldsymbol{\lambda}^{\boldsymbol{x}} \mathrm{e}^{-H(\boldsymbol{x})} \, \mathrm{d}\boldsymbol{x}.$$

While $\mu_{\boldsymbol{\lambda}\mathbb{1}_{\Lambda'}}[\Lambda']$ and $\mu_{\boldsymbol{\lambda}\mathbb{1}_{\Lambda'}}$ seem similar, the former is a distribution on $(\mathcal{N}_{\Lambda'}, \mathfrak{N}_{\Lambda'})$ whereas the latter is defined on $(\mathcal{N}_{\Lambda}, \mathfrak{N}_{\Lambda})$.

3.2 Bernoulli factories

In designing our sampling algorithm, it will be useful to consider the following Bernoulli factory problem. We are given access to a sampler for $\mathrm{Ber}(p)$ and for $\mathrm{Ber}(q)$, that is samplers of Bernoulli random variables with parameters p and q respectively, where we further assume $p < q$. We want to sample a random variable $Z \sim \mathrm{Ber}\left(\frac{p}{q}\right)$.

Most work on Bernoulli factories studies their running time in terms of the number of coin flips required. In our setting, the time needed to generate each of these coin flips is random variable. Fortunately, suitable independence assumptions hold in our setting allowing us to prove the following lemma.

▶ **Lemma 3.1.** *Fix some $p, q \in [0, 1]$ such that $q - p \geq \epsilon$ for some $\epsilon > 0$. Further assume that we have oracle access to a sampler from $\mathrm{Ber}(p)$ and $\mathrm{Ber}(q)$ in the following sense:*

1. *every sample from $\mathrm{Ber}(p)$ (resp. $\mathrm{Ber}(q)$) is independent from all previous samples;*
2. *the expected running time for obtaining a sample from $\mathrm{Ber}(p)$ (resp. $\mathrm{Ber}(q)$), conditioned on previously obtained samples, is uniformly bounded by some $t \in \mathbb{R}_{\geq 0}$.*

Then we can sample from $\mathrm{Ber}\left(\frac{p}{q}\right)$ in $\mathrm{O}\left(t\epsilon^{-2}\right)$ expected time.

4 The algorithm

Let $\Lambda = [0, L)^d$ and consider a Gibbs point processes on Λ with uniform activity $\boldsymbol{\lambda}(x) \equiv \lambda$ for some $\lambda \in \mathbb{R}_{>0}$ and repulsive potential ϕ with finite range $r \in \mathbb{R}_{>0}$. Throughout the analysis of our algorithm, it will be useful to focus on configurations $\eta \in \mathcal{N}_{\Lambda}$ such that $\phi(x, y) < \infty$ for all $\{x, y\} \in \binom{\eta}{2}$, in which case we call η a *feasible configuration*.

Before stating our algorithm, we first formalize how we divide Λ into smaller boxes, following the description given in Section 2. For a r and L as above, let $N = \lceil L/r \rceil$. We set $\mathcal{V} = \{0, \dots, N-1\}^d$ to be the set of box indices and associate each box index $\boldsymbol{v} = (v_1, \dots, v_d) \in \mathcal{V}$ with the region $\Lambda_{\boldsymbol{v}} = \big([v_1 r, (v_1 + 1)r) \times \dots \times [v_d r, (v_d + 1)r)\big) \cap \Lambda$. As in Section 2, we extend this notation to sets of box indices $S \subseteq \mathcal{V}$ by setting $\Lambda_S = \bigcup_{\boldsymbol{v} \in S} \Lambda_{\boldsymbol{v}}$. Further, recall that, for $\boldsymbol{v} \in \mathcal{V}$, we write $\mathbb{B}_k(\boldsymbol{v})$ for the set of boxes $\boldsymbol{w} \in \mathcal{V}$ with $\|\boldsymbol{v} - \boldsymbol{w}\|_{\infty} < k$. As mentioned earlier, our algorithm tries to update in each step the point configuration on a subset of boxes $B \subseteq \mathcal{V}$. To this end, for $S \subseteq \mathcal{V}$, $\boldsymbol{v} \in S, r \in \mathbb{R}_{>0}$ and $\ell \in \mathbb{N}$, we define

$$B(S, \boldsymbol{v}, \ell) := \{\boldsymbol{v}\} \cup (\mathbb{B}_{\boldsymbol{v}}(\ell) \setminus S).$$

We refer to the parameter ℓ as the *update radius*. Finally, recall that we write $\partial S = \left(\bigcup_{\boldsymbol{v} \in S} \mathbb{B}_1(\boldsymbol{v})\right) \setminus S$ for the outer boundary of $S \subseteq \mathcal{V}$.

Whether the algorithm updates the point configuration in iteration t depends on the outcome of a Bernoulli random variable F_t, called the *Bayes filter*. We introduce the following definition.

▶ **Definition 4.1.** *Fix a repulsive potential ϕ of range $r \in \mathbb{R}_{>0}$, an activity $\lambda \in \mathbb{R}_{>0}$ and some $\ell \in \mathbb{N}$. We call a function $C : 2^{\mathcal{V}} \times \mathcal{V} \times \mathcal{N}_\Lambda \to [0,1]$ a* Bayes filter correction *if, for all non-empty $S \subseteq \mathcal{V}$ and $\boldsymbol{v} \in S$, it holds that*

1. *$C(S, \boldsymbol{v}, \cdot)$ is $\mathfrak{R}_{\Lambda_S}$-measurable (in particular $C(S, \boldsymbol{v}, \eta) = C(S, \boldsymbol{v}, \eta \cap \Lambda_S)$ for all $\eta \in \mathcal{N}_\Lambda$),*
2. *there is some $\varepsilon > 0$ such that for $B = B(S, \boldsymbol{v}, \ell)$, $H = (S \cup B)^{\mathrm{c}}$ and all feasible $\eta \in \mathcal{N}_\Lambda$ it holds that*

$$\varepsilon \le C(S, \boldsymbol{v}, \eta) \le \inf_{\substack{\xi \in \mathcal{N}_{\Lambda_H} \\ \xi \cup (\eta \cap \Lambda_S) \text{ is feasible}}} \left\{ \frac{Z_{\Lambda_{B \setminus \{\boldsymbol{v}\}}}\left(\lambda_{\xi \cup (\eta \cap \Lambda_S)}\right)}{Z_{\Lambda_B}\left(\lambda_{\xi \cup (\eta \cap \Lambda_{S \setminus \{\boldsymbol{v}\}})}\right)} \right\}.$$

Our perfect sampling procedure is stated in Algorithm 1.

Algorithm 1 Perfect sampling algorithm for repulsive Gibbs point processes.

```
   Data: region $\Lambda = [0, L)^d$, repulsive potential $\phi$ of range at most $r \in \mathbb{R}_{>0}$, activity
         $\lambda \in \mathbb{R}_{>0}$, update radius $\ell \in \mathbb{N}$
 1 set $t = 0$, $\mathcal{U}_t = \mathcal{V}$, $X_t = \emptyset$
 2 while $\mathcal{U}_t \neq \emptyset$ do
 3     draw $\boldsymbol{u}_t \in \mathcal{U}_t$ uniformly at random
 4     set $B = B(\mathcal{U}_t, \boldsymbol{u}_t, \ell)$
 5     draw $F_t$ from Ber$\left(C(\mathcal{U}_t, \boldsymbol{u}_t, X_t) \cdot \frac{Z_{\Lambda_B}(\lambda_{X_t \cap \Lambda_{\partial B}})}{Z_{\Lambda_{B \setminus \{\boldsymbol{u}_t\}}}(\lambda_{X_t \cap \Lambda_{\partial B \cup \{\boldsymbol{u}_t\}}})}\right)$ where $C$ is a Bayes
        filter correction as in Definition 4.1
 6     if $F_t$ = 1 then
 7         draw $Y$ from $\mu_{\lambda_{X_t \cap (\Lambda_B)^{\mathrm{c}}} \mathbb{1}_{\Lambda_B}}[\Lambda_B]$
 8         set $X_{t+1} = (X_t \setminus \Lambda_B) \cup Y$
 9         set $\mathcal{U}_{t+1} = \mathcal{U}_t \setminus \{\boldsymbol{u}_t\}$
10     else
11         set $\mathcal{U}_{t+1} = \mathcal{U}_t \cup \partial B$
12     increase $t$ by 1
13 return $X_t$
```

Before we get to the question of how to sample an appropriate Bayes filter in step 5, the following statement ensures that the algorithm produces the correct output distribution.

▶ **Theorem 4.2.** *Let $T = \inf_{t \in \mathbb{N}}\{\mathcal{U}_t = \emptyset\}$. Then T is almost surely finite and for all $t \in \mathbb{N}_0$ with $\mathbb{P}[t \ge T] > 0$ and all $A \in \mathfrak{R}_\Lambda$, it holds that $\mathbb{P}[X_t \in A \mid t \ge T] = \mu_\lambda(A)$.*

We proceed to exemplify how we use Bernoulli factories to sample the Bayes filter. For brevity, we focus on the hard-sphere model here. The more general case of bounded-range repulsive potential can be found in the full version of the paper [3].

Bayes filter for the hard-sphere model

Recall that for the hard-sphere model, we have $\phi(x, y) = \infty$ if $\mathrm{dist}(x, y) < r$ and 0 otherwise. Given a non-empty set of boxes $S \subseteq \mathcal{V}$, $\boldsymbol{v} \in S$ and a feasible configuration $\eta \in \mathcal{N}_\Lambda$, we want to construct a Bayes filter correction $C(S, \boldsymbol{v}, \eta)$ that allows us to efficiently sample the filter.

To this end, set $B = B(S, \boldsymbol{v}, \ell)$ and $H = (S \cup B)^{\mathrm{c}}$. Our construction makes use of two ingredients. Firstly, we argue that, instead of minimization over the uncountable set of boundary conditions $\mathcal{N}_{\Lambda_H}$, it suffices to minimize over subsets of the finite set $(\delta_1\mathbb{Z})^d \cap \Lambda_{H\cap\partial B}$ for a sufficiently small $\delta_1 > 0$. Secondly, choosing a sufficiently small $\delta_2 > 0$, we show that we can approximate the involved partition functions using the function

$$\hat{Z}(S, \eta, \delta_2) = \sum_{\gamma \subseteq (\delta_2\mathbb{Z})^d \cap \Lambda_S} \lambda^{|\gamma|} \delta_2^{|\gamma|} \cdot D(\gamma) \cdot D(\gamma \mid \eta \cap \Lambda_{\partial S}), \tag{5}$$

where $D(\gamma) = \prod_{\{x,y\} \in \binom{\gamma}{2}} \mathbb{1}_{\mathrm{dist}(x,y) \geq r}$ and $D(\gamma \mid \eta) = \prod_{x \in \gamma} \prod_{y \in \eta} \mathbb{1}_{\mathrm{dist}(x,y) \geq r}$.

The following lemma then gives a way to construct a Bayes filter correction for the hard-sphere model.

► **Lemma 4.3.** *For non-empty $S \subseteq \mathcal{V}$, $\boldsymbol{v} \in S$, feasible $\eta \in \mathcal{N}_\Lambda$ and $\varepsilon, \delta_1, \delta_2 > 0$ define*

$$C_\varepsilon(S, \boldsymbol{v}, \eta) := \mathrm{e}^{-\varepsilon} \cdot \min_{\gamma \subseteq (\delta_1\mathbb{Z})^d \cap \Lambda_{H\cap\partial B}} \frac{\hat{Z}(B \setminus \{\boldsymbol{v}\}, \gamma \cup (\eta \cap \Lambda_S), \delta_2)}{\hat{Z}\big(B, \gamma \cup (\eta \cap \Lambda_{S\setminus\boldsymbol{v}}), \delta_2\big)},$$

where $B = B(S, \boldsymbol{v}, \ell)$ and $H = (S \cup B)^{\mathrm{c}}$. For δ_1, δ_2 sufficiently small, depending only on d, r, ℓ and ε, it holds that $C_\varepsilon(S, \boldsymbol{v}, \eta)$ is a Bayes filter correction.

In fact, we will not use C_ε directly for our Bayes filter, but a slightly scaled version $0 < \mathrm{e}^{-\varepsilon} C_\varepsilon$, which is again a Bayes filter correction. The additional slack allows us to efficiently sample the Bayes filter by using a Bernoulli factory, as we argue in the next lemma.

► **Lemma 4.4.** *Let $S \subseteq \mathcal{V}$ be non-empty, $\boldsymbol{v} \in S$ and $\eta \in \mathcal{N}_\Lambda$ be feasible, and set $B = B(S, \boldsymbol{v}, \ell)$. For all $\varepsilon > 0$ we can sample a Bernoulli random variable with success probability*

$$\mathrm{e}^{-\varepsilon} C_\varepsilon(S, \boldsymbol{v}, \eta) \cdot \frac{Z_{\Lambda_B}(\lambda_{\eta\cap\Lambda_{\partial B}})}{Z_{\Lambda_{B\setminus\{\boldsymbol{v}\}}}\big(\lambda_{\eta\cap\Lambda_{\partial B\cup\{\boldsymbol{v}\}}}\big)}$$

with expected running time only depending on ε, ℓ, r, λ and d.

The core idea of the above lemma to express $\frac{Z_{\Lambda_B}(\lambda_{\eta\cap\Lambda_{\partial B}})}{Z_{\Lambda_{B\setminus\{\boldsymbol{v}\}}}\big(\lambda_{\eta\cap\Lambda_{\partial B\cup\{\boldsymbol{v}\}}}\big)}$ as a fraction of probabilities. Together with the fact that $\mathrm{e}^{-\varepsilon} C_\varepsilon(S, \boldsymbol{v}, \eta) < 1$, this allows us to write the success probability of the Bayes filter as a fraction of probabilities $\frac{p}{q}$. Arguing that $p < q$, and that we can sample $\mathrm{Ber}(p)$ and $\mathrm{Ber}(q)$ efficiently allows us to apply Lemma 3.1 to prove Lemma 4.4.

While the above suffices to perform each iteration of Algorithm 1 efficiently, we still need to bound the number of iterations. For this, we derive a lower bound on the success probability of the Bayes filter with correction $\mathrm{e}^{-\varepsilon} C_\varepsilon(\cdot)$ for a particular choice of ε, using the assumption of strong spatial mixing.

► **Lemma 4.5.** *Consider a hard-sphere model that exhibits (a, b)-strong spatial mixing up to λ. Then there are constants a', b', only depending on a, b, r, λ and d, such that for all non-empty $S \subseteq \mathcal{V}$, $\boldsymbol{v} \in S$ and feasible $\eta \in \mathcal{N}_\Lambda$ it holds that*

$$\mathrm{e}^{-\mathrm{e}^{-\ell}} C_{\mathrm{e}^{-\ell}}(S, \boldsymbol{v}, \eta) \cdot \frac{Z_{\Lambda_B}(\lambda_{\eta\cap\Lambda_{\partial B}})}{Z_{\Lambda_{B\setminus\{\boldsymbol{v}\}}}\big(\lambda_{\eta\cap\Lambda_{\partial B\cup\{\boldsymbol{v}\}}}\big)} \geq 1 - a'\mathrm{e}^{-b'\ell}.$$

Lemma 4.5 allows us to control the success probability of the Bayes filter in terms of ℓ. Combining the results above gives the following theorem.

▶ **Theorem 4.6.** *Consider Algorithm 1 on a hard-sphere model with* $C(\cdot) = \mathrm{e}^{-\mathrm{e}^{-\ell}} C_{\mathrm{e}^{-\ell}}(\cdot)$ *as Bayes filter correction in line 5. We can run the algorithm in almost-surely finite running time and, on termination, it outputs a sample from the hard-sphere Gibbs measure* μ_λ *on* Λ*. Moreover, if the hard-sphere model satisfies* (a, b)*-strong spatial mixing and if* ℓ *is chosen as a sufficiently large constant, depending on* a*,* b*,* r*,* λ *and* d*, then we can run the algorithm in expected time* $\mathrm{O}(|\Lambda|)$*.*

References

1 Michael Aizenman and Richard Holley. Rapid convergence to equilibrium of stochastic Ising models in the Dobrushin Shlosman regime. *Percolation theory and ergodic theory of infinite particle systems*, pages 1–11, 1987.

2 Berni Julian Alder and Thomas Everett Wainwright. Phase transition for a hard sphere system. *The Journal of Chemical Physics*, 27(5):1208–1209, 1957.

3 Konrad Anand, Andreas Göbel, Marcus Pappik, and Will Perkins. Perfect sampling for hard spheres from strong spatial mixing. *arXiv preprint*, 2023. `arXiv:2305.02450`.

4 Konrad Anand and Mark Jerrum. Perfect sampling in infinite spin systems via strong spatial mixing. *SIAM Journal on Computing*, 51(4):1280–1295, 2022.

5 Søren Asmussen, Peter W Glynn, and Hermann Thorisson. Stationarity detection in the initial transient problem. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 2(2):130–157, 1992.

6 Etienne P Bernard and Werner Krauth. Two-step melting in two dimensions: first-order liquid-hexatic transition. *Physical Review Letters*, 107(15):155704, 2011.

7 Etienne P Bernard, Werner Krauth, and David B Wilson. Event-chain Monte Carlo algorithms for hard-sphere systems. *Physical Review E*, 80(5):056704, 2009.

8 Steffen Betsch and Günter Last. On the uniqueness of Gibbs distributions with a non-negative and subcritical pair potential. In *Annales de l'Institut Henri Poincare (B) Probabilites et statistiques*, volume 59(2), pages 706–725. Institut Henri Poincaré, 2023.

9 Siddharth Bhandari and Sayantan Chakraborty. Improved bounds for perfect sampling of k-colorings in graphs. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 631–642, 2020.

10 Zongchen Chen, Kuikui Liu, Nitya Mani, and Ankur Moitra. Strong spatial mixing for colorings on trees and its algorithmic applications. *arXiv preprint*, 2023. `arXiv:2304.01954`.

11 Hofer Temmel Christoph. Disagreement percolation for the hard-sphere model. *Electronic Journal of Probability*, 24:1–22, 2019.

12 David Dereudre. Introduction to the theory of Gibbs point processes. In *Stochastic Geometry*, pages 181–229. Springer, 2019.

13 Persi Diaconis. The Markov Chain Monte Carlo revolution. *Bulletin of the American Mathematical Society*, 46(2):179–205, 2009.

14 Shaddin Dughmi, Jason Hartline, Robert D Kleinberg, and Rad Niazadeh. Bernoulli factories and black-box reductions in mechanism design. *Journal of the ACM (JACM)*, 68(2):1–30, 2021.

15 Martin Dyer, Alistair Sinclair, Eric Vigoda, and Dror Weitz. Mixing in time and space for lattice spin systems: A combinatorial view. *Random Structures & Algorithms*, 24(4):461–479, 2004.

16 Michael Engel, Joshua A Anderson, Sharon C Glotzer, Masaharu Isobe, Etienne P Bernard, and Werner Krauth. Hard-disk equation of state: First-order liquid-hexatic transition in two dimensions with three simulation methods. *Physical Review E*, 87(4):042134, 2013.

17 Stefan Felsner and Lorenz Wernisch. Markov chains for linear extensions, the two-dimensional case. In *SODA*, pages 239–247, 1997.

18 Weiming Feng, Heng Guo, and Yitong Yin. Perfect sampling from spatial mixing. *Random Structures & Algorithms*, 61(4):678–709, 2022.

19 Weiming Feng and Yitong Yin. On local distributed sampling and counting. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 189–198, 2018.

20 Roberto Fernández, Aldo Procacci, and Benedetto Scoppola. The analyticity region of the hard sphere gas. Improved bounds. *J. Stat. Phys.*, 5:1139–1143, 2007.

21 Pablo A Ferrari, Roberto Fernández, and Nancy L Garcia. Perfect simulation for interacting point processes, loss networks and Ising models. *Stochastic Processes and their Applications*, 102(1):63–88, 2002.

22 David Gamarnik, Dmitriy Katz, and Sidhant Misra. Strong spatial mixing of list coloring of graphs. *Random Structures & Algorithms*, 46(4):599–613, 2015.

23 Nancy L Garcia. Perfect simulation of spatial processes. *Resenhas do Instituto de Matemática e Estatística da Universidade de São Paulo*, 4(3):283–325, 2000.

24 J Groeneveld. Two theorems on classical many-particle systems. *Phys. Letters*, 3, 1962.

25 Heng Guo and Mark Jerrum. Perfect simulation of the hard disks model by partial rejection sampling. *Annales de l'Institut Henri Poincaré D*, 8(2):159–177, 2021.

26 Heng Guo, Mark Jerrum, and Jingcheng Liu. Uniform sampling through the Lovász local lemma. *Journal of the ACM (JACM)*, 66(3):1–31, 2019.

27 Olle Häggström and Karin Nelander. Exact sampling from anti-monotone systems. *Statistica Neerlandica*, 52(3):360–380, 1998.

28 Olle Häggström, Marie-Colette N.M. van Lieshout, and Jesper Møller. Characterization results and Markov chain Monte Carlo algorithms including exact simulation for some spatial point processes. *Bernoulli*, 5(4):641–658, 1999.

29 Thomas P Hayes and Cristopher Moore. Lower bounds on the critical density in the hard disk model via optimized metrics. *arXiv preprint*, 2014. arXiv:1407.1930.

30 Kun He, Xiaoming Sun, and Kewen Wu. Perfect sampling for (atomic) Lovász Local Lemma. *arXiv preprint*, 2021. arXiv:2107.03932.

31 Kun He, Chunyang Wang, and Yitong Yin. Sampling Lovász Local Lemma for general constraint satisfaction solutions in near-linear time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 147–158. IEEE, 2022.

32 Kun He, Kewen Wu, and Kuan Yang. Improved bounds for sampling solutions of random CNF formulas. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3330–3361. SIAM, 2023.

33 Tyler Helmuth, Will Perkins, and Samantha Petti. Correlation decay for hard spheres via Markov chains. *The Annals of Applied Probability*, 32(3):2063–2082, 2022.

34 Christoph Hofer-Temmel and Pierre Houdebert. Disagreement percolation for Gibbs ball models. *Stochastic Processes and their Applications*, 129(10):3922–3940, 2019.

35 Richard Holley. Possible rates of convergence in finite range, attractive spin systems. *Part. Syst. Random Media Large Deviat.*, 41:215, 1985.

36 Mark Huber. Spatial birth–death swap chains. *Bernoulli*, 18(3):1031–1041, 2012.

37 Mark Huber. Nearly optimal Bernoulli factories for linear functions. *Combin. Probab. Comput.*, 25(4):577–591, 2016.

38 Mark Huber, Elise Villella, Daniel Rozenfeld, and Jason Xu. Bounds on the artificial phase transition for perfect simulation of hard core Gibbs processes. *Involve, a Journal of Mathematics*, 5(3):247–255, 2013.

39 Masaharu Isobe. Hard sphere simulation in statistical physics—methodologies and applications. *Molecular Simulation*, 42(16):1317–1329, 2016.

40 Vishesh Jain, Ashwin Sah, and Mehtaab Sawhney. Perfectly sampling $k \geq (8/3 + o(1))\Delta$-colorings in graphs. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1589–1600, 2021.

41 Matthew Jenssen, Marcus Michelen, and Mohan Ravichandran. Quasipolynomial-time algorithms for repulsive Gibbs point processes. *arXiv preprint*, 2022. arXiv:2209.10453.

42 Mark Jerrum and Alistair Sinclair. The Markov chain Monte Carlo method: an approach to approximate counting and integration. *Approximation algorithms for NP-hard problems*, pages 482–520, 1996.

43 Ravi Kannan, Michael W. Mahoney, and Ravi Montenegro. Rapid mixing of several Markov chains for a hard-core model. In *Algorithms and computation*, volume 2906 of *Lecture Notes in Comput. Sci.*, pages 663–675. Springer, Berlin, 2003.

44 Frank P Kelly and Brian D Ripley. A note on Strauss's model for clustering. *Biometrika*, pages 357–360, 1976.

45 Wilfrid S Kendall. Perfect simulation for the area-interaction point process. In *Probability towards 2000*, pages 218–234. Springer, 1998.

46 Wilfrid S Kendall and Jesper Møller. Perfect simulation using dominating processes on ordered spaces, with application to locally stable point processes. *Advances in Applied Probability*, pages 844–865, 2000.

47 Botao Li, Yoshihiko Nishikawa, Philipp Höllmer, Louis Carillo, AC Maggs, and Werner Krauth. Hard-disk pressure computations – A historic perspective. *The Journal of Chemical Physics*, 157(23):234111, 2022.

48 Jingcheng Liu, Alistair Sinclair, and Piyush Srivastava. Correlation decay and partition function zeros: Algorithms and phase transitions. *SIAM Journal on Computing*, pages FOCS19–200, 2022.

49 Laszlo Lovasz and Peter Winkler. Exact mixing in an unknown Markov chain. *The Electronic Journal of Combinatorics*, pages R15–R15, 1995.

50 Hartmut Löwen. Fun with hard spheres. In *Statistical physics and spatial statistics*, volume 554, pages 295–331. Springer, 2000.

51 Pinyan Lu and Yitong Yin. Improved FPTAS for multi-spin systems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques: 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*, pages 639–654. Springer, 2013.

52 Fabio Martinelli. Lectures on Glauber dynamics for discrete spin models. In *Lectures on probability theory and statistics*, pages 93–191. Springer, 1999.

53 Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

54 Marcus Michelen and Will Perkins. Potential-weighted connective constants and uniqueness of Gibbs measures. *arXiv preprint*, 2021. `arXiv:2109.01094`.

55 Marcus Michelen and Will Perkins. Analyticity for classical gasses via recursion. *Communications in Mathematical Physics*, pages 1–22, 2022.

56 Marcus Michelen and Will Perkins. Strong spatial mixing for repulsive point processes. *Journal of Statistical Physics*, 189(1):9, 2022.

57 Sarat B Moka, Dirk P Kroese, et al. Perfect sampling for Gibbs point processes using partial rejection sampling. *Bernoulli*, 26(3):2082–2104, 2020.

58 Jesper Møller. A review of perfect simulation in stochastic geometry. *Lecture Notes-Monograph Series*, pages 333–355, 2001.

59 Jesper Møller and Rasmus Plenge Waagepetersen. *Statistical inference and simulation for spatial point processes*. CRC Press, 2003.

60 Duncan J Murdoch and Peter J Green. Exact sampling from a continuous state space. *Scandinavian Journal of Statistics*, 25(3):483–502, 1998.

61 Serban Nacu and Yuval Peres. Fast simulation of new coins from old. *The Annals of Applied Probability*, 15(1A):93–115, 2005.

62 Oliver Penrose. Convergence of fugacity expansions for fluids and lattice gases. *Journal of Mathematical Physics*, 4(10):1312–1320, 1963.

63 James Gary Propp and David Bruce Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures & Algorithms*, 9(1-2):223–252, 1996.

64 James Gary Propp and David Bruce Wilson. How to get a perfectly random sample from a generic Markov chain and generate a random spanning tree of a directed graph. *Journal of Algorithms*, 27(2):170–217, 1998.

65 Dana Randall. Rapidly mixing Markov chains with applications in computer science and physics. *Computing in Science & Engineering*, 8(2):30–41, 2006.

66 Guus Regts. Absence of zeros implies strong spatial mixing. *Probability Theory and Related Fields*, pages 1–21, 2023.

67 David Ruelle. Correlation functions of classical gases. *Annals of Physics*, 25:109–120, 1963.

68 David Ruelle. *Statistical mechanics: Rigorous results.* World Scientific, 1999.

69 Alistair Sinclair, Piyush Srivastava, Daniel Štefankovič, and Yitong Yin. Spatial mixing and the connective constant: Optimal bounds. *Probability Theory and Related Fields*, 168(1-2):153–197, 2017.

70 Yinon Spinka. Finitary codings for spatial mixing Markov random fields. *Ann. Probab.*, 48(3):1557–1591, 2020.

71 David J Strauss. A model for clustering. *Biometrika*, 62(2):467–475, 1975.

72 Daniel W Stroock and Boguslaw Zegarlinski. The logarithmic Sobolev inequality for discrete spin systems on a lattice. *Communications in Mathematical Physics*, 149(1):175–193, 1992.

73 Marie-Colette N.M. van Lieshout. *Markov Point Processes and Their Applications.* Imperial College Press, 2000.

74 Dror Weitz. Counting independent sets up to the tree threshold. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, STOC 2006, pages 140–149. ACM, 2006.

75 William W. Wood, F. Raymond Parker, and Jack David Jacobson. Recent monte carlo calculations of the equation of state of lenard-jones and hard sphere molecules. *Il Nuovo Cimento (1955-1965)*, 9:133–143, 1958.

Subset Sum in Time $2^{n/2}/\mathrm{poly}(n)$

Xi Chen
Columbia University, New York, NY, USA

Yaonan Jin
Columbia University, New York, NY, USA

Tim Randolph
Columbia University, New York, NY, USA

Rocco A. Servedio
Columbia University, New York, NY, USA

Abstract

A major goal in the area of exact exponential algorithms is to give an algorithm for the (worst-case) n-input Subset Sum problem that runs in time $2^{(1/2-c)n}$ for some constant $c > 0$. In this paper we give a Subset Sum algorithm with worst-case running time $O(2^{n/2} \cdot n^{-\gamma})$ for a constant $\gamma > 0.5023$ in standard word RAM or circuit RAM models. To the best of our knowledge, this is the first improvement on the classical "meet-in-the-middle" algorithm for worst-case Subset Sum, due to Horowitz and Sahni, which can be implemented in time $O(2^{n/2})$ in these memory models [16].

Our algorithm combines a number of different techniques, including the "representation method" introduced by Howgrave-Graham and Joux [17] and subsequent adaptations of the method in Austrin, Kaski, Koivisto, and Nederlof [4], and Nederlof and Węgrzycki [20], and "bit-packing" techniques used in the work of Baran, Demaine, and Pătraşcu [5] on subquadratic algorithms for 3SUM.

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

Keywords and phrases Exact algorithms, subset sum, log shaving

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.39

Category RANDOM

Related Version *Full Version*: `https://arxiv.org/abs/2301.07134`

Funding *Xi Chen*: Supported in part by NSF grants IIS-1838154, CCF-2106429 and CCF-2107187.
Rocco A. Servedio: Supported in part by NSF awards IIS-1838154, CCF-2106429, and CCF-2211238, and by the Simons Collaboration on Algorithms and Geometry.

Acknowledgements We would like to thank Martin Dietzfelbinger for pointing out the work [12], as well as several anonymous referees for helpful suggestions.

1 Introduction

One of the most well-known and simple-to-state NP-complete problems is the *Subset Sum* problem. An instance of Subset Sum consists of a list $X = (x_1, \dots, x_n)$ of n positive integer values and a positive integer target t, and the output is either a subset $S \subseteq X$ such that $\sum_{x_i \in S} x_i = t$ or a report that no such subset exists. Subset Sum was one of the original 21 problems proved NP-complete in Karp's seminal paper [18] and has been the subject of intensive study from many different perspectives for at least five decades.

This paper is motivated by the following open problem in the theory of exact exponential time algorithms: how quickly can we solve worst-case instances of Subset Sum? Exhaustive search over all possible solutions yields a trivial $2^n \cdot \mathrm{poly}(n)$-time algorithm. In 1974, Horowitz and Sahni introduced the "meet-in-the-middle" technique, which gives an algorithm that can be implemented in $O(2^{n/2})$ time in standard RAM models [16]. Since then, obtaining

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 39; pp. 39:1–39:18
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

a $2^{(1/2-c)n}$-time algorithm for some constant $c > 0$ has emerged as a major goal in the exact exponential time algorithms community (explicitly mentioned in [23, 11, 3, 4, 20] and numerous other works) which has attracted the attention of many researchers.

Intriguing progress has been made on a number of variants of the core worst-case Subset Sum problem. More than forty years ago Schroeppel and Shamir [22] improved the $2^{n/2} \cdot \mathrm{poly}(n)$ space complexity of the meet-in-the-middle algorithm by giving an algorithm that runs in $2^{n/2} \cdot \mathrm{poly}(n)$ time and $2^{n/4} \cdot \mathrm{poly}(n)$ space. An exciting recent breakthrough by Nederlof and Węgrzycki [20] further improved this space complexity to $2^{0.249999n}$. In [17] Howgrave-Graham and Joux gave an algorithm which can solve *average-case* Subset Sum instances in time $2^{0.337n}$,[1] and this was later improved to $2^{0.291n}$ in the work of Becker et al. [6]. The closely related *Equal Subset Sum* problem, which looks for two subsets with the same sum, can be solved exponentially faster than suggested by meet-in-the-middle [19] and also yields further improvements in the average case [10]. However, to the best of our knowledge, there have been no improvements on the worst-case $O(2^{n/2})$ runtime of the meet-in-the-middle algorithm for Subset Sum since it was first introduced almost fifty years ago.

Our contribution: Worst-case Subset Sum in $2^{n/2}/\mathrm{poly}(n)$ time. Given the longstanding difficulty of achieving a $2^{(1/2-c)n}$-time worst-case algorithm for Subset Sum, it is natural to consider the relaxed goal of achieving *some* nontrivial speedup of the meet-in-the-middle algorithm. In this paper we achieve this goal; more precisely, we give three different randomized algorithms for worst-case Subset Sum, each of which runs in time $O(2^{n/2} \cdot n^{-\gamma})$ for a specific constant $\gamma > 0$ in a standard word RAM or circuit RAM model (described in detail in Section 1.1 below). Our fastest algorithm, which combines techniques from our other two algorithms, runs in time $O(2^{n/2} \cdot n^{-0.5023})$.

The improvements we achieve over the $O(2^{n/2})$ runtime of the meet-in-the-middle algorithm for Subset Sum are analogous to "log-shaving" improvements on the runtimes of well-known and simple polynomial-time algorithms for various problems which have resisted attempts at polynomial-factor improvements. There is a substantial strand of research along these lines (see [9, 8] for a non-exhaustive overview); indeed, Abboud and Bringmann [1] have recently stated that: "A noticeable fraction of Algorithms papers in the last few decades improve the runtime of well-known algorithms for fundamental problems by logarithmic factors." In our setting, since the well-known and simple algorithm for Subset Sum (namely, meet-in-the-middle) runs in exponential time, saving a $\mathrm{poly}(n)$ factor, as we do, is analogous to "log-shaving". Indeed, as we discuss in Section 1.2 below, our first and most straightforward algorithm is based on "bit-packing" techniques that were used by Baran, Demaine, and Pătraşcu [5] to shave log factors from the standard $O(n^2)$-time algorithm for the 3SUM problem. We find it somewhat surprising that the "log-shaving" perspective has not previously appeared in the literature on Subset Sum, and we hope that our work will lead to further (and more substantial) runtime improvements for Subset Sum and other problems with well-known and simple exponential-time algorithms.

▶ Remark 1. We note that by a straightforward reduction, an algorithm for 4SUM running in time $O(n^2/\log(n)^\alpha)$ for any constant $\alpha > 0$ would immediately imply a Subset Sum algorithm running in time $O(2^{n/2}/n^\alpha)$, which would be a result comparable to ours. However, while log-shaving results for 3SUM are known [5, 14], giving an $o(n^2)$ algorithm for 4SUM is a well-known open problem.

[1] See the last paragraph of [6] for a discussion of the runtime of [17].

1.1 Our Computational Model

A Subset Sum instance is parameterized by the number of inputs n and the size of the target value t (without loss of generality, $x_1, x_2, \ldots, x_n \leq t$). Thus it is natural to adopt a memory model with word length $\ell = \Theta(\log t)$ such that each input integer can be stored in a single word. This is the framework used in the work of Pisinger [21], which studies dynamic programming approaches for Subset Sum in the word RAM model (see [21, Equation (1)]). We also note that this memory model is analogous to the standard RAM model that is commonly used for problems such as 3SUM (see e.g. [5]), where it is assumed that each input value is at most $\mathrm{poly}(n)$ and hence fits into a single $O(\log n)$-bit machine word.

This framework lets us consider arbitrary input instances of Subset Sum with no constraints on the size of the input integers. If $t = 2^{o(n)}$, standard dynamic programming algorithms [7] solve the problem in time $O(nt) = 2^{o(n)}$, which supersedes our $\mathrm{poly}(n)$-factor improvements over meet-in-the-middle; hence throughout the paper we assume $t = 2^{\Omega(n)}$. It is arguably most natural to think about instances in which $t = 2^{\Theta(n)}$, in which case $\ell = \Theta(n)$, and we encourage the first-time reader to imagine $\ell = \Theta(n)$ for easy digestion. More precisely, we make the assumption throughout the paper that the word size $\ell = \mathrm{poly}(n)$, although some of our results even hold for extremely large word sizes and are footnoted accordingly.

We consider runtime in two standard variants of the RAM model. The first is *circuit RAM*; in this model, any operation that maps a constant number of words to a single word and has a $\mathrm{poly}(\ell)$-size circuit with unbounded fan-in gates can be performed in time proportional to the depth of the circuit. Consequently, in the circuit RAM model, AC^0 operations on a *constant* number of words can be performed in *constant* time, and multiplying, performing modular division, etc., on two ℓ-bit words can be performed in time $O(\log \ell)$. The second is *word RAM*, in which the usual arithmetic operations, including multiplication, are assumed to take unit time, but arbitrary AC^0 operations are not atomic operations on words. We present each of our algorithms for the stronger circuit RAM model,[2] and explain adaptations that give corresponding word RAM algorithms.

1.2 Results, Techniques, and Organization

In Section 2 we establish our notation and review some background results and observations that will be used throughout the paper.

Sections 3–5 give our three new algorithms, which augment the standard meet-in-the-middle approach in different ways to achieve their respective runtime improvements. Each of our algorithms is a randomized decision algorithm that runs in the time bound claimed below and on every input instance outputs the correct answer with probability at least 3/4. Further, each of our algorithms has one-sided error, i.e., it never makes a mistake when it outputs "yes".

Our first and simplest algorithm, presented in Section 3, achieves a runtime of $\widetilde{O}(2^{n/2} \cdot \ell^{-1/2}) \leq \widetilde{O}(2^{n/2} \cdot n^{-1/2})$ in the circuit RAM model and $\widetilde{O}(2^{n/2} \cdot n^{-1/2})$ in the word RAM model, for all $\ell = \mathrm{poly}(n)$.[3] It works by adapting the *bit-packing* trick, a technique developed by Baran, Demaine, and Pătraşcu [5] for the 3SUM problem, for the `Meet-in-the-Middle` algorithm. The idea is to compress the two lists of partial subset sums used in `Meet-in-the-Middle` by packing hashes of multiple values into a machine

[2] Note that any algorithm in the word RAM model can be simulated in the circuit RAM model with no more than an $O(\log \ell)$-factor slowdown.

[3] The notation $\widetilde{O}(\cdot)$ suppresses $\mathrm{polylog}(\ell) = \mathrm{polylog}(n)$ factors.

word, while preserving enough information to make it possible to run (an adaptation of) `Meet-in-the-Middle` on the lists of hashed and packed values. This results in a runtime savings over performing `Meet-in-the-Middle` on the original lists (without hashing and packing), because processing a pair of words, each containing multiple hashed values, takes constant expected time in the circuit RAM model and can be memoized to take constant time in the word RAM model.

Our second algorithm, given in Section 4, achieves a runtime of $O(2^{n/2} \cdot \ell^{-\gamma}) \leq O(2^{n/2} \cdot n^{-\gamma})$ for some constant $\gamma > 0.01$ in the circuit RAM model and $O(2^{n/2} \cdot n^{-\gamma})$ in the word RAM model, for all $\ell = \text{poly}(n)$. Although the time savings is smaller than our first algorithm, we believe that this algorithm is conceptually interesting since it avoids bit-packing and instead combines `Meet-in-the-Middle` with two techniques devised in prior work on Subset Sum. The first of these is the "representation method" introduced by Howgrave-Graham and Joux [17]. Roughly speaking, the idea of this method is to (i) increase the size of the search space in such a way that a single solution has many "representations" in the space of enhanced solutions, and then (ii) search over only a fraction of the enhanced solution space. A consequence of expanding the solution space, though, is that a number of "pseudosolutions", solutions that contain certain input elements more than once, are introduced. This leads us to the second technique, i.e. the use of a fast subroutine for the Orthogonal Vectors (OV) problem (recall that OV is the problem of deciding whether two lists of $\{0, 1\}$-vectors contain a pair of vectors, one from each list, that are orthogonal). The fast OV subroutine lets us efficiently rule out pseudosolutions while running (an adaptation of) `Meet-in-the-Middle` on a fraction of the enhanced solution space.

In Section 5 we give our fastest algorithm, which uses a delicate combination of the techniques from Sections 3 and 4 to obtain a runtime of $O(2^{n/2} \cdot n^{-0.5023})$ for all $\ell = \text{poly}(n)$. While the runtime improvement over Section 3 is not large, this algorithm demonstrates that by leveraging insights specific to the Subset Sum problem, we can achieve time savings beyond what is possible with more "generic" log shaving techniques. Finally, in Section 6 we briefly discuss directions for future work.

Note that this version of the work is an extended abstract. Readers interested in a more complete presentation may wish to consult the full version.[4]

2 Preliminaries

To ease readability, we adopt the following notational conventions throughout the paper: lowercase Roman letters (ℓ, n, etc.) denote variables; lowercase Greek letters (ε, α, etc.) denote numerical constants; capital Roman letters (L, W, etc.) denote sets, multisets, or lists; and calligraphic capital letters ($\mathcal{W}$, $\mathcal{Q}$, etc.) denote collections of sets of numbers.

Logarithms. When written without a specified base, $\log(\cdot)$ denotes the base-2 logarithm.

Big-O Notation. We augment big-O notation with a tilde ($\widetilde{O}$, $\widetilde{\Omega}$, $\widetilde{\Theta}$ etc.) to suppress $\text{polylog}(\ell) = \text{polylog}(n)$ factors. For example, we have $\widetilde{O}(2^n) = O(2^n \cdot \text{polylog}(n))$ and $\widetilde{\Omega}(n) = \Omega(\frac{n}{\text{polylog}(n)})$.

Probability Notation. Random variables are written in boldface. In particular, we write "$\boldsymbol{x} \sim S$" to indicate that an element $\boldsymbol{x}$ is sampled uniformly at random from a finite multiset S.

[4] `https://arxiv.org/abs/2301.07134`

Set Notation. We write $[a : b]$ for the set of integers $\{a, a+1, \dots, b\}$ and $[a]$ for $\{1, 2, \dots, a\}$. We write $\mathbb{P}[a : b]$ for the set of all primes in the interval $[a : b]$.

Given a multiset or list Y of integers, we adopt several shorthands: $\Sigma(Y) := \sum_{y \in Y} y$ denotes the sum, $W(Y) := \{\Sigma(T) \mid T \subseteq Y\}$ denotes the set of *distinct* sub-multiset sums, and L_Y denotes the list containing the elements of $W(Y)$ sorted in increasing order.

Also, we often write $f(Y)$ for the multiset or list obtained by applying operation f element-wise, such as $Y + \alpha = \{y + \alpha \mid y \in Y\}$ and $\alpha Y = \{\alpha y \mid y \in Y\}$. The only exception is $(Y \bmod p)$, which denotes the set of *distinct* residues $\{(y \bmod p) \mid y \in Y\}$.

Stirling's Approximation for Binomial Coefficients. We use the following well-known consequence of Stirling's approximation (see e.g. [13, Lemma 16.19]): For each integer $j \in [0 : n/2]$,

$$\sum_{i \le j} \binom{n}{i} \;\le\; 2^{H(j/n) \cdot n}, \tag{1}$$

where $H(y) := -y \log(y) - (1 - y) \log(1 - y)$ is the binary entropy function.

Pseudolinear Hashing. Recall that $\ell = \mathrm{poly}(n)$ is the word length in our memory model. Given an integer $m \le \ell$, we write $\boldsymbol{h}_m$ to denote the random hash function defined as

$$\boldsymbol{h}_m(y) \;:=\; (\boldsymbol{u} \cdot y \pmod{2^\ell}) \gg \ell - m. \tag{2}$$

Here the input y is an ℓ-bit integer, $\boldsymbol{u}$ is selected uniformly at random from all odd ℓ-bit integers, and $\gg$ denotes a bit shift to the right, i.e., dividing $\boldsymbol{u} \cdot y \pmod{2^\ell}$ by $2^{\ell - m}$ and then truncating the result so that only the higher-order m bits remain.

This hash function $\boldsymbol{h}_m(y)$ can be evaluated in time $O(\log \ell) = O(\log n)$ for $\ell = \mathrm{poly}(n)$ in the circuit RAM model, i.e., essentially the time to multiply, or constant time $O_n(1)$ in the word RAM model. Further, $\boldsymbol{h}_m$ has the following useful properties [12, 5].

▶ **Lemma 2** (Pseudolinear Hashing [12, 5]). *The following hold for the hash function $\boldsymbol{h}_m$:*

1. ***Pseudolinearity.*** *For any two ℓ-bit integers y, z and any outcome of $\boldsymbol{h}_m$,*

$$\boldsymbol{h}_m(y) + \boldsymbol{h}_m(z) \;\in\; \boldsymbol{h}_m(y + z) - \{0, 1\} \pmod{2^m}.$$

2. ***Pseudouniversality.*** *For any two ℓ-bit integers y, z with $y \neq z$,*

$$\mathbf{Pr}\left[\boldsymbol{h}_m(y) = \boldsymbol{h}_m(z)\right] \;=\; O(2^{-m}).$$

2.1 Preliminary Results

For the purposes of this paper, we may assume that the input target t have size $2^{\Omega(n)}$:

▶ **Observation 3** (Assumptions about Input Instances). *Given a Subset Sum instance (X, t) with $t \le 2^{0.499n}$, the standard dynamic programming algorithm [7] takes time $O(nt) = 2^{0.499n + o(n)}$, much faster than the $\mathrm{poly}(n)$-factor speedups we are targeting, and cannot be improved to time $t^{1-\varepsilon} \cdot 2^{o(n)}$ unless SETH fails [2]. Hence without loss of generality, we assume $t = 2^{\Omega(n)}$ and $\ell = \Omega(\log t) = \Omega(n)$ throughout the paper.*

We further observe that the randomized *decision* algorithms that we give can be converted into randomized *search* algorithms with the same asymptotic runtimes using standard search-to-decision reductions.

Procedure `Sorted-Sum-Enumeration`(Y).

Input: An integer multiset $Y = \{y_1, \dots, y_k\}$.

Output: A sorted list $L_Y := L_k$ of $W(Y)$, all distinct subset sums for Y.

```
Initialize $L_0 := (0)$. Then for each $i \in [k]$:
    Create the sorted list $L'_{i-1} := L_{i-1} + y_i$.
    Create the sorted list $L_i$ obtained by merging $L_{i-1}$, $L'_{i-1}$ and removing duplicates.
```

Figure 1 Efficiently enumerating subset sums of $k \geq 1$ integers.

A basic primitive for our algorithms is the sorted list L_Y containing the elements of $W(Y)$, all *distinct* subset sums, for a given multiset Y of size $|Y| = k$. Figure 1 shows an $O(2^k)$-time folklore algorithm that enumerates L_Y:

▶ **Lemma 4** (Sorted Sum Enumeration; Folklore). `Sorted-Sum-Enumeration` *runs in time* $O(2^k)$.

Proof. We essentially adapt the classic *merge sort* algorithm. Since $|L_{i-1}| = |L'_{i-1}| \leq |L_i| \leq 2|L_{i-1}|$ for each $i \in [k]$, the runtime of `Sorted-Sum-Enumeration` is

$$\textstyle\sum_{i\in[k]} O(|L'_{i-1}| + |L_i|) \;=\; \sum_{i\in[k]} O(|L_i|) \;=\; \sum_{i\in[k]} O(2^i) \;=\; O(2^k). \qquad ◀$$

We can improve this runtime analysis if $|W(Y)|$ is smaller than 2^k by a $\mathrm{poly}(k)$ factor:

▶ **Lemma 5** (Sorted Sum Enumeration for Small $W(Y)$). *If Y is a multiset of $|Y| = k$ integers with $|W(Y)| \leq 2^k \cdot k^{-\varepsilon}$ for some constant $\varepsilon > 0$,* `Sorted-Sum-Enumeration` *runs in time $O(2^k \cdot k^{-\varepsilon} \cdot \log k)$.*

Proof. If $|W(Y)| \leq 2^k \cdot k^{-(1+\varepsilon)}$, then since $|L_i| \leq |L_k| = |W(Y)|$ for each $i \in [k]$, it is easy to check that the algorithm runs in time $O(k \cdot |W(Y)|) = O(2^k \cdot k^{-\varepsilon})$; so suppose that $|W(Y)| \geq 2^k \cdot k^{-(1+\varepsilon)}$. Using the bound $|L_i| \leq 2^i$ for $i \leq \log|W(Y)|$ and the bound $|L_i| \leq |W(Y)|$ for $\log|W(Y)| < i \leq k$, the runtime of `Sorted-Sum-Enumeration` is upper bounded by

$$\begin{aligned} \textstyle\sum_{i\in[k]} O(|L_i|) &= \textstyle\sum_{i\leq\log|W(Y)|} O(2^i) + \sum_{\log|W(Y)|<i\leq k} O(|W(Y)|) \\ &= O(|W(Y)|) + (1+\varepsilon)\cdot\log k \cdot O(|W(Y)|) \\ &= O(|W(Y)|\cdot\log k) \;=\; O(2^k\cdot k^{-\varepsilon}\cdot\log k). \qquad ◀ \end{aligned}$$

▶ Remark 6. Lemma 5 gives a tight analysis of the algorithm when $|W(Y)| = 2^k \cdot k^{-\varepsilon}$, as can be seen by considering the particular size-k multiset $Y = (2^0, 2^1, \dots, 2^{k-\varepsilon\log k-1}, 1, 1, \dots, 1)$.

Finally, Figure 2 shows the classic meet-in-the-middle algorithm for Subset Sum, by Horowitz and Sahni [16], which serves as our baseline for comparison.

▶ **Lemma 7.** *The worst-case runtime of* `Meet-in-the-Middle` *is $O(2^{n/2})$.*

Proof. This follows immediately from Lemma 4 as $|L_A|, |L_B| \leq 2^{n/2}$. ◀

```
Procedure Meet-in-the-Middle($X$, $t$).

Input: An integer multiset $X = \{x_1, x_2, \ldots, x_n\}$ and an integer target $t$.

Output: "yes" if $(X, t)$ is a Subset Sum instance that has a solution, "no" otherwise.

0. Fix any partition of $X = A \cup B$ such that $|A| = |B| = n/2$.
1. Enumerate the sorted lists $L_A$ and $L_B$ using Sorted-Sum-Enumeration.
2. Initialize two pointers at the smallest value in $L_A$ and the largest value in $L_B$.
   If these two values sum to the target $t$, then return "yes";
   if they sum to less than $t$, then increment the pointer into $L_A$ and repeat;
   and if they sum to more than $t$, then increment the pointer into $L_B$ and repeat.
   If either pointer goes past the end of its list, then return "no".
```

Figure 2 The classic `Meet-in-the-Middle` algorithm [16].

3 $\Omega(n^{0.5}/\log n)$-Factor Speedup via Bit Packing

In this section we analyze our first and simplest algorithm, `Bit-Packing` (see Figure 3).[5]

▶ **Theorem 8.** `Bit-Packing` *is a zero-error randomized algorithm for the Subset Sum problem with expected runtime* $O(2^{n/2} \cdot \ell^{-1/2} \cdot \log \ell) \le O(2^{n/2} \cdot n^{-1/2} \cdot \log n)$ *in the circuit RAM model.*[6]

`Bit-Packing` works by packing ℓ/m hashed values into a single word via our pseudolinear hash function $\boldsymbol{h}_m$, for $m = 3 \log \ell$, while preserving enough information to run `Meet-in-the-Middle` on the lists of hashed and packed values. This allows us to compare two length-(ℓ/m) sublists of L_A and L_B in constant expected time in the circuit RAM model, since each hashed and packed sublist fits into a constant number of words, instead of time $O(\ell/m)$ like the original `Meet-in-the-Middle` algorithm. So far, this is essentially the approach taken by [5] in their bit-packing algorithm for 3SUM. However, in our context the $O(\ell/m)$ speedup described above is offset by the following issue: if we follow the original `Meet-in-the-Middle` setup and take $|A| = |B| = \frac{n}{2}$, the lists L_A and L_B may have length $\Omega(2^{n/2})$, increasing the runtime.

To deal with this, we set aside a small set $D \subseteq X$ of $|D| = \log \ell$ many input elements and solve the remaining subinstance $X \setminus D$ for each shifted target $t' \in (t - W(D))$. Removing the elements in D shortens the lists L_A and L_B, which are now formed from the elements in $X \setminus D$, and allows us to enumerate and pack them quickly. Balancing the overhead of solving each of these subinstances against the savings described earlier, we get the claimed speedup $\Omega(\ell^{1/2}/\log \ell) \ge \Omega(n^{1/2}/\log n)$.

Proof of Correctness for `Bit-Packing`. The algorithm outputs "yes" only when a triple $(a, b, t') \in L_A \times L_B \times (t - W(D))$ with $a + b = t'$ is found, so it never returns a false positive.

[5] As explained in Section 1.1, we assume $\ell = \mathrm{poly}(n)$ throughout; however, our results for `Bit-Packing` apply for superpolynomial word length as long as $\ell = O(2^{n/3})$. We leave the extensional modifications to the proofs for this regime as an exercise for the interested reader.

[6] By halting `Bit-Packing` and returning "no" if its runtime exceeds $C \cdot 2^{n/2} \cdot \ell^{-1/2} \cdot \log \ell$ for a large enough constant $C > 0$, we get an one-sided error algorithm with success probability $\ge 3/4$, as claimed in Section 1.2.

```
Procedure Bit-Packing($X$, $t$).

Input: An integer multiset $X = \{x_1, x_2, \ldots, x_n\}$ and an integer target $t$.

Setup: Draw a random hash function $\boldsymbol{h}_m$ with $m = 3\log \ell$ (cf. Equation (2)).

0. Fix any partition of $X = A \cup B \cup D$ such that $|D| = \log \ell$ and $|A| = |B| = \frac{n-|D|}{2}$.
1. Create the set $W(D)$ and the sorted lists $L_A$, $L_B$ using Sorted-Sum-Enumeration.
2. Create lists $\boldsymbol{h}_m(L_A)$ and $\boldsymbol{h}_m(L_B)$ by applying $\boldsymbol{h}_m$ element-wise to $L_A$ and $L_B$.
   Let $\boldsymbol{H}_A$ be the list obtained from $\boldsymbol{h}_m(L_A)$ by packing $(\ell/m)$ elements of $\boldsymbol{h}_m(L_A)$
   into each $\ell$-bit word of $\boldsymbol{H}_A$ (preserving the sorted ordering) and likewise for $\boldsymbol{H}_B$.
3. For each $t' \in (t - W(D))$:
4.     Initialize indices $i := 0$ and $j := |\boldsymbol{H}_B| - 1$. While $i < |\boldsymbol{H}_A|$ and $j \geq 0$:
5.         If the indexed words $\boldsymbol{H}_A[i]$ and $\boldsymbol{H}_B[j]$ contain a pair of hashes
   $(\boldsymbol{h}_m(a'), \boldsymbol{h}_m(b'))$
           such that $\boldsymbol{h}_m(a') + \boldsymbol{h}_m(b') \in \boldsymbol{h}_m(t') - \{0, 1\} \pmod{2^m}$,
           use Meet-in-the-Middle to search for a solution
           $(a, b) \in L_A[i\ell/m : (i+1)\ell/m - 1] \times L_B[j\ell/m : (j+1)\ell/m - 1]$
           such that $a + b = t'$. Halt and return "yes" if a solution is found.
6.         If $L_A[(i+1)\ell/m - 1] + L_B[j\ell/m] < t'$, increment $i \leftarrow i + 1$.
           Otherwise, decrement $j \leftarrow j - 1$.
7. Return "no" (i.e., no solution was found for any $t' \in (t - W(D))$).
```

Figure 3 The Bit-Packing algorithm.

It remains to show that for any $t' \in (t - W(D))$, we are guaranteed to find a shifted solution $(a, b) \in L_A \times L_B$ such that $a + b = t'$, if one exists. Without loss of generality, we consider two sublists $L_A[i\ell/m : (i+1)\ell/m - 1]$ and $L_B[j\ell/m : (j+1)\ell/m - 1]$ that contain such a shifted solution (a, b) and correspond to two packed words $\boldsymbol{H}_A[i]$ and $\boldsymbol{H}_B[j]$ for some indices i and j. The existence of such a shifted solution $a+b = t'$ combined with the condition in Line 6 ensures that the algorithm will not step past either the packed word $\boldsymbol{H}_A[i]$ or $\boldsymbol{H}_B[j]$ before reaching the other one, so the algorithm will compare these two packed words at some point. Following Lemma 2, we have $\boldsymbol{h}_m(a) + \boldsymbol{h}_m(b) \in \boldsymbol{h}_m(t') - \{0, 1\} \pmod{2^m}$, satisfying the condition in Line 5. Thus we are guaranteed to find the shifted solution (a, b) by running Meet-in-the-Middle to check all pairs in $L_A[i\ell/m : (i+1)\ell/m - 1] \times L_B[j\ell/m : (j+1)\ell/m - 1]$. ◀

Proof of Runtime for Bit-Packing. Recalling the assumption $\ell = \mathrm{poly}(n)$:

- Line 1 takes time $O(2^{|A|} + 2^{|B|} + 2^{|D|}) = O(2^{n/2} \cdot \ell^{-1/2} + \ell) = O(2^{n/2} \cdot \ell^{-1/2})$ by Lemma 4, for the choices of $|A|$, $|B|$, and $|D|$.
- Line 2 takes time $(|L_A| + |L_B|) \cdot O(\log \ell) = O(2^{n/2} \cdot \ell^{-1/2} \cdot \log \ell)$, where $O(\log \ell)$ bounds the time for each evaluation of the hash function $\boldsymbol{h}_m$.
- Line 3 (the outer loop) is performed for at most $|W(D)| \leq 2^{|D|} = \ell$ iterations.
- Line 4 (the inner loop) is performed for at most $(|L_A| + |L_B|) \cdot \frac{1}{\ell/m} = O(2^{n/2} \cdot \ell^{-3/2} \cdot \log \ell)$ iterations, since each iteration either increments $i \leftarrow i + 1$ or decrements $j \leftarrow j - 1$.
- Line 5: (i) Checking whether the "If" condition holds for any two words $\boldsymbol{H}_A[i]$ and $\boldsymbol{H}_B[j]$ requires a single AC^0 operation on three words, taking constant time in the circuit RAM model. (ii) Finding a solution (a, b) using Meet-in-the-Middle on the two length-(ℓ/m) sublists takes time $O(\ell/m) = O(\ell/\log \ell)$.

The “If” test is passed (i) at most once for a correct solution and (ii) each time we encounter a hash collision. Note that the sequence of pairs of words $(\boldsymbol{H}_A[i], \boldsymbol{H}_B[j])$ we compare is completely determined by L_A and L_B and is unaffected by the outcome of the random hash function $\boldsymbol{h}_m$. Thus by Lemma 2, each of the $(\ell/m)^2$ hash pairs in $(\boldsymbol{H}_A[i], \boldsymbol{H}_B[j])$ incurs a collision with probability $O(2^{-m})$. By a union bound, the expected time taken for Line 5 because of hash collisions is at most $(\ell/m)^2 \cdot O(2^{-m}) \cdot O(\ell/m) = O(1/\log^3(\ell)) = o_n(1)$, since $m = 3\log \ell$ and $\ell = \Omega(n)$.

- Line 6 clearly takes time $O_n(1)$.

Consequently, Bit-Packing has expected runtime

$$\begin{aligned}
\mathrm{TIME}(n, \ell) &= O(2^{n/2} \cdot \ell^{-1/2} + \ell) + O(2^{n/2} \cdot \ell^{-1/2} \cdot \log \ell) && \text{Lines 1 and 2} \\
&\quad + 1 \cdot \big(O_n(1) + O(\ell/\log \ell) + O_n(1)\big) && \text{Lines 3 to 6} \\
&\quad + \ell \cdot O(2^{n/2} \cdot \ell^{-3/2} \cdot \log \ell) \cdot \big(O_n(1) + o_n(1) + O_n(1)\big) && \text{Lines 3 to 6} \\
&= O(2^{n/2} \cdot \ell^{-1/2} \cdot \log \ell).
\end{aligned}$$ ◀

▶ **Observation 9** (Adapting Bit-Packing to Word RAM). *In the word RAM model, multiplication and evaluation of our pseudolinear hash function $\boldsymbol{h}_m$ each take constant time. Hence, for any word length $\ell = \Omega(n)$ we can get a variant of* Bit-Packing *with expected runtime $O(2^{n/2} \cdot n^{-1/2} \cdot \log n)$, essentially by performing* Bit-Packing *as if the word length were $\ell' := 0.1n$. By a similar conversion from [5]:*

Run Bit-Packing *as if the word length were $\ell' = 0.1n$, which results in the modified parameters $m' = 3\log \ell'$, $|D'| = \log \ell'$, and $|A'| = |B'| = (n - |D'|)/2$ etc., except for two modifications:*

1. *Line 2 packs $q' := \min\{\ell', \ell\}/m' = \Theta(\frac{n}{\log n})$ many m'-bit hashes into each word of $\boldsymbol{H}_{A'}$, $\boldsymbol{H}_{B'}$, so every (ℓ'/m') hashes are stored in $\ell'/(m'q') = \Theta_n(1)$ words rather than a single word.*
2. *Before Line 5, create a table that memoizes the result of every comparison of two ℓ'-bit strings in time $(2^{\ell'})^2 \cdot \mathrm{poly}(\ell') = O(2^{0.21n})$. This table can then be accessed via a $2\ell'/(m'q') = \Theta_n(1)$-word index in constant time. Line 5 replaces the constant-time AC^0 circuit RAM operation on two ℓ'-bit strings with a constant-time lookup into this table.*

Compared with running Bit-Packing *itself for $\ell' = 0.1n$, the only difference of this variant is that $\boldsymbol{H}_{A'}$ and $\boldsymbol{H}_{B'}$ are stored in $\Theta_n(1)$ times as many words, so the correctness is easy to check. The expected time taken for the collisions in each execution of Line 5 is $(q')^2 \cdot O(2^{-m'}) \cdot O(q') = o_n(1)$. Thus, the overall runtime is as claimed:*

$$\underbrace{O(2^{|A'|} + 2^{|B'|} + 2^{|D'|} + 2^{0.21n})}_{\textit{Lines 1 and 2}} + \underbrace{O(q') + 2^{|D'|} \cdot O((2^{|A'|} + 2^{|B'|})/q')}_{\textit{Lines 3 to 6; solution versus collisions}}$$
$$= O(2^{n/2} \cdot n^{-1/2} \cdot \log n).$$

4 $\Omega(n^{0.01})$-Factor Speedup via Orthogonal Vectors and the Representation Method

Our second algorithm, Representation-OV (see Figure 4), achieves a speedup of $\Omega(\ell^\gamma) \geq \Omega(n^\gamma)$ over Meet-in-the-Middle for a constant $\gamma > 0.01$.[7] While this is a smaller speedup than that achieved by the Bit-Packing algorithm, the Representation-OV algorithm does

[7] Similar to Section 3, while we investigate Representation-OV in the regime $\ell = \mathrm{poly}(n)$, analogous results apply for word length ℓ as large as $O(2^{cn})$ for an absolute constant $c > 0$.

not use “bit tricks”. Instead, `Representation-OV` combines `Meet-in-the-Middle` with the *representation method* of Howgrave-Graham and Joux [17], so as to reduce the Subset Sum problem to many small instances of the Orthogonal Vectors (OV) problem: namely, instances with $O(\ell/\log\ell)$ many binary vectors of dimension $\Theta(\log\ell) = \Theta(\log n)$. Such instances of OV can be solved quickly through a single AC^0 word operation in the circuit RAM model (or through constantly many operations in the word RAM model after an initial memoization step), which leads to our speedup.

The high-level idea behind our algorithm is to partition the input $X = A \cup B \cup C$ into two large subsets[8] A and B and one small subset C of size $|C| = \Theta(\log\ell)$, and to run `Meet-in-the-Middle` on two lists formed from subsets of $(A \cup C)$ and $(B \cup C)$. We describe in more detail below just how these lists are formed, but roughly speaking they are created by modifying the representation method (in a way somewhat similar to the algorithm of Nederlof and Węgryzcki [20]) to ensure that the lists are not too long. Further, to eliminate the false positives due to overlapping subsets of C, we exploit a fast implementation of a function that computes a batch of small instances of Orthogonal Vectors. (To see the relevance of the Orthogonal Vectors problem in this context, note that a solution to an instance of OV on k-dimensional Boolean vectors corresponds to two disjoint subsets of the set $[k]$.) Before giving more details we provide some helpful notation:

Notation and setup. We write $\mathcal{Q}(C) := \{T \mid T \subseteq C \text{ and } |T| \leq \frac{|C|}{4}\}$ to denote the collection of all quartersets for C. While $\mathcal{Q}(C)$ is useful for intuition, in fact, as explained below, our algorithm will use a slight variant of it: we define $\mathcal{Q}^{+\varepsilon_2}(C)$ to be the collection of all subsets of size at most $(1+\varepsilon_2)\frac{|C|}{4}$, where $\varepsilon_2 > 0$ is a small constant that is fixed in the detailed description of the algorithm.

We denote by `OV` the Boolean function on $2\ell = \mathrm{poly}(n)$ many input bits that takes as input two lists $(x^1, \ldots, x^{\ell/|C|})$, $(y^1, \ldots, y^{\ell/|C|})$ of (at most) $\ell/|C|$ many binary vectors each, where each binary vector x^i, $y^j \in \{0,1\}^{|C|}$, and returns 1 if and only if the two lists contain an orthogonal pair, i.e., a pair (i, j) such that $x^i_k \cdot y^j_k = 0$ for every $1 \leq k \leq |C|$. It is easy to see that `OV` is an AC^0 operation on two ℓ-bit words, and thus it takes constant time in the circuit RAM model.

We return to the intuitive overview of our approach. At a high level, the representation method works by first expanding the search space of possible solutions; for our algorithm this is done by writing down the list $\mathcal{Q}^{+\varepsilon_2}(C)$ of all “near-quartersets”. If a certain Subset Sum solution $S \subseteq X$ satisfies $|S \cap C| \leq \frac{|C|}{2}$, the restricted solution $S \cap C = Q_1 \cup Q_2$ is the union of *many* different pairs of disjoint quartersets, namely $Q_1, Q_2 \in \mathcal{Q}(C)$ with $Q_1 \cap Q_2 = \emptyset$. In fact, we work on the list of near-quartersets, $\mathcal{Q}^{+\varepsilon_2}(C)$, rather than $\mathcal{Q}(C)$, so as to cover all disjoint pairs (Q_1, Q_2) with $|Q_1| + |Q_2| \approx |C|/2$.

We then *filter* the list $\mathcal{Q}^{+\varepsilon_2}(C)$: given a random prime modulus $\boldsymbol{p}$ we extract those near-quartersets that fall into two particular residue classes that sum to $\Sigma(S \cap C) \pmod{\boldsymbol{p}}$. With appropriate preprocessing checks and parameter settings, this significantly reduces the search space while ensuring that we retain some disjoint pair of near-quartersets Q_1, $Q_2 \in \mathcal{Q}^{+\varepsilon_2}(C)$ that recover the restricted solution, $Q_1 \cup Q_2 = S \cap C$ with $Q_1 \cap Q_2 = \emptyset$.

Finally, we use a modified `Meet-in-the-Middle` procedure to search for a solution, i.e., two sum-subset couples $(a, Q_1) \in (L_A \times \mathcal{Q}^{+\varepsilon_2}(C))$, $(b, Q_2) \times (L_B \times \mathcal{Q}^{+\varepsilon_2}(C))$ with $a + \Sigma(Q_1) + b + \Sigma(Q_2) = \Sigma(S \cap A) + \Sigma(S \cap B) + \Sigma(S \cap C) = \Sigma(S) = t$, verifying $Q_1 \cap Q_2 = \emptyset$ via the Boolean function `OV`.

[8] Technically, sub-multisets. We make the same simplification hereafter.

```
Procedure Representation-OV($X$, $t$).

Input: An integer multiset $X = \{x_1, x_2, \ldots, x_n\}$ and an integer target $t$.

Setup: Constants $\varepsilon_1 \approx 0.1579$ and $\varepsilon_2 \approx 0.2427$, the solutions to Equations (3) and (4).
       Parameters $\beta \approx 1.1186$, $\lambda \approx 0.0202$, $s(\ell)$, and $k(n, \ell)$, all to be specified in the proof.
       A uniform random prime modulus $\boldsymbol{p} \sim \mathbb{P}[\ell^{1+\beta/2} : 2\ell^{1+\beta/2}]$.
0. Fix any partition of $X = A \cup B \cup C$ such that $|C| = \beta \log(\frac{\ell}{\beta \log \ell})$ and
   $|A| = |B| = \frac{n-|C|}{2}$.
1. Use Lemma 13 to solve $(X, t)$ if there exists a solution $S \subseteq X$ with
   $|S \cap C| \notin (1 \pm \varepsilon_1)\frac{|C|}{2}$.
2. Use Lemma 12 to solve $(X, t)$ if $|W(C)| \leq 2^{|C|} \cdot \ell^{-\lambda}$.
3. Create the sorted lists $L_A$ and $L_B$ using Sorted-Sum-Enumeration. Let
   $\{\boldsymbol{L}_{A,\,i}\}_{i \in [\boldsymbol{p}]}$ be the sorted sublists given by $\boldsymbol{L}_{A,\,i} := \{a \in L_A \mid a \equiv_{\boldsymbol{p}} i\}$ and likewise
   for $\{\boldsymbol{L}_{B,\,i}\}_{i \in [\boldsymbol{p}]}$.
   Create the collection $\mathcal{Q}^{+\varepsilon_2}(C) = \{Q \mid Q \subseteq C \text{ and } |Q| \leq (1+\varepsilon_2)\frac{|C|}{4}\}$.
4. Repeat Lines 5 to 6 either $s(\ell)$ times or until a total of $k(n, \ell)$ many sum-subset
   couples have been created in Line 5 (whichever comes first):
5.     Sample a uniform random residue $\boldsymbol{r} \sim [\boldsymbol{p}]$. Then use Lemma 15 and the subroutine
       Residue-Couple-List to create the sorted lists $\boldsymbol{R}_{A,\,\boldsymbol{r}}$ and $\boldsymbol{R}_{B,\,\boldsymbol{r}}$:
       $\boldsymbol{R}_{A,\,\boldsymbol{r}} = \{(a' := a + \Sigma(Q_1),\, Q_1) \mid (a,\, Q_1) \in L_A \times \mathcal{Q}^{+\varepsilon_2}(C) \text{ with } a' \equiv_{\boldsymbol{p}} \boldsymbol{r}\}$ and
       $\boldsymbol{R}_{B,\,\boldsymbol{r}} = \{(b' := b + \Sigma(Q_2),\, Q_2) \mid (b,\, Q_2) \in L_B \times \mathcal{Q}^{+\varepsilon_2}(C) \text{ with } b' \equiv_{\boldsymbol{p}} (t - \boldsymbol{r})\}$.
   ▷ In fact, $\boldsymbol{R}_{A,\,\boldsymbol{r}}$ and $\boldsymbol{R}_{B,\,\boldsymbol{r}}$ are stored in a succinct format by Residue-Couple-List.
6.     Use Lemma 15 (based on Meet-in-the-Middle and the Boolean function OV)
       to search the sorted lists $\boldsymbol{R}_{A,\,\boldsymbol{r}}$ and $\boldsymbol{R}_{B,\,\boldsymbol{r}}$ for a solution $(a',\, Q_1)$, $(b',\, Q_2)$ such that
       $a' + b' = t$ and $Q_1 \cap Q_2 = \emptyset$. Halt and return "yes" if a solution is found.
7. Return "no" (i.e., no solution was found).          ▷ Possibly a false negative.
```

Figure 4 The Representation-OV algorithm.

▶ **Theorem 10.** Representation-OV *is a one-sided error randomized algorithm for the Subset Sum problem (with no false positives) with worst-case runtime* $O(2^{n/2} \cdot \ell^{-\gamma}) \leq O(2^{n/2} \cdot n^{-\gamma})$*, for some constant* $\gamma > 0.01$*, and a success probability of* $1/3$ *in the circuit RAM model.*

The proofs of correctness and runtime below rely on several auxiliary lemmas, namely Lemmas 12–15. These lemmas, along with an adaptation of the result to the Word RAM model, can be found in Appendix A.

Proof of Correctness for Representation-OV. The constants $\varepsilon_1 \approx 0.1579$ and $\varepsilon_2 \approx 0.2427$ given in the description of the algorithm are the solutions to the following equations:

$$\frac{1-H((1-\varepsilon_1)/2)}{(1-\varepsilon_1)/2} \;=\; 1 - H(\tfrac{1-\varepsilon_2}{2}), \tag{3}$$

$$1 + \varepsilon_1 - 3H(\tfrac{1-\varepsilon_1}{2}) \;=\; -2H(\tfrac{1+\varepsilon_2}{4}). \tag{4}$$

Also, we set $\beta := \frac{1}{H((1+\varepsilon_2)/4)} \approx 1.1186$ and $\lambda := (1 - 10^{-5}) \cdot \frac{1-\varepsilon_1}{2} \cdot \beta \cdot (1 - H(\frac{1-\varepsilon_2}{2})) \approx 0.0202$.

Lines 1 and 2 preprocess the instance $(X,\, t)$, solving it deterministically via Lemmas 12 and 13 unless both Conditions (1) and (2) hold:

Condition (1). $(X,\, t)$ is either a "yes" instance with $|S \cap C| \in [(1-\varepsilon_1)\frac{|C|}{2} : \frac{|C|}{2}]$ for each solution $S \subseteq X$,[9] or a "no" instance.

Condition (2). $|W(C)| > 2^{|C|} \cdot \ell^{-\lambda}$. Note that this implies $|W(T)| > 2^{|T|} \cdot \ell^{-\lambda}$ for each $T \subseteq C$.

Lines 3 to 7 accept only if a solution $t = a' + b' = (a + \Sigma(Q_1)) + (b + \Sigma(Q_2))$ is found in Line 6, for which $a \in L_A$, $b \in L_B$, and $Q_1,\, Q_2 \in \mathcal{Q}^{+\varepsilon_2}(C)$ are disjoint. As a consequence, the algorithm never reports false positives.

It remains to show that Lines 4 to 6 accept a "yes" instance $(X,\, t)$ with probability at least $1/3$ when $(X,\, t)$ satisfies Conditions (1) and (2). Consider a solution $S \subseteq X$ and the following set W' containing distinct sums of all "ε_2-balanced" subsets of $S \cap C$:

$$W' \;:=\; \{\, \Sigma(Q) \mid Q \subseteq (S \cap C) \text{ and } |Q| \in (1 \pm \varepsilon_2)\tfrac{|S \cap C|}{2} \,\}.$$

Line 3 creates the residue sublists $L_A = \{\boldsymbol{L}_{A,\,i}\}_{i \in [\boldsymbol{p}]}$ and $L_B = \{\boldsymbol{L}_{B,\,i}\}_{i \in [\boldsymbol{p}]}$. We say that a residue $i \in [\boldsymbol{p}]$ is *good* if it satisfies $i - \Sigma(S \cap A) \in (W' \bmod \boldsymbol{p})$, namely there exists a subset $Q_1 \subseteq (S \cap C)$ of size $|Q_1| \in (1 \pm \varepsilon_2)\frac{|S \cap C|}{2}$ such that $\Sigma(S \cap A) + \Sigma(Q_1) \equiv_{\boldsymbol{p}} i$. On sampling a good residue $\boldsymbol{r} = i$ in Line 5, both Q_1 and $Q_2 := (S \cap C) \setminus Q_1$ are of size at most $(1+\varepsilon_2)\frac{|S \cap C|}{2} \le (1+\varepsilon_2)\frac{|C|}{4}$, so they are included in the collection $\mathcal{Q}^{+\varepsilon_2}(C)$ and, respectively, in the lists $\boldsymbol{R}_{A,\,\boldsymbol{r}}$ and $\boldsymbol{R}_{B,\,\boldsymbol{r}}$ created in Line 5. Then using Lemma 15, we are ensured to find the solution $S = (S \cap A) \cup (S \cap B) \cup (Q_1 \cup Q_2)$.

Hence it suffices to **(i)** lower bound the probability that at least one of the $s(\ell)$ samples $\boldsymbol{r} \sim [\boldsymbol{p}]$ is good, and **(ii)** upper bound the probability that these samples generate a total of $k(n,\, \ell)$ or more sum-subset couples.

(i). We claim that the size of set W' is at least $\Omega(2^{|S \cap C|} \cdot \ell^{-\lambda}) \ge \widetilde{\Omega}(\ell^{(1-\varepsilon_1)\cdot\beta/2-\lambda})$ and is at most $2^{|S \cap C|} \le \ell^{\beta/2}$. The lower bound on the size comes from a combination of two observations. First, the set $(S \cap C)$ has at least $2^{|S \cap C|} \cdot \ell^{-\lambda}$ many distinct subset sums, by Condition (2). Second, the number of subsets $Q \subseteq (S \cap C)$ of size $|Q| \notin (1 \pm \varepsilon_2)\frac{|S \cap C|}{2}$ is at most $2 \cdot 2^{H(\frac{1-\varepsilon_2}{2})\cdot|S \cap C|} = o(2^{|S \cap C|} \cdot \ell^{-\lambda})$, given Stirling's approximation (Equation (1)) and the technical condition

$$\tfrac{|S \cap C|}{\log \ell} \cdot (1 - H(\tfrac{1-\varepsilon_2}{2})) \;\ge\; (1 - o_n(1)) \cdot \tfrac{1-\varepsilon_1}{2} \cdot \beta \cdot (1 - H(\tfrac{1-\varepsilon_2}{2})) \;>\; \lambda,$$

which is true for our choice of λ.

The upper and lower bounds on $|W'|$ allow us to apply Lemma 14: with probability at least $3/4$ over the modulus $\boldsymbol{p} \sim \mathbb{P}[\ell^{1+\beta/2} : 2\ell^{1+\beta/2}]$, there are $|W' \bmod \boldsymbol{p}| = \Omega(|W'|) = \widetilde{\Omega}(\ell^{(1-\varepsilon_1)\cdot\beta/2-\lambda})$ many good residues. Conditioned on this event, taking

$$s(\ell) \;:=\; \widetilde{\Theta}(\ell^{1+\lambda+\varepsilon_1\cdot\beta/2}) \;\ge\; \widetilde{\Omega}\left(\tfrac{\boldsymbol{p}}{|(W' \bmod \boldsymbol{p})|}\right)$$

many samples $\boldsymbol{r} \sim [\boldsymbol{p}]$ yields at least one good residue with probability $\ge 2/3$.

[9] Given Line 1, a solution $S \subseteq X$ or its complementary $(X \setminus S)$, as a solution to the complementary instance $(X,\, \Sigma(X) - t)$, must have this property.

(ii). All candidate lists $\{\boldsymbol{R}_{A,\,i}\}_{i\in[\boldsymbol{p}]}$, $\{\boldsymbol{R}_{B,\,i}\}_{i\in[\boldsymbol{p}]}$ together have a total of

$$(|L_A| + |L_B|)\cdot|\mathcal{Q}^{+\varepsilon_2}(C)| \le 2\cdot 2^{(n-|C|)/2}\cdot 2^{|C|/\beta} = O(2^{n/2}\cdot \ell^{1-\beta/2})$$

sum-subset couples, by construction (Line 3), the choices of $|A|$, $|B|$, $|C|$, and Stirling's approximation (Equation (1)). For any outcome of the random modulus $\boldsymbol{p} = p = \Theta(\ell^{1+\beta/2})$, a number of $s(\ell)$ samples $\boldsymbol{r}\sim[p]$ generate a total of $O(2^{n/2}\cdot\ell^{1-\beta/2})\cdot s(\ell)/p = \widetilde{O}(2^{n/2}\cdot \ell^{-((1-\varepsilon_1/2)\cdot\beta-(1+\lambda))})$ sum-subset couples in expectation. By setting a large enough cutoff

$$k(n,\,\ell) \;:=\; \widetilde{\Theta}(2^{n/2}\cdot\ell^{-((1-\varepsilon_1/2)\cdot\beta-(1+\lambda))}),$$

the probability that $k(n,\,\ell)$ or more sum-subset couples are generated is at most $1/6$.

Overall, our algorithm succeeds with probability $\ge (3/4)\cdot(2/3)-(1/6) = 1/3$. ◀

Proof of Runtime for `Representation-OV`. Recalling the assumption $\ell = \mathrm{poly}(n)$:

- Line 1 takes time $\widetilde{O}(2^{n/2}\cdot\ell^{-(1-H((1-\varepsilon_1)/2))\cdot\beta/2})$, by Lemma 13.
- Line 2 takes time $\widetilde{O}(2^{n/2}\cdot\ell^{-\lambda/2})$, by Lemma 12.
- Line 3 takes time $O(2^{|A|}+2^{|B|}+2^{|C|})\cdot O(\log\ell) = \widetilde{O}(2^{n/2}\cdot\ell^{-\beta/2})$, by Lemma 4, the choices of $|A|$, $|B|$, $|C|$, and that the modulo operation in the circuit RAM model takes time $O(\log\ell)$.
- Lines 4 to 6 take time $\widetilde{O}(k(n,\,\ell)) = \widetilde{O}(2^{n/2}\cdot\ell^{-((1-\varepsilon_1/2)\cdot\beta-(1+\lambda))})$, because a single iteration (Lemma 15) takes time $\widetilde{O}(|\boldsymbol{R}_{A,\,\boldsymbol{r}}|+|\boldsymbol{R}_{B,\,\boldsymbol{r}}|)$ and by construction we create a total of at most $\sum_{\boldsymbol{r}}(|\boldsymbol{R}_{A,\,\boldsymbol{r}}|+|\boldsymbol{R}_{B,\,\boldsymbol{r}}|)\le k(n,\,\ell)$ many sum-subset couples.

The runtime of Line 3 is dominated by that of Line 1, so the bottleneck occurs in Line 1, Line 2, or Lines 4 to 6. For the choices of constants given in the algorithm, we achieve a speedup of $\Omega(\ell^{\gamma})$ for any constant $\gamma\in(0,\,\gamma_*)$, where the

$$\gamma_* \;:=\; \min\Big\{\lambda/2,\quad (1-H(\tfrac{1-\varepsilon_1}{2}))\cdot\beta/2,\quad (1-\varepsilon_1/2)\cdot\beta-(1+\lambda)\Big\} \;=\; \lambda/2 \;\approx\; 0.0101. \blacktriangleleft$$

5 Subset Sum in Time $O(2^{n/2}\cdot n^{-0.5023})$

Our last algorithm is a delicate combination of `Bit-Packing` and `Representation-OV`. Our main result, Theorem 11, demonstrates that problem-specific features of Subset Sum can be exploited to obtain an additional nontrivial time savings beyond what can be achieved by augmenting `Meet-in-the-Middle` with generic bit-packing tricks. Although the bookkeeping to analyze the algorithm of this section is somewhat intricate, many of the ideas in this section are previewed in Section 4.

▶ **Theorem 11.** `Packed-Representation-OV`*is a one-sided error randomized algorithm for the Subset Sum problem (with no false positives) with worst-case runtime* $O(2^{n/2}\cdot n^{-(1/2+\gamma)})$*, for some constant* $\gamma > 0.0023$*, and success probability at least* $1/12$ *in the circuit RAM model.*

This extended abstract presentation omits the proof of Theorem 11 and the adaptation of Theorem 11 to the word RAM model. For the proof, the reader is referred to the full version of the paper.[10] Below, we describe a difficulty that arises in the attempt to directly combine the two building block algorithms, as well as the new AC^0 operation used in the approach.

First, we describe a difficulty that arises when attempting to combine our two previous algorithms. `Bit-Packing` saves time by removing a subset $D\subseteq X$ from the input, then running a `Meet-in-the-Middle` variant on the resulting subinstance multiple times, while

[10] `https://arxiv.org/abs/2301.07134`

`Representation-OV` runs a `Meet-in-the-Middle` variant on multiple subinstances indexed by residue classes modulo a random prime $\boldsymbol{p}$. This presents a problem: to get the time savings from bit packing, we would like to reuse subinstances multiple times, but to get the time savings from the representation method we need to build separate subinstances with respect to each residue class $\pmod{\boldsymbol{p}}$ that contains elements of $W(D)$. To solve this problem, we construct D in a way that ensures the elements of $W(D)$ fall into few residue classes $\pmod{\boldsymbol{p}}$. Specifically, we fix $\boldsymbol{p}$ and consider two cases. In one case , the elements of X fall into few residue classes $\pmod{\boldsymbol{p}}$ and it is possible to choose a small set D such that the elements of D (and $W(D)$) fall into very few residue classes $\pmod{\boldsymbol{p}}$. In this case we need to construct just $|W(D) \bmod \boldsymbol{p}| = \mathrm{polylog}(n)$ distinct subinstances. In the other case , the elements of X fall into many residue classes $\pmod{\boldsymbol{p}}$. Here a carefully selected D satisfies the weaker bound $|W(D) \bmod \boldsymbol{p}| = \widetilde{O}(n^{\delta})$ for a small constant $\delta > 0$, increasing the number of subinstances we need to construct. However, the fact that the elements of X fall into many residue classes $\pmod{\boldsymbol{p}}$ lets us select a larger set C such that the subset sums in $W(S \cap C)$ distribute well $\pmod{\boldsymbol{p}}$ for any solution S, offsetting the increase in runtime.

Like the AC^0 operation `OV` in the `Representation-OV` algorithm, the core of our new algorithm is another AC^0 operation, `Hash-OV`, that solves Orthogonal Vectors on small instances. Like `OV`, `Hash-OV` takes as input two ℓ-bit words containing multiple bit vectors of length $O(\log n)$, but now the bit vectors in either word may come from two or more lists, and each list is indexed by the m-bit hash $\boldsymbol{h}_m(s)$ of a corresponding subset sum s, for $m = 3\log \ell$. Thus `Hash-OV` may solve multiple small Orthogonal Vectors instances at once.

6 Extensions and Future Work

Our results open up several natural directions for future investigation:

- **Derandomization:** All of the $2^{n/2}/\mathrm{poly}(n)$-time algorithms we have given for Subset Sum use randomness. Can our results be extended to achieve deterministic algorithms with worst-case running time $2^{n/2}/\mathrm{poly}(n)$?
- **Counting:** It is straightforward to modify the `Meet-in-the-Middle` algorithm to output a count of the number of Subset Sum solutions in time $O(2^{n/2})$ (essentially, by keeping track of the multiplicity with which each value occurs in each list L_A, L_B). Can our techniques be extended to give counting algorithms for Subset Sum that run in time $2^{n/2}/\mathrm{poly}(n)$? In time $2^{n/2}/n^{0.501}$?
- **Faster runtimes:** Finally, an obvious goal is to quantitatively strengthen our results by developing faster algorithms for worst-case Subset Sum. It would be particularly interesting to achieve running times of the form $2^{n/2}/f(n)$ for some $f(n) = n^{\omega(1)}$.

References

1 Amir Abboud and Karl Bringmann. Tighter Connections Between Formula-SAT and Shaving Logs. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 8:1–8:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

2 Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. Seth-based lower bounds for subset sum and bicriteria path. *ACM Transactions on Algorithms (TALG)*, 18(1):1–22, 2022.

3 Per Austrin, Petteri Kaski, Mikko Koivisto, and Jesper Nederlof. Subset sum in the absence of concentration. In *32nd International Symposium on Theoretical Aspects of Computer Science (STACS 2015)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2015.

4 Per Austrin, Mikko Koivisto, Petteri Kaski, and Jesper Nederlof. Dense subset sum may be the hardest. *33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016)*, pages 13:1–13:14, 2016.

5 Ilya Baran, Erik D Demaine, and Mihai Patraşcu. Subquadratic algorithms for 3SUM. In *Workshop on Algorithms and Data Structures*, pages 409–421. Springer, 2005.

6 Anja Becker, Jean-Sébastien Coron, and Antoine Joux. Improved generic algorithms for hard knapsacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 364–385. Springer, 2011.

7 Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

8 Timothy Chan. The art of shaving logs. Presentation at WADS 2013, slides available at https://tmc.web.engr.illinois.edu/talks/wads13_talk.pdf, 2013.

9 Timothy M. Chan. The art of shaving logs. In Frank Dehne, Roberto Solis-Oba, and Jörg-Rüdiger Sack, editors, *Algorithms and Data Structures – 13th International Symposium, WADS 2013, London, ON, Canada, August 12-14, 2013. Proceedings*, volume 8037 of *Lecture Notes in Computer Science*, page 231. Springer, 2013.

10 Xi Chen, Yaonan Jin, Tim Randolph, and Rocco A Servedio. Average-case subset balancing problems. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 743–778. SIAM, 2022.

11 Marek Cygan, Fedor Fomin, Bart M.P. Jansen, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Open problems for fpt school. Available at https://fptschool.mimuw.edu.pl/opl.pdf, 2014.

12 Martin Dietzfelbinger, Torben Hagerup, Jyrki Katajainen, and Martti Penttonen. A reliable randomized algorithm for the closest-pair problem. *Journal of Algorithms*, 25(1):19–51, 1997.

13 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. `doi:10.1007/3-540-29953-X`.

14 Allan Grønlund and Seth Pettie. Threesomes, degenerates, and love triangles. *J. ACM*, 65(4):22:1–22:25, 2018. `doi:10.1145/3185378`.

15 Godfrey Harold Hardy, Edward Maitland Wright, et al. *An introduction to the theory of numbers*. Oxford university press, 1979.

16 Ellis Horowitz and Sartaj Sahni. Computing partitions with applications to the knapsack problem. *Journal of the ACM (JACM)*, 21(2):277–292, 1974.

17 Nick Howgrave-Graham and Antoine Joux. New generic algorithms for hard knapsacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 235–256. Springer, 2010.

18 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, pages 85–103. Plenum Press, New York, 1972.

19 Marcin Mucha, Jesper Nederlof, Jakub Pawlewicz, and Karol Wegrzycki. Equal-subset-sum faster than the meet-in-the-middle. 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 73:1–73:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. `doi:10.4230/LIPIcs.ESA.2019.73`.

20 Jesper Nederlof and Karol Wegrzycki. Improving Schroeppel and Shamir's algorithm for subset sum via orthogonal vectors. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1670–1683. ACM, 2021.

21 David Pisinger. Dynamic programming on the word RAM. *Algorithmica*, 35(2):128–145, 2003.

22 Richard Schroeppel and Adi Shamir. A $T = O(2^{n/2}), S = O(2^{n/4})$ algorithm for certain NP-complete problems. *SIAM journal on Computing*, 10(3):456–464, 1981.

23 Gerhard J Woeginger. Open problems around exact algorithms. *Discrete Applied Mathematics*, 156(3):397–405, 2008.

A Auxiliary Lemmas for Representation-OV

A first “preprocessing lemma” stems from the observation that a not-too-large subset $Y \subseteq X$ with few distinct subset sums (i.e., $|W(Y)| \ll 2^{|Y|}$) can help speed up Meet-in-the-Middle. This idea goes back to [3, 4], although the condition of “few distinct subset sums” we use refers to sets with polynomially rather than exponentially fewer subset sums than the maximum possible number.

▶ **Lemma 12** (Speedup via Additive Structure). *Let $(X,\, t)$ be an n-integer Subset Sum instance. Given as input $(X,\, t)$ and a subset $Y \subseteq X$ of size $|Y| \le n/2$ such that $|W(Y)| \le 2^{|Y|}\cdot\ell^{-\varepsilon}$ for some constant $\varepsilon > 0$, the instance $(X,\, t)$ can be solved deterministically in time $\widetilde{O}(2^{n/2}\cdot\ell^{-\varepsilon/2})$.*

Proof. Fix any size-$(\frac{n+\varepsilon\log\ell}{2})$ subset A such that $Y \subseteq A \subseteq X$; we have $|W(A)| \le 2^{|A\setminus Y|}\cdot|W(Y)| \le 2^{n/2}\cdot\ell^{-\varepsilon/2}$. We also have $|W(X\setminus A)| \le 2^{|X\setminus A|} = 2^{n/2}\cdot\ell^{-\varepsilon/2}$. By Lemmas 4 and 5, it takes overall time $O(2^{n/2}\cdot\ell^{-\varepsilon/2}\cdot\log n)$ in the regime $\ell = \mathrm{poly}(n)$ to create the sorted lists L_A, $L_{X\setminus A}$ and to run Meet-in-the-Middle. ◀

Another useful preprocessing lemma shows that the existence of a solution that is “unbalanced” vis-a-vis a given small subset yields a speedup:

▶ **Lemma 13** (Speedup via Unbalanced Solutions). *Let $(X,\, t)$ be an n-integer Subset Sum instance that has a solution. Given as input $(X,\, t)$ and a subset $Y \subseteq X$ of size $|Y| = \beta\log(\ell/(\beta\log\ell))$ for β as specified in* Representation-OV *such that some solution $S \subseteq X$ satisfies $|S\cap Y| \notin (1\pm\varepsilon)\frac{|Y|}{2}$ for some constant $\varepsilon > 0$, the solution S can be found deterministically in time $\widetilde{O}(2^{n/2}\cdot\ell^{-\delta/2})$, where the constant $\delta := (1 - H(\frac{1-\varepsilon}{2}))\cdot\beta$.*

Proof. We can assume without loss of generality that the solution S satisfies $|S\cap Y| \le \frac{|Y|}{2}$ (since either the original instance $(X,\, t)$ or the complementary instance $(X,\, \Sigma(X) - t)$ must satisfy this property, and we can attempt both instances and only double the runtime). Hence we can suppose $|S\cap Y| \le (1-\varepsilon)\frac{|Y|}{2}$. Then the sorted list L'_Y of the set $\{\Sigma(T) \mid T \subseteq Y \text{ and } |T| \le (1-\varepsilon)\frac{|Y|}{2}\}$ can be created in time $O(2^{|Y|}) = O(\ell^{\beta}) = \mathrm{poly}(n)$ using Sorted-Sum-Enumeration (restricted to subsets of sizes $< (1-\varepsilon)\frac{|Y|}{2}$).

Fix any size-$(\frac{n+(\delta/\beta)|Y|}{2})$ subset A with $Y \subseteq A \subseteq X$. In the regime $\ell = \mathrm{poly}(n)$, we can use L'_Y and Lemma 5 to create the sorted list L'_A of $\{\Sigma(T) \mid T \subseteq A \text{ and } |T\cap Y| < (1-\varepsilon)\frac{|Y|}{2}\}$ in time

$$\widetilde{O}(|L'_A|) \le \widetilde{O}(2^{|A\setminus Y|}\cdot|L'_Y|) \le \widetilde{O}(2^{|A\setminus Y|}\cdot 2^{H(\frac{1-\varepsilon}{2})\cdot|Y|}) \le \widetilde{O}(2^{n/2}\cdot\ell^{-\delta/2}),$$

following Stirling’s approximation (Equation (1)) and the choices of $|Y|$ and $|A|$. Moreover, we can use Lemma 4 to create the sorted list $L_{X\setminus A}$ in time $O(2^{|X\setminus A|}) = \widetilde{O}(2^{n/2}\cdot\ell^{-\delta/2})$.

Provided $|S\cap Y| < (1-\varepsilon)\frac{|Y|}{2}$, running Meet-in-the-Middle on L'_A and $L_{X\setminus A}$ solves (X,t) and takes time $O(|L'_A| + |L_{X\setminus A}|) = \widetilde{O}(2^{n/2}\cdot\ell^{-\delta/2})$. The overall runtime is $\widetilde{O}(2^{n/2}\cdot\ell^{-\delta/2})$. ◀

The next lemma specifies a parameter space within which any set of distinct integers is likely to fall into many residue classes modulo a random prime. This allows us to reduce the search space by considering only solutions that fall into certain residue classes.

▶ **Lemma 14** (Distribution of Integer Sets modulo Random Primes). *Fix a set Y of at most $|Y| \le \ell^{\beta/2}$ distinct ℓ-bit integers for β as specified in* Representation-OV*. For a uniform random modulus $\boldsymbol{p} \sim \mathbb{P}[\ell^{1+\beta/2} : 2\ell^{1+\beta/2}]$, the residue set has size $|(Y \bmod \boldsymbol{p})| = \Theta(|Y|)$ with probability at least $3/4$.*

```
Subroutine Residue-Couple-List($\{L_{A,\,i}\}_{i \in [p]}$, $\mathcal{Q}^{+\varepsilon}(C)$, $r$).

Input: A collection of $p = \mathrm{poly}(\ell)$ sorted sublists $L_A = \bigcup_{i \in [p]} L_{A,\,i}$, for some subset
$A \subseteq X$, indexed by residue class modulo $p$.

Output: A sorted sum-subset list $R_{A,\,r}$ with elements in the sum-collection format
$(a',\, \mathcal{Q}_{a'})$.

1. For each $Q \in \mathcal{Q}^{+\varepsilon}(C)$, create the sum-subset sublist $R_Q := f_Q(L_{A,\,j(Q)})$ by applying
   $f_Q$, the element-to-couple operation $a \mapsto (a' := a + \Sigma(Q),\, Q)$, to the particular input
   sublist of index $j(Q) := ((r - \Sigma(Q)) \bmod p)$.   ▷ Each $R_Q$ is sorted by the sums $a'$.
2. Let $R_{A,\,r}$ be the list obtained by merging $\{R_Q\}_{Q \in \mathcal{Q}^{+\varepsilon}(C)}$, sorted by sums
   $a' = a + \Sigma(Q)$.
   For each distinct sum $a'$, compress all couples $(a',\, Q_1)$, $(a',\, Q_2)$, ... with the same
   first element $= a'$ into a single data object $(a',\, \mathcal{Q}_{a'} := \{Q_1, Q_2, \dots\})$.
           ▷ Note that for each sum $a'$ we have $a' \equiv_p r$ and $|\mathcal{Q}_{a'}| \le |\mathcal{Q}^{+\varepsilon}(C)|$.
```

Figure 5 The Residue-Couple-List subroutine.

Proof. By the prime number theorem [15, Equation (22.19.3)], there are at least $\frac{\ell^{1+\beta/2}}{(1+\beta/2)\cdot\log \ell}$ primes in $\mathbb{P}[\ell^{1+\beta/2} : 2\ell^{1+\beta/2}]$ (for any sufficiently large ℓ). Given any two distinct integers $y \neq z \in Y$, the difference $|y - z| \le 2^\ell$ has at most $\frac{\ell}{(1+\beta/2)\cdot\log \ell}$ distinct prime factors in $\mathbb{P}[\ell^{1+\beta/2} : 2\ell^{1+\beta/2}]$. Thus under a uniform random choice of modulus $\boldsymbol{p} \sim \mathbb{P}[\ell^{1+\beta/2} : 2\ell^{1+\beta/2}]$, the second frequency moment $\boldsymbol{f}_{(2)} := |\{(y,\, z) \mid y,\, z \in Y \text{ with } y \equiv_{\boldsymbol{p}} z\}|$ has the expectation

$$\mathbf{E}_{\boldsymbol{p}}[\boldsymbol{f}_{(2)}] = |\{y = z \in Y\}| + |\{y \neq z \in Y\}| \cdot \ell^{-\beta/2} \;=\; |Y| + (|Y|^2 - |Y|) \cdot \ell^{-\beta/2} \;\le\; 2|Y|.$$

Therefore, with an arbitrarily high constant probability, we have $\boldsymbol{f}_{(2)} = O(|Y|)$ and, by the Cauchy-Schwarz inequality $|Y \bmod \boldsymbol{p}| \cdot \boldsymbol{f}_{(2)} \ge |Y|^2$,

a residue set of size $|Y \bmod \boldsymbol{p}| = \Omega(|Y|)$. ◀

▶ **Lemma 15** (Sorted Lists $\boldsymbol{R}_{A,\,\boldsymbol{r}}$ and $\boldsymbol{R}_{B,\,\boldsymbol{r}}$; Lines 5 and 6). *In* Representation-OV*:*
(i) *Line 5 uses the subroutine* Residue-Couple-List *to create the sorted lists $\boldsymbol{R}_{A,\,\boldsymbol{r}}$, $\boldsymbol{R}_{B,\,\boldsymbol{r}}$ in time $\widetilde{O}(|\boldsymbol{R}_{A,\,\boldsymbol{r}}| + |\boldsymbol{R}_{B,\,\boldsymbol{r}}|)$. Moreover,* **(ii)** *Line 6 finds a solution pair $(a',\, Q_1) \in \boldsymbol{R}_{A,\,\boldsymbol{r}}$, $(b',\, Q_2) \in \boldsymbol{R}_{B,\,\boldsymbol{r}}$, if one exists, in time $O(|\boldsymbol{R}_{A,\,\boldsymbol{r}}| + |\boldsymbol{R}_{B,\,\boldsymbol{r}}|)$.*

Proof. Line 5 creates $\boldsymbol{R}_{A,\,\boldsymbol{r}}$ and $\boldsymbol{R}_{B,\,\boldsymbol{r}}$ using the subroutine Residue-Couple-List (see Figure 5). First, we claim that Residue-Couple-List takes time $\widetilde{O}(|\boldsymbol{R}_A|)$ in the regime $\ell = \mathrm{poly}(n)$:

- Line 1 takes time $\Sigma_{Q \in \mathcal{Q}^{+\varepsilon_2}(C)} O(|\boldsymbol{R}_Q|) = O(|\boldsymbol{R}_{A,\,\boldsymbol{r}}|)$, since all shifts $\Sigma(Q)$ for $Q \in \mathcal{Q}^{+\varepsilon_2}(C)$ can be precomputed and memoized when the collection $\mathcal{Q}^{+\varepsilon_2}(C)$ is created in Line 3.
- Line 2 builds one sorted list $\boldsymbol{R}_{A,\,\boldsymbol{r}}$ from $|\mathcal{Q}^{+\varepsilon_2}(C)| \le 2^{|C|/\beta} = \ell/(\beta \log \ell) = \mathrm{poly}(n)$ sorted sublists $\{\boldsymbol{R}_Q\}_{Q \in \mathcal{Q}^{+\varepsilon_2}(C)}$, taking time $O(|\boldsymbol{R}_{A,\,\boldsymbol{r}}| \cdot \log n)$ via the classic *merge sort* algorithm.

After Line 5 creates the sorted lists $\boldsymbol{R}_{A,\,\boldsymbol{r}} = \{(a',\, \boldsymbol{\mathcal{Q}}_{a'})\}$ and $\boldsymbol{R}_{B,\,\boldsymbol{r}} = \{(b',\, \boldsymbol{\mathcal{Q}}_{b'})\}$, Line 6 can run Meet-in-the-Middle based on the (ordered) indices a' and b' in time $O(|\boldsymbol{R}_{A,\,\boldsymbol{r}}| + |\boldsymbol{R}_{B,\,\boldsymbol{r}}|)$. This ensures that we discover every pair $(a',\, \boldsymbol{\mathcal{Q}}_{a'})$, $(b',\, \boldsymbol{\mathcal{Q}}_{b'})$ such that $a' + b' = t$.

Such a pair yields a solution if and only if it contains two disjoint near-quartersets $Q_1 \in \boldsymbol{\mathcal{Q}}_{a'}$, $Q_2 \in \boldsymbol{\mathcal{Q}}_{b'}$ with $Q_1 \cap Q_2 = \emptyset$, which we can check in constant time by one call of the Boolean function `OV`. Namely, each near-quarterset $Q \in \mathcal{Q}^{+\varepsilon_2}(C)$ is stored in $|C| < \beta \log \ell$ bits, so each collection $\boldsymbol{\mathcal{Q}}_{a'}$, $\boldsymbol{\mathcal{Q}}_{b'} \subseteq \mathcal{Q}^{+\varepsilon_2}(C)$ can be stored a single word $|C| \cdot |\mathcal{Q}^{+\varepsilon_2}(C)| \leq \ell$. Thus one call of `OV` suffices to check a given pair $(a', \boldsymbol{\mathcal{Q}}_{a'})$, $(b', \boldsymbol{\mathcal{Q}}_{b'})$. Overall, Line 6 takes time $O(|\boldsymbol{R}_{A,\boldsymbol{r}}| + |\boldsymbol{R}_{B,\boldsymbol{r}}|)$. ◀

▶ **Observation 16** (Adapting `Representation-OV` to Word RAM). *In the word RAM model, it may take superconstant time to evaluate the Boolean function* `OV`*. Similar to the strategy used in Section 3, our word RAM variant avoids this issue by performing* `Representation-OV` *as if the word length were $\ell' := 0.1n$ (using sets A', B', C', etc., with appropriate size modifications), except for three modifications:*

1. *In lines 3 and 2, on creating a (sub)collection $\mathcal{Q} \subseteq \mathcal{Q}^{+\varepsilon_2}(C')$ as a bit string, store it in at most $\leq \lceil |\mathcal{Q}^{+\varepsilon_2}(C')| \cdot |C'|/\ell \rceil \leq \lceil \ell'/\ell \rceil = \Theta(1)$ words (since a single word with ℓ bits may be insufficient).*
2. *In Line 3, after creating the collection $\mathcal{Q}^{+\varepsilon_2}(C')$, create a lookup table* `OV'` *that memoizes the input-output result of the Boolean function* `OV` *on each subcollection pair $\mathcal{Q}_{a'}$, $\mathcal{Q}_{b'} \subseteq \mathcal{Q}^{+\varepsilon_2}(C')$ in time $(2^{|\mathcal{Q}^{+\varepsilon_2}(C')|})^2 \cdot \mathrm{poly}(|\mathcal{Q}^{+\varepsilon_2}(C')|) \leq (2^{\ell'})^2 \cdot \mathrm{poly}(\ell') = O(2^{0.21n})$. This table can then be accessed using a $2\lceil \ell'/\ell \rceil = \Theta(1)$-word index in constant time.*
3. *Line 6 replaces the Boolean function* `OV` *(namely a constant-time AC^0 circuit RAM operation) with constant-time lookup into* `OV'`*.*

Compared with running `Representation-OV` *itself for $\ell' = 0.1n$, the only difference of this variant is that $\boldsymbol{R}_{A,\boldsymbol{r}}$ and $\boldsymbol{R}_{B,\boldsymbol{r}}$ are stored in $\lceil \ell'/\ell \rceil = \Theta(1)$ times as many words, given $\ell = \Omega(n)$. Hence, it is easy to check the correctness and the runtime $O(2^{n/2} \cdot \ell'^{-\gamma} \cdot \lceil \ell'/\ell \rceil) = O(2^{n/2} \cdot n^{-\gamma})$, for the same constant $\gamma > 0.01$.*

On Optimization and Counting of Non-Broken Bases of Matroids

Dorna Abdolazimi ✉
University of Washington, Seattle, WA, USA

Kasper Lindberg ✉
University of Washington, Seattle, WA, USA

Shayan Oveis Gharan ✉
University of Washington, Seattle, WA, USA

Abstract

Given a matroid $M = (E, \mathcal{I})$, and a total ordering over the elements E, a broken circuit is a circuit where the smallest element is removed and an NBC independent set is an independent set in $\mathcal{I}$ with no broken circuit. The set of NBC independent sets of any matroid M define a simplicial complex called the broken circuit complex which has been the subject of intense study in combinatorics. Recently, Adiprasito, Huh and Katz showed that the face of numbers of any broken circuit complex form a log-concave sequence, proving a long-standing conjecture of Rota.

We study counting and optimization problems on NBC bases of a generic matroid. We find several fundamental differences with the independent set complex: for example, we show that it is NP-hard to find the max-weight NBC base of a matroid or that the convex hull of NBC bases of a matroid has edges of arbitrary large length. We also give evidence that the natural down-up walk on the space of NBC bases of a matroid may not mix rapidly by showing that for some family of matroids it is NP-hard to count the number of NBC bases after certain conditionings.

2012 ACM Subject Classification Theory of computation → Random walks and Markov chains; Theory of computation → Computational complexity and cryptography

Keywords and phrases Complexity, Hardness, Optimization, Counting, Random walk, Local to Global, Matroids

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.40

Category RANDOM

Funding *Dorna Abdolazimi*: Research supported by NSF grant CCF-2203541, and Air Force Office of Scientific Research grant FA9550-20-1-0212.
Kasper Lindberg: Research supported by NSF grant CCF-2203541.
Shayan Oveis Gharan: Research supported by NSF grant CCF-2203541, Air Force Office of Scientific Research grant FA9550-20-1-0212, and Simons Investigator grant.

1 Introduction

A matroid $M = (E, \mathcal{I})$ is consists of a finite ground set E and a collection $\mathcal{I}$ of subsets of E, called independent sets, satisfying:

- **Downward closure:** If $S \subseteq T$ and $T \in \mathcal{I}$, then $S \in \mathcal{I}$.
- **Exchange axiom:** If $S, T \in \mathcal{I}$ and $|T| > |S|$, then there exists an element $i \in T \setminus S$ such that $S \cup \{i\} \in \mathcal{I}$.

The rank of a set $S \subseteq E$ is the size of the largest independent set contained in S. All maximal independent sets of M, called the bases of M, have the same size r, which is called the rank of M.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 40; pp. 40:1–40:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Sampling and counting problems on matroids have captured the interest of many researchers for several decades with applications to reliability [6], liquidity of markets [19], etc. A recent breakthrough in this field proved that the down-up walk on the bases of a matroid mixes rapidly to the (uniform) stationary distribution and can be used to count the number of bases of a matroid [3, 7], resolving the conjecture of Mihail and Vazirani from 1989 [18]. The down-up walk is easy to describe: Start with an arbitrary base B and repeatedly execute the following two steps:

1. Choose a uniformly random element $i \in B$ and delete it.
2. Among all bases (of M) that contain $B \smallsetminus \{i\}$, choose one uniformly at random.

A central question that has puzzled researchers since then is sampling a non-broken (circuit) basis (NBC basis) of a matroid [4].

A set $C \subseteq E$ is a *circuit* iff $C \setminus \{e\} \in \mathcal{I}$ for any $e \in C$. A *broken circuit* (with respect to a total ordering $\mathcal{O}$) is a set $C \setminus \{e\}$, where $C \subseteq E$ is a circuit and e is the **smallest** element of C with respect to $\mathcal{O}$. An independent set $S \subseteq E$ is a *non-broken* independent set (NBC independent set) if it contains no broken circuits. The NBC independent sets are closely related to several interesting combinatorial objects. The number of NBC independent sets of size k in a graphic matroid is equal to the absolute value of the $(n-1)-k$-th coefficient of the chromatic polynomial of the underlying graph where n is the number of vertices. As a corollary the following facts hold:

► **Fact 1.** *The following facts are well-known about the counts of NBC bases/independent sets of different family of matroids.*

- *The number of all NBC independent sets of a graphic matroid is equal to the the number of acyclic orientations of the graph [22].*
- *The number of all NBC independent sets of a co-graphic matroid is equal to the number of strongly connected orientations of the graph (see e.g., [13]).*
- *The number of non-broken spanning trees of a graph is equal to the number of parking functions with respect to a unique source vertex [4]*
- *The number of NBC independent of sets of linear matroid with vectors $v_1, \dots, v_n$ is equal to the number of regions defined by the intersection of the orthogonal hyperplanes (see e.g., [23]).*

We emphasize that although the set of NBC independent sets/bases of a matroid are functions of the underlying total order $\mathcal{O}$, the counts of the number NBC independent sets of rank k for any $0 \leq k \leq r$ are invariant under $\mathcal{O}$ [23]. We remark that, to the best of our knowledge as of this date, none of the above counting problems are known to be computationally tractable.

Given a matroid M with an arbitrary total ordering $\mathcal{O}$, one can analogously run the down-up walk only on the NBC bases of M. It is not hard to see that this chain is irreducible and converges to the uniform stationary distribution. Following the work of [3] it was conjectured that the down-up walk on the NBC bases of any matroid mixes rapidly [1].

► **Conjecture 2.** *For any matroid M, and any total ordering $\mathcal{O}$ of the elements of M, the down-up walk on the NBC bases of a matroid mixes in polynomial time.*

[1] In fact, this conjecture was raised an an open problem in several recent workshops UC Santa Barbara workshop on New tools for Optimal Mixing of Markov Chains: Spectral Independence and Entropy Decay, and Simon's workshop on Geometry of Polynomials

It turns out that the above conjecture, if true, would be give a generalization of the result of [3], because of the following fact.

▶ **Fact 3** ([5])**.** *For any matroid M one can construct another matroid M' with an ordering $\mathcal{O}$ with only one extra element such that there is a bijection between bases of M and non-broken bases of M'.*

Furthermore, if the above conjecture is true, then since matroids are closed under truncation, one can also count the number of all NBC independent sets of M, thus resolving all of the open problems in Fact 1.

A promising reason to expect these problems to be tractable in the first place is the remarkable work of Adiprasito, Huh and Katz [1] who proved the Rota's conjecture showing that the face numbers of a broken circuit complex (see below for definition) of any matroid forms a log-concave sequence. For comparison, it is well-known that the coefficients of the matching polynomial of any graph form a log-concave sequence and the classical algorithm of Jerrum-Sinclair [15] gives an efficient algorithm to count the number of matchings of any graph (although to this date we still don't know an efficient algorithm to count the number of **perfect** matchings of general graphs).

1.1 Background

The existing analyses of the mixing time of the down-up walk for bases of matroids, crucially rely on the theory of high dimensional simplicial complxes [3, 17], which has found many intriguing applications in several areas of computer science and math in the past few years [14].

Simplicial Complex. A *simplicial complex* X on a finite ground set U is a downwards closed set system, i.e. if $\tau \in X$ and $\sigma \subset \tau \subseteq U$, then $\sigma \in X$. The elements of X are called faces, and the maximal faces are called facets. We say X is a pure d-dimensional complex if all of its facets are of size d. We denote the set of facets by $X(d)$. A weighted simplicial complex (X, π) is a simplicial complex X paired with a probability distribution π on its facets. The global walk (down-up walk) $P^\vee$ on the facets of a d-dimensional complex (X, π) is defined as follows: starting at a facet τ, we transition to the next facet τ' by the following two steps:

1. Select a uniformly random element $x \in \tau$ and remove x from τ.
2. Select a random facet τ' containing $\tau \setminus \{x\}$ with probability proportional to $\pi(\tau')$.

Broken Circuit Complex. For a concrete example, it turns out that the set NBC independent sets of any matroid M (with respect to any ordering $\mathcal{O}$) form a **pure** simplicial complex that is known as the *broken circuit complex.* We denote this complex by $\mathrm{BC}(M, \mathcal{O})$. We state purity as the following fact.

▶ **Fact 4.** *For every NBC independent set I, there exists an NBC base B such that $I \subseteq B$.*

The face numbers of the complex $BC(M, \mathcal{O})$ is the sequence $n_0, n_1, \ldots, n_r$ where n_i is the number of NBC independent sets of rank i. As alluded to above this sequence is in variant over $\mathcal{O}$. The down-up walk over this complex equipped with a uniform distribution over its facets is the same as the down-up walk over NBC bases we explained before.

The link of a face $\tau \in X$ is the simplicial complex $X_\tau \coloneqq \{\sigma \setminus \tau : \sigma \in X, \sigma \supset \tau\}$. For each face τ, we define the induced distribution π_τ on the facets of X_τ as

$$\pi_\tau(\eta) = \Pr_{\sigma \sim \pi}[\sigma \supset \eta \mid \sigma \supset \tau]. \tag{1}$$

Local Walks. For any face τ of size $0 \le k \le d-2$, the local walk for τ is a Markov chain on the ground set of X_τ with transition probability matrix P_τ is defined as

$$P_\tau(x,y) = \frac{1}{d-k-1} \Pr_{\sigma \sim \pi_\tau}[y \in \sigma \mid \sigma \supset \tau \cup \{x\}]. \tag{2}$$

for distinct x, y in the ground set of X_τ.

The following theorem shows that the spectral expansion of the global walk $P^\vee$ on a simplicial complex can be bounded through bounding the local spectral expansion of the complex.

▶ **Theorem 5** (Local-to-Global Theorem [10, 17, 9, 2]). *Say a d-dimensional weighted simplicial complex (X, π) is a $(\gamma_0, \dots, \gamma_{d-2})$-local spectral expander if for every face τ of size $0 \le k \le d-2$, the second largest eigenvalue of P_τ is at most γ_k, i.e., $\lambda_2(P_\tau) \le \gamma_k$.*

Given a weighted simplical complex (X, π_d) that is a $(\gamma_0, \dots, \gamma_{d-2})$-local spectral expander, the down-up walk which samples from π has spectral gap lower bounded by

$$1 - \lambda_2(P^\vee) \ge \frac{1}{d} \prod_{j=0}^{d-2} (1 - \gamma_j)$$

To prove that the down-up walk mixes rapidly on the bases of any matroid, [3] proved that the independent set complex of any matroid M is a $(0, 0, \dots, 0)$-local spectral expander. Building on this, a natural method to prove Conjecture 2 is to show that the broken circuit complex of any matroid M of rank r and for any total ordering is a $(\gamma_0, \dots, \gamma_{r-2})$-local spectral expander for $\gamma_i \le \frac{O(1)}{r-i}$.

▶ **Conjecture 6.** *For any matroid M of rank r and any ordering $\mathcal{O}$ the broken circuit complex of M is a $(\gamma_0, \dots, \gamma_{r-2})$-local spectral expander for some $\gamma_i \le \frac{O(1)}{r-i}$*

1.2 Our results

Our main result is to disprove Conjecture 6 in a very strong form, namely for the class of (truncated) graphic matroids.

▶ **Theorem 7.** *There exists an infinite sequence of (truncated) graphic matroids $M_1, M_2, \dots$ with orderings $\mathcal{O}_1, \mathcal{O}_2, \dots$, such that for every $n \ge 1$, M_n has poly(n) elements, and there exists a face τ of the broken circuit complex of $X = \mathrm{BC}(M_n, \mathcal{O})$ for which the down-up walk on the facets of the link X_τ has a spectral gap of at most $n^{-\Omega(n)}$.*

In fact, we even prove a stronger statement

▶ **Theorem 8.** *Given a matroid $M = (E, \mathcal{I})$ and a total ordering $\mathcal{O}$ and a set $S \subseteq E$, unless RP=NP, there is no FPRAS for counting the number of NBC bases of M that contain S.*

Although this theorem does not refute Conjecture 2, it shows that one probably need different techniques (or probably a different chain) to sample/count NBC bases of a matroid. Indeed, one may even need a different proof for the performance of down-up walk to sample ordinary bases of matroids.

To complement our main results we also prove that, unlike optimization on bases of a matroid, optimization is NP-hard on the NBC bases of matroids. Moreover, unless NP= RP, there is no FPRAS for computing the sum of the weights of all NBC bases of a matroid subject to an external field, while the same computation over the bases of a matroid has a FPRAS.

▶ **Theorem 9.** *Given a matroid $M = (E, \mathcal{I})$ with $|E| = n$ elements, an arbitrary total ordering $\mathcal{O}$, and weights $w_1, \dots, w_n$, it is NP-hard to find the maximum weight NBC basis of M, where the weight of a NBC basis B is $\sum_{i \in B} w_i$.*

▶ **Theorem 10.** *Given a matroid $M = (E, \mathcal{I})$ with $|E| = n$ elements, a total ordering $\mathcal{O}$, and weights $\{1 \le \lambda_e \le O(n)\}_{e \in E}$, unless NP = RP, there is no FPRAS for computing the partition function of the λ-external field applied to uniform distribution of NBC independent sets, i.e., there is no FPRAS for computing :*

$$\sum_{B\ NBC\ Base} \prod_{e \in B} \lambda_e.$$

It is well known that a 0/1-polytope (i.e. the convex hull of a subset $S \subseteq \{0,1\}^n$) has all vertices of equal hamming weight r and edges of ℓ_2 length $\sqrt{2}$ iff the polytope is a matroid base polytope of rank r [12]. Moreover, assuming the Mihail-Vazirani conjecture, there is efficient algorithm to sample a uniformly random vertex of a 0/1-polytope with constant sized edge length [18].

We show that, unlike matroids, the NBC Base polytope, i.e. the convex hull of the indicator vectors of all NBC bases of a matroid M, has edges of arbitrarily long length.

▶ **Theorem 11.** *For any n, there exists a graphic matroid M with n elements and a total ordering $\mathcal{O}$ such that the convex hull of all NBC bases of M has edges of ℓ_2 length at least $\Omega(\sqrt{n})$.*

2 Preliminaries

Given a graph $G = (V, E)$, we denote the number of independent sets of size i of G by $i_k(G)$ For every set $S \subseteq V$, we define $N(S) := \{v \notin S : \exists u \in S, \{u, v\} \in E\}$ as the set of neighbors of S in G.

▶ **Definition 12** (Conductance)**.** *Given a weighted d-regular graph $G = (V, E, w)$, with weights $w : E \to \mathbb{R}_{\ge 0}$, for $S \subseteq V$, the conductance of S is defined as*

$$\phi(S) = \frac{w(S, \overline{S})}{d|S|},$$

where $w(S, \overline{S})$ is the sum of the weights of edges in the cut $(S, \overline{S})$. Note that since G is regular, the weighted degree of every vertex is d. The conductance of G is defined as

$$\phi(G) = \min_{S:|S| \le |V|/2} \phi(S).$$

Given a weighted graph $G = (V, E, w)$, the simple random walk is the following stochastic process: Given $X_0 = v \in V$, for every $u \sim v$, we have $X_1 = u$ with probability $\frac{w_{\{u,v\}}}{d_w(v)}$ and we let P be the transition probability matrix of the walk.

The following theorem is well-known and follows from the easy side of the Cheeger's inequality.

▶ **Theorem 13.** *For any regular graph $G = (V, E)$ and any set $S \subseteq V$ and $|S| \le |V|/2$*

$$\frac{1 - \lambda_2(P)}{2} \le \phi(G) \le \phi(S) \le \frac{|N(S)|}{|S|}$$

where $1 - \lambda_2(P)$ is the spectral gap of the simple random walk on G.

A graphic matroid $M = (E, \mathcal{I})$ is a matroid defined on the edges of a graph $G = (V, E)$ and its independent sets are all subsets of edges that do not contain any cycle. It is easy to verify that circuits of M correspond to cycles of G.

▶ **Definition 14** (Matroid Truncation). *Let $M = (E, \mathcal{I})$ be a matroid of rank r. The truncation of M to rank $r' \le r$ removes all independent sets of size strictly greater than r'. It is easy to see that the truncation of any matroid M to any $r' \le r$ is also a matroid.*

Let M' be the truncation to rank r' of a graphic matroid of rank r defined on the edges of a graph G. The bases of M' correspond to forests with r' edges and the circuits of M' are the circuits of G along with all spanning forests of size $r' + 1$.

The following fact about polytopes follows from convexity.

▶ **Fact 15.** *For any polytope $P \subseteq \mathbb{R}^d$ with vertices $v_1, \ldots, v_n \in \mathbb{R}^d$, $\{v_i, v_j\}$ is an edge of P iff there exists a weight function $w \in \mathbb{R}^d$ such that*

$$\langle w, v_i \rangle = \langle w, v_j \rangle > \langle w, v_k \rangle,$$

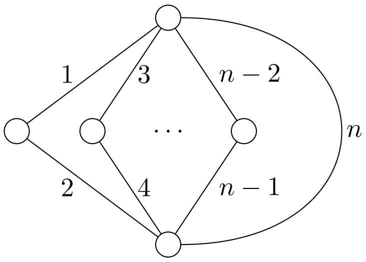
for any $k \ne i, j$.

3 Results

We start with proving Theorem 11.

▶ **Theorem 11.** *For any n, there exists a graphic matroid M with n elements and a total ordering $\mathcal{O}$ such that the convex hull of all NBC bases of M has edges of ℓ_2 length at least $\Omega(\sqrt{n})$.*

Proof. Let n be odd. Consider the following graphic matroid M (with n edges), with the ordering $\mathcal{O}$: $1 < 2 < \cdots < n$ defined by the edges of the following graph:



We show that for $B = \{n\} \cup \{2i - 1 : 1 \le i \le \frac{n-1}{2}\}$ and $B' = \{1\} \cup \{2i : 1 \le i \le \frac{n-1}{2}\}$, $\{B, B'\}$ forms an edge in the NBC matroid base polytope denoted as P_M. We define a $w \in \mathbb{R}^n$ and then use Fact 15 to prove the statement. Let $w_n = \frac{n+1}{2}$, and for any $1 \le i \le \frac{n-1}{2}$, let $w_{2i} = 1$ and $w_{2i-1} = 0$. It is easy to check that the function $\langle w, \mathbf{1}_B \rangle = \langle w, \mathbf{1}_{B'} \rangle = \frac{n+1}{2}$ and $\langle w, \mathbf{1}_{B''} \rangle < \frac{n+1}{2}$ for all NBC basis $B'' \ne B, B'$. Therefore $\{B, B'\}$ forms and edge in P_M. The statements follows from the fact that $\|\mathbf{1}_B - \mathbf{1}_{B'}\|_2 = \sqrt{n}$. ◀

Next, we prove Theorem 9 via a reduction from the MAX-INDEP-SET problem: Given a graph $G = (V, E)$, a weight function $w : V \to \mathbb{R}_{\ge 0}$, and an integer k, decide whether G has an independent set of weight at least k or not.

Note that independent sets of G and independents sets of a BC complex/matroid are two different notions. To complete the proof we use the following well-known hardness result.

▶ **Theorem 16** ([16]). *MAX-INDEP-SET is NP-complete.*

▶ **Theorem 9.** *Given a matroid $M = (E, \mathcal{I})$ with $|E| = n$ elements, an arbitrary total ordering $\mathcal{O}$, and weights $w_1, \ldots, w_n$, it is NP-hard to find the maximum weight NBC basis of M, where the weight of a NBC basis B is $\sum_{i \in B} w_i$.*

Proof. We prove this by a reduction from MAX-INDEP-SET. Let $G = (V, E)$ be a graph, a vertex weight function $w : V \to \mathbb{R}_{\geq 0}$ and k an integer. Construct a new graph $G' = (V', E')$ from G by first copying G and then adding a new vertex z and edges $e_v = \{z, v\}$ for all $v \in V$. We define $w' : E' \to \mathbb{R}_{\geq 0}$ as $w'(e_v) = w(v)$ for every $v \in V$, and $w'(e) = 0$ for every $e \in E$. Moreover, consider the following total ordering $\mathcal{O}$ on E':

$$E < \{e_v : v \in V\},$$

where the ordering within each set is arbitrary. Let M be the graphic matroid defined by the edges of G', we will be look at bases/independent sets of $\mathrm{BC}(M, \mathcal{O})$.

▷ Claim 17. There exists an independent set of G of weight at least k iff there exists an NBC basis of M with weight at least k.

We prove the claim in a straightforward manner. Suppose there is an independent set $I \subseteq V$ of G with $w(I) \geq k$ and consider the set $I' \subseteq E'$ defined by $I' = \{e_v : v \in I\}$. By definition, $w'(I') \geq k$. We argue that I' does not contain any broken circuit. Assume otherwise that there is a broken circuit $C \setminus \{e\} \subseteq I'$. Since C corresponds to a cycle in G' and $C \setminus \{e\}$ is contained in I', it is not hard to see that $C \setminus \{e\} = \{e_v, e_{v'}\}$ for some $v, v' \in I$ and $e = \{v, v'\}$ is an edge in G. But this is a contradiction with the fact that I is an independent set of G.

Hence I' is a NBC independent set. Since the broken circuit complex is pure (see Fact 4), there exists an NBC basis B containing I' which has weight $w'(B) \geq w'(I') \geq k$.

For the other direction, suppose we have a NBC basis $B' \subseteq E'$ of weight $w'(k) \geq k$, and define $I \subseteq V$ by $I = \{v : e_v \in B'\}$. Since all edges coming from E have zero weight, $w(I) = w'(B') \geq k$. To see that I is an independent set of G', note that if there is an edge $\{v, v'\}$ for some $v, v' \in I$, we have $e_v, e_{v'} \in B'$, then $\{e_v, e_{v'}\}$ forms a broken circuit according to the ordering $\mathcal{O}$. Therefore I is an independent set of G of weight at least k. ◀

It's important to note that the above proof works under the crucial assumption that the order $\mathcal{O}$ is chosen carefully based on the weights (and in some sense in the same order of the weights).

We can amplify the ideas in the previous construction to also argue Theorem 7. This is done by constructing a Broken Circuit complex for which the down-up walk of a carefully chosen link has inverse exponentially small spectral gap.

▶ **Theorem 7.** *There exists an infinite sequence of (truncated) graphic matroids $M_1, M_2, \ldots$ with orderings $\mathcal{O}_1, \mathcal{O}_2, \ldots$, such that for every $n \geq 1$, M_n has poly(n) elements, and there exists a face τ of the broken circuit complex of $X = \mathrm{BC}(M_n, \mathcal{O})$ for which the down-up walk on the facets of the link X_τ has a spectral gap of at most $n^{-\Omega(n)}$.*

Proof. Take the complete bipartite graph $G = K_{n,n} = (A, B, E = A \times B)$, with $|A| = |B| = n$. Also, let $V = A \cup B$. Let $\ell \geq 1$ be a parameter that we choose later, and construct a new graph

$$G' = (V' = V \cup \{y, z\} \cup \{z_{v,i} : v \in V, i \in [\ell]\}, E' = E \cup \{e_0\} \cup \{e_{v,i}, f_{v,i} : v \in V, i \in [\ell]\})$$

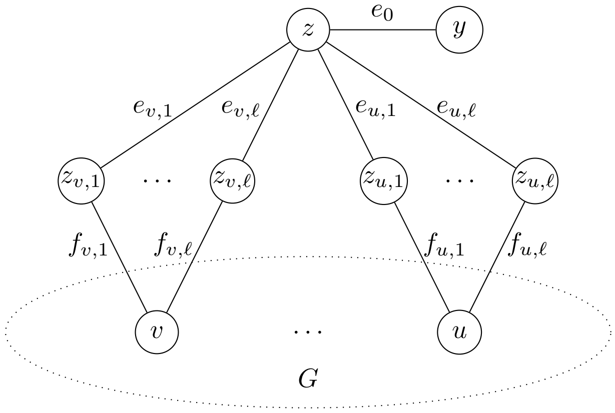


Figure 1 A schematic of the graph G' in the proofs of Theorem 7 and Theorem 8 where $G = K_{n,n}$ is the complete bipartite graph in the former and it is a hard instance of $\sharp$INDEP-SET-INC$(7, \frac{2}{19})$ in latter.

where $e_0 = \{y, z\}, e_{v,i} = \{z, z_{v,i}\}, f_{v,i} = \{z_{v,i}, v\}$ (see Figure 1). For a sanity check, note that $|V| = 2n$ and $|V'| = 2\ell n + 2n + 2$.

Let $M = (E', \mathcal{I})$ be the graphic matroid defined by G' truncated to rank $2\ell n + n + 1$, i.e., the bases of M are forests of G' with exactly $2\ell n + n + 1$ edges. Now, consider the following total ordering $\mathcal{O}$ on E':

$$e_0 < E < \{e_{v,i} : v \in V, i \in [\ell]\} < \{f_{v,i} : v \in V, i \in [\ell]\},$$

where the ordering within each set is arbitrary.

Moreover, let $X := \mathrm{BC}(M, \mathcal{O})$, and define

$$\tau = \{e_{v,i} : v \in V, i \in [\ell]\}.$$

For simplicity of notation, let $F_A := \{f_{v,i} : v \in A, i \in [\ell]\}$ and $F_B := \{f_{v,i} : v \in B, i \in [\ell]\}$.

▷ Claim 18. For any facet S of X_τ, either $S \cap F_A = \emptyset$, or $S \cap F_B = \emptyset$,

This follows from the fact that G is a complete bipartite graph and edges in E are smaller than $e_{v,i}$'s and $f_{u,j}$'s; so if $S \cap F_A, S \cap F_B \neq \emptyset$, then it has a broken circuit.

Therefore, the set of facets of X_τ can be partitioned into $2n + 1$ sets $(\cup_{i=1}^n \mathcal{S}_{A,i}) \cup (\cup_{i=1}^n \mathcal{S}_{B,i}) \cup \mathcal{S}_0$, where $\mathcal{S}_{A,i}$ is the set of all facets S with $|S \cap F_A| = i$, $\mathcal{S}_{B,i}$ is the set of all facets S with $|S \cap F_B| = i$, and $\mathcal{S}_0$ is the set of all facets with $|S \cap (F_A \cup F_B)| = 0$. Let $\mathcal{S}_A := \cup_{i=1}^n \mathcal{S}_{A,i}$ and similarly define $\mathcal{S}_B$. We show that $\frac{|N(\mathcal{S}_A)|}{|\mathcal{S}_A|} \leq n^{-\Omega(n)}$, where $N(\mathcal{S}_A)$ is the set of neighbors of $\mathcal{S}_A$ in the down-up walk $P_\tau^\vee$ on the facets of τ. WLOG we can assume that $|\mathcal{S}_A|$ is at most half of all facets. Applying Theorem 13, this would imply that $1 - \lambda_2(P_\tau^\vee) \leq n^{-\Omega(n)}$.

First, note that for every facet $S \in \mathcal{S}_A$ and $T \in \mathcal{S}_B \setminus \mathcal{S}_{B,1}$, we get $P^\vee(S, T) = 0$ since $|S \Delta T| > 2$. So, $N(\mathcal{S}_A) \subseteq \mathcal{S}_{B,1} \cup \mathcal{S}_0$. First, notice $|\mathcal{S}_0| \leq \binom{|E|}{n} \leq n^{2n}$. Furthermore, $|\mathcal{S}_{B,1}| \leq \binom{n}{1} \ell \binom{|E|}{n-1} \leq \ell n^{2n}$.

This follows from the fact that any facet in $\mathcal{S}_{B,1}$ can be written as $\{f_{v,i_v}\} \cup \{e_0\} \cup K$ for some $v \in A$, $i_v \in [\ell]$, and subset $K \subseteq E$ of size $n - 1$.

Lastly, $|\mathcal{S}_A| \geq |\mathcal{S}_{A,n}| = \ell^n$. This is because every choice of $\{i_v\}_{v \in A}$ corresponds to a set in $\mathcal{S}_{A,n}$ whose sets are of the form $\{f_{v,i_v} : v \in V\} \cup \{e_0\}$. These sets all don't contain a broken circuit because the circuits introduced through truncation are exactly the forests with

$2\ell n + n + 2$ edges. However, any proper superset of $\{f_{v,i_v} : v \in V\} \cup \{e_0\}$ must include e_0, so looking at the circuit introduced by the superset, the corresponding broken circuit will always remove e_0. Putting it all together,

$$1 - \lambda_2(P_\tau^\vee) \leq \frac{|N(\mathcal{S}_A)|}{|\mathcal{S}_A|} \leq \frac{n^{2n}(1+\ell)}{\ell^n} \underset{\text{assuming } \ell \geq n^3}{\leq} n^{-\Omega(n)}$$

as desired. ◀

We prove Theorem 10 and Theorem 8 by a reduction from ♯INDEP-SET-INC($7, \frac{2}{19}$), defined as the following.

▶ **Definition 19** (♯INDEP-SET-INC($7, \frac{2}{19}$)). *Given a 7-regular graph $G = (V, E)$ that satisfies $i_k(G) \leq i_{\lfloor \frac{2|V|}{19} \rfloor}(G)$ for any $k < \lfloor \frac{2|V|}{19} \rfloor$, where $i_k(G)$ are the independent sets of G of size k, count the number of independent sets of size $\lfloor \frac{2|V|}{19} \rfloor$.*

▶ **Theorem 20.** *Unless* NP = RP*, there is no randomized algorithm with constant approximation ratio for ♯INDEP-SET-INC*($7, \frac{2}{19}$)*.*

We leave the proof of this for the appendix.

Now, we are ready to prove Theorem 8. The high-level structure of the proof is similar to the proof of Theorem 7 where we apply a similar gadget to graphs on which it is hard to count independent sets (as opposed to the complete bipartite graph).

▶ **Theorem 8.** *Given a matroid $M = (E, \mathcal{I})$ and a total ordering $\mathcal{O}$ and a set $S \subseteq E$, unless RP=NP, there is no FPRAS for counting the number of NBC bases of M that contain S.*

Proof. For simplicity of notion, let $\alpha := \frac{2}{19}$. We prove by a reduction from ♯INDEP-SET-INC($7, \frac{2}{19}$). Take any arbitrary 7-regular graph $G = (V, E)$ whose number of independent sets of size $\lfloor \alpha |V| \rfloor$ is at least the number of its independent sets of size k for any $k < \lfloor \alpha |V| \rfloor$. Let $n := |V|$ and N be the number of independent sets of size $\lfloor \alpha n \rfloor$ of G. Also, define $\ell \geq 1$ to be a parameter that we choose later.

Now, construct a new graph

$$G' = (V' = V \cup \{y, z\} \cup \{z_{v,i} : v \in V, i \in [\ell]\}, E' = E \cup \{e_0\} \cup \{e_{v,i}, f_{v,i} : v \in V, i \in [\ell]\})$$

where $e_0 = \{y, z\}, e_{v,i} = \{z, z_{v,i}\}, f_{v,i} = \{z_{v,i}, v\}$ (see Figure 1). Let $M = (E', \mathcal{I})$ be the graphic matroid defined by G truncated at rank $\ell n + \lfloor \alpha n \rfloor + 1$, i.e., the bases of M are forests of G' with exactly $\ell n + \lfloor \alpha n \rfloor + 1$ edges. Now, consider the following ordering $\mathcal{O}$ on E':

$$e_0 < E < \{e_{v,i} : v \in V, i \in [\ell]\} < \{f_{v,i} : v \in V, i \in [\ell]\},$$

where the ordering within each set is arbitrary. Moreover, let $X := \mathrm{BC}(M, \mathcal{O})$, and define

$$\tau = \{e_{v,i} : v \in V, i \in [\ell]\}.$$

We claim that the number of facets of X_τ is at least $\ell^{\lfloor \alpha n \rfloor} N$ and at most $2\ell^{\lfloor \alpha n \rfloor} N$. So, a 1.5-approximation to the number facets of X_τ, i.e., the number NBC bases of M that contain τ, gives a 3-approximation to N, the number of independent sets of size $\lfloor \alpha n \rfloor$ of G.

We use the following crucial observation:

▷ Claim 21. For any facet S of X_τ, $\{v : \exists f_{v,i} \in S\}$ is an independent set of G and for any $f_{v,i}, f_{v,j} \in S$ we have $i = j$.

Conversely, for any $S \subseteq \{f_{v,i} : v \in V, i \in [\ell]\}$, such that the set $\{v : \exists f_{v,i} \in S\}$ is an independent set of size $\lfloor \alpha n \rfloor$ of G, and $f_{v,i}, f_{v,j} \in S \implies i = j$, we have $S \cup \{e_0\}$ is a facet of X_τ.

The proof simply follows from the fact that edges of E are smaller than $e_{v,i}$'s, and $f'_{u,j}s$ in $\mathcal{O}$. By the second part of the claim, we can write

$$|X_\tau(\lfloor \alpha n \rfloor + 1)| = \ell^{\lfloor \alpha n \rfloor} N + |\{S \in X_\tau(\lfloor \alpha n \rfloor + 1) : S \cap E \neq \emptyset\}| \geq \ell^{\lfloor \alpha n \rfloor} N. \tag{3}$$

Define $i_k := i_k(G)$ as the number of independent sets of size k of graph G. By the first part of the above claim we can write,

$$|\{S \in X_\tau(\lfloor \alpha n \rfloor + 1) : S \cap E \neq \emptyset\}| \leq \sum_{k=0}^{\lfloor \alpha n \rfloor - 1} \ell^k \cdot i_k \cdot \binom{|E|}{\lfloor \alpha n \rfloor - k} \tag{4}$$

$$\leq \sum_{k=0}^{\lfloor \alpha n \rfloor - 1} \ell^k \cdot i_k \cdot |E|^{\lfloor \alpha n \rfloor - k} \tag{5}$$

$$\underset{\text{using } i_k \leq N}{\leq} N|E|^{\lfloor \alpha n \rfloor} \sum_{k=0}^{\lfloor \alpha n \rfloor - 1} (\ell/|E|)^k \tag{6}$$

$$\underset{\text{assuming } \ell \geq 2|E|}{\leq} N|E|^{\lfloor \alpha n \rfloor} (\ell/|E|)^{\lfloor \alpha n \rfloor} \leq N\ell^{\lfloor \alpha n \rfloor} \tag{7}$$

Putting these together with (3) concludes the proof. ◀

▶ **Theorem 10.** *Given a matroid $M = (E, \mathcal{I})$ with $|E| = n$ elements, a total ordering $\mathcal{O}$, and weights $\{1 \leq \lambda_e \leq O(n)\}_{e \in E}$, unless NP = RP, there is no FPRAS for computing the partition function of the λ-external field applied to uniform distribution of NBC independent sets, i.e., there is no FPRAS for computing :*

$$\sum_{B \ NBC \ Base} \prod_{e \in B} \lambda_e.$$

Proof. For simplicity of notion, let $\alpha := \frac{2}{19}$. The proof is similar to the proof of Theorem 8 by a reduction from $\sharp$INDEP-SET-INC($7, \frac{2}{19}$). Take any arbitrary 7-regular graph $G = (V, E)$ with $n := |V|$ vertices whose number of independent sets of size $\lfloor \alpha |V| \rfloor$ is at least the number of its independent sets of size k for any $k < \lfloor \alpha |V| \rfloor$. Construct a new graph

$$G' = (V' = V \cup \{y, z\}, E' = E \cup \{e_0 = \{y, z\}\} \cup \{e_v = \{v, z\} : v \in V\})$$

Let $M = (E', \mathcal{I})$ be the graphic matroid given by G' truncated to rank $\lfloor \alpha n \rfloor + 1$ and consider the following ordering $\mathcal{O}$ on E'':

$$e_0 < E < \{e_v : v \in V\},$$

where as usual the ordering within each set is arbitrary. Define weights $\lambda : E' \to \mathbb{R}_{\geq 0}$ as follows:

$$\lambda_e = \begin{cases} \ell & \text{if } e = e_v \text{ for some } v \in V, \\ 1 & \text{o.w.} \end{cases},$$

for some ℓ that we choose later. We argue that

$$\lambda^{\lfloor \alpha n \rfloor} N \leq \sum_B \prod_{e \in B} \lambda_e \leq 2\lambda^{\lfloor \alpha n \rfloor} N.$$

where here (and henceforth) the sum is over B's that are NBC bases of M, and therefore a 1.5-approximation to the partition function, i.e., the quantity in the middle, is a 3-approximation to N. Similar to the previous theorem we have the following claim.

▷ Claim 22. For any NBC base B of M, we have $\{v : e_v \in B\}$ is an independent set of G. Conversely, for any independent set I of G of size $|I| = \lfloor \alpha n \rfloor$, $\{e_0\} \cup \{e_v : v \in I\}$ is a NBC base of M.

So,

$$\begin{aligned}\sum_{B}\prod_{e\in B}\lambda_e &= \sum_{B:B\cap E\neq\emptyset}\prod_{e\in B}\lambda_e + \sum_{B:B\cap E=\emptyset}\prod_{e\in B}\lambda_e \\ &= \sum_{B:B\cap E\neq\emptyset}\prod_{e\in B}\lambda_e + \ell^{\lfloor\alpha n\rfloor}|\{S\subseteq V : S \text{ independent set of } G, |S| = \lfloor\alpha n\rfloor\}|\end{aligned} \tag{8}$$

Define i_k as the number of independent sets of size k of graph G. We have

$$\sum_{B:B\cap E\neq\emptyset}\prod_{e\in B}\lambda_e \le \sum_{k=0}^{\lfloor\alpha n\rfloor-1}\ell^k i_k\binom{|E|}{\lfloor\alpha n\rfloor - k} \underset{\substack{\text{using } i_k\le N,\\ \text{assuming } \ell\ge 2|E|}}{\le} \ell^{\lfloor\alpha n\rfloor}N$$

where the last inequality follows from the same calculations as in Equation (4). ◀

References

1 Karim Adiprasito, June Huh, and Eric Katz. Hodge theory for combinatorial geometries, 2018. arXiv:1511.02888.

2 Vedat Levi Alev and Lap Chi Lau. Improved analysis of higher order random walks and applications. In *Proceedings of the 52nd Annual ACM Symposium on Theory of Computing (STOC)*, 2020.

3 Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials ii: high-dimensional walks and an fpras for counting bases of a matroid. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1–12, 2019.

4 Brian Benson, Deeparnab Chakrabarty, and Prasad Tetali. g-parking functions, acyclic orientations and spanning trees, 2010. arXiv:0801.1114.

5 Tom Brylawski. The broken-circuit complex. *Trans. Amer. Math. Soc.*, 234(2):417–433, 1977.

6 Charles J Colbourn and William R Pulleyblank. Matroid steiner problems, the tutte polynomial and network reliability. *Journal of Combinatorial Theory, Series B*, 47(1):20–31, 1989. doi:10.1016/0095-8956(89)90062-2.

7 Mary Cryan, Heng Guo, and Giorgos Mousa. Modified log-sobolev inequalities for strongly log-concave distributions, 2020. arXiv:1903.06081.

8 Ewan Davies and Will Perkins. Approximately counting independent sets of a given size in bounded-degree graphs. *arXiv preprint*, 2021. arXiv:2102.04984.

9 Yotam Dikstein, Irit Dinur, Yuval Filmus, and Prahladh Harsha. Boolean Function Analysis on High-Dimensional Expanders. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018)*, volume 116 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 38:1–38:20, 2018.

10 I. Dinur and T. Kaufman. High dimensional expanders imply agreement expanders. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 974–985, 2017.

11 Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Inapproximability of the partition function for the antiferromagnetic ising and hard-core models. *Combinatorics, Probability and Computing*, 25(4):500–559, 2016.

12 I.M Gelfand, R.M Goresky, R.D MacPherson, and V.V Serganova. Combinatorial geometries, convex polyhedra, and schubert cells. *Advances in Mathematics*, 63(3):301–316, 1987. doi:10.1016/0001-8708(87)90059-4.

13 Emeric Gioan and Michel Las Vergnas. The active bijection for graphs. *Advances in Applied Mathematics*, 104:165–236, 2019. doi:10.1016/j.aam.2018.11.001.

14 Roy Gotlib and Tali Kaufman. No where to go but high: A perspective on high dimensional expanders, 2023. arXiv:2304.10106.

15 Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM Journal on Computing*, 18(6):1149–1178, 1989. doi:10.1137/0218077.

16 Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972. doi:10.1007/978-1-4684-2001-2_9.

17 Tali Kaufman and Izhar Oppenheim. High order random walks: Beyond spectral gap. In *APPROX/RANDOM*, pages 47:1–47:17, 2018.

18 Milena Mihail and Umesh Vazirani. On the expansion of 0-1 polytopes. *preprint*, 3461, 1989.

19 Geoffrey Ramseyer, Ashish Goel, and David Mazières. Liquidity in credit networks with constrained agents. In *Proceedings of The Web Conference 2020*. ACM, April 2020. doi:10.1145/3366423.3380276.

20 Allan Sly. Computational transition at the uniqueness threshold. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 287–296. IEEE, 2010.

21 Allan Sly and Nike Sun. Counting in two-spin models on d-regular graphs. *Annals of Probability*, 42(6):2383–2416, 2014.

22 Richard P. Stanley. Acyclic orientations of graphs. *Discrete Mathematics*, 5(2):171–178, 1973. doi:10.1016/0012-365X(73)90108-8.

23 Richard P Stanley et al. An introduction to hyperplane arrangements. *Geometric combinatorics*, 13(389-496):24, 2004.

A Proof of Theorem 20

In this section we prove Theorem 20. We use a reduction from the problem of computing the partition function of the Hardcore model when the fugacity is above the critical threshold. Define $\sharp$HC(Δ, λ) as follows: given a Δ-regular graph $G = (V, E)$, compute the partition function $Z_G(\lambda) = \sum_I \lambda^{|I|}$, where the sum is taken over the family of independent sets $I \subseteq V$ of G. The critical threshold is defined as $\lambda_c(\Delta) := \frac{(\Delta-1)^{\Delta-1}}{(\Delta-2)^{\Delta}}$.

▶ **Theorem 23** ([20, 21, 11]). *The following holds for any fixed $\epsilon > 0$, integer $\Delta \geq 3$ and $\lambda > \lambda_c(\Delta)$: unless NP=RP, for any $\lambda > \lambda_c(\Delta)$ there is no polynomial-time algorithm for for approximating $\sharp$HC(Δ, λ) up to a $1+\epsilon$ multiplicative factor.*

We give a polynomial-time algorithm that given a $e^{\pm\epsilon/2}$-approximation for $\sharp$INDEP-SET-INC$(7, \frac{2}{19})$ (see Definition 19), approximates $\sharp$HC$(7, \frac{2}{3})$ up to a $e^{\pm\epsilon}$-multiplicative error. Since $\frac{2}{3} > \lambda_c(7) = \frac{6^6}{5^7} \geq 0.6$, this finishes the proof of Theorem 20. Our reduction is a modification of Theorem 16 in [8].

▶ **Theorem 24.** *There exists a polynomial-time algorithm that for any given $\epsilon \leq 1$, satisfies the following properties:*

1. *Given an instance $G = (V, E)$ of $\sharp$HC$(7, \frac{2}{3})$, the algorithm constructs an instance $G' = (V', E')$ of the problem $\sharp$INDEP-SET-INC$(7, \frac{2}{19})$ with size polynomial in $|G|$.*
2. *Given a $e^{\pm\epsilon/2}$-multiplicative approximation to the number of independent sets of size $\lfloor \frac{2|V'|}{19} \rfloor$ of G', a $e^{\pm\epsilon}$-approximation of $Z_G(\frac{2}{3})$ can be computed in polynomial time.*

Proof. Given a 7-regular graph $G = (V, E)$, we define G' as the disjoint union of G with $r := \frac{c^2 n^2}{\epsilon}$ copies of the complete graph K_8, where $n = |V|$, for some $c > 1$ that we choose later. For simplicity of notation, let $N := |V'| = n + 8r$, $\alpha := \frac{2}{19}$, $\lambda := \frac{2}{3}$. It is enough to show that G' is an instance of $\sharp$INDEP-SET-INC$(7, \frac{2}{19})$ and

$$e^{-\epsilon/2} \frac{i_{\lfloor \alpha N \rfloor}(G')}{\binom{r}{\lfloor \alpha N \rfloor} 8^{\lfloor \alpha N \rfloor}} \leq Z_G(\lambda) \leq e^{\epsilon/2} \frac{i_{\lfloor \alpha N \rfloor}(G')}{\binom{r}{\lfloor \alpha N \rfloor} 8^{\lfloor \alpha N \rfloor}}, \tag{9}$$

where as usual $i_k(G)$ is the number of independent sets of size k in G, and

$$\binom{r}{\lfloor \alpha N \rfloor} 8^{\lfloor \alpha N \rfloor} = i_{\lfloor \alpha N \rfloor}(rK_8).$$

Here, rK_8 is a shorthand for the graph which is a disjoint union of r copies of K_8. We first show that Equation (9) holds. Note that

$$i_{\lfloor \alpha N \rfloor}(G') = \sum_{j=0}^{n} i_j(G) i_{\lfloor \alpha N \rfloor - j}(rK_8) = i_{\lfloor \alpha N \rfloor}(rK_8) \sum_{j=0}^{n} i_j(G) \frac{i_{\lfloor \alpha N \rfloor - j}(rK_8)}{i_{\lfloor \alpha N \rfloor}(rK_8)}.$$

Thus, to show Equation (9), it is enough to prove that for every $1 \le j \le n$,

$$e^{-\epsilon/2} \cdot \frac{i_{\lfloor \alpha N \rfloor - j}(rK_8)}{i_{\lfloor \alpha N \rfloor}(rK_8)} \le \lambda^j \le e^{\epsilon/2} \cdot \frac{i_{\lfloor \alpha N \rfloor - j}(rK_8)}{i_{\lfloor \alpha N \rfloor}(rK_8)}. \tag{10}$$

We can write

$$\frac{i_{\lfloor \alpha N \rfloor - j}(rK_8)}{i_{\lfloor \alpha N \rfloor}(rK_8)} = \frac{\binom{r}{\lfloor \alpha N \rfloor - j} 8^{\lfloor \alpha N \rfloor - j}}{\binom{r}{\lfloor \alpha N \rfloor} 8^{\lfloor \alpha N \rfloor}} = \frac{1}{8^j} \prod_{i=0}^{j-1} \frac{\lfloor \alpha N \rfloor - i}{r - \lfloor \alpha N \rfloor + j - i}. \tag{11}$$

To prove the upper bound, first note that

$$\frac{\alpha N}{r - \alpha N + j} \underset{\alpha N \ge 8\alpha r}{\ge} \frac{8\alpha r}{r(1 - 8\alpha) + n} \underset{\substack{n = \sqrt{\epsilon r}/c \\ \alpha = 2/19}}{=} \frac{16}{3} \left(\frac{1}{1 + \frac{19\sqrt{\epsilon}}{3c\sqrt{r}}} \right). \tag{12}$$

This implies that $\frac{\alpha N}{r - \alpha N + j} \ge 1$. So, $\frac{\alpha N}{r - \alpha N + j} \le \frac{\lfloor \alpha N \rfloor - i}{r - \lfloor \alpha N \rfloor + j - i}$ for every $i < r - \lfloor \alpha N \rfloor + j$. Thus,

$$\begin{aligned} \frac{1}{8^j} \cdot \prod_{i=0}^{j-1} \frac{\lfloor \alpha N \rfloor - i}{r - \lfloor \alpha N \rfloor + j - i} &\ge \frac{1}{8^j} \cdot \left(\frac{\alpha N}{r - \alpha N + j} \right)^j \\ &\underset{\substack{Equation\ (12) \\ j \le n = \sqrt{\epsilon r}/c}}{\ge} \frac{1}{8^j} \cdot (\tfrac{16}{3})^j \left(\frac{1}{1 + \frac{19\sqrt{\epsilon}}{3c\sqrt{r}}} \right)^{\sqrt{\epsilon r}/c} \ge (\tfrac{2}{3})^j e^{-\epsilon/2} = \lambda^j e^{-\epsilon/2}, \end{aligned}$$

for a large enough $c > 1$. Combining this with Equation (11), we get the upper bound in Equation (10).

To prove the lower bound, note that

$$\begin{aligned} \frac{1}{8^j} \cdot \prod_{i=0}^{j-1} \frac{\lfloor \alpha N \rfloor - i}{r - \lfloor \alpha N \rfloor + j - i} &\underset{j - i \ge 0}{\le} \frac{1}{8^j} \cdot \left(\frac{\lfloor \alpha N \rfloor}{r - \lfloor \alpha N \rfloor} \right)^j \\ &\underset{\lfloor \alpha N \rfloor = \lfloor \frac{16r}{19} + \frac{2\sqrt{\epsilon r}}{19c} \rfloor}{\le} \frac{1}{8^j} \cdot \left(\frac{\frac{16r}{19}(1 + \frac{\sqrt{\epsilon}}{8c\sqrt{r}})}{\frac{3r}{19}(1 - \frac{2\sqrt{\epsilon}}{3c\sqrt{r}})} \right)^j \\ &\le (\tfrac{2}{3})^j e^{\epsilon/2} = \lambda^j \cdot e^{\epsilon/2}, \end{aligned}$$

for a large enough $c > 1$. Combining this with Equation (11), the lower bound in Equation (10), thus (9) follows.

It remains to show that G' is an instance of ♯INDEP-SET-INC$(7, \frac{2}{19})$, i.e. $i_k(G') \le i_{\lfloor \alpha N \rfloor}(G')$ for any $k < \lfloor \alpha N \rfloor$.

For any $k < \lfloor \alpha N \rfloor$, and any independent set S in the original graph G, let $T_{S,k}$ be the set of all independent sets of size k of G' whose intersection with the vertices of G is S. It is enough to show that there exists a constant n_0 such that if $n \geq n_0$, then we have $|T_{S,k}| \leq |T'_{S,\lfloor \alpha N \rfloor}|$ for every independent set $S \subseteq V$ of G and $k < \lfloor \alpha N \rfloor$. We prove a stronger statement that there exists a constant n_0 such that if $n \geq n_0$, then for any fixed independent set $S \subseteq V$, $|T_{S,k}|$ is increasing as a function of k for all $k \leq \lfloor \alpha N \rfloor$. It is enough to show that $\frac{|T_{S,k}|}{|T_{S,k-1}|} \geq 1$ for any $|S| \leq k \leq \alpha N$. Note that $|T_{S,k}| = \binom{r}{k-|S|} 8^{k-|S|}$. So we have

$$\frac{|T_{S,k}|}{|T_{S,k-1}|} = \frac{\binom{r}{k-|S|} 8^{k-|S|}}{\binom{r}{k-1-|S|} 8^{k-1-|S|}} = 8 \cdot \frac{r-k+|S|+1}{k-|S|} \geq 8 \cdot \frac{r-k}{k} \geq 8 \frac{\frac{3r}{19} - n}{\frac{16r}{19} + n},$$

where the last inequality comes from the fact that $k \leq \frac{2N}{19} = \frac{2}{19}(8r+n) \leq \frac{16r}{19} + n$. But since $r = \frac{c^2 n^2}{\epsilon}$, there is a constant n_0 such that for $n \geq n_0$, we have $\frac{\frac{3r}{19} - n}{\frac{16r}{19} + n} \geq \frac{1}{8}$. This shows that $\frac{|T_{S,k}|}{|T_{S,k-1}|} \geq 1$, which finishes the proof. ◀

Low-Degree Testing over Grids

Prashanth Amireddy
School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA

Srikanth Srinivasan
Department of Computer Science, Aarhus University, Denmark

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

Abstract

We study the question of local testability of low (constant) degree functions from a product domain $\mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ to a field $\mathbb{F}$, where $\mathcal{S}_i \subseteq \mathbb{F}$ can be arbitrary constant sized sets. We show that this family is locally testable when the grid is "symmetric". That is, if $\mathcal{S}_i = \mathcal{S}$ for all i, there is a probabilistic algorithm using constantly many queries that distinguishes whether f has a polynomial representation of degree at most d or is $\Omega(1)$-far from having this property. In contrast, we show that there exist asymmetric grids with $|\mathcal{S}_1| = \cdots = |\mathcal{S}_n| = 3$ for which testing requires $\omega_n(1)$ queries, thereby establishing that even in the context of polynomials, local testing depends on the structure of the domain and not just the distance of the underlying code.

The low-degree testing problem has been studied extensively over the years and a wide variety of tools have been applied to propose and analyze tests. Our work introduces yet another new connection in this rich field, by building low-degree tests out of tests for "junta-degrees". A function $f : \mathcal{S}_1 \times \cdots \times \mathcal{S}_n \to \mathcal{G}$, for an abelian group $\mathcal{G}$ is said to be a junta-degree-d function if it is a sum of d-juntas. We derive our low-degree test by giving a new local test for junta-degree-d functions. For the analysis of our tests, we deduce a small-set expansion theorem for spherical/hamming noise over large grids, which may be of independent interest.

2012 ACM Subject Classification Theory of computation → Probabilistic computation

Keywords and phrases Property testing, Low-degree testing, Small-set expansion, Local testing

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.41

Category RANDOM

Related Version *Full Version*: https://arxiv.org/pdf/2305.04983.pdf

Funding *Prashanth Amireddy*: Supported in part by a Simons Investigator Award and NSF Award CCF 2152413 to Madhu Sudan.
Srikanth Srinivasan: Supported by a start-up package from Aarhus University. Work done while visiting the Meta-Complexity program at the Simons Institute for the Theory of Computing, UC Berkeley.
Madhu Sudan: Supported in part by a Simons Investigator Award and NSF Award CCF 2152413.

1 Introduction

The main problem considered in this paper is "low-degree testing over grids". Specifically given a degree parameter $d \in \mathbb{Z}^{\geqslant 0}$ and proximity parameter $\delta > 0$ we would like to design a tester (a randomized oracle algorithm) that is given oracle access to a function $f : \mathcal{S}_1 \times \cdots \times \mathcal{S}_n \to \mathbb{F}$ where $\mathbb{F}$ is a field and $\mathcal{S}_1, \ldots, \mathcal{S}_n \subseteq \mathbb{F}$ are arbitrary finite sets, and accepts if f is a polynomial of degree at most d while rejecting with constant probability (say $1/2$) if f is δ-far (in relative Hamming distance) from every degree d polynomial. The main goal here is to identify settings where the test makes $O(1)$ queries when $d, 1/\delta$ and $\max_{i \in [n]}\{|\mathcal{S}_i|\}$ are all considered constants. (In particular the goal is to get a query complexity independent of n.)

Low-degree testing

The low-degree testing problem over grids is a generalization of the classical low-degree testing problem which corresponds to the special case where $\mathbb{F}$ is a finite field and $\mathcal{S}_1 = \cdots = \mathcal{S}_n = \mathbb{F}$. Versions of the classical problem were studied in the early 90s [5, 6, 11] in the context of program checking and (multi-prover) interactive proofs. The problem was formally defined and systematically studied by Rubinfeld and Sudan [24] and played a central role in the PCP theorem [2, 3] and subsequent improvements. While the initial exploration of low-degree testing focused on the case where $d \ll |\mathbb{F}|$ (and tried to get bounds that depended polynomially, or even linearly, on d), a later series of works starting with that of Alon, Kaufman, Krivelevich, Litsyn and Ron [1] initiated the study of low degree testing in the setting where $d > |\mathbb{F}|$. [1] studied the setting of $\mathbb{F} = \mathbb{F}_2$ and this was extended to the setting of other constant sized fields in [17, 19]. An even more recent sequence of works [10, 14, 15, 18] explores so-called "optimal tests" for this setting and these results have led to new applications to the study of the Gowers uniformity norm, proofs of XOR lemmas for polynomials [10], and novel constructions of small set expanders [8].

Part of the reason for the wide applicability of low-degree testing is the fact that evaluations of polynomials form error-correcting codes, a fact that dates back at least to the work of Ore [22]. Ore's theorem (a.k.a. the Schwartz-Zippel lemma) however applies widely to the evaluations of polynomial on entire "grids", i.e., sets of the form $\mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ and bounds the distance between low-degree functions in terms of the degree d and minimum set size $\min_i\{|\mathcal{S}_i|\}$. This motivated Bafna, Srinivasan and Sudan [7] to introduce the low-degree testing problem over grids. They proposed and analyzed a low-degree test for the special case of the Boolean grid, i.e., where $|\mathcal{S}_1| = \cdots = |\mathcal{S}_n| = 2$. This setting already captures the setting considered in [1] while also including some novel settings such as testing the Fourier degree of Boolean functions (here the domain is $\{-1, +1\}^n$ while the range is $\mathbb{R}$). The main theorem in [7] shows that there is a tester with constant query complexity, thus qualitatively reproducing the theorem of [1] (though with a worse query complexity than [1] which was itself worse than the optimal result in [10]), while extending the result to many new settings.

In this work we attempt to go beyond the restriction of a Boolean grid. We discuss our results in more detail shortly, but the main outcome of our exploration is that the problem takes on very different flavors depending on whether the grid is symmetric ($\mathcal{S}_1 = \cdots = \mathcal{S}_n$) or not. In the former case, we get constant complexity testers for constant $|\mathcal{S}_i|$ whereas in the latter setting we show that even when $|\mathcal{S}_i| = 3$ low-degree (even $d = 1$) testing requires superconstant query complexity. (See Theorem 3 for details.) In contrast to previous testers, our tester goes via "junta-degree-tests", a concept that has been explored in the literature but not as extensively as low-degree tests, and not been connected to low-degree tests in the past. We describe this problem and our results for this problem next.

Junta-degree testing

A function $f : \mathcal{S}_1 \times \cdots \times \mathcal{S}_n \to \mathcal{G}$ for an arbitrary set $\mathcal{G}$ is said to be a d-junta if it depends only on d of the n variables. When $\mathcal{G}$ is an abelian group, a function $f : \mathcal{S}_1 \times \cdots \times \mathcal{S}_n \to \mathcal{G}$ is said to be of junta-degree d if it is the sum of d-juntas (where the sum is over $\mathcal{G}$).[1] In the special case where $|\mathcal{S}_i| = 2$ for all i and $\mathcal{G}$ is a field, junta degree coincides with the

[1] While in principle the problem could also be considered over non-abelian groups, in such a case it not clear if there is a fixed bound on the number of juntas that need to be summed to get to a function of bounded junta-degree.

usual notion of degree. More generally every degree d polynomial has junta degree d, while a function of junta-degree d is a polynomial of degree at most $d \cdot \max_i\{|\mathcal{S}_i|\}$. Thus junta-degree is softly related to algebraic degree and our work provides a step towards low-degree testing via the problem of junta-degree testing.

Junta-degree testing considers the task of testing if a given function has junta-degree at most d or if it is far from all functions of junta-degree at most d. While this problem has not been considered in full generality before, two works do consider this problem for the special case of $d = 1$. Dinur and Golubev [13] considered this problem in the setting where $\mathcal{G} = \mathbb{F}_2$, while Bogdanov and Prakriya [12] consider this for general abelian groups. This special case corresponds to the problem of testing if a function is a direct sum, thus relating to other interesting classes of properties studied in testing. Both works give $O(1)$ query testers in their settings, but even the case of $d = 2$ remained open.

In our work we give testers for this problem for general constant d in the general asymmetric domain setting with the range being an arbitrary finite group $\mathcal{G}$, though with the restriction that the maximum set size $|\mathcal{S}_i|$ is bounded. We then use this tester to design our low-degree test over symmetric grids. We turn to our results below. Even though our primary motivation in studying low-*junta*-degree testing is to ultimately use it for low-degree testing, we note that junta-degree testing even for the case of $\mathcal{G}$ being the additive group of $\mathbb{R}$ (or $\mathbb{C}$) and $\mathcal{S}_i = \Omega$ (which is some finite set) for all i, is by itself already interesting as in this case, junta-degree corresponds to the "degree" of the Fourier representation of the function (in any basis). Low-*Fourier*-degree functions and such approximations form a central object in complexity theory and computational learning theory, at least when the domain size is $|\Omega| = 2$. The problem of *learning* low-Fourier-degree functions in particular has received much attention over the years [20, 21], and hence *testing* the same family, over general domains Ω, is an interesting corollary of our results, especially since our techniques are more algebraic than analytic (modulo the usage of a hypercontractivity theorem).

1.1 Our results

We start by stating our theorem for junta-degree testing. (For a formal definition of a tester, see Definition 7).

▶ **Theorem 1.** *The family of junta-degree-d functions from $\mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ to $\mathcal{G}$ is locally testable with a non-adaptive one-sided tester that makes $O_{s,d}(1)$ queries to the function being tested, where $s = \max_i |\mathcal{S}_i|$.*

In the special case where $|\mathcal{S}_i| = s$ for all i, the tester makes $s^{O(s^2 d)}$ queries.

In particular, if we treat all the parameters above except n as constant, this gives a test that succeeds with high probability by making only a constant number of queries. Taking $(\mathcal{G}, +) = (\mathbb{R}, +)$ or $(\mathbb{C}, +)$, the above theorem results in a local tester for Fourier-degree:

▶ **Corollary 2.** *The family of functions $f : \Omega^n \to \mathbb{R}$ of Fourier-degree at most d is locally testable in $s^{O(s^2 d)} = O_{s,d}(1)$ queries, where $s = |\Omega|$.*[2]

We now turn to the question of testing whether a given function $f : \mathcal{S}^n \to \mathbb{F}$ is *degree-d*, i.e., whether there is a polynomial of degree at most d agreeing with f, or δ-far from it. Here $\mathcal{S}$ can be any arbitrary finite subset of the field. Note that being junta-degree-d is a necessary condition for f being degree-d. Combining the above JUNTA-DEG with an additional test (called WEAK-DEG), we can test low-degree functions over a field, or rather over *any subset* of a field.

[2] The same result also holds if the co-domain is $\mathbb{C}$ instead of $\mathbb{R}$.

▶ **Theorem 3.** *For any subset $\mathcal{S} \subseteq \mathbb{F}$ of size s, the family of degree-d functions from $\mathcal{S}^n$ to $\mathbb{F}$ is locally testable with a non-adaptive, one-sided tester that makes $(sd)^{O(s^3 d)} = O_{s,d}(1)$ queries to the function being tested.*

The special case of $\mathcal{S} = \mathbb{F} = \mathbb{F}_q$ (finite field of size q) is especially interesting. Although this was already established for general finite fields first by Kaufman and Ron [19] and an optimal query complexity (in terms of d, for constant prime q) was achieved by Haramaty, Shpilka and Sudan [15], we nevertheless present it as a corollary of Theorem 3.

▶ **Corollary 4** (Kaufman and Ron [19]). *The family of degree-d functions $f : \mathbb{F}_q^n \to \mathbb{F}_q$ is locally testable in $(qd)^{O(q^3 d)} = O_{q,d}(1)$ queries.*

Turning our attention to more general product domains, we show that while junta-degree testing is still locally testable over there more general grids, testing degree in constantly many queries, even for $d = 1$, is intractable for all sufficiently large fields $\mathbb{F}$.

▶ **Theorem 5.** *For a growing parameter n, there exists a field $\mathbb{F}$ and its subsets $\mathcal{S}_1, \dots, \mathcal{S}_n$ of constant size (i.e., 3) such that testing the family of degree-1 functions $f : \mathcal{S}_1 \times \cdots \times \mathcal{S}_n \to \mathbb{F}$ requires $\Omega(\log n)$ queries.*

▶ Remark 6. A recent work of Arora, Bhattacharyya, Fleming, Kelman and Yoshida [4] considers low-degree testing over the reals and tests whether a given $f : \mathbb{R}^n \to \mathbb{R}$ is degree-d or ε-far with respect to a distribution $\mathcal{D}$. They give a test with query complexity independent of n for their problem ([4, Theorem 1.1]). This seems to contradict our result which seems to include the special case of their setting for $\mathcal{D} = \mathrm{Unif}(\mathcal{S}_1 \times \cdots \times \mathcal{S}_n)$ and $\mathbb{F} = \mathbb{R}$, where Theorem 5 shows that a dependence on n is necessary. The seeming contradiction is resolved by noting that the models in our paper and that of [4] are quite different. In particular, while in our setting the function f can only be queried on the support of the distribution $\mathcal{D}$ (namely $\mathcal{S}_1 \times \cdots \times \mathcal{S}_n$), in [4] the function can be queried at any point in $\mathbb{R}^n$ and the distribution $\mathcal{D}$ only shows up when defining the distance between two functions. (So in their model a function f that happens to agree with a degree d polynomial on the support of $\mathcal{D}$ but disagrees outside the support may be rejected with positive probability, while in our model such a function must be accepted with probability one.)

1.2 Technical contributions

All low-degree tests roughly follow the following pattern: Given a function f on n variables $x_1, \dots, x_n$ they select some $k = O_d(1)$ new variables $y = (y_1, \dots, y_k)$ and substitute $x_i = \sigma_i(y)$, where σ_i's are simple random functions, to get an $O(1)$-variate function $g(y) = f(\sigma(y))$; and then verify g is a low-degree polynomial in k variables by brute force. When the domain is $\mathbb{F}_q^n$ for some field $\mathbb{F}_q$, σ_i's can be chosen to be an affine form in y – this preserves the domain and ensures degree of g is at most the degree of f, thus at least ensuring completeness. While soundness of the test was complex to analyze, a key ingredient in the analysis is that for any pair of points $a \neq b \in \mathbb{F}_q^k$, $\sigma(a)$ and $\sigma(b)$ are uniform independent elements of $\mathbb{F}_q^n$ (over the randomness of σ). At least in the case where f is roughly $1/q^k$ distance from the degree d family, this ensures that with constant probability g will differ from a degree d polynomial in exactly one point making the test reject. Dealing with cases where f is much further away is the more complex part that we won't get into here.

When the domain is not $\mathbb{F}_q^n$ affine substitutions no longer preserve the domain and so we can't use them in our tests. In the cases of the domain being $\{-1, +1\}^n$, [7] used much simpler affine substitutions of the form $x_i = c_i y_{j(i)}$ where $c_i \in \{-1, +1\}$ uniformly and independently over i and $j(i) \in \{1, \dots, k\}$ uniformly though not independently over i. Then [7] iteratively reduce the number of variables as follows: When only r variables $x_1, \ldots, x_r$ remain, they pick two uniformly random indices $i \neq j \in \{1, \ldots, r-1\}$ and identify x_j with x_i, and then rename the $r-1$ remaining variables as $x_1, \ldots, x_{r-1}$. At the end when $r = k$, they pick a random bijection between $x_1, \ldots, x_k$ and $y_1, \ldots, y_k$. This iterative identification eventually maps every variable x_i to some variable $y_{j(i)}$. The nice feature of this identification scheme is it leads to a sequence of functions $f_n, f_{n-1}, ..., f_k$ with f_r being a function of r variables on the same domain, and of degree at most d if $f = f_n$ has degree at most d. If however we start with f_n being *very* far from degree d polynomials, there must exist r such that f_r is very far from high-degree functions while f_{r-1} is only *moderately* far. The probability of a bad event can be bounded (via some algebraic arguments) by $O(d^2/r^2)$. This step is the key to this argument and depends on the fact that f_{r-1} involves very small changes to f_r. Summing over r then gives the constant probability that the final function f_k (or equivalently g) is far from degree d polynomials. This still leaves [7] with the problem of dealing with functions f that are close to codewords: Here they use the fact that this substitution ensures that $\sigma(a)$ is distributed uniformly in $\{-1,+1\}^n$ for every $a \in \{-1,+1\}^k$. It is however no longer true that $\sigma(b)$ is uniform conditioned on $\sigma(a)$ for $b \neq a$, but it is still the case that if b is moderately far in Hamming distance from a then $\sigma(b)$ has sufficient entropy conditioned on $\sigma(a)$. (Specifically $\sigma(b)$ is distributed uniformly on a sphere of distance $\Omega(n)$ from $\sigma(a)$.) This entropy, combined with appropriate small-set expansion bounds on the Boolean hypercube, and in particular a spherical hypercontractivity result due to Polyanskiy [23]), ensures that if f is somewhat close to a low-degree polynomial then g is far from every degree d polynomial on an appropriately chosen subset of $\{-1,+1\}^k$ and so the test rejects.

To extend this algorithm and analysis to the setting on non-Boolean domains we are faced with two challenges: (1) We cannot afford to negate variables (using the random variables $c(i)$ above) when the domain is not $\{-1,+1\}$ – we can only work with identification of variables (or something similar). (2) The increase in the domain size forces us to seek a general spherical hypercontractivity result on non-Boolean alphabets and this is not readily available. Overcoming either one of the restrictions on its own seems plausible, but doing it together (while also ensuring that the sequence of restrictions/identifications do not make the distance to the family being tested to abruptly drop in distance as we go from $f_n, f_{n-1}, \ldots$ to f_k) turns out to be challenging and this is where we find it critical to go via junta-degree testing.

As a first step in our proof we extend the approach of [7] to junta-degree testing over the domain $\mathcal{S}^n$ for arbitrary finite $\mathcal{S}$. (It is relatively simple to extend this further to the case of $\mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ – we don't discuss that here.) This is achieved by using substitutions of the form $x_i = \pi_i(y_{j(i)})$ where $\pi_i : \mathcal{S} \to \mathcal{S}$ is a random bijection. While this might increase the degree of the function, this preserves the junta-degree (or reduces it) and makes it suitable for analysis of the junta-degree test, which we now describe: Following the template of a low-degree test stated at the beginning of this subsection, the junta-tester would simply check whether $g(y) = f(\sigma(y))$ is of junta-degree at most d where σ is the random function induced by the identifications $j(.)$ and permutations π_i of variables. The permutations π_i here serve the same purpose as the coefficients c_i's do in the substitutions $x_i = c_i y_{j(i)}$ of [7] which is to ensure that for any $a \in \mathcal{S}^k$, $\sigma(a)$ is uniformly distributed in $\mathcal{S}^n$. With this idea in place extending the analysis of [7] to our setting ends up with a feasible path, except we had to address a few more differences; one such challenge is that in the analysis the rejection probability of junta-degree test on functions that are close to being junta-degree-d, we will need to analyze the effect of a spherical noise operator on grids (i.e., a subset of coordinates of fixed size is chosen uniformly at random and each coordinate in that subset is changed to a different value uniformly at random). While [23] shows that such a noise operator has the desired hypercontractivity behavior, and the corresponding small-set expansion theorem was used in the test of [7], this was only for a Boolean alphabet. In this paper, when the alphabet size $s = |\mathcal{S}|$ is more than 2, by doing Fourier analysis over $\mathbb{Z}_s^n$, we are able to relate it to the more standard Bernoulli i.i.d. noise operator for which we do have a small-set expansion theorem available – we believe this can be of independent interest.[3]

The other differences of our junta-degree test analysis compared to that of [7] are mainly to account for the fact that we are aiming for junta-degree testing over any (abelian) group whereas the low-degree testing ideas of [7] and other prior work utilize the properties of polynomials over fields. We also give a cleaner proof as compared to [7] for the fact that the sequence of functions of fewer and fewer variables obtained by the random identifications (along with permutations) does not abruptly decrease in distance to the junta-degree-d family like we pointed out earlier (see "large-distance lemma" Lemma 16).

We then return to the task of low-degree testing: For this we design a new test: We first test the given oracle for junta-degree d, then if it passes, we pick a fresh random identification scheme setting $x_i = y_{j(i)}$ for uniform independent $j(i) \in \{1, \ldots, k\}$ and verify (by brute-force) that the resulting k variate function has degree at most d. The advantage with this two stage tester is that in the second stage the given function is already known to be close to a polynomial of degree at most sd where $s = \max_i\{|\mathcal{S}_i|\}$. This makes the testing problem closer to a polynomial identity testing problem, though the problem takes some care to define, and many careful details to be worked out in the analysis. A particular challenge arises from the fact that the first phase only proves that our function is only *close* to a low-degree polynomial and may not be low-degree exactly – so in the second stage we have to be careful to sample the function on essentially uniform inputs. This prevents us from using all of $\mathcal{S}^k$ when looking at the restricted function $g(y)$, but only allows us to use balanced inputs in $\mathcal{S}^k$ (where a balanced input has an equal number of coordinates with each value $v \in \mathcal{S}$). In turn understanding what the lowest degree of function can be given its values on a balanced set leads to new algebraic questions. Section 4 gives a full proof of the low-degree test and analysis spelling out the many technical questions and our solutions to those.

The final testing-related result we prove is an impossibility result, showing that while low-degree functions are locally testable over $\mathcal{S}^n$, this cannot be extended to general grids $\mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ for large enough fields (Theorem 5). From a coding theory perspective, this reveals that local testability of even polynomial evaluations codes requires more structure than simply having a large distance. To sketch the idea, let $d = 1$ and $\mathbb{F}$ be any field of size at least $n + 2$ with distinct elements $\{0, 1, a_1, \ldots, a_n\}$. For $i \in [n]$, let $\mathcal{S}_i = \{0, 1, a_i\}$. We will refer to $0, 1$ as Boolean elements and the remaining as non-Boolean ones. Let $\zeta(b) = b$ if b is Boolean and $\star$ otherwise. As degree-1 functions over $\mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ form a linear subspace over $\mathbb{F}$, by a result due to Ben-Sasson, Harsha and Raskhodnikova [9] any test can be converted to a one-sided, non-adaptive one without changing the number of queries or the error by more than a factor of 2. Thus, we may assume that the test (call it TEST) is of the following form: TEST samples a matrix $M \in \mathbb{F}^{\ell \times n}$ according to a distribution $\mathcal{D}$ with rows $x^{(1)}, \ldots, x^{(\ell)} \in \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ and accepts f if and only if $P(f(x^{(1)}), \ldots, f(x^{(\ell)}))$ is true where P is some fixed predicate. We will show that if TEST accepts degree-1 functions with probability 1 and rejects $\Omega(1)$-far functions with probability $\Omega(1)$, then $\ell = \Omega(\log n)$. By the one-sidedness, we must have for all $M \in \sup(\mathcal{D})$ that if $f_M := (f(x^{(1)}), \ldots, f(x^{(\ell)})) \in \text{colspace}(M)$, then TEST accepts, where sup denotes the support and colspace denotes the column space. Picking $i \in [n]$ uniformly at random, note that the function $g(x) := x_i(x_i - 1)$ is $\Omega(1)$-far from degree-1.

[3] We note that the hypercontractivity setting we are considering and analyzing in this part is not sufficient to get a direct analysis of low-degree testing. Such an analysis would require hypercontractivity for more delicate noise models than the simpler "q-ary symmetric" models we analyze here.

Then we argue that the "evaluation vector" $g_M := (g(x^{(1)}), \ldots, g(x^{(\ell)}))$ lies in the column space of M with high probability (over i) if $\ell = o(\log n)$, so Test faultily accepts g. The idea is that if $\ell = o(\log n)$ then there are at least two columns (say $i \neq j \in [n]$) of M that are identical under the ζ mapping, so subtracting one from the other gives a vector in $\mathbb{F}^\ell$ that is (up to a constant factor) equal to g_M. This is because for $k \in [\ell]$, the k-th coordinate of the difference vector is $a_i - a_j$ (some constant) if $\zeta(x_i^{(k)}) = \star$ and 0 otherwise. Similarly the k-th coordinate of g_M is $a_i(a_i - 1)$ (some constant) if $\zeta(x_i^{(k)}) = \star$ and 0 otherwise.

2 Preliminaries

We denote $[n] = \{1, \ldots, n\} \subseteq \mathbb{Z}$, $[m..n] = [n] \setminus [m-1]$ and $\mathbb{Z}_s = \mathbb{Z}/s\mathbb{Z} = \{0, 1, \ldots, s-1\}$ for $s \geqslant 2$. Throughout the paper, let $(\mathcal{G}, +)$ be an arbitrary abelian group and $(\mathbb{F}, +, \cdot)$ an arbitrary field. $\mathbb{F}_q^n$ is a vector space over the finite field of q elements, to which we associate an inner product (bilinear form) as: $\langle x, y\rangle = \sum_{i=1}^n x_i \cdot y_i$.

For any finite set $\mathcal{S}$ and $a \in \mathcal{S}^n$ we denote the Hamming weight of a by $\#a = \{i \in [n] : a_i \neq 0\}$, assuming $\mathcal{S}$ contains an element called 0. If $I \subseteq [n]$, we use a^I to denote the tuple a restricted to the coordinates of I, i.e., $a^I = (a_i)_{i \in I}$. Similarly $\mathcal{S}^I = \{a^I : a \in \mathcal{S}^n\}$. For disjoint subsets $I, J \subseteq [n]$, and $a \in \mathcal{S}^I$ and $b \in \mathcal{S}^J$, we denote their concatenation by $a \circ b \in \mathcal{S}^{I \cup J}$. Denoting a product domain/grid by $\overline{\mathcal{S}} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$, we let $\overline{\mathcal{S}}^I = \times_{i \in I} \mathcal{S}_i$ denote the Cartesian product of sets restricted to the coordinates of I.

We use $\binom{[n]}{\leqslant d}$ to denote the set of subsets of $[n]$ of size at most d. For m a multiple of s, let "balanced set" $\mathcal{B}(\mathcal{S}, m) \subseteq \mathcal{S}^m$ be the set of points that contain exactly m/s many repetitions of each element of $\mathcal{S}$. Abusing notation, sometimes we may think of $\mathcal{B}(\mathcal{S}, m)^{m'}$ as a subset of $\mathcal{S}^{mm'}$ by flattening the tuple of m-tuples. The *group-integer multiplication* operation $\cdot : \mathcal{G} \times \mathbb{Z} \to \mathcal{G}$ is defined by $g \cdot m = g + \cdots + g$ ($|m|$ times) if $m \geqslant 0$ and $-g - \cdots - g$ ($|m|$ times) otherwise.

2.1 Local testability

The *distance* between $f : \overline{\mathcal{S}} \to \mathcal{G}$ and a family of functions $\mathcal{F}$ with the same domain $\overline{\mathcal{S}}$ is $\delta_{\mathcal{F}}(f) = \min_{g \in \mathcal{F}} \delta(f, g)$, where $\delta(f, g) = \Pr_{x \sim \overline{\mathcal{S}}}[f(x) \neq g(x)]$. We say that f is δ-*far* from $\mathcal{F}$ if $\delta_{\mathcal{F}}(f) \geqslant \delta$. When $\mathcal{F}$ is the family of junta-degree-d functions, we denote $\delta_{\mathcal{F}}(.)$ by simply $\delta_d(.)$. Similarly f is δ-*close* to $\mathcal{F}$ if $\delta_{\mathcal{F}}(f) \leqslant \delta$.

▶ **Definition 7** (Local testability). *A randomized algorithm $\mathcal{A}$ with an oracle access to a function $f : \overline{\mathcal{S}} \to \mathcal{G}$ as its input, is said to be* q-local *if it performs at most q queries for any given f. For a family of functions $\mathcal{F}$ with domain $\overline{\mathcal{S}}$ and co-domain $\mathcal{G}$, we say that $\mathcal{F}$ is* q-locally testable *for $q = q(\mathcal{F})$ if there exists a q-local test $\mathcal{A}$ that accepts f with probability* 1 *if $f \in \mathcal{F}$, and rejects f with probability at least $\delta_{\mathcal{F}}(f)/2$ if $f \notin \mathcal{F}$. Further if $q(\mathcal{F}) = O(1)$ (i.e., independent of the number of variables n), we simply refer to $\mathcal{F}$ as being* locally testable.

One can define more general two-sided error and adaptive tests, but in the context of this paper, the above definition for local testability is without loss of generality as we know from the work of Ben-Sasson, Harsha and Raskhodnikova [9] that for *linear properties*[4], any "test" can be transformed to be one-sided and non-adaptive without altering the query complexity (locality) and success probability by more than constant factors.

[4] i.e., for families $\mathcal{F}$ for which $f \in \mathcal{F}$ and $g \in \mathcal{F}$ implies $c_1 f + c_2 g \in \mathcal{F}$ for all $c_1, c_2 \in \mathbb{F}$.

The family of functions $\mathcal{F}$ of our study (namely "junta-degree-d" and "degree-d" to be formally defined shortly) are parameterized by $s = \max_i |\mathcal{S}_i|$ and an integer d which we treat as constants. All tests we are going to present are $O_{s,d}(1)$-local, one-sided and non-adaptive. However, the probability of rejection in case of $f \notin \mathcal{F}$ is only $\Omega_{s,d}(\delta_{\mathcal{F}}(f))$; nevertheless by repeating the test an appropriate $O_{s,d}(1)$ number of times, we get a $O_{s,d}(1)$-local test for $\mathcal{F}$ that succeeds with probability $\delta_{\mathcal{F}}(f)/2$ when $f \notin \mathcal{F}$ and with probability 1 when $f \in \mathcal{F}$.

2.2 Junta-polynomials and polynomials

For this section, we let $\overline{\mathcal{S}} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ denote an arbitrary finite product domain (or grid) and $s = \max_i \{s_i\}$, where $s_i = |\mathcal{S}_i|$.

▶ **Definition 8** (Junta-degree). *A function $f : \overline{\mathcal{S}} \to \mathcal{G}$* [5] *is said to be* junta-degree-d *if*

$$f(x) = f_1(x^{D_1}) + \cdots + f_t(x^{D_t})$$

for some $t \in \mathbb{Z}, D_j \in \binom{[n]}{\leqslant d}$ and functions $f_j : \overline{\mathcal{S}}^{D_j} \to \mathcal{G}$ for $j \leqslant t$. If $t = 1$, we call f a d-junta.

The junta-degree *of f is the minimum $d \geqslant 0$ such that f is junta-degree-d.*

For junta-degree testing over arbitrary grids $\overline{\mathcal{S}} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$, we may assume that $\mathcal{S}_i = \mathbb{Z}_{s_i}$ without loss of generality, where $s_i = |\mathcal{S}_i|$. The following claims about junta-polynomials are analogous to standard facts about multi-variate polynomials over a field.

▷ Claim 9. Any junta-degree-d function $f : \mathbb{Z}_s^n \to \mathcal{G}$ can be uniquely[6] expressed as

$$f(x_1, \ldots, x_n) = \sum_{\substack{a \in \mathbb{Z}_s{}^n \\ \#a \leqslant d}} g_a \cdot \prod_{i \in [n]:\ a_i \neq 0} \delta_{a_i}(x_i), \tag{1}$$

where $g_a \in \mathcal{G}$ and $\delta_b : \mathbb{Z}_s \to \mathbb{Z}$ is defined as $\delta_b(y) = 1$ if $b = y$ and 0 otherwise.

▶ **Definition 10** (Junta-polynomial). *We will call such a representation as a* junta-polynomial*, and the* degree *of a junta-polynomial is defined as $\max_{a \in \mathbb{Z}_s{}^n : g_a \neq 0} \#a$. It can be seen that the degree of a junta-polynomial is exactly equal to the junta-degree of the function it computes, assuming that the degree of the identically 0 junta-polynomial is 0.*

We will refer to the summands in (1) as terms*, the constants g_a as* coefficients*, the integer products $\prod_{i \in [n]:\ a_i \neq 0} \delta_{a_i}(x_i)$ as* monomials*. We say that a is a* root *of a junta-polynomial P if $P(a) = 0$ and a is a* non-root *otherwise.*

▷ Claim 11. Any non-zero junta-polynomial $P : \mathbb{Z}_s^n \to \mathcal{G}$ of degree at most d has at least s^{n-d} non-roots.

We will now discuss standard facts about formal polynomials. Let $\mathbb{F}$ be a field and $\mathcal{S} \subseteq \mathbb{F}$ be of size $s \geqslant 2$. For a polynomial $P(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ the individual degree of x_i is the largest degree x_i takes in any (non-zero) monomial of P. The individual degree of P is the largest individual degree of any variable x_i. We say that P is *degree-d* if its degree is at most d. We say that $f : \mathcal{S}^n \to \mathbb{F}$ is degree-d iff there is a degree-d polynomial $P \in \mathbb{F}[x_1, \ldots, x_n]$ computing f. For the analysis of our degree-tester, we also need a notion of degree-d for non-product domains: for any $\mathcal{T} \subseteq \mathcal{S}^n$, we say that $f : \mathcal{T} \to \mathbb{F}$ is degree-d if there is a degree-d polynomial $P \in \mathbb{F}[x_1, \ldots, x_n]$ computing f.

[5] Here we treat a tuple of sets as the domain of the function

[6] up to the commutativity of the Σ (group addition) and Π (integer multiplication) operations

▷ Claim 12. Any degree-d function $f : \mathcal{S}^n \to \mathbb{F}$ has a unique polynomial representation with degree at most d and individual degree at most $s-1$.

By setting $d = n(s-1)$ (or ∞) in the above claim, we see that the set of all functions from $\mathcal{S}^n$ to $\mathbb{F}$ is a vector space over $\mathbb{F}$ of dimension s^n – the monomials with individual degree at most $s-1$ form a basis. More generally, for any $\mathcal{T} \subseteq \mathcal{S}^n$ the set of functions from $\mathcal{T}$ to $\mathbb{F}$ forms a vector space of dimension $|\mathcal{T}|$ with an inner product defined for $f, g : \mathcal{T} \to \mathbb{F}$ as $\langle f, g\rangle = \sum_{x\in\mathcal{T}} f(x)\cdot g(x)$. For any d, the set of degree-d functions is a subspace of this vector space.

It is easy to see that if $f : \mathcal{S}^n \to \mathbb{F}$ is degree-d, then it is also junta-degree-d (w.r.t. to the additive group of $\mathbb{F}$). Conversely, if $f : \mathcal{S}^n \to \mathbb{F}$ is junta-degree-d, then it is degree-$(s-1)d$: this follows by applying Claim 12 to the d-junta components of f. If $s = 2$, the degree is exactly equal to the junta-degree.

Let $\delta'_d(f)$ denote the distance of f to the degree-d family.

2.3 Fourier analysis

▶ **Definition 13** (Fourier representation). *Any function $f : \mathbb{Z}_s^n \to \mathbb{C}$ can be uniquely expressed as*

$$f(x) = \sum_{\alpha\in\mathbb{Z}_s^n} \widehat{f}(\alpha)\chi_\alpha(x) \tag{2}$$

where the characters *are defined as $\chi_\alpha(x) = \prod_{i\in[n]} \chi_{\alpha_i}(x_i)$ where $\chi_\beta(y) = \omega^{\beta y \mod s}$ for $\beta, y \in \mathbb{Z}_s$ and $\omega \in \mathbb{C}$ is a (fixed) primitive s-th root of unity.*

▶ **Definition 14** (Noise operators). *For $\nu \in [0,1]$ and $x \in \mathbb{Z}_s^n$, we define $N_\nu(x)$* [7] *to be the distribution over $\mathbb{Z}_s^n$ where each coordinate of x is unchanged with probability $1-\nu$, and changes to a* different *value uniformly at random with probability ν. Similarly, the* spherical noise *corresponds to $S_\nu(x)$ where a subset $J \subseteq [n]$ of fixed size νn is chosen uniformly at random and the coordinates outside J are unchanged and those within J are changed to a uniformly different value. Let $\mathcal{D}_\nu$ denote the probability distribution over $\mathbb{Z}_s$ with mass $1-\nu$ at 0 and $\nu/(s-1)$ at all the other points. Let $\mathcal{E}_\nu$ denote the uniform distribution over $\{y \in \mathbb{Z}_s^n : \#y = \nu n\}$. For $\mu_1 \sim \mathcal{D}_\nu^{\otimes n}$ and $\mu_2 \sim \mathcal{E}_\nu$, note that $N_\nu(x)$ and $x + \mu_1$ are identically distributed; so are $S_\nu(x)$ and $x + \mu_2$.*

3 Low-junta-degree testing

We note that junta-degree-d functions with domain $\mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ such that $|\mathcal{S}_i| = s$ for all i are "equivalent" to those with domain $\mathbb{Z}_s^n$ as one can fix an arbitrary ordering of elements in each $\mathcal{S}_i$ and treat the function as being over $\mathbb{Z}_s^n$: this does not change the junta-degree. Hence, we will fix $\mathcal{S}_i = \mathbb{Z}_s$. The more general case of unequal domain sizes is handled in the full version of the paper.

We claim that the following test works to check if a given function $f : \mathbb{Z}_s^n \to \mathcal{G}$ is junta-degree-d.

[7] This is different from the standard usage N_ρ where ρ denotes the probability of "retention" and not of noise.

The junta-degree test (Junta-deg)

For a parameter $k = O_{s,d}(1)$ that is yet to be fixed, the junta-degree test (which we shall refer to as Junta-deg) for $f : \mathbb{Z}_s^n \to \mathcal{G}$ is the following algorithm with $I = [n]$, $r = n$ and $f_r = f$:

Test $\boldsymbol{T_{I,r}(f_r)}$: gets query access to $\boldsymbol{f_r : \mathbb{Z}_s^I \to \mathcal{G}}$ where $\boldsymbol{I \subseteq [n]}$ is of size $\boldsymbol{r}$.

1. If $r \leqslant k$, accept iff f_r is junta-degree-d (check this by querying f_r at all points in its domain). Otherwise,
2. Choose $i \neq j \in I$ and a permutation $\pi_j : \mathbb{Z}_s \to \mathbb{Z}_s$ independently and uniformly at random. Let $I' = I \setminus \{j\}$.
3. Apply the test $T_{I',r-1}(f_{r-1})$ where $f_{r-1} : \mathbb{Z}_s^{I'} \to \mathcal{G}$ is the function obtained by setting $x_j = \pi_j(x_i)$ in f_r: that is, $f_{r-1}(a^{I'}) := f_r(a^{I'} \circ (\pi_j(a_i))^{\{j\}})$ for $a \in \mathcal{S}^{I'}$.

The query complexity of the Junta-deg test is $s^k = O_{s,d}(1)$ regardless of the randomness within the test. Furthermore, if the function f happens to be a junta-degree-d function, then the test Junta-deg always accepts it, since permuting variables and substituting some variables with other variables does not change the junta-degree, so Step 1 succeeds. In this section, we will show that if $\delta := \delta_d(f) > 0$, then $\Pr[\text{Junta-deg rejects } f] \geqslant \varepsilon\delta$ for appropriate $\varepsilon = \Omega_{s,d}(1)$.

We follow the same approach as [7] (which itself follows [10]) and argue that if $\delta_d(f)$ is "small", then we will be able to prove $\Pr[\text{Junta-deg rejects } f] \geqslant \varepsilon{\cdot}\delta_d(f)$ and if not, at least we will be able to find some $r \in [k+1..n]$ such that $\delta_d(f_r)$ is small enough (but importantly, not too small). Then, we apply the small-distance analysis to that f_r.

We state here the two main lemmas to prove that the correctness of the junta-degree tester. Here, the parameters $\varepsilon_0 \leqslant \varepsilon_1$ and ε will be chosen to be at least $s^{-O(k)}$. In the context of the test $T_{I,r}(f_r)$ described above, we will set $k = \psi s^2 d$ for a sufficiently large but constant ψ to be fixed in the proofs of the below lemmas[8].

▶ **Lemma 15** (Small-distance lemma). *For any $I \subseteq [n]$ of size $r > k$, if $\delta = \delta_d(f_r) \leqslant \varepsilon_1$, then*

$$\Pr[T_{I,r} \textit{ rejects } f_r] \geqslant \varepsilon\delta.$$

▶ **Lemma 16** (Large-distance lemma). *For any $I \subseteq [n]$ of size $r > k$, if $\delta_d(f_r) > \varepsilon_1$, then*

$$\Pr_{i,j,\pi_j}[\delta_d(f_{r-1}) \leqslant \varepsilon_0] \leqslant k^2/2r(r-1).$$

Assuming the above two lemmas to be true, the proof of Theorem 1, at least for symmetric domains $\mathcal{S}_1 \times \cdots \times \mathcal{S}_n = \mathbb{Z}_s^n$, follows the same approach as in [7] and we omit it here. We also defer the proof of the large-distance lemma to the full version (and Appendix B). The case of junta-degree testing over *general* grids can be reduced to that of symmetric grids and we refer the reader to the full version for details.

3.1 Small-distance lemma

Proof of Lemma 15. We will "unroll" the recursion of the Junta-deg test and state it more directly as follows: Fix an arbitrary $r > k$. As r is fixed, we denote f_r by f (not to be confused with the initial function on n variables). For the proof, we will need the following alternate description of $T_{I,r}$ (subsequently, we shall drop the subscript I). Here, $\sigma : [r] \to [k]$ is a map chosen according to the following random process:

[8] For $d = 0$, we can take $k = \psi s^2$ so that it is non-zero.

- For $i = 1$ to k, set $\sigma(i) = i$.
- For $i = k+1$ to r, set $\sigma(i) = j$ with probability $|\{i' < i : \sigma(i') = j\}| / (i-1)$, for each $j \in [k]$.

The only property we need about the above distribution of σ is that it is "well-spread", which was already shown in [7] as the following lemma.

▶ **Lemma 17** (Corollary 6.9 in [7]). *With probability at least $1/2^{O(k)}$, we have $\left|\sigma^{-1}(j)\right| \geqslant r/4k$ for all $j \in [k]$ – we call such a σ* good.

Test $T_r(f)$: gets query access to $f : \mathcal{S}^{[r]} \to \mathcal{G}$ with variables $x_1, \ldots, x_r$.

1. Choose a tuple of permutations of $\mathbb{Z}_s$, $\pi = (\pi_1, \ldots, \pi_r)$ u.a.r.
2. Choose a bijection $\mu : [r] \to [r]$ u.a.r.
3. Choose a map $\sigma : [r] \to [k]$ according to the distribution described above Lemma 17.
4. For $y = (y_1, \ldots, y_k) \in \mathbb{Z}_s^k$, define $x_{\pi\sigma\mu}(y) = \left(\pi_1(y_{\sigma(\mu^{-1}(1))}), \ldots, \pi_r(y_{\sigma(\mu^{-1}(r))})\right)$.
5. Accept iff $f'(y) := f(x_{\pi\sigma\mu}(y))$ is junta-degree-d.

Let $\delta = \delta_d(f_r) = \delta(f, P) \leqslant \varepsilon_1$ where $P : \mathbb{Z}_s^{[r]} \to \mathcal{G}$ is junta-degree-d and $E \subseteq \mathcal{S}^r$ be the points where f and P differ. Our objective is to show that

$$\Pr_{\pi,\sigma,\mu}[T_r \text{ rejects } f] = \Pr_{\pi,\sigma,\mu}[f' \text{ is not junta-degree-}d] \geqslant \varepsilon\delta. \tag{3}$$

Let the functions $f' : \mathcal{S}^k \to \mathcal{G}$ and $P' : \mathcal{S}^k \to \mathcal{G}$ be defined by $f'(y) = f(x_{\pi\sigma\mu}(y))$ and $P'(y) = P(x_{\pi\sigma\mu}(y))$ respectively (these functions depend on π, σ, μ) and $E' \subseteq \mathcal{S}^k$ be the points where these two restricted functions differ.

To proceed further, we will need a subset U of $\mathbb{Z}_s^k$ with the following properties (we defer the proof to Appendix A and the full version):

▷ Claim 18. Let $w = \lceil \log(8\psi s^2)d \rceil < k$. There exists a set $U \subseteq \mathbb{Z}_s^k$ of size 2^w such that

1. (Code) For all $y \neq y' \in U$,

   $$k/4 \leqslant \Delta(y, y') \leqslant 3k/4$$

   where $\Delta(y, y')$ denotes the number of coordinates where y and y' differ.
2. (Hitting) No two junta-degree-d functions $P : \mathcal{S}^k \to \mathcal{G}$ and $Q : \mathcal{S}^k \to \mathcal{G}$ can differ at exactly one point in U.

Let $V = \{x_{\pi\sigma\mu}(y) : y \in U\} \subseteq \mathbb{Z}_s^r$. Because the mapping $y \mapsto x_{\pi\sigma\mu}(y)$ is one-one conditioned on σ being good, it holds that $|V \cap E| = |U \cap E'|$ under this conditioning. Now suppose the randomness is such that $|U \cap E'| = 1$. Then, since no two junta-degree-d functions can disagree at exactly one point in U (Property 2 of Claim 18), it must be the case that f' be of junta-degree greater than d (as P', being a restriction of a junta-degree-d function is already junta-degree-d). Therefore, for (3) we can set $\varepsilon := \Pr_\sigma[\sigma \text{ good}] \geqslant 1/2^{O(k)}$ and show

$$\Pr_{\pi,\sigma,\mu}[|U \cap E'| = 1 \mid \sigma \text{ good}] = \Pr_{\substack{\pi,\mu \\ \sigma \text{ good}}}[|U \cap E'| = 1] \geqslant \delta.$$

By a simple inclusion-exclusion, the above probability is

$$\Pr_{\substack{\pi,\mu \\ \sigma \text{ good}}}[|V \cap E| = 1] \geqslant \sum_{y \in U} \Pr_{\substack{\pi,\mu \\ \sigma \text{ good}}}[x_{\pi\sigma\mu}(y) \in E] - \sum_{y \neq y' \in U} \Pr_{\substack{\pi,\mu \\ \sigma \text{ good}}}[x_{\pi\sigma\mu}(y) \in E \text{ and } x_{\pi\sigma\mu}(y') \in E] \tag{4}$$

For any $y \in U$, $x_{\pi\sigma\mu}(y) = \big(\pi_1(y_{\sigma(\mu^{-1}(1))}), \ldots, \pi_r(y_{\sigma(\mu^{-1}(r))})\big)$ is uniformly distributed over $\mathbb{Z}_s^r$ since $\pi_1, \ldots, \pi_r$ are random permutations of $\mathbb{Z}_s$. Hence the first part of (4) is

$$\sum_{y \in U} \Pr[x_{\pi\sigma\mu}(y) \in E] = |U| \cdot \frac{|E|}{s^r} = |U| \cdot \delta. \tag{5}$$

For any fixed $y \neq y' \in U$ and good σ, using Property 1 of Claim 18 for points in U we claim that the random variables $x := x_{\pi\sigma\mu}(y)$ and $x' := x_{\pi\sigma\mu}(y')$ are related as follows:

▷ Claim 19. $x' \sim S_\nu(x)$, for some $\nu \in [1/32, 31/32]$.

For the second term of (4),

$$\Pr_{\substack{\pi,\mu \\ \sigma \text{ good}}}[x_{\pi\sigma\mu}(y) \in E \text{ and } x_{\pi\sigma\mu}(y') \in E] = \Pr_{\substack{x \sim \mathbb{Z}_s^r \\ x' \sim S_\nu(x)}}[x \in E \text{ and } x' \in E]$$

(for some $\nu \in [1/32, 31/32]$ depending on σ, using Claim 19)

$$\leqslant C \cdot \delta^{1+\lambda} \quad \text{for some constant } C \text{ and } \lambda = 1/2^{14} \log s. \tag{6}$$

(Using spherical noise small-set expansion (Theorem 23))

Plugging the bounds (5) and (6) back in (4), we get

$$\Pr_{\substack{\pi,\mu, \\ \sigma \text{ good}}}[|V \cap E| = 1] \geqslant |U|\,\delta - |U|^2\, C\delta^{1+\lambda} \geqslant |U|\,\delta/2 \geqslant \delta.$$

The above inequalities follow from $|U| = 2^w$ and $\delta \leqslant \varepsilon_1$; this is where we set $\varepsilon_1 := (1/2C\,|U|)^{1/\lambda} = (1/2C2^w)^{2^{14}\log s} \geqslant 1/s^{O(\log(8\psi s^2)d)} \geqslant 1/s^{O(k)}$. Hence we conclude that

$$\Pr_{\pi,\sigma,\mu}[T_r \text{ rejects } f] \geqslant \Pr_{\sigma}[\sigma \text{ good}] \cdot \Pr_{\substack{\pi,\mu \\ \sigma \text{ good}}}[|U \cap E'| = 1] \geqslant \varepsilon \Pr_{\substack{\pi,\mu \\ \sigma \text{ good}}}[|V \cap E| = 1] \geqslant \varepsilon\delta. \qquad ◀$$

4 Low-degree testing

We will describe our low-degree test now.

The degree test (Deg)

Given query access to $f : \mathcal{S}^n \to \mathbb{F}$, the following test (called Deg) works to test whether f is degree-d. We may assume that $s = |\mathcal{S}| \geqslant 2$ as f is a constant function otherwise.

Test Deg(f): gets query access to $f : \mathcal{S}^n \to \mathbb{F}$.
- Run Junta-deg(f) to check if f is junta-degree-d.
- Run Weak-deg(f).
- Accept iff both the above tests accept.

In the above description, the sub-routine Weak-deg corresponds to the following test.

Test Weak-deg(f)**: gets query access to** $f : \mathcal{S}^n \to \mathbb{F}$**.**

- Choose a map $\mu : [n] \to [K]$ u.a.r. where $K = t(d+1)$ and $t = s^3$.
- For $y = (y_1, \ldots, y_K) \in \mathcal{S}^K$, define $x_\mu(y) = (y_{\mu(1)}, \ldots, y_{\mu(n)})$.
- Define the function $f' : \mathcal{B}(\mathcal{S}, t)^{K/t} \to \mathbb{F}$ as $f'(y) = f(x_\mu(y))$, where $\mathcal{B}(\mathcal{S}, t)$, defined in Section 2, is the "balanced" subset of $\mathcal{S}^t$.
- Accept iff f' is degree-d.

We now move on to the analysis of this test, proving Theorem 3.

Proof of Theorem 3. Let $g : \mathcal{S}^n \to \mathbb{F}$ be a closest junta-degree-d function to f i.e., $\delta_d(f) = \delta(f, g)$. We shall assume that $\delta_d(f) \leqslant \varepsilon_2$ for a small enough $\varepsilon_2 = K^{-O(K)}$, and neither f nor g are degree-d. Otherwise, we will be able to appeal to the correctness of Junta-deg.

Let $E \subseteq \mathcal{S}^n$ be the points where f and g differ; thus $|E|/s^n = \delta_d(f) \leqslant \varepsilon_2$. Let

$$V_\mu = \left\{ x_\mu(y) : y \in \mathcal{B}(\mathcal{S}, t)^{K/t} \right\}.$$

Suppose $\mu : [n] \to [K]$ is such that $V_\mu \cap E = \emptyset$ and Weak-deg rejects g. Then Weak-deg does not distinguish between f and g and hence rejects f as well. We will show that both these events occur with good probability. For the first probability, we will upper bound

$$\begin{aligned}\Pr_\mu\left[V_\mu \cap E \neq \emptyset\right] &= \Pr_\mu\left[\exists y \in \mathcal{B}(\mathcal{S}, t)^{K/t} : x_\mu(y) \in E\right] \\ &\leqslant \left|\mathcal{B}(\mathcal{S}, t)^{K/t}\right| \cdot \Pr_\mu\left[\text{For fixed arbitrary } y \in \mathcal{B}(\mathcal{S}, t)^{K/t}, x_\mu(y) \in E\right].\end{aligned}$$

Note that since all points in $\mathcal{B}(\mathcal{S}, t)^{K/t}$ contain an equal number of occurrences of all the elements of $\mathcal{S}$, $x_\mu(y)$ is uniformly distributed in $\mathcal{S}^n$ for a uniformly random μ. Hence, the above probability is

$$\Pr_\mu\left[V_\mu \cap E \neq \emptyset\right] \leqslant |\mathcal{S}|^K \cdot \frac{|E|}{s^n} \leqslant s^K \varepsilon_2 < \frac{1}{2K^d}. \qquad (\text{by setting } \varepsilon_2 := 1/4s^K K^d \geqslant K^{-O(K)})$$

We show that Weak-deg indeed rejects g with good probability.

▷ Claim 20. $\Pr_\mu[\text{Weak-deg rejects } g] \geqslant 1/K^d$.

Assuming this claim,

$$\begin{aligned}&\Pr[\text{Deg rejects } f] \geqslant \Pr[\text{Weak-deg rejects } f] \\ &\qquad \geqslant \Pr[\text{Weak-deg rejects } g] - \Pr[V_\mu \cap E \neq \emptyset] \geqslant \frac{1}{2K^d} \geqslant \frac{\delta_d'(f)}{2K^d}.\end{aligned}$$

This finishes the analysis of the low-degree test assuming Claim 20. ◀

4.1 Soundness of Weak-deg

Proof of Claim 20. We will need the following lemma about the vector space formed by functions over $\mathcal{B}(\mathcal{S}, t)^{K/t} \subseteq \mathcal{S}^K$.

▶ **Lemma 21.** *For $\mathcal{T} \subseteq \mathcal{S}^K$, the vector space of functions from $\mathcal{T}$ to $\mathbb{F}$ has a basis $\{m_1, \ldots, m_\ell\}$ such that for any $f : \mathcal{T} \to \mathbb{F}$ of the form $f = c_1 m_1 + \ldots c_\ell m_\ell$ for some $c_i \in \mathbb{F}$, we have*

$$f \text{ is degree-}d \iff \forall i,\ c_i = 0 \text{ or } m_i \text{ is degree-}d. \qquad (7)$$

Let $g' : \mathcal{B}(\mathcal{S},t)^{K/t} \to \mathbb{F}$ be defined as $g'(y) = g(x_\mu(y))$ where $x_\mu(y) = (y_{\mu(1)}, \dots, y_{\mu(n)})$. Then, recall that WEAK-DEG rejects g iff g' is not degree-d. We use Lemma 21 above to fix a suitable basis $m_1, \dots, m_\ell$ for functions from $\mathcal{T} = \mathcal{B}(\mathcal{S},t)^{K/t}$ to $\mathbb{F}$. Then we can write each $g'(y)$ obtained above uniquely as

$$g'(y) = \sum_{i=1}^{\ell} c_i \cdot m_i$$

where the coefficients c_i are some functions of μ. We will treat $\mu : [n] \to [K]$ as a random element of $[K]^n$.

We argue that each c_i is junta-degree-d (as a function of μ). To see this, we recall that g is junta-degree-d. Consider the case when g is a function of only $x_{i_1}, \dots, x_{i_s}$ for some $s \leqslant d$. In this case, clearly the polynomial g' depends only on $\mu_{i_1}, \dots, \mu_{i_s}$. In particular, each c_i is just an s-junta. Extending the argument by linearity, we see that for any g that is junta-degree-d, the underlying coefficients c_i of $g'(y)$ are junta-degree-d polynomials in the co-ordinates of μ.

Now assume that there exists a $\mu^* : [n] \to [K]$ such that $g'(y)$ is not degree-d (we will show the existence of such a "good" μ in the next subsection). Thus, by Lemma 21 there exists $i^* \in [\ell]$ such that m_{i^*} is not degree-d and $c_{i^*}(\mu^*) \neq 0$. In particular, the function c_{i^*} is non-zero.

We have argued that there is an m_{i^*} in the basis such that the associated coefficient c_{i^*} is a non-zero junta-degree-d polynomial. In particular, Claim 11 implies that the probability that $c_i^*(\mu) \neq 0$ for a random μ is at least $1/K^d$. Therefore, using Lemma 21,

$$\Pr_\mu[\text{WEAK-DEG rejects } g] = \Pr_\mu[g' \text{ is not degree-}d] \geqslant \Pr_\mu[c_{i^*}(\mu) \neq 0] \geqslant 1/K^d. \quad ◀$$

4.2 Existence of a good map μ

We will show for any function $g : \mathcal{S}^n \to \mathbb{F}$ that is not degree-d, there exists a map $\mu : [n] \to [K]$ such that the function $g'(y) = g(x_\mu(y))$ defined for $y \in \mathcal{B}(\mathcal{S},t)^{K/t}$ is also not degree-d. This is easy to prove if the domain of g' were to be $\mathcal{S}^K$, but is particularly tricky in our setting.

Let $D = d + 1$. We will give a map $\mu : [n] \to [t] \times [D] \equiv [tD] = [K]$ instead. Let P be the polynomial with individual degree at most $s - 1$ representing g; suppose the degree of P is $d' > d$ and let $m(x) = c \cdot x_{i_1}^{a_1} \cdots x_{i_\ell}^{a_\ell}$ be a monomial of $P(x)$ of degree d' for some non-zero $c \in \mathbb{F}$, where $a_j \geqslant 1$ for all j and $i_1, \dots, i_\ell$ are some distinct elements of $[n]$ and $\ell \leqslant d$ as g is junta-degree-d. Then, we define μ as follows for $i \in [n]$:

$$\mu(i) = \begin{cases} (1,j), & \text{if } i = i_j \text{ for some } j \in [\ell] \\ (1,D), & \text{otherwise.} \end{cases}$$

It is easy to inspect that $P(x_\mu(y))$ (call it $Q(y)$) is a polynomial in variables $y_{(1,1)}, \dots, y_{(1,D)}$, and is of degree $d' > d$ – this is because the monomial $m(x)$, upon this substitution turns to

$$m(x_\mu(y)) = c \cdot y_{(1,1)}^{a_1} \cdots y_{(1,\ell)}^{a_\ell},$$

which cannot be cancelled by $m'(x_\mu(y))$ for any other monomial $m'(x)$ of $P(x)$, as if m' contains the variable x_i for some $i \notin \{i_1, \dots, i_\ell\}$ then $m'(x_\mu(y))$ contains the variable $y_{(1,D)}$ and on the other hand if m' only contains variables x_i for some $i \in \{i_1, \dots, i_\ell\}$, then the individual degree of $y_{(1,j)}$ in the two substitutions differs for some j. Hence, the degree of $Q(y)$ is $a_1 + \cdots + a_\ell = d'$. As we can express the function $y_{(1,D)}^a$ for $a > s - 1$ as a polynomial in $y_{1,D}$ of individual degree at most $s-1$, we can further transform $Q(y)$ so that it has individual degree at most $s-1$, while maintaining the properties that it still only contains the variables $y_{(1,1)}, \ldots, y_{(1,D)}$ (i.e., the first "row") and has degree d' and computes the function $g'(y)$. The following claim then completes the proof of the existence of a good μ by setting $w = D$ and $d' = d'$.

▷ Claim 22. For formal variables $y \equiv (y_1, \ldots, y_w) \equiv (y_{(i,j)})_{(i,j)\in[t]\times[w]}$, let $Q(y)$ be a polynomial of degree $d' \geqslant 0$ containing only the variables from the first row. Then the degree of $Q(y)$ as a function over $\mathcal{B}(\mathcal{S}, t)^w$ is exactly d'.

The proof of the above claim is deferred to Appendix C.

5 Small-set expansion for spherical noise

In this section, we will prove a small-set expansion theorem for spherical noise, which we have used for (6) in the proof of the small-distance lemma of junta-degree testing. Let $f : \mathbb{Z}_s^n \to \mathbb{C}$ be arbitrary. For $\nu \in [0,1]$ the Bernoulli noise operator is defined as $N_\nu f(x) = \mathbb{E}_{y\sim N_\nu(x)}[f(y)] = \mathbb{E}_{y\sim \mathcal{D}_\nu^{\otimes n}}[f(x+y)]$. Similarly, the spherical noise operator is defined for $\nu \in [0,1]$ such that $\nu n \in \mathbb{Z}$: $S_\nu f(x) = \mathbb{E}_{y\sim S_\nu(x)}[f(y)] = \mathbb{E}_{y\sim\mathcal{E}_\nu}[f(x+y)]$. For the rest of this section, let $\rho \in [0,1]$ and $\nu = (1-1/s)(1-\rho) \in [0,1]$; it is easy to check that if each coordinate of x is retained with probability ρ and randomized (uniformly over $\mathbb{Z}_s$) with probability $1-\rho$, the resulting string is distributed according to $N_\nu(x)$.

The goal of this section is to show for $s \geqslant 3$, that we can reduce the problem of small-set expansion for spherical noise to that of Bernoulli noise, for which such a theorem is already known.

▶ **Theorem 23** (Small-set expansion for spherical noise). *Let $s \geqslant 3$ and $A \subseteq \mathbb{Z}_s^n$ be such that $\Pr_{x\sim\mathbb{Z}_s^n}[x \in A] = \delta$. Then, for any $\nu \in [1/32, 1]$*

$$\Pr_{\substack{x\sim\mathbb{Z}_s^n\\ y\sim S_\nu(x)}}[x \in A \text{ and } y \in A] \leqslant 2 \cdot \delta^{1+\lambda}, \tag{8}$$

where $\lambda = \frac{1}{2^{14}\log s}$.

▶ Remark. When the size of the domain s is equal to 2 and $\nu \in [1/32, 31/32]$, the above statement still holds (with the factor 2 replaced with some other constant factor C) as proved by [23] (or Corollary 2.8 in [7]).

Proof of Theorem 23. Let $f : \mathbb{Z}_s^n \to \mathbb{C}$ be the indicator function of A and consider its Fourier representation as in Definition 13: $f(x) = \sum_{\alpha\in\mathbb{Z}_s^n} \widehat{f}(\alpha)\chi_\alpha(x)$. Then the probability in (8) is equal to

$$\Pr_{\substack{x\sim\mathbb{Z}_s^n\\ y\sim S_\nu(x)}}[x \in A \text{ and } y \in A] = \sum_{\alpha\in\mathbb{Z}_s^n} \left|\widehat{f}(\alpha)\right|^2 \mathop{\mathbb{E}}_{y\sim\mathcal{E}_\nu}[\chi_\alpha(y)]\,. \tag{9}$$

We will show that for any $\alpha \in \mathbb{Z}_s^n$, the quantity $\mathbb{E}_{y\sim\mathcal{E}_\nu}[\chi_\alpha(y)]$ above is upper bounded by $2 \cdot \widetilde{\rho}^{\#\alpha}$ for some constant $\widetilde{\rho}$. We have

$$\mathop{\mathbb{E}}_{y\sim\mathcal{E}_\nu}[\chi_\alpha(y)] = \mathop{\mathbb{E}}_{y\sim\mathcal{E}_\nu}[\chi_{\alpha_1}(y_1)\cdots\chi_{\alpha_n}(y_n)] = \mathop{\mathbb{E}}_{\substack{I\sim\binom{[n]}{\nu n}\\ \mu\sim\mathbb{Z}_s\setminus\{0\}\\ y\sim 0^{\overline{I}}\circ\mu^I}}\left[\prod_{i\in I}\chi_{\alpha_i}(y_i)\prod_{i\notin I}\chi_{\alpha_i}(y_i)\right]$$

(where I and μ are independent)

$$= \mathop{\mathbb{E}}_{I,\mu,y}\left[\prod_{i\in I}\chi_{\alpha_i}(\mu_i)\right] \qquad \text{(where } I \sim \tbinom{[n]}{\nu n}, \mu \sim \mathbb{Z}_s \setminus \{0\}, \text{ and } y \sim 0^{\overline{I}} \circ \mu^I\text{)}$$

$$= \mathop{\mathbb{E}}_{I}\left[\prod_{i\in I}\mathop{\mathbb{E}}_{\mu_i\sim\mathbb{Z}_s\setminus\{0\}}[\chi_{\alpha_i}(\mu_i)]\right]. \tag{10}$$

Now we note that the inner term

$$\mathop{\mathbb{E}}_{\mu_i\sim\mathbb{Z}_s\setminus\{0\}}[\chi_{\alpha_i}(\mu_i)] = \begin{cases} 1, \text{ if } \alpha_i = 0, \text{ and} \\ \frac{1}{s-1}\left(\sum_{\mu_i\in\mathbb{Z}_s\setminus\{0\}}\chi_{\alpha_i}(\mu_i)\right) = \frac{1}{s-1}\left(s\,\mathbb{E}_{\mu_i\sim\mathbb{Z}_s}[\chi_{\alpha_i}(\mu_i)] - 1\right) = \frac{-1}{s-1} \text{ otherwise.}\end{cases} \tag{11}$$

Therefore, denoting the coordinates of α with non-zero entries by $J \subseteq [n]$, plugging (11) into (10) gives

$$\mathop{\mathbb{E}}_{y\sim\mathcal{E}_\nu}[\chi_\alpha(y)] = \mathop{\mathbb{E}}_{I\sim\binom{[n]}{\nu n}}\left[\left(\frac{-1}{s-1}\right)^{|J\cap I|}\right] \leqslant \mathop{\mathbb{E}}_{I\sim\binom{[n]}{\nu n}}\left[\left(\frac{1}{2}\right)^{|J\cap I|}\right] \qquad \text{(as } s \geqslant 3\text{)}$$

$$\leqslant \mathop{\Pr}_{I\sim\binom{[n]}{\nu n}}[|J\cap I| < \nu k/2]\cdot 1 + \mathop{\mathbb{E}}_{I\sim\binom{[n]}{\nu n}}\left[\left(\frac{1}{2}\right)^{|J\cap I|} \middle| \, |J\cap I| \geqslant \nu k/2\right].$$

Denoting $|J| = \#\alpha$ by k, we observe that $|J\cap I|$ is distributed according to the hypergeometric distribution of k draws (without replacement) from a population of size n and νn many success states. Hence, by a tail bound [16] $\Pr[|J\cap I| < \nu k/2] \leqslant e^{-\nu^2 k/2}$ and using $\nu \geqslant 1/32$, we get that $\mathbb{E}_{y\sim\mathcal{E}_\nu}[\chi_\alpha(y)] \leqslant \Pr_{I\sim\binom{[n]}{\nu n}}[|J\cap I| < \nu k/2]\cdot 1 + \mathbb{E}_{I\sim\binom{[n]}{\nu n}}\left[\left(\frac{1}{2}\right)^{|J\cap I|} \middle| \, |J\cap I| \geqslant \nu k/2\right] \leqslant 2\cdot\widetilde{\rho}^k$ for $\widetilde{\rho} := 2^{-2^{-11}}$.

Plugging the above bound in (9), letting $\widetilde{\nu} = (1-1/s)(1-\widetilde{\rho})$ and $q = 2+\varepsilon = 2 + \frac{1}{2^{12}\log s}$, we get

$$\mathop{\Pr}_{\substack{x\sim\mathbb{Z}_s^n\\ y\sim S_\nu(x)}}[x \in A \text{ and } y \in A] \leqslant 2\sum_{\alpha\in\mathbb{Z}_s^n}\left|\widehat{f}(\alpha)\right|^2 \widetilde{\rho}^{\#\alpha} = 2\mathop{\Pr}_{\substack{x\sim\mathbb{Z}_s^n\\ y\sim N_{\widetilde{\nu}}(x)}}[x \in A \text{ and } y \in A] \leqslant 2\delta^{2-2/q},$$

where the last step uses the small-set expansion theorem corresponding to Bernoulli noise (e.g. Theorem 10.25 in [21]). ◀

References

1 Noga Alon, Tali Kaufman, Michael Krivelevich, Simon Litsyn, and Dana Ron. Testing reed-muller codes. *IEEE Trans. Inf. Theory*, 51(11):4032–4039, 2005. doi:10.1109/TIT.2005.856958.

2 Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998. doi:10.1145/278298.278306.

3 Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998. doi:10.1145/273865.273901.

4 Vipul Arora, Arnab Bhattacharyya, Noah Fleming, Esty Kelman, and Yuichi Yoshida. Low degree testing over the reals. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 738–792. SIAM, 2023. doi:10.1137/1.9781611977554.ch31.

5 László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 21–31. ACM, 1991. doi:10.1145/103418.103428.

6 László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Comput. Complex.*, 1:3–40, 1991. doi:10.1007/BF01200056.

7 Mitali Bafna, Srikanth Srinivasan, and Madhu Sudan. Local decoding and testing of polynomials over grids. *Random Struct. Algorithms*, 57(3):658–694, 2020. doi:10.1002/rsa.20933.

8 Boaz Barak, Parikshit Gopalan, Johan Håstad, Raghu Meka, Prasad Raghavendra, and David Steurer. Making the long code shorter. *SIAM J. Comput.*, 44(5):1287–1324, 2015. doi:10.1137/130929394.

9 Eli Ben-Sasson, Prahladh Harsha, and Sofya Raskhodnikova. Some 3cnf properties are hard to test. *SIAM J. Comput.*, 35(1):1–21, 2005. doi:10.1137/S0097539704445445.

10 Arnab Bhattacharyya, Swastik Kopparty, Grant Schoenebeck, Madhu Sudan, and David Zuckerman. Optimal testing of reed-muller codes. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 488–497. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.54.

11 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.

12 Andrej Bogdanov and Gautam Prakriya. Direct sum and partitionability testing over general groups. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 33:1–33:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.33.

13 Irit Dinur and Konstantin Golubev. Direct sum testing: The general case. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019, September 20-22, 2019, Massachusetts Institute of Technology, Cambridge, MA, USA*, volume 145 of *LIPIcs*, pages 40:1–40:11. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.40.

14 Elad Haramaty, Noga Ron-Zewi, and Madhu Sudan. Absolutely sound testing of lifted codes. *Theory Comput.*, 11:299–338, 2015. doi:10.4086/toc.2015.v011a012.

15 Elad Haramaty, Amir Shpilka, and Madhu Sudan. Optimal testing of multivariate polynomials over small prime fields. *SIAM J. Comput.*, 42(2):536–562, 2013. doi:10.1137/120879257.

16 Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding*, pages 409–426, 1994.

17 Charanjit S. Jutla, Anindya C. Patthak, Atri Rudra, and David Zuckerman. Testing low-degree polynomials over prime fields. *Random Struct. Algorithms*, 35(2):163–193, 2009. doi:10.1002/rsa.20262.

18 Tali Kaufman and Dor Minzer. Improved optimal testing results from global hypercontractivity. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 98–109. IEEE, 2022. doi:10.1109/FOCS54457.2022.00017.

19 Tali Kaufman and Dana Ron. Testing polynomials over general fields. *SIAM J. Comput.*, 36(3):779–802, 2006. doi:10.1137/S0097539704445615.

20 Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. *J. ACM*, 40(3):607–620, 1993. doi:10.1145/174130.174138.

21 Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014. URL: http://www.cambridge.org/de/academic/subjects/computer-science/algorithmics-complexity-computer-algebra-and-computational-g/analysis-boolean-functions.

22 Øystein Ore. Über höhere kongruenzen. *Norsk Mat. Forenings Skrifter*, 1(7):15, 1922.

23 Yury Polyanskiy. Hypercontractivity of spherical averages in hamming space. *SIAM J. Discret. Math.*, 33(2):731–754, 2019. doi:10.1137/15M1046575.

24 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.

A Existence of the code U

In order to prove Claim 18 we will need the following:

▷ Claim 24. There exists a matrix $M \in \mathbb{F}_2^{k\times w}$ such that $U := \{Mz : z \in \mathbb{F}_2^w\} \subseteq \mathbb{F}_2^k$ is if size 2^w and:

- For all $y \neq y' \in U$, we have $k/4 \leqslant \Delta(y, y') \leqslant 3k/4$.
- There exists a function $\chi : U \to \{\pm 1\}$ such that for all $I \in \binom{[k]}{\leqslant d}$, we have

$$\sum_{y\in U:\ y^I=1^I} \chi(y) = 0.$$

Proof. We will show that picking M uniformly at random satisfies both the items with positive probability. For Item 1, if suffices if for all $y \neq 0^k$ in U, $k/4 \leqslant \#y \leqslant 3k/4$; that is, for all $z \in \mathbb{F}_2^a \setminus \{0^{d+1}\}$, $k/4 \leqslant \#Mz \leqslant 3k/4$. For any fixed $z \neq 0^a$, we note that $y = Mz$ is uniformly distributed over $\mathbb{F}_2^k$ as $M \sim \mathbb{F}_2^{k\times w}$. Hence, by a Chernoff bound, we have $\Pr_M[\#Mz \notin [k/4, 3k/4]] \leqslant 2e^{-k/24}$. A union bound over all $z \in \mathbb{F}_2^w \setminus \{0^w\}$ gives

$$\Pr_M\left[\neg\left(\forall y \neq y' \in U,\ k/4 \leqslant \Delta(y, y') \leqslant 3k/4\right)\right] \leqslant 2^w \cdot 2e^{-k/24} < 1/2.$$

However, it is a known fact that a uniformly chosen rectangular matrix has full rank with probability at least 1/2. Therefore, with positive probability there must be a matrix M such that it is full rank and Item 1 holds. We fix such an M and prove Item 2.

For $y \in U$, as M is full rank there exists a unique $z \in \mathbb{F}_2^w$ such that $Mz = y$. Then we define

$$\chi(y) := (-1)^{\langle z,\eta\rangle} = (-1)^{z_1\eta_1+\cdots+z_w\eta_w},$$

where $\eta \in \mathbb{F}_2^w$ is an arbitrary vector such that it is not in the $\mathbb{F}_2^w$-span of any d rows of M. Such an η always exists as the number of vectors that *can* be expressed as a linear combination of d rows of M is at most

$$\binom{k}{d}2^d \leqslant \left(\frac{ek}{d}\right)^d 2^d = \left(2e\psi s^2\right)^d < 2^w,$$

the total number of vectors in $\mathbb{F}_2^w$.

Let $M_1, \ldots, M_k \in \mathbb{F}_2^w$ denote the rows of M. For any $I \in \binom{[k]}{\leqslant d}$ and $y = Mz$, the condition $y^I = 1^I$ is equivalent to: $\langle z, M_i\rangle = 1$ for all $i \in I$. Hence we have

$$\sum_{y\in U:\ y^I=1^I} \chi(y) = \sum_{z\in\mathbb{F}_2^a:\ \forall i\in I,\ \langle z,M_i\rangle=1} (-1)^{\langle z,\eta\rangle} \tag{12}$$

As η is linearly independent with $\{M_i\}_{i\in I}$, there exists $\eta' \neq 0^a$ such that $\langle \eta', M_i\rangle = 0$ for all $i \in I$ and $\langle \eta', \eta\rangle = 1$: this is because we can treat these conditions as a system of linear equations over $\mathbb{F}_2$.

Note that for any $z \in \mathbb{F}_2^w$, $\langle z, M_i\rangle = 1$ if and only if $\langle z + \eta', M_i\rangle = \langle z, M_i\rangle + \langle \eta', M_i\rangle = 1$. Since $z \neq z + \eta'$, we may partition the summation (12) into buckets of size 2, each bucket corresponding to z and $z + \eta'$ for some z. For each such bucket, the sum is

$$(-1)^{\langle z,\eta\rangle} + (-1)^{\left\langle z+\eta',\eta\right\rangle} = (-1)^{\langle z,\eta\rangle} + (-1)^{\langle z,\eta\rangle+\left\langle \eta',\eta\right\rangle} = (-1)^{\langle z,\eta\rangle}\left(1 + (-1)^{\left\langle \eta',\eta\right\rangle}\right) = 0,$$

so the overall sum is also 0. ◁

Using this result, we will prove Claim 18:

Proof of Claim 18. Identifying the 0's (resp. 1's) in $\mathbb{F}_2$ and $\mathbb{Z}_s$, we will rephrase the above claim as U being a subset of $\mathbb{Z}_s^k$ instead of $\mathbb{F}_2^k$: Then, this set $U \subseteq \{0,1\}^k \subseteq \mathbb{Z}_s^k$ immediately satisfies Item 1 of Claim 18. For Item 2, it suffices to show that for any non-zero junta-degree-d function $P : \mathbb{Z}_s^k \to \mathcal{G}$,

$$\sum_{y \in U} P(y) \cdot \chi(y) = 0 \tag{13}$$

where $\chi : \{0,1\}^k \to \mathbb{Z}$ is from Claim 24. Towards a contradiction, suppose that a junta-degree-d function P has exactly one point in $y^* \in U$ such that $P(y^*) \neq 0$. Then using (13), $0 + \cdots + 0 + P(y^*) \cdot \chi(y^*) + 0 + \cdots + 0 = 0$ and as $\chi(y^*) = \pm 1$, $P(y^*) = 0$, a contradiction.

To prove (13), we expand P into its junta-polynomial representation:

$$\begin{aligned}\sum_{y \in U} P(y) \cdot \chi(y) &= \sum_{y \in U} \sum_{\substack{a \in {\mathbb{Z}_s}^k \\ \#a \leqslant d}} g_a \cdot \left(\prod_{i \in [k]:\ a_i \neq 0} \delta_{a_i}(y_i) \right) \chi(y) \\ &= \sum_{\substack{a \in {\mathbb{Z}_s}^k \\ \#a \leqslant d}} g_a \cdot \left(\sum_{y \in U} \chi(y) \prod_{i \in [k]:\ a_i \neq 0} \delta_{a_i}(y_i) \right)\end{aligned}$$

For any $a \in \mathbb{Z}_s^k$, letting $I := \{i \in [k] : a_i \neq 0\}$, the inner factor is

$$\sum_{y \in U} \chi(y) \prod_{i \in [k]:\ a_i \neq 0} \delta_{a_i}(y_i) = \sum_{y \in U:\ y^I = a^I} \chi(y).$$

Now if a contains any coordinates taking values other than 0 and 1, the above sum is 0 since all the coordinates of $y \in U$ are either 0 or 1. On the other hand, if $a \in \{0,1\}^k$, then $a^I = 1^I$ and Claim 24 is applicable, again giving a sum of 0. Therefore,

$$\sum_{y \in U} P(y) \cdot \chi(y) = \sum_{\substack{a \in \mathcal{S}^k \\ \#a \leqslant d}} g_a \cdot \left(\sum_{y \in U} \chi(y) \prod_{i \in [k]:\ a_i \neq 0} \delta_{a_i}(y_i) \right) = \sum_{\substack{a \in \mathcal{S}^k \\ \#a \leqslant d}} g_a \cdot 0 = 0. \qquad \triangleleft$$

B Proof of the large-distance lemma

Proof of Lemma 16. For this proof, we may assume without loss of generality that $I = [r]$ as relabelling the variables does not affect the probability of a random restriction (i.e., $x_j = \pi_j(x_i)$) being ε_0-close to junta-degree-d. We will prove the contrapositive: assuming $\delta_d(f_{r-1}) \leqslant \varepsilon_0$ for more than $k^2/2r(r-1)$ fraction of choices of (i, j, π_j) (call these *bad* restrictions), we will construct a junta-degree-d function P such that $\delta(f_r, P) \leqslant \varepsilon_1$. Like in [7, 10], the high level idea is to "stitch" together low-junta-degree functions corresponding to the restrictions f_{r-1} (which we shall call $P^{(h)}$) into a low-junta-degree function that is close to f_r.

As there are more than $\frac{k^2}{2r(r-1)} r(r-1)s! = k^2 s!/2$ many bad tuples (i, j, π_j), by pigeonhole principle, there must be some permutation $\pi : \mathbb{Z}_s \to \mathbb{Z}_s$ such that the number of bad tuples of the form (i, j, π) is more than $k^2/2$. In fact, we can say something more: Consider the directed graph G_{bad} over vertices $[r]$ with a directed edge (i, j) for each bad tuple (i, j, π).

As the number of edges in G_{bad} is at least $k^2/2$, by the pigeon-hole principle, we can conclude that there is a *matching* or a *star*[9] in G_{bad} of size $L := k/4$. For the rest of the proof, we will handle both the cases in parallel as the differences are minor.

Suppose we are in the matching case and the corresponding bad tuples are

$$(i_1, j_1, \pi), \dots (i_L, j_L, \pi),$$

where $i_1, \dots, i_L, j_1, \dots, j_L$ are all distinct. Let id denote the identity permutation of $\mathbb{Z}_s$. Consider the function $\widetilde{f}_r(x_1, \dots, x_r)$ obtained by replacing the variables $x_{i_1}, \dots, x_{i_L}$ in f_r with $\pi(x_{i_1}), \dots, \pi(x_{i_L})$ respectively. Then, $\delta_d(\widetilde{f}_r) = \delta_d(f_r)$ and (i_h, j_h, π) is a bad restriction for f_r if and only if (i_h, j_h, id) is a bad restriction for $\widetilde{f}_r$, for all $h \in [L]$. Moreover, if $\widetilde{f}_r$ satisfies $\delta(\widetilde{P}, \widetilde{f}_r) \leqslant \varepsilon_1$, then there also exists a junta-degree-d P such that $\delta(P, f_r) \leqslant \varepsilon_1$ (obtained from $\widetilde{P}$ by applying the inverse permutation π^{-1} to $x_{i_1}, \dots, x_{i_L}$). Therefore, without loss of generality we may assume that $\pi = id$ to construct a junta-degree-d function P such that $\delta(P, f_r) \leqslant \varepsilon_1$. A similar reduction holds in the star case.

We may further assume w.l.o.g. that the matching case corresponds to the tuples

$$(L+1, 1, id), (L+2, 2, id), \dots (2L, L, id)$$

and the star case corresponds to

$$(r, 1, id), (r, 2, id), \dots, (r, L, id).$$

For $h \in [L]$, we define

$$R_h := \begin{cases} \{x \in \mathbb{Z}_s^r : x_{L+h} = x_h\} \text{ in the matching case,} \\ \{x \in \mathbb{Z}_s^r : x_h = x_r\} \text{ in the star case.} \end{cases}$$

as the points that agree with the h-th bad restriction (i, j, π) in the matching or star case correspondingly. Let R'_h denote the complement of R_h. Then for any function $P : \mathbb{Z}_s^r \to \mathcal{G}$,

$$\Pr_{x \sim \mathbb{Z}_s^r}[f_r(x) \neq P(x)] \leqslant \Pr_x\left[x \notin \bigcup_{h \leqslant L} R_h\right] + \sum_{h \leqslant L} \Pr_x\left[x \in R_h \setminus \bigcup_{h' < h} R'_h\right] \cdot \Pr_x\left[f_r(x) \neq P(x) \,\middle|\, x \in R_h \setminus \bigcup_{h'<h} R_{h'}\right] \tag{14}$$

To estimate the above probabilities, we note that in both the matching or the star case, $\Pr_{x \sim \mathbb{Z}_s^r}\left[x \notin \bigcup_{h \leqslant L} R_h\right] = \Pr_x\left[x \in \bigcap_{h \leqslant L} R'_h\right] = \left(1 - \frac{1}{s}\right)^L$, and $\Pr_{x \sim \mathbb{Z}_s^r}\left[x \in R_h \setminus \bigcup_{h'<h} R_{h'}\right] = \Pr_x\left[x \in R_h \cap \bigcap_{h'<h} R'_{h'}\right] = \frac{1}{s}\left(1 - \frac{1}{s}\right)^{h-1}$.

For $h \in [L]$, let $f_{r-1}^{(h)} : \mathbb{Z}_s^r \to \mathcal{G}$ be the restricted function corresponding to the h-th bad tuple, treated as a function of all the r many variables (rather than $r-1$). Let $P^{(h)}$ denote the junta-degree-d function that is of distance at most ε_0 from $f_{r-1}^{(h)}$. We will use the following claim that there is a junta-degree-d function P that agrees with $P^{(h)}$ over R_h, for all h.

▷ Claim 25. There exists a junta-degree-d function P such that $P(x) = P^{(h)}(x)$ for all $h \in [L]$ and $x \in R_h$.

[9] A matching is a set of disjoint edges and a star is a set of edges that share a common start vertex, or a common end vertex.

We defer its proof to the full version and only mention here that the idea is to first show for $h \neq h' \leqslant L$ that $P^{(h)}|_{h'} = P^{(h')}|_h$, since both these functions agree (with $f^{(h)}_{r-1}$ and $f^{(h')}_{r-1}$) over a "large" subset $R_h \cap R_{h'}$ of their domain. Then one can interpolate the restricted functions into a junta-degree-d function P. Then for such P and any $h \leqslant L$,

$$\begin{aligned}\Pr_x\left[f_r(x) \neq P(x) \middle| x \in R_h \setminus \bigcup_{h'<h} R_{h'}\right] &\leqslant \frac{\Pr_x\left[f_r(x) \neq P(x) \mid x \in R_h\right]}{\Pr\left[x \in R_h \setminus \bigcap_{h'<h} R_{h'} \mid x \in R_h\right]} \\ &= \frac{\Pr_x\left[f_r(x) \neq P(x) \mid x \in R_h\right]}{\left(1-\frac{1}{s}\right)^{h-1}} \\ &\leqslant \left(\frac{s}{s-1}\right)^{h-1} \varepsilon_0.\end{aligned}$$

Then we can bound (14) as $\Pr_{x\sim\mathbb{Z}_s^r}\left[f_r(x) \neq P(x)\right] \leqslant \left(1-\frac{1}{s}\right)^L + \frac{L\varepsilon_0}{s} \leqslant \varepsilon_1/2 + \varepsilon_1/2 = \varepsilon_1$ as we can set $\varepsilon_0 := 2s\varepsilon_1/k \geqslant 1/s^{O(k)}$ and

$$\left(1-\frac{1}{s}\right)^{k/4} \leqslant e^{-k/4s} = e^{-\psi sd/4} \leqslant \frac{1}{2}\left(\frac{1}{2C2^{\lceil \log(8\psi s^2)d\rceil}}\right)^{2^{14}\log s} = \frac{\varepsilon_1}{2}.$$

(for the last inequality, we can take ψ to be a sufficiently large constant)

◀

C Proof of Claim 22

Proof of Claim 22. The proof is by induction on w. The base case $w = 1$ is crucial and it is equivalent to the following claim:

▷ Claim 26. For $0 \leqslant d' \leqslant s-1$, the function $f_{d'} : \mathcal{B}(\mathcal{S}, t) \to \mathbb{F}$ defined as $f_{d'}(z) = z_1^{d'}$ for $z = (z_1, \ldots, z_t) \in \mathcal{B}(\mathcal{S}, t)$ has degree exactly d'.

Assuming the above claim to be true, let $w > 1$ be arbitrary. As $d' = 0$ is trivial to handle, we will assume that $d' \geqslant 1$. Hence, Q contains at least one monomial m of degree d' and containing some variable $y_{(1,j)}$ with individual degree $a \in [s-1]$. Without loss of generality, suppose $j = w$. Since Claim 26 states that the function $y^a_{(1,w)}$ is linearly independent of degree-$(a-1)$ functions over $\mathcal{B}(\mathcal{S}, t)$, there exists a function $C : \mathcal{B}(\mathcal{S}, t) \to \mathbb{F}$ such that for any $f : \mathcal{B}(\mathcal{S}, t) \to \mathbb{F}$

$$\langle C, f\rangle = \begin{cases} 1 \text{ if } f = y^a_{(1,w)} \text{ i.e., } a\text{-th power of the last coordinate} \\ 0 \text{ if } f \text{ is degree-}(a-1). \end{cases} \tag{15}$$

Now we decompose Q as a polynomial over variables in the first $w-1$ columns and coefficients being the monomials over variables in the last column: that is

$$Q(y) = \sum_{\alpha\in[0..a]} Q'_\alpha(y_1, \ldots, y_{w-1}) \cdot y^\alpha_{(1,w)}, \tag{16}$$

where y_j represents the variables in the j-th column $Q'_a \neq 0$ has degree $d' - a$. Here, we are using the fact that Q only contains variables from the first row.

Towards a contradiction, suppose there is some degree-$(d'-1)$ polynomial $R(y)$ such that $Q(y) = R(y)$ for all $y \in \mathcal{B}(\mathcal{S}, t)^w$. We may decompose R as follows:

$$R(y) = \sum_{\alpha \in [0..s-1]^t} R'_\alpha(y_1, \ldots, y_{w-1}) \cdot y_w^\alpha$$

where $y_w^\alpha = y_{(1,w)}^{\alpha_1} \cdots y_{(t,w)}^{\alpha_t}$ and for all α, either $R'_\alpha = 0$ or is of degree (as a formal polynomial) at most $d' - 1 - |\alpha|_1$, where $|\alpha|_1 = \alpha_1 + \cdots + \alpha_t$.

Fixing $y_1, \ldots, y_{w-1} \in \mathcal{B}(\mathcal{S}, t)$ to arbitrary values and treating $Q(y)$ and $R(y)$ as functions of y_w, we get

$$\begin{aligned}
\langle C, Q(y_1, \ldots, y_{w-1}, y_w) \rangle &= \left\langle C, \sum_{\alpha \in [0..a]} Q'_\alpha(y_1, \ldots, y_{w-1}) \cdot y_{(1,w)}^\alpha \right\rangle \\
&= \sum_{\alpha \in [0..a]} Q'_\alpha(y_1, \ldots, y_{w-1}) \cdot \left\langle C, y_{(1,w)}^\alpha \right\rangle \\
&= Q'_a(y_1, \ldots, y_{w-1}). && \text{(using (15))}
\end{aligned}$$

Similarly,

$$\begin{aligned}
\langle C, R(y_1, \ldots, y_{w-1}, y_w) \rangle &= \left\langle C, \sum_{\alpha \in [0..s-1]^t} R'_\alpha(y_1, \ldots, y_{w-1}) \cdot y_w^\alpha \right\rangle \\
&= \sum_{\alpha \in [0..s-1]^t :\ |\alpha|_1 \geqslant a} R'_\alpha(y_1, \ldots, y_{w-1}) \cdot \langle C, y_w^\alpha \rangle && \text{(using (15))}
\end{aligned}$$

As a polynomial in the variables of $y_1, \ldots, y_{w-1}$, the final expression above is of degree at most $d' - 1 - |\alpha|_1 \leqslant d' - 1 - a$. However, as a function it is equivalent to Q'_a, which has a strictly higher degree, $d' - a$. This contradicts the induction hypothesis. ◁

Improved Local Computation Algorithms for Constructing Spanners

Rubi Arviv
Efi Arazi School of Computer Science, Reichman University, Herzliya, Israel

Lily Chung
MIT, Cambridge, MA, USA

Reut Levi
Efi Arazi School of Computer Science, Reichman University, Herzliya, Israel

Edward Pyne
MIT, Cambridge, MA, USA

Abstract

A spanner of a graph is a subgraph that preserves lengths of shortest paths up to a multiplicative distortion. For every k, a spanner with size $O(n^{1+1/k})$ and stretch $(2k+1)$ can be constructed by a simple centralized greedy algorithm, and this is tight assuming Erdős girth conjecture.

In this paper we study the problem of constructing spanners in a local manner, specifically in the Local Computation Model proposed by Rubinfeld et al. (ICS 2011).

We provide a randomized Local Computation Agorithm (LCA) for constructing $(2r-1)$-spanners with $\tilde{O}(n^{1+1/r})$ edges and probe complexity of $\tilde{O}(n^{1-1/r})$ for $r \in \{2,3\}$, where n denotes the number of vertices in the input graph. Up to polylogarithmic factors, in both cases, the stretch factor is optimal (for the respective number of edges). In addition, our probe complexity for $r = 2$, i.e., for constructing a 3-spanner, is optimal up to polylogarithmic factors. Our result improves over the probe complexity of Parter et al. (ITCS 2019) that is $\tilde{O}(n^{1-1/2r})$ for $r \in \{2,3\}$. Both our algorithms and the algorithms of Parter et al. use a combination of neighbor-probes and pair-probes in the above-mentioned LCAs.

For general $k \geq 1$, we provide an LCA for constructing $O(k^2)$-spanners with $\tilde{O}(n^{1+1/k})$ edges using $O(n^{2/3}\Delta^2)$ neighbor-probes, improving over the $\tilde{O}(n^{2/3}\Delta^4)$ algorithm of Parter et al.

By developing a new randomized LCA for graph decomposition, we further improve the probe complexity of the latter task to be $O(n^{2/3-(1.5-\alpha)/k}\Delta^2)$, for any constant $\alpha > 0$. This latter LCA may be of independent interest.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Local Computation Algorithms, Spanners

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.42

Category RANDOM

Related Version *Full Version*: `https://arxiv.org/abs/2105.04847v2`

Funding *Lily Chung*: Supported by an Akamai Presidental Fellowship.
Reut Levi: Supported by ISF Grant No. 1867/20
Edward Pyne: Supported by an Akamai Presidental Fellowship.

1 Introduction

A spanner is a sparse structure that is a subgraph of the input graph and preserves, up to a predetermined multiplicative factor, the pairwise distance of vertices. Formally, a k-spanner of a graph $G = (V, E)$ is a graph $G' = (V, E')$ such that $E' \subseteq E$, in which the distance between any pair of vertices in G' is at most k times longer than the corresponding distance in G. k is referred to as the *stretch* of the spanner.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 42; pp. 42:1–42:23
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Spanners have numerous applications in a wide variety of fields such as communication networks [4, 29, 30], biology [5] and robotics [11, 16]. Consequently, the problem of constructing spanners has been studied extensively in several models, such as the distributed model [6, 12, 13, 14, 15, 17, 31], streaming algorithms [1, 22] and dynamic algorithms [10, 9].

This problem was also considered in the realm of sublinear algorithms and in particular in the model of *Local computation algorithms* (LCAs) introduced by Rubinfeld et al. [32] (see also Alon et al. [2] and survey in [24]). In this model the goal is to avoid computing the entire output and instead to compute parts of the output on demand. This model is suitable for the case that not only the input is massive but also the output. Moreover, LCAs support queries from different users while preserving consistency with a single valid solution (although there might be several valid solutions) across different queries. The notion of computing the output locally goes back to local algorithms, locally decodable codes and local reconstruction algorithms. LCAs can be viewed as a generalization of these frameworks.

Recently, several works [26, 25, 23, 28] considered the problem of constructing spanners in the LCA model. The formulation of the problem in this model is as defined next.

▶ **Definition 1** ([2, 26]). *An* LCA $\mathcal{A}$ *for graph* spanners *is a (randomized) algorithm with the following properties.* $\mathcal{A}$ *has access to the adjacency list oracle* $\mathcal{O}^G$ *of the input graph* G*, a tape of random bits, and local read-write computation memory. When given an input (query) edge* $(u, v) \in E$*,* $\mathcal{A}$ *accesses* $\mathcal{O}^G$ *by making probes, then returns* YES *if* (u, v) *is in the spanner* H*, or returns* NO *otherwise. This answer must only depend on the query* (u, v)*, the graph* G*, and the random bits. For a fixed tape of random bits, the answers given by* $\mathcal{A}$ *to all possible edge queries, must be consistent with one particular sparse spanner.*

For specific details regarding the types of probes supported in the LCA model, we refer the reader to Section 2.

1.1 Our Results

We provide LCAs that with high probability construct the following spanners.

1. A 3-spanner with $\tilde{O}(n^{1+1/2})$ edges. The probe and time complexity of the algorithm is $\tilde{O}(n^{1/2})$ which is optimal up to polylogarithmic factors (and constitutes the first optimal algorithm for general graphs). The size-stretch trade-off is optimal as well (up to polylogarithmic factors). This improves over the algorithm of Parter et al. [28] whose probe and time complexity is $\tilde{O}(n^{3/4})$.
2. A 5-spanner with $\tilde{O}(n^{1+1/3})$ edges (the size-stretch trade-off is optimal up to polylogarithmic factors). The probe and time complexity of the algorithm is $\tilde{O}(n^{2/3})$. This improves over the algorithm of Parter et al. [28] whose probe and time complexity is $\tilde{O}(n^{5/6})$.
3. An $O(k^2)$-spanner with $\tilde{O}(n^{1+1/k})$ edges with high probability. The probe and time complexity of the algorithm is $O(n^{2/3}\Delta^2)$ where Δ denotes the maximum degree of the input graph. This improves over the algorithm of Parter et al. [28] whose probe and time complexity is $\tilde{O}(n^{2/3}\Delta^4)$. Our algorithm (and the algorithm of [28]) uses only neighbor probes for this task.
4. By additionally taking advantage of adjacency probes we further improve the probe and time complexity of the latter algorithm to be $O(n^{2/3-(1.5-\alpha)/k}\Delta^2)$, for any constant $\alpha > 0$. This result utilizes a new, efficient local computation algorithm for decomposing a graph into subgraphs with improved maximum degree that may be of independent interest.

1.2 Our algorithms and techniques

We next describe our algorithms in high-level. Our LCAs for constructing 3-spanners and 5-spanners share similarities with the LCAs in [28] (which are inspired by the algorithm of Baswana and Sen for constructing spanners [7]). The main novelty of our algorithms is in selecting several sets of centers, each designed to cluster different type of vertices. The basic idea is that for high-degree vertices we need to select less centers. Consequently, we can allow more edges per pair of vertex-cluster or cluster-cluster which decreases the probe complexity. To support this approach we also change the way each vertex finds its center. See more details in Subsections 1.3 and 1.4.

Our algorithm for constructing $O(k^2)$-spanners consists of two parts. The first, which is described in high-level is Subsection 1.5, closely follows the construction in [28]. The main novelty in this algorithm is in the way we partition the Voronoi cells, which are formed with respect to randomly selected centers, into clusters of smaller size. In addition, we make other adjustments in order to save an additional factor of Δ in the probe and time complexity. The second is a new LCA for decomposing a graph into subgraphs of smaller maximum degree. As the first algorithm depends quadratically on the degree, this allows for further savings. We elaborate on this algorithm, which may be of independent interest, in Subsection 1.6.

1.3 Algorithm for constructing 3-spanners

We begin with describing our algorithm for constructing 3-spanners from a global point of view. The local implementation of this global algorithm is relatively straight-forward.

The high level idea is as follows. We consider a partition of the vertices into *heavy* and *light* according to their degrees. All the edges incident to light vertices are added to the spanner. We now focus on the heavy vertices. As a first step, a random subset of vertices is selected. We refer to these vertices as *centers*. With high probability, every heavy vertex has a center in its neighborhood. Assuming this event occurs, each heavy vertex joins a *cluster* of at least one of the centers in its neighborhood. A cluster is composed from a center and a subset of its neighbors. On query $\{u, v\}$, where both u and v are heavy, we consider two cases.

1. u and v belong to the same cluster. In this case we add the edge $\{u, v\}$ to the spanner only in case u is the center of v or vice versa.
2. Otherwise, u and v belong to different clusters. Assume without loss of generality that the degree of v is not greater than the degree of u. We divide the edges incident to v into fixed size *buckets* and add the edge $\{u, v\}$ only if it has minimum rank amongst the edges that are incident to the cluster of u.

In order to make the above high-level description concrete we need to set up some parameters and describe how the centers are selected and how each vertex finds its center. We begin by defining vertices with degrees larger than $\sqrt{n}$ as heavy. Thus by adding all the edges incident to light vertices we add at most $O(n^{3/2})$ edges.

The selection of the centers proceeds as follows. We define $t = \Theta(\log \sqrt{n})$ sets of centers, which are picked uniformly at random, $S_1, \ldots, S_t$ such that the size of S_1 is $\Theta(\sqrt{n})$ and the size of S_{i+1} is roughly half of the size of S_i. Thus, overall, the number of centers is $\tilde{O}(\sqrt{n})$.

We next describe how each heavy vertex finds its center. We partition the heavy vertices into t sets, $V_1, \ldots, V_t$ according to their degrees. The set V_1 contains all the vertices with degree in $[\sqrt{n} + 1, 2\sqrt{n}]$ and in general for every $i \in [t]$, the set V_i contains all the vertices with degree in $[2^{i-1}\sqrt{n} + 1, 2^i\sqrt{n}]$. The centers for vertices in the set V_i are taken from the set S_i. With high probability, for every $i \in [t]$, each vertex $v \in V_i$ has at least one vertex from S_i in its neighborhood and at most $O(\log n)$. Thus, with high probability, each heavy vertex belongs to at least one cluster and at most $O(\log n)$ clusters.

Given a heavy vertex $v \in V_i$, the centers of v are found by going over all the vertices in S_i, u, and checking if $\{u, v\}$ is an edge in the graph. Since the total number of centers is $\tilde{O}(\sqrt{n})$, the probe and time complexity of finding the center of a given vertex is $\tilde{O}(\sqrt{n})$.

It remains to set the size of the buckets. Let $\{u, v\} \in E$ be such that $v \in V_i$ and $\deg(u) \leq \deg(v)$. Since $v \in V_i$, it follows that $\deg(v) \leq 2^i\sqrt{n}$. Since $|S_i| \leq c\sqrt{n}\log n/2^i$ for some constant c, by setting the size of the buckets to be $\sqrt{n}$ we obtain that the total number of edges between heavy vertices that belong to different clusters is $\tilde{O}(n^{3/2})$, as desired.

From the fact that the size of the buckets is $\sqrt{n}$ it follows that the total probe and time complexity of our algorithms is $\tilde{O}(\sqrt{n})$. From the fact that the diameter of every cluster is 2 we obtain that for every edge $\{u, v\}$ which we remove from the graph, there exists a path of length at most 3 between u and v. Hence, the stretch factor of our spanner is 3, as desired.

1.4 Algorithm for constructing 5-spanners

We extend the ideas from the previous section to obtain our algorithm for constructing 5-spanners as follows. We partition the vertices in the graph into three sets: *heavy*, *medium*, and *light*. The set of light vertices is defined to be the set of all vertices with a degree at most $n^{1/3}$ and the set of heavy vertices is defined to be the set of all vertices of degree at least $n^{2/3}$. The set of the medium vertices is defined to be all vertices that are not light nor heavy.

As before, we add to the spanner all the edges incident to light vertices and cluster all the heavy vertices into a cluster of diameter 2. The difference is that now when we partition the heavy vertices into sets according to their degrees the first set consists of all vertices with a degree in $[n^{2/3} + 1, 2n^{2/3}]$.

We partition the set of medium vertices into two sets according to the following random process. Each medium vertex v samples uniformly at random $\Theta(\log n)$ of its neighbors. If one of the vertices in the sample is heavy then v joins the cluster of the heavy vertex in the sample that has minimum rank. Otherwise we say that v is *bad*. This forms clusters of diameter 4.

In a similar manner to the process described above we define another a new collection of sets of centers for the bad vertices such that the total number of such centers is $\tilde{O}(n^{2/3})$ and each bad vertex belongs to at least one cluster and at most $O(\log n)$ clusters. The new centers are selected (randomly) only from the set of vertices which are not heavy. We call the corresponding clusters *light-clusters* since they contain at most $n^{2/3}$ vertices and have diameter 2. Since the total number of light-clusters is $\tilde{O}(n^{2/3})$ we can afford to take an edge between every pair of adjacent light-clusters. Moreover, we partition each light-cluster into buckets of size $n^{1/3}$ and take an edge between every pair of adjacent buckets. Since each bad vertex belongs to $O(\log n)$ light-clusters, the total number of pairs of buckets is $\tilde{O}(n^{4/3})$. The time and probe complexity of finding all the edges incident to two buckets is $\tilde{O}(n^{2/3})$, as desired.

To analyse the stretch factor we partition the edges we remove into three types. The first type of edges are edges between vertices in the same cluster. The second type of edges are edges between a vertex v and a cluster C which is not light, in which case there at least one edge in the spanner which is incident to both v and C. The third type of edges are edges which are incident to a pair of light-clusters, in which case there exists at least one edge in the spanner which is incident to each pair of such clusters. Thus, overall the stretch factor is 5, as claimed.

1.5 Algorithm for constructing $O(k^2)$-spanners

The high-level idea of the algorithm for constructing $O(k^2)$-spanners, which we describe from a global point of view, follows that of [28]. The vertices of the graph are first partitioned into $\tilde{O}(n^{2/3})$ Voronoi cells which are formed with respect to a randomly selected set of $\tilde{O}(n^{2/3})$ centers. We can assume that each Voronoi cell has diameter $O(k)$ by using a separate algorithm to handle *remote* vertices which may not be close to a center. Each Voronoi cell is then partitioned into clusters of size $L = \tilde{O}(n^{1/3})$. In addition each Voronoi cell is marked with probability $1/n^{1/3}$ which respectively also marks all the clusters of the cell. Each non-marked cluster connects to all the adjacent marked clusters using a single edge. This forms *clusters-of-clusters* around marked clusters. Instead of connecting every pair of adjacent clusters A and B, which we can not afford, our goal is to connect A with the cluster-of-clusters of some marked cluster C adjacent to B. Since we can not afford to reconstruct the cluster-of-clusters of C, we instead find the identity of all the Voronoi cells which are adjacent to C and try to connect A with at least one of these cells. We show that this is indeed the case although A may not be connected directly to any one of these cells. By applying an inductive argument we show that the number of hops between A and B is $O(k)$, where traversing from one Voronoi cell to another is considered one hop. Since the diameter of each Voronoi cell is $O(k)$ we obtain an overall stretch factor of $O(k^2)$.

We improve on the LCA of [28] in two main ways. The first main improvement is a new method for partitioning the Voronoi cells into clusters of size L, allowing the cluster containing a vertex to be reconstructed using $O(\Delta^2 L^2)$ probes instead of $O(\Delta^3 L^2)$. The second improvement relates to the problem of connecting a cluster A to the cluster-of-clusters of some marked cluster C adjacent to B. In particular there is an issue of which marked cluster C should be chosen, since it is too expensive to reconstruct every marked cluster adjacent to B. The LCA of [28] processes a single cluster of each adjacent marked Voronoi cell to B, of which there may be as many as $\tilde{O}(\Delta)$. We instead devise a rule by which B is *engaged* with a single marked cluster adjacent to it, and show that in fact it suffices to only consider this one cluster. Combining these improvements reduces the total number of probes from $\tilde{O}(n^{2/3}\Delta^4)$ to $O(n^{2/3}\Delta^2)$.

1.6 Algorithm for graph decomposition

To further reduce the runtime of Theorem 31, we develop a new local computation algorithm to decompose graphs into subgraphs with smaller degree. Observe that for a graph G, for subgraphs $G_1, \ldots, G_t$, if we have k-spanners $H_i \subset G_i$ for every i, the union $\bigcup_{i\in[t]} H_i$ is a k-spanner for G. As the runtime of Theorem 31 depends on the maximum degree Δ, we develop an efficient LCA to break G into t graphs, each with maximum degree $O(\max\{\Delta/t, \log n\})$, where t is a parameter to be chosen. Given v and an index $i \in [t]$, the LCA returns in time $O(\Delta/\sqrt{t})$ all neighbors of v in G_i (i.e. it supports `ALL_NBR` queries to each subgraph). We believe this algorithm may have other applications.

To apply this algorithm in the spanner framework, we compose the LCA for $O(k^2)$ spanners with the LCA for graph decomposition. This is more subtle than generic sequential composition of algorithms, as we must ensure the per-query overhead is mild. We do this by observing the $O(k^2)$-spanner algorithm only ever makes all neighbor queries, and so the decomposition LCA spends $O(\Delta/\sqrt{t})$ work per query the spanner LCA makes to the graph. In particular, as the spanner LCA makes $O(n^{2/3}\Delta)$ all neighbor queries, our new runtime is $O(n^{2/3}\Delta^2/t^{3/2})$ given our choice of t. As decomposing G into t subgraphs increases the size of the output spanner by a factor of t, we ultimately balance parameters and obtain a probe and time complexity of $O(n^{2/3-(1.5-\alpha)/k}\Delta^2)$ for any $\alpha > 0$.

1.6.1 Decomposing the Graph

To build this graph, consider assigning each edge of $G = (V, E)$ two colors $i, j \in [R]$, one from each endpoint. In particular, v assigns its first Δ/R edges to receive color 1, the next Δ/R to receive color 2, etc. Then if an edge (u, v) has received colors i, j from both endpoints, we can let the overall edge color be (i, j) where we assume w.l.o.g that $u < v$. Observe that if each color corresponds to a subgraph, then this decomposition breaks G into R^2 subgraphs. Moreover, given i, j and a vertex v, we can quickly enumerate blocks i and j from v and determine which edges lie in the specified subgraph. However, as these blocks may be poorly aligned, this as described results in a maximum subgraph degree of Δ/R rather than Δ/R^2. Instead, we have each vertex choose a random shift, and assign labels to its blocks according to this shift. Then via standard concentration bounds the maximum degree of every subgraph is as desired. We remark that as we must enumerate every element of bucket i and j from v to find edges with label (i, j), the worst-case time for an individual neighbor query can be up to $\Omega(\Delta/R)$. However, we only need to do this once to answer an all-neighbors query, so as long as all the neighbors are desired we can efficiently amortize this cost.

1.7 The number of random bits

All our algorithms are randomized and hence use random bits. For results 1-2 we use randomness in the selection of centers and representatives. In result 3 we use randomness in the selection of centers, marked clusters, and random ranking of edges.

When the centers and representatives are selected independently, the arguments for proving the guarantees on the sparsity of the spanner follow from standard concentration bounds. As shown by Parter at al. [28], by using a less standard analysis one is able to prove that the same guarantees on the sparsity hold even when the random bits are only $\Theta(\log n)$-wise independent (which requires only $O(\log^2 n)$ truly random bits). Furthermore, by using an intricate analysis, they showed that the guarantees on the stretch factor continue to hold as well. In this writing, we do not repeat the analysis in [28] since it lends itself quite easily to our setting.

For result 4, similar techniques to those of [28] allow the result to be implemented using polylog(n)-wise independence as well [1].

1.8 Related work

As mentioned above, the work which is the most closely related to our work is by Parter et el. [28]. In addition to the upper bounds mentioned in Section 1.1 they also observe that it is possible to obtain an LCA for constructing 5-spanners with $\tilde{O}(n^{1+1/k})$ edges and probe complexity $\tilde{O}(n^{1-1/(2k)})$ for the special case in which the minimum degree is known to be at least $n^{1/2-1/(2k)}$ (this builds on the fact that by picking $\tilde{O}(n^{(1+1/k)/2})$ centers, w.h.p. each vertex has a center in its neighborhood). In addition to upper bounds, they also provide a lower bound of $\Omega(\min\{\sqrt{n}, \frac{n^2}{m}\})$ probes for the simpler task of constructing a spanning graph with $o(m)$ edges, where m denotes the number of edges in the input graph.

Our work also builds on the upper bound in [23], designed originally for bounded degree graphs, which provide a spanner with $(1 + \epsilon)n$ edges on expectation, where ϵ is a parameter, stretch factor $O(\log^2 n \cdot \text{poly}(\Delta/\epsilon))$ and probe complexity of $O(\text{poly}(\Delta/\epsilon) \cdot n^{2/3})$. The work in [23] is a follow-up of [26, 25] which initiated the study of LCAs for constructing ultra-sparse (namely, with $(1 + \epsilon)n$ edges) spanning subgraphs.

[1] More specifically, by using the concentration bound from Fact 5.3 in [28] on the sum of d-wise independent random variables.

2 Preliminaries

The input graph $G = (V, E)$ is a simple undirected graph with $|V| = n$ vertices and a bound on the degree Δ. Both parameters n and Δ are known to the algorithm. Each vertex $v \in V$ is represented as a unique ID from $[n]$.

A local algorithm has access to the *adjacency list oracle* $\mathcal{O}^G$ which provides answers to the following probes (in a single step):

- **Degree probe:** Given $v \in V$, returns the degree of v, denoted by $\deg(v)$.
- **Neighbour probe:** Given $v \in V$ and an index i, returns the i^{th} neighbor of v if $i \leq \deg(v)$. Otherwise, $\perp$ is returned. Additionally, for $v \in V$, we define the all-neighbors query, denoted by $\texttt{ALL\_NBR}(v)$, which returns all the neighbors of v. Clearly, this query can be implemented by $\deg(v) + 1$ neighbor probes.
- **Adjacency probe:** Given an ordered pair $\langle u, v \rangle$ where $u \in V$ and $v \in V$, if v is a neighbor of u then i is returned where v is the i^{th} neighbor of u. Otherwise, $\perp$ is returned.

We denote the distance between two vertices u and v in G by $\mathrm{d}(u, v)$ and the set of neighbours of v in G by $N_G(v)$. We denote by $N_G(v)[i]$ the i-th neighbour of v in G. For vertex $v \in V$ and an integer k, let $\Gamma_k(v, G)$ denote the set of vertices at distance at most k from v. When the graph G is clear from the context, we shall use the shorthand $d(u, v)$, $N(v)$ and $\Gamma_k(v)$ for $d_G(u, v)$, $N_G(v)$ and $\Gamma_k(v, G)$, respectively. We define a ranking r of the edges as follows: $r(u, v) < r(u', v')$ if and only if $\min\{u, v\} < \min\{u', v'\}$ or $\min\{u, v\} = \min\{u', v'\}$ and $\max\{u, v\} < \max\{u', v'\}$.

We shall use the following definitions in our algorithms for constructing 3-spanners and 5-spanners.

▶ **Definition 2.** *We say that a vertex $v \in V$ is in* class $i \in \mathbb{N}$ w.r.t. Δ *if* $\deg(v) \in [2^{i-1}\Delta + 1, 2^i\Delta]$.

▶ **Definition 3.** *We say that an index $i \in \mathbb{N}$ is in* bucket $j \in \mathbb{N}$ w.r.t. Δ *if* $i \in [(j-1) \cdot \Delta + 1, j \cdot \Delta]$.

2.1 Probes in the LCA model

Since the introduction of the model in [32], there have been several formulations concerning, mainly, the measure of performance, the way the input is accessed, and whether preprocessing is allowed. In particular, when the input is a graph, there is the question of whether the LCA can probe the graph anywhere (i.e. ask for the neighbors of an arbitrary vertex). In contrast to message-passing models such as CONGEST and distributed LOCAL algorithms, in LCAs the standard assumption is that indeed the LCA can access the graph anywhere and more specifically that each vertex in the input graph is represented as a unique ID from $[n] = \{1, \ldots, n\}$. To support this claim, we refer the reader to the ultra-formal definition in [20] (Definition 12.11) as well as [3, 18].

We note that the utility of making far-probes [2] was studied in [21], in which the authors showed that for a large family of problems, this extra power is not so useful. Indeed, this extra power is not always used by LCAs. For example, in the recent result of Ghaffari [19], which provides an LCA for the problem of Maximal Independent Set, the assumption is that the IDs are taken from $[n^{10}]$. Nonetheless, we stress that this extra power is an important feature of

[2] Namely, probing vertices for which we do not yet know a path from the query vertex.

the LCA model, which, in particular, distinguishes it from message-passing models (see more on the difference between LCAs and distributed LOCAL algorithms in Section 4.1 in [24]) and comes into play in problems which have a more global nature. For example, this extra power is utilized in Prop. 12.13 in [20] for graph coloring and in [27] for approximate Maximum-Matching. The latter LCA is used in Behnezhad et al. [8] to obtain a state-of-the-art sublinear algorithm for the extensively studied problem of approximate Maximum-Matching.

3 LCA for constructing 3-spanners

In this section, we prove the following theorem. Due to space limitations, we defer the claims regarding probe and time complexities as well as the stretch factor to the appendix.

▶ **Theorem 4.** *There exists an LCA that given access to an n-vertex simple undirected graph G, constructs a 3-spanner of G with $\tilde{O}(n^{1+1/2})$ edges whose probe complexity and time complexity are $\tilde{O}(n^{1/2})$.*

Our algorithm is listed as Algorithm 1. As mentioned-above our algorithm proceeds by forming clusters around centers and connecting the different clusters. To make the description of our algorithm complete we begin with describing the selection of centers.

We define $t \stackrel{\text{def}}{=} \log\sqrt{n}$ sets of centers $S_1, \ldots, S_t$. For every $i \in [t]$, we pick u.a.r. x_i vertices to be in S_i where $x_1 = \sqrt{n}\log n$ and $x_{i+1} = x_i/2$ for every $i \in [t-1]$. The rest of the details of the algorithm appear in Algorithm 1. We next prove the correctness of the algorithm.

Recall that we refer to a vertex whose degree is greater than $\sqrt{n}$ as heavy. The next claim states that with high probability every heavy vertex has at least one center and $O(\log n)$ centers in its neighborhood.

▷ Claim 5. With high probability, for every $i \in [t]$ and every vertex $v \in V$ that is in class i w.r.t. $\sqrt{n}$ it holds that $N(v) \cap S_i \neq \emptyset$ and that $|N(v) \cap S_i| = O(\log n)$.

Algorithm 1 LCA for constructing 3-spanners.

```
Input: Access to an undirected graph $G = (V, E)$ and a query $\{u, v\} \in E$ where we assume w.l.o.g. that $\deg(u) \geq \deg(v)$.
Output: Returns whether $\{u, v\}$ belongs to the spanner or not.
1. If $\deg(v) \leq n^{1/2}$ then return YES (recall that $\deg(u) \geq \deg(v)$).
2. Otherwise, let $c$ denote the class of $u$ w.r.t. $\sqrt{n}$ (see Definition 2).
3. If $v \in S_c$ then return YES.
4. Otherwise, let $\mathcal{C} \stackrel{\text{def}}{=} S_c \cap N(u)$. If $\mathcal{C} = \emptyset$ then return YES.
5. Let $i$ denote the index of $u$ in $N(v)$ and let $b$ denote the bucket of $i$ w.r.t. $\sqrt{n}$ (see Definition 3).
6. For each $x \in \mathcal{C}$:
   a. Go over every $j < i$ such that $j$ is in bucket $b$ and return YES if for every such $j$, $N(v)[j]$ does not belong to the cluster of $x$.
7. Return NO.
```

▷ Claim 6. With high probability, the stretch factor of the spanner constructed by Algorithm 1 is 3.

Proof. Let $\{u, v\}$ be an edge in E such that $\deg(u) \geq \deg(v)$. We will show that there exists a path of length at most 3 between u and v in the the spanner constructed by Algorithm 1 denoted by $G' = (V, E')$. If $\deg(v) \leq \sqrt{n}$ then $\{u, v\} \in E'$ and we are done. Otherwise, if there exists a cluster C such that u and v are both belong to C then in G' they are both connected by an edge to the center of C. Thus there exists a path of length at most 2 between u and v in G'. Otherwise, let C' be a cluster for which u belongs to. We claim that v is adjacent to C' in G'. This follows by induction on the index of u in $N(v)$ and Sub-Step 6a. ◁

▷ Claim 7. The probe and time complexity of Algorithm 1 is $O(\sqrt{n} \log n)$.

Proof. Steps 1-2 can be implemented by making a single degree probe. Their time complexity is $O(1)$. Step 3 can be implemented by accessing the random coins. To implement Step 4 we need to go over all the vertices in S_c (we may assume w.l.o.g. that we generate all the centers in advance as there are only $O(\sqrt{n} \log n)$ centers) and check whether they are in $N(u)$ (by making a single adjacency probe). Thus the probe (and time) complexity of this step is $O(\sqrt{n} \log n)$. Step 5 can be implemented by a single adjacency probe. The total number of vertices we check in Sub-Step 6a is bounded by the size of $\mathcal{C}$ times the size of a bucket which is $\sqrt{n}$. For each vertex we check we make a single neighbor and then we check whether it belongs to the cluster of a specific center. The latter can be implemented by making a single degree probe and a single adjacency probe. By Claim 5, the size of $\mathcal{C}$ is bounded by $O(\log n)$, thus the probe (and time) complexity of Step 6 is $O(\sqrt{n} \log n)$. The claim follows. ◁

▷ Claim 8. With high probability, the number of edges of the spanner constructed by Algorithm 1 is $\tilde{O}(n^{1+1/2})$.

Proof. The number of edges added to E' due to Step 1 is at most $n^{3/2}$. By the bound on the number of centers, the number of edges added to E' due to Step 3 is $O(n^{3/2} \log n)$. To analyse the number of edges added to E' due to Step 6 consider an edge $\{u, v\}$ such that $\deg(u) \geq \deg(v)$, $\deg(v) > \sqrt{n}$ and $v \notin S_c$, where c denotes the class of u w.r.t. $\sqrt{n}$. Since u is in class c it follows that $\deg(u) \leq 2^c\sqrt{n}$. Since $\deg(v) \leq \deg(u)$ it follows that $N(v)$ has at most 2^c buckets. By Sub-Step 6a, for any cluster C, the number of edges in E' that are incident to v and a vertex from C is at most 2^c (since we add to E' at most a single edge for each bucket of $N(v)$). Since the number of clusters of class c is $O(\sqrt{n} \log n/2^c)$, the total number of clusters of class greater or equal to c is $O(\sqrt{n} \log n/2^c)$ as well. Therefore, the total number of edges that are incident to v and added to E' due to Step 6 is $O(\sqrt{n} \log n)$. By Claim 5, w.h.p., the number of edges that are added due to Step 4 is 0. We conclude that the $|E'| = O(n^{3/2} \log n)$, as desired. ◁

4 LCA for constructing 5-spanners

In this section, we prove the following theorem.

▶ **Theorem 9.** *There exists an LCA that given access to an n-vertex simple undirected graph G, constructs a 5-spanner of G with $\tilde{O}(n^{1+1/3})$ edges whose probe complexity and time complexity are $\tilde{O}(n^{2/3})$.*

Our algorithm for constructing 5-spanners also proceeds by forming clusters around centers and connecting the different clusters. For the sake of presentation, we first describe our local algorithm from a global point of view (see algorithm 2). In Section 4.1 we describe the local implementation of this algorithm.

As in the algorithm for constructing 3-spanners, the clusters are formed around randomly selected centers only that now we have two types of clusters (and centers), *heavy-clusters* and *light-clusters* that will be described in the sequel.

The selection of the first type of centers

The selection of the first type of centers proceeds as follows. We define $a \overset{\text{def}}{=} \log n^{1/3}$ sets of centers $S^1_1, \ldots, S^1_a$. For every $i \in [a]$, we pick u.a.r. y_i vertices to be in S^1_i where $y_1 = n^{1/3} \log n$ and $y_{i+1} = y_i/2$ for every $i \in [a-1]$. The clusters which are formed around the first type of centers are the *heavy-clusters*. The formation of the heavy-clusters is described in Step 2 of Algorithm 2.

The selection of the second type of centers

The selection of the second type of centers proceeds as follows. We define $b \overset{\text{def}}{=} \log n^{1/3}$ sets of centers $S^2_1, \ldots, S^2_b$. For every $i \in [b]$, we pick u.a.r. x_i vertices to be in S^2_i where $x_1 = n^{2/3} \log n$ and $x_{i+1} = x_i/2$ for every $i \in [b-1]$.

The clusters which are formed around the second type of centers are the *light-clusters*. The formation of the light-clusters is described in Step 3 of Algorithm 2.

The way we connect the different clusters is described in Steps 4 and 5.

In the next couple of claims we prove that with high probability every vertex v such that $\deg(v) > n^{1/3}$ joins at least one cluster and at most $O(\log n)$ clusters. To do so, we partition the vertices with degree greater than into $n^{1/3}$ into 3 sets. The first set, denoted by H, is the set of vertices, v, such that $\deg(v) \geq n^{2/3}$. The second set is the set of vertices, v, such that $n^{1/3} < \deg(v) < n^{2/3}$ for which at least half of the vertices in $N(v)$ have degree at least $n^{2/3}$. We denote this set by M_1. M_2 consists of the remaining vertices. Namely, M_2 is the set of vertices, v, such that $n^{1/3} < \deg(v) < n^{2/3}$ and for which less than half of the vertices in $N(v)$ have degree at least $n^{2/3}$.

The implication of the next claim is that w.h.p. every vertex in H joins at least one heavy-cluster and at most $O(\log n)$ heavy-clusters.

▷ Claim 10. With high probability, for every $v \in H$ it holds that $N(v) \cap S^1_c \neq \emptyset$ and that $|N(v) \cap S^1_c| = O(\log n)$ where $c \in [a]$ is the class of v w.r.t. $n^{2/3}$.

The implication of the next claim (when combined with Claim 10) is that w.h.p. every vertex in M_1 joins, via a representative, at least one heavy-cluster and at most $O(\log n)$ heavy-clusters.

▷ Claim 11. With high probability, for every $v \in M_1$ it holds that v has a representative.

Proof. Let $v \in M_1$. Consider Step 2b of Algorithm 2. Since at least half of the neighbors of v have degree at least $n^{2/3}$, it follows that w.h.p. $R_v \neq \emptyset$ and so v has a representative. ◁

The implication of the next claim is that w.h.p. every vertex in M_2 that does not have a representative joins at least one light-cluster and at most $O(\log n)$ light-clusters.

▷ Claim 12. With high probability, for every $v \in M_2$ it holds that $N(v) \cap S^2_c \neq \emptyset$ and that $|N(v) \cap S^2_c| = O(\log n)$ where $c \in [b]$ is the class of v w.r.t. $n^{1/3}$.

The following corollary follows directly from Claims 10–12.

Algorithm 2 Global algorithm for constructing 5-spanners.

```
Input: A graph $G = (V, E)$.
Output: Constructs a 5-spanner of $G$, $G' = (V, E')$.
1. For every $v$ such that $\deg(v) \leq n^{1/3}$ add to $E'$ all the edges that are incident to $v$.
2. Forming heavy-clusters:
   a. For each vertex $v$ such that $\deg(v) \geq n^{2/3}$ we define the centers of $v$ to be $N(v) \cap S_c^1$ where $c$ is the class of $v$ w.r.t. $n^{2/3}$ (see Definition 2). For every center $s$ of $v$, $v$ joins the cluster of $s$ by adding the edge $\{s, v\}$ to $E'$.
   b. Each vertex $v$ such that $n^{1/3} < \deg(v) < n^{2/3}$ sample u.a.r. $y \overset{\text{def}}{=} \Theta(\log n)$ of its neighbors. Let $R_v$ denote this set. The *representative* of $v$ is defined to be the vertex, $r$, of minimum id in $R_v$ such that $\deg(r) \geq n^{2/3}$ (if such vertex exists). If $v$ has a representative, $r$, then the edge $\{v, r\}$ is added to $E'$ (and hence $v$ joins all the clusters of $r$).
3. Forming light-clusters:
   a. For each vertex $v$ such that $n^{1/3} < \deg(v) < n^{2/3}$ for which $v$ does not have a representative we define the centers of $v$ to be $N(v) \cap S_c^2$ where $c$ is the class of $v$ w.r.t. $n^{1/3}$ (see Definition 2). For every center $s$ of $v$, $v$ joins the cluster of $s$ by adding the edge $\{s, v\}$ to $E'$.
4. Connecting vertices to adjacent heavy-clusters:
   a. Let $\{u, v\}$ be such that $\deg(u) \geq \deg(v)$ and $u$ belongs to a heavy-cluster. For each cluster $C$ that $u$ belongs to, do:
      i. Partition the interval $[\deg(v)]$ into sequential intervals, which we refer to as buckets, of size $n^{2/3}$: $b_1, \ldots, b_s$ (where only $b_s$ may have size which is smaller than $n^{2/3}$).
      ii. For each $i \in [s]$, go over every $j \in b_i$ in increasing order and check if $N(v)[j]$ belongs to $C$. If such $j$ is found, add $\{v, N(v)[j]\}$ to $E'$ and move to the next bucket.
5. Connecting adjacent light-clusters:
   a. Let $\{u, v\}$ be such that both $u$ and $v$ belong to different light-clusters. For each light clusters $C_u$ and $C_v$ that $u$ and $v$ belong to, respectively, do:
      i. Let $s_u$ and $s_v$ denote the centers of $C_u$ and $C_v$, respectively. Let $c_u$ and $c_v$ denote the classes of $u$ and $v$ w.r.t. $n^{1/3}$, respectively.
      ii. Partition the vertices in $N(s_u)$ that belong to the cluster $C_u$ (namely, the neighbors of $s_u$ that are in class $c_u$ w.r.t. $n^{1/3}$) into subsets of size $n^{1/3}$ greedily by their index in $N(s_u)$, $S_1^u, \ldots, S_t^u$ (all the subsets are of size $n^{1/3}$ except from perhaps $S_t^u$).
      iii. Repeat Step 5(a)ii for the vertices in $N(s_v)$ that belong to $C_v$ and let $S_1^v, \ldots, S_r^v$ denote the resulting subsets.
      iv. For each $i \in [t]$ and $j \in [r]$, add the edge of minimum rank in $E(S_i^u, S_j^v)$ to $E'$ (if such edge exists).
```

▶ **Corollary 13.** *With high probability every vertex v such that $\deg(v) > n^{1/3}$ joins at least one cluster and at most $O(\log n)$ clusters.*

▷ Claim 14. With high probability, $|E'| = \tilde{O}(n^{1+1/3})$.

Proof. The number of edges added to E' due to Step 1 is at most $n^{1+1/3}$. By Claims 10 and 12 the number of edges added to E' due to Steps 2a and 3a is $\tilde{O}(n)$. Since each vertex has at most one representative the number of edges added to E' due to Step 2b is at most n.

Consider $\{u, v\}$ such that $\deg(u) \geq \deg(v)$ and u belongs to a heavy-cluster C. According to Step 4(a)ii we connect v to C by adding to E' at most $\lceil \deg(v)/n^{2/3} \rceil$ edges (at most one edge for each bucket of $N(v)$).

If $\deg(u) \leq n^{2/3}$ then $\deg(v) \leq n^{2/3}$ as well and so $\lceil \deg(v)/n^{2/3} \rceil \leq 1$. Therefore the total number of edges that are incident to v and added to E' due to Step 4(a)ii is bounded by the total number of centers of the first type which is $\tilde{O}(n^{1/3})$.

Otherwise, let c denote the class of u w.r.t. $n^{2/3}$, then by definition $\deg(u) \leq 2^c \cdot n^{2/3}$. Therefore by our assumption $\deg(v) \leq 2^c \cdot n^{2/3}$ as well. The number of centers in S^1_c is $n^{1/3} \log n/2^c$ and so the total number of centers in $\bigcup_{c \leq i \leq a} S^1_i$ is $O(n^{1/3} \log n/2^c)$. Observe that the number of edges which are incident to v and added to E' due to Step 4(a)ii is at most $\lceil \deg(v)/n^{2/3} \rceil$ times the number of centers in $\bigcup_{c \leq i \leq a} S^1_i$. Thus the total number of edges that are incident to v and added to E' due to Step 4(a)ii is $\tilde{O}(n^{1/3})$ in this case as well. Therefore, the total number of edges which are added to E' in Step 4(a)ii is $\tilde{O}(n^{1+1/3})$.

By Claim 12 it follows that the total number of subsets partitioning the light clusters is $\tilde{O}(n^{2/3})$ as the size of each subset is $n^{1/3}$ except for at most $\tilde{O}(n^{2/3})$ subsets, and since each vertex may belong to $O(\log n)$ different clusters. Since in Step 5(a)iv we add at most a single edge between a pair of subsets the total number of edges added to E' due to this step is $\tilde{O}(n^{1+1/3})$. This concludes the proof of the claim. ◁

▷ Claim 15. With high probability, the stretch factor of the spanner constructed by Algorithm 2 is 5.

Proof. Let $\{u, v\}$ be an edge which is not included in E'. By Step 1 of the algorithm it follows that the degree of both u and v is greater than $n^{1/3}$. By Corollary 13 w.h.p. all vertices with degree greater than $n^{1/3}$ join at least one cluster. In the rest of the proof we condition on the event that both u and v join at least one cluster.

Assume w.l.o.g. that $\deg(u) \geq \deg(v)$. If both u and v belong to the same cluster (either heavy or light) then there exists a path of length at most 4 in G' between u and v as the diameter of each cluster is at most 4.

If u belongs to a heavy cluster, C, then by Step 4(a)ii of the algorithm it follows that there exists at least one edge in E' which is incident to v and a vertex in C. Since the diameter of C is at most 4 it follows that there exists a path in G' from v to u.

Otherwise, both u and v belong to different light clusters C_u and C_v. By Step 5(a)iv, there exists at least one edge in E' which is incident to a vertex in C_u and a vertex in C_v. Since the diameter of a light cluster is at most 2 we obtain that there exists a path in G' from u to v of length at most 5. This concludes the proof of the claim. ◁

4.1 The local implementation

In this section we prove the following claim. In the proof of the claim we also describe the local implementation of Algorithm 2.

▷ Claim 16. The probe and time complexity of the local implementation of Algorithm 2 is $\tilde{O}(n^{2/3} \log n)$.

Proof. On query $\{u, v\}$ we first probe the degree of u and v and return YES if either u or v have degree which is at most $n^{1/3}$. Otherwise, assume w.l.o.g. that $\deg(u) \geq \deg(v)$. we consider the following cases.

First case: $\deg(u) \geq n^{2/3}$. In this case we find the centers of u by going over all the centers, s, in S^1_c where c is the class of u w.r.t. $n^{2/3}$ and preforming the adjacency probe $\langle u, s\rangle$. If v belongs to the set of centers of u then we return YES. Overall, since the number of centers of the first type is $\tilde{O}(n^{1/3})$, finding the centers of u requires $\tilde{O}(n^{1/3})$ probes and time.

We then find the bucket b of u in $N(v)$ w.r.t. $n^{2/3}$ (see Definition 3) by preforming the adjacency probe $\langle v, u\rangle$. Let i denote the index of u in $N(v)$. For each center of u, s and for each $j \in b$ such that $j < i$, we check if $N(v)[j]$ belongs to the cluster of s. In order to do so we first probe the degree of $y = N(v)[j]$. If $\deg(y) \geq n^{2/3}$ then v is in the cluster of s if and only if it is a neighbour of s and is in class c w.r.t. $n^{2/3}$ where c is such that s belongs to S^1_c. If $\deg(y) < n^{2/3}$ then we first find the representative of y and if it has a representative we check if it belongs to the cluster of s. Since we have to check this for at most $n^{2/3}$ vertices and for $O(\log n)$ centers, overall the probe and time complexity of preforming this task is $\tilde{O}(n^{2/3})$.

Second case: $n^{1/3} < \deg(u) < n^{2/3}$ and either u or v have a representative. In this case we proceed as in the previous case only that we preform all the checks with respect to the centers of the representative of u (and/or the representative of v). Since finding the representative of a vertex requires $O(\log n)$ probes and time the probe and time complexity in this case is $\tilde{O}(n^{2/3})$ as well.

Third case: $n^{1/3} < \deg(u) < n^{2/3}$ and both u and v do not have representatives. This corresponds to the case in which both u and v belong to light clusters. In order to find the centers of u we simply go over all vertices, y, in $N(u)$ and check if y is in S^2_c where c denotes the class of u w.r.t. $n^{1/3}$. We repeat the same process for v. Since checking if a vertex belongs to S^2_c can be done in $O(\log n)$ time (we can generate all the centers in advance and store them in a binary search tree) this task requires $\tilde{O}(n^{2/3})$ probes and time.

Finally, for each pair of centers s_u and s_v of u and v, respectively, we go over all the neighbours of s_u and s_v and determine for each one, according to its degree, whether it belongs to the cluster of s_u and s_v, respectively. We then find the subsets that u and v belong to as defined in Steps 5(a)ii and 5(a)iii and return YES if and only of $\{u, v\}$ is the edge of minimum rank that connects these subsets.

The above three cases cover all possible scenarios which implies that the time (and probe) complexity of the local implementation of Algorithm 2 is $\tilde{O}(n^{2/3})$ as claimed. ◁

5 Graph Decomposition Via Ranking

We give the formal statement of the LCA of Item 4. We note that our decomposition gives a stronger promise than the maximum degree of each subgraph being bounded, in that we actually bound the degree vertex-wise. In particular, up to poly-logarithmic factors, the average degree of every subgraph is equal to the overall average degree divided by the number of subgraphs with high probability.

▶ **Theorem 17.** *There exists an LCA that, given a parameter $R \leq \sqrt{\Delta}$ and access to an n-vertex simple undirected graph $G = (V, E)$ with maximum degree Δ, decomposes G into edge-disjoint subgraphs $G_1, \ldots, G_{R^2}$ such that:*

1. *Given $(u, v) \in E$, the $i \in [R^2]$ such that $(u, v) \in G_i$ can be computed in time and space $O(1)$.*
2. *Given $v \in V$ and $i \in [R^2]$, $\texttt{ALL\_NBR}_i(v)$ can be computed in time and space $O(d_G(v)/R)$.*
3. *With high probability, the degree of v in G_i is $O(\max\{\log(n), d_G(v)/R^2\})$ for every $v \in V$ and $i \in [R^2]$. In particular, for every i the maximum degree of G_i is $O(\max\{\log(n), \Delta/R^2\})$ with high probability.*

We first describe the decomposition in a global manner. We refer to each subgraph as a color. We assign each edge in G one of R^2 colors, such that the degree and ALL_NBR query times are as claimed. We will identify the set of colors with $[R] \times [R]$, and assume $R^2 \leq \Delta$ since otherwise the statement is trivial. For convenience, let $d(v) := d_G(v)$ and $d_i(v) := d_{H_i}(v)$.

Each vertex v draws a random value $r_v \sim [R]$. Furthermore, for every vertex v, let the first $\lceil d(v)/R \rceil$ neighbors of v be $B_1(v)$, the second be $B_2(v)$, etc. Note that this divides the out-edges into R blocks. For an edge (u, v) with $u < v$ in blocks $B_i(u), B_j(v)$ respectively, the color of the edge is the pair $(i + r_u \mod R,\ j + r_v \mod R)$. Let $G_{a,b}$ for $a, b \in [R]$ be the subgraph consisting of all edges with color (a, b).

We can then combine Theorem 17 with Theorem 31 to give the final result.

▶ **Corollary 18.** *There exists an LCA that given access to an n-vertex simple undirected graph G with maximum degree Δ, constructs an $O(k^2)$-spanner with $\tilde{O}(n^{1+1/k})$ edges whose probe complexity and time complexity are $O(n^{2/3-(1.5-\alpha)/k}\Delta^2)$, for any constant $\alpha > 0$.*

5.1 Decomposition Implementation

▷ Claim 19. Given $(u, v) \in G$, we can determine the color (a, b) of the edge in time and space $O(1)$.

Proof. By making two adjacency probes, we can determine the indices of edge (u, v) in u and v. Then we can compute which blocks contain this edge using two degree queries and a constant number of arithmetic operations, and then compute the final color by looking up the random shifts of u and v. ◁

▷ Claim 20. With high probability, for every $v \in V$ and $(a, b) \in [R] \times [R]$ we have $d_{a,b}(v) = O(\max\{\log(n), d(v)/R^2\})$.

Proof. Fix an arbitrary vertex $v \in V$ and color $(a, b) \in [R] \times [R]$. Fix its shift of r_v of v arbitrarily. Let $S = B_{a-r_v}(v) \cup B_{b-r_v}(v)$ be the set of all edges incident to v (in G) in blocks $a - r_v$ and $b - r_v$. Then S is a superset of the set of edges incident to v with color (a, b), and $|S| = 2d(v)/R$.

For an arbitrary edge $e = (v, u)$ in S, let X_e be the indicator random variable which is 1 exactly when the color of e is (a, b). Let k be the block index of e in u so that $e \in B_k(u)$. There are four cases to consider regarding e:

- Case 1: $v < u$ and $e \in B_{a-r_v}(v)$. Then e has color (a, b) if and only if r_u is equal to $b - k$, which occurs with probability exactly $1/R$.
- Case 2: $v < u$ and $e \notin B_{a-r_v}(v)$. In this case e never has color (a, b).
- Case 3: $v > u$ and $e \in B_{b-r_v}(v)$. Similarly to case 1, e has color (a, b) if and only if r_u is equal to $a - k$, which occurs with probability $1/R$.
- Case 4: $v > u$ and $e \notin B_{b-r_v(v)}$. Similarly to case 2, e never has color (a, b).

In any case, we have $P[X_e = 1] \leq 1/R$ for all $e \in S$. Furthermore, the variables in $\{X_e\}_{e \in S}$ are independent random variables: for distinct edges $e, e' \in S$ where $e = (u, v)$ and $e = (u', v)$, X_e and $X_{e'}$ are independent since the random variables $r_u, r_{u'}$ are independent.

Letting $X = \sum_{e \in S} X_e$ and picking an arbitrary constant $c \geq 2$, we find

$$E[X] \leq 2d(v)/R^2 \leq c \max\{\log n, d(v)/R^2\} =: \mu.$$

By the multiplicative Chernoff's bound we have

$$\Pr[X > 3\mu] \leq \exp(-\mu) \leq \exp(-c \log n) = n^{-c},$$

and so the total number of neighbors of v with color (a, b) is $O(\max\{\log n, d(v)/R^2\})$ with high probability. Finally, a union bound over all n vertices and $R^2 \leq \Delta$ colors completes the proof. ◁

▷ Claim 21. $\texttt{ALL\_NBR}_i(v)$ can be computed in time and space $O(d(v)/R)$.

Proof. Given $v \in V$ and a color $i = (a, b)$, let $S = B_{a-r_v}(v) \cup B_{b-r_v}(v)$ as before. Note that these correspond to the blocks which have received labels a and b respectively, given the random shift of vertex v. We make $2d(v)/R$ neighbor probes to determine all elements of S, then $2d(v)/R$ adjacency probes to determine the indices of every edge in the other endpoint. Then for each edge, we can check in time $O(1)$ (by examining the random shift of the other endpoint) if the label is (a, b). ◁

Proof of Corollary 18. Let $\beta \in (0, 1]$ be such that $3/(2 + \beta) = 1.5 - \alpha$. Given a query if edge $(u, v) \in G$ is in the spanner, we apply the LCA of Theorem 17 with parameter[3] $R = \lceil n^{1/(k(2+\beta))} \rceil$ to determine the i such that $(u, v) \in G_i$. We then apply Theorem 31 with parameter $k' = \lceil k(1 + \frac{2}{\alpha}) \rceil$ to the graph G_i and query if (u, v) is contained in the spanner, and return the answer. Note that we ultimately obtain a $O(k'^2) = O(k^2)$-spanner for every subgraph (and thus for the overall graph), and the number of edges is bounded as $\tilde{O}(R^2 n^{1+1/k'}) \leq \tilde{O}\left(n^{1+\frac{2}{k(2+\alpha)}+\frac{1}{k(1+2/\alpha)}}\right) = \tilde{O}(n^{1+1/k})$. Furthermore, the time complexity is $O(n^{2/3}\Delta^2 R^{-3}) \leq O\left(n^{\frac{2}{3}-\frac{1.5-\alpha}{k}}\Delta^2\right)$. ◀

References

1 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 5–14. ACM, 2012. doi:10.1145/2213556.2213560.

2 Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient local computation algorithms. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1132–1139. SIAM, 2012. doi:10.1137/1.9781611973099.89.

3 Sepehr Assadi and Aditi Dudeja. Lecture 5 in advanced algorithms ii – sublinear algorithms. URL: https://people.cs.rutgers.edu/~sa1497/courses/cs514-s20/lec5.pdf.

4 Baruch Awerbuch. Complexity of network synchronization. *Journal of the ACM (JACM)*, 32(4):804–823, 1985.

[3] We have $R^2 \in O(n^{1/k})$; in order to apply Theorem 17 this should be at most Δ. We can assume this since if Δ is $O(n^{1/k})$ then the graph is already sparse to begin with.

5 Hans-Jürgen Bandelt and Andreas Dress. Reconstructing the shape of a tree from observed dissimilarity data. *Advances in applied mathematics*, 7(3):309–343, 1986.

6 Surender Baswana, Sumeet Khurana, and Soumojit Sarkar. Fully dynamic randomized algorithms for graph spanners. *ACM Trans. Algorithms*, 8(4):35:1–35:51, 2012. `doi:10.1145/2344422.2344425`.

7 Surender Baswana and Sandeep Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Struct. Algorithms*, 30(4):532–563, 2007. `doi:10.1002/rsa.20130`.

8 Soheil Behnezhad, Mohammad Roghani, and Aviad Rubinstein. Sublinear time algorithms and complexity of approximate maximum matching. *CoRR*, abs/2211.15843, 2022. `doi:10.48550/arXiv.2211.15843`.

9 Aaron Bernstein, Sebastian Forster, and Monika Henzinger. A deamortization approach for dynamic spanner and dynamic maximal matching. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1899–1918. SIAM, 2019. `doi:10.1137/1.9781611975482.115`.

10 Greg Bodwin and Sebastian Krinninger. Fully dynamic spanners with worst-case update time. In *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, volume 57 of *LIPIcs*, pages 17:1–17:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. `doi:10.4230/LIPIcs.ESA.2016.17`.

11 Paul Chew. There is a planar graph almost as good as the complete graph. In Alok Aggarwal, editor, *Proceedings of the Second Annual ACM SIGACT/SIGGRAPH Symposium on Computational Geometry, Yorktown Heights, NY, USA, June 2-4, 1986*, pages 169–177. ACM, 1986. `doi:10.1145/10515.10534`.

12 Bilel Derbel and Cyril Gavoille. Fast deterministic distributed algorithms for sparse spanners. *Theor. Comput. Sci.*, 399(1-2):83–100, 2008. `doi:10.1016/j.tcs.2008.02.019`.

13 Bilel Derbel, Cyril Gavoille, and David Peleg. Deterministic distributed construction of linear stretch spanners in polylogarithmic time. In *Distributed Computing, 21st International Symposium, DISC 2007, Lemesos, Cyprus, September 24-26, 2007, Proceedings*, volume 4731 of *Lecture Notes in Computer Science*, pages 179–192. Springer, 2007. `doi:10.1007/978-3-540-75142-7_16`.

14 Bilel Derbel, Cyril Gavoille, David Peleg, and Laurent Viennot. On the locality of distributed sparse spanner construction. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Principles of Distributed Computing, PODC 2008, Toronto, Canada, August 18-21, 2008*, pages 273–282. ACM, 2008. `doi:10.1145/1400751.1400788`.

15 Bilel Derbel, Cyril Gavoille, David Peleg, and Laurent Viennot. Local computation of nearly additive spanners. In *Distributed Computing, 23rd International Symposium, DISC 2009, Elche, Spain, September 23-25, 2009. Proceedings*, volume 5805 of *Lecture Notes in Computer Science*, pages 176–190. Springer, 2009. `doi:10.1007/978-3-642-04355-0_20`.

16 David P Dobkin, Steven J Friedman, and Kenneth J Supowit. Delaunay graphs are almost as good as complete graphs. *Discrete & Computational Geometry*, 5(4):399–407, 1990.

17 Michael Elkin and Ofer Neiman. Efficient algorithms for constructing very sparse spanners and emulators. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 652–669. SIAM, 2017. `doi:10.1137/1.9781611974782.41`.

18 Guy Even, Moti Medina, and Dana Ron. Best of two local models: Centralized local and distributed local algorithms. *Inf. Comput.*, 262:69–89, 2018. `doi:10.1016/j.ic.2018.07.001`.

19 Mohsen Ghaffari. Local computation of maximal independent set. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 – November 3, 2022*, pages 438–449. IEEE, 2022. `doi:10.1109/FOCS54457.2022.00049`.

20 Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. `doi:10.1017/9781108135252`.

21 Mika Göös, Juho Hirvonen, Reut Levi, Moti Medina, and Jukka Suomela. Non-local probes do not help with many graph problems. In Cyril Gavoille and David Ilcinkas, editors, *Distributed Computing – 30th International Symposium, DISC 2016, Paris, France, September 27-29, 2016. Proceedings*, volume 9888 of *Lecture Notes in Computer Science*, pages 201–214. Springer, 2016. doi:10.1007/978-3-662-53426-7_15.

22 Michael Kapralov and David P. Woodruff. Spanners and sparsifiers in dynamic streams. In *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, pages 272–281. ACM, 2014. doi:10.1145/2611462.2611497.

23 Christoph Lenzen and Reut Levi. A centralized local algorithm for the sparse spanning graph problem. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 87:1–87:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.87.

24 Reut Levi and Moti Medina. A (centralized) local guide. *Bull. EATCS*, 122, 2017. URL: http://eatcs.org/beatcs/index.php/beatcs/article/view/495.

25 Reut Levi, Dana Ron, and Ronitt Rubinfeld. A local algorithm for constructing spanners in minor-free graphs. In Klaus Jansen, Claire Mathieu, José D. P. Rolim, and Chris Umans, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France*, volume 60 of *LIPIcs*, pages 38:1–38:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.APPROX-RANDOM.2016.38.

26 Reut Levi, Dana Ron, and Ronitt Rubinfeld. Local algorithms for sparse spanning graphs. *Algorithmica*, 82(4):747–786, 2020. doi:10.1007/s00453-019-00612-6.

27 Reut Levi, Ronitt Rubinfeld, and Anak Yodpinyanee. Local computation algorithms for graphs of non-constant degrees. *Algorithmica*, 77(4):971–994, 2017. doi:10.1007/s00453-016-0126-y.

28 Merav Parter, Ronitt Rubinfeld, Ali Vakilian, and Anak Yodpinyanee. Local computation algorithms for spanners. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 58:1–58:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ITCS.2019.58.

29 David Peleg and Jeffrey D Ullman. An optimal synchronizer for the hypercube. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 77–85, 1987.

30 David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *Journal of the ACM (JACM)*, 36(3):510–530, 1989.

31 Seth Pettie. Distributed algorithms for ultrasparse spanners and linear size skeletons. *Distributed Comput.*, 22(3):147–166, 2010. doi:10.1007/s00446-009-0091-7.

32 Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. In *Innovations in Computer Science – ICS 2011, Tsinghua University, Beijing, China, January 7-9, 2011. Proceedings*, pages 223–238. Tsinghua University Press, 2011.

A LCA for constructing $O(k^2)$-spanners

In this section, we present our LCA for constructing $O(k^2)$-spanners.

A.1 The algorithm that works under a promise

We begin by describing a global algorithm for constructing an $O(k^2)$-spanner which works under the following promise on the input graph $G = (V, E)$. Let $L \stackrel{\text{def}}{=} cn^{1/3} \log n$, where c is a constant that will be determined later. For every $v \in V$, let $i_v \stackrel{\text{def}}{=} \min_r\{|\Gamma_r(v)| \geq L\}$. We are promised that $\max_{v \in V}\{i_v\} \leq k$. In words, we assume that the k-hop neighborhood of every vertex in G contains at least L vertices.

In addition, we assume without loss of generality that $k = O(\log n)$ as already for $k = \log n$ our construction yields a spanner with $\tilde{O}(n)$ edges on expectation.

Our algorithm builds on the partition of V which is described next.

Centers. Pick a set $S \subset V$ by independently including each vertex v in S with probability $n^{-1/3} \log n$, so that $|S| = \Theta(n^{2/3} \log n)$ w.h.p. We shall refer to the vertices in S as *centers*. For each vertex $v \in V$, its *center*, denoted by $c(v)$, is the center which is closest to v amongst all centers (break ties between centers according to the id of the center).

Voronoi cells. The *Voronoi cell* of a vertex v, denoted by $\mathrm{Vor}(v)$, is the set of all vertices u for which $c(u) = c(v)$. Additionally, we assign to each cell a random rank, so that there is a uniformly random total order on the cells; note carefully that the rank of a cell thus differs from the rank of its center (which is given by its identifier, which is not assigned randomly). We remark that we can determine the rank of the cell from the shared randomness and the cell's identifier, for which we simply use the identifier of its center.

The Voronoi cells are partitioned into clusters which are classified into a couple of categories as described next.

Singleton Clusters. For each Voronoi cell, consider the BFS tree spanning it, which is rooted at the respective center. For every $v \in V$, let $p(v)$ denote the *parent* of v in this BFS tree. If v is a center then $p(v) = v$. For every $v \in V \setminus S$, let $T(v)$ denote the subtree of v in the above-mentioned BFS tree when we remove the edge $\{v, p(v)\}$; for $v \in S$, $T(v)$ is simply the entire tree. Now consider a Voronoi cell. If the cell contains at most L vertices, then the *cluster* of all the vertices in the Voronoi cell is the cell itself. Otherwise, there are two cases. If $T(v)$ contains more than L vertices, then we say that v is *heavy* and define the cluster of v to be the singleton $\{v\}$. Otherwise, we say that v is *light* and its cluster is defined as follows.

Non-singleton clusters. Observe that if v is light then it has a unique ancestor u (including v) such that u is not heavy and $p(u)$ is heavy. We define the cluster of v to consist of $T(u)$ and possibly additional subtrees, $T(u')$, where u' is a also a child of $p(u)$ (in $T(p(u))$, as described next.

We begin with some definitions and notations. In order to determine the cluster of u (which is also the cluster of v) consider transforming the heavy vertex $r = p(u)$ into a binary tree which we call *the auxiliary tree of* r, B_r, as follows. B_r is rooted at r and has i complete layers where i is such that $2^i < \deg(r)$ and $2^{i+1} \geq \deg(r)$. These layers consist of *auxiliary vertices*, namely they do not correspond to vertices in G. We then add another layer to B_r consisting of the neighbors of r, sorted from left to right according to their index in $N(r)$. Note that except from the root and the vertices at the last layer of B_r, all vertices in B_r are auxiliary vertices. This completes the definition of B_r. For each vertex $x \in B_r$ we define $B_r(x)$ to be the subtree of B_r rooted at x. We define $S(x) \overset{\text{def}}{=} B_r(x) \cap N(r)$, namely $S(x)$ is the set of vertices of $N(r)$ which are in the subtree of B_r rooted at x. The descendants of x, denoted by the set $D(x)$, are defined to be the union of the vertices in $T(y)$ for every $y \in S(x)$, namely $D(x) \overset{\text{def}}{=} \bigcup_{y \in S(x)} T(y)$. The weight of x is defined to be the number of vertices in $D(x)$, namely, $w(x) \overset{\text{def}}{=} |D(x)|$.

We are now ready to define the cluster of u. Let $z(u)$ be the unique ancestor of u in B_r (including r), z, for which $w(z) \leq L$ and $w(p(z)) > L$ (where $p(z)$ denotes the parent of z in B_r). The cluster of u (and v) is defined to be the set $D(z)$. This completes the description of how the Voronoi cell is partitioned into clusters.

Special vertices. In order to bound the number of clusters (see Section A.3) we shall use the following definitions.

▶ **Definition 22** (Special vertex). *We say that a vertex u is* special *if $|T(u)| > L$ and for every child of u in $T(u)$, t, it holds that $|T(t)| \leq L$.*

Analogously we define special auxiliary vertex as follows.

▶ **Definition 23** (Special auxiliary vertex). *We say that an auxiliary vertex y is a* special auxiliary vertex *if either of the following conditions hold:*

1. *y is a parent of a (non auxiliary) vertex v which is heavy. In this case we say that y is a type (a) special vertex.*
2. *$w(y) > L$ and for every child of y, t, it holds that $w(t) \leq L$. In this case we say that y is a type (b) special vertex.*

For a cluster C, let $c(C)$ denote the center of the vertices in C (all the vertices in the cluster have the same center). Let $\text{Vor}(C)$ denote the Voronoi cell of the vertices in C.

A.2 The Edge Set

Our spanner, $G' = (V, E')$, initially contains, for each Voronoi cell, Vor, the edges of the BFS tree that spans Vor, i.e., the BFS tree rooted at the center of Vor spanning the subgraph induced by Vor. Clearly, the spanner spans the subgraph induced on every Voronoi cell. Next, we describe which edges we add to E' in order to connect adjacent clusters of different Voronoi cells.

Marked Clusters and Clusters-of-Clusters

Each center is *marked* independently with probability $p \overset{\text{def}}{=} 1/n^{1/3}$. If a center is marked, then we say that its Voronoi cell is marked and all the clusters in this cell are marked as well.

Cluster-of-clusters. For every marked cluster, C, define the *cluster-of-clusters* of C, denoted by $\mathcal{C}(C)$, to be the set of clusters which consists of C and all the clusters which are adjacent to C. Let B be a non-marked cluster which is adjacent to at least one marked cluster. Let Y denote the set of all edges such that one endpoint is in B and the other endpoint belongs to a marked cluster. The cluster B is *engaged* with the marked cluster C which is adjacent to B and for which the edge of minimum rank in Y has its other endpoint in C.

The Edges between Clusters

By saying that we *connect* two adjacent subsets of vertices A and B, we mean that we add the minimum ranked edge in $E(A, B)$ to E'. For a cluster A, define its *adjacent centers* $\text{Cen}(\partial A) \overset{\text{def}}{=} \{c(v) \,|\, u \in A \wedge \{u, v\} \in E\} \setminus \{c(A)\}$, i.e., the set of centers of Voronoi cells that are adjacent to A. This definition explicitly excludes $c(A)$, as there is no need to connect A to its own Voronoi cell.

We next describe how we connect the clusters. The high-level idea is to make sure that for every adjacent clusters A and B we connect A with the cluster engaging B (perhaps not directly) and vise versa. For clusters which are not adjacent to any marked cluster and hence not engaged with any cluster we make sure to keep them connected to all adjacent Voronoi cells. Formally:

1. We connect every cluster to every adjacent marked cluster.

2. Each cluster A that is not engaged with any marked cluster (i.e., no cell adjacent to A is marked) we connect to each adjacent cell.
3. Suppose cluster A is adjacent to cluster B, where B is adjacent to a marked cell. Denote by C the (unique) marked cluster that B is engaged with. We connect A with B if the following conditions hold:
   a. the minimum ranked edge in $E(A, \mathrm{Vor}(B))$ is also in $E(A, B)$
   b. $c(B)$ is amongst the $n^{1/k} \log n$ lowest ranked centers in $\mathrm{Cen}(\partial A) \cap \mathrm{Cen}(\partial C)$

A.3 Sparsity

▷ Claim 24. The number of clusters, denoted by s, is at most $|S| + O(nk \log \Delta)/L)$.

▷ Claim 25. The number of edges in E' is $O(n^{1+1/k} \cdot k^2 \log^3 n)$ with high probability.

Proof. Deferred to full version. ◁

A.4 Connectivity and Stretch

▷ Claim 26. G' is connected.

▷ Claim 27. Denote by G_{Vor} the graph obtained from G by contracting Voronoi cells and by G'_{Vor} its subgraph obtained when doing the same in G'. If the cells' ranks are uniformly random, w.h.p. G'_{Vor} is a spanner of G_{Vor} of stretch $O(k)$.

Proof. Deferred to the full version. ◁

▷ Claim 28. W.h.p., G' is a spanner of G of stretch $O(k^2)$.

Proof. Due to the promise on G, w.h.p. the spanning trees on Voronoi cells have depth $O(k)$. Hence, the claim holds for any edge within a Voronoi cell. Moreover, for an edge connecting different Voronoi cells, by Lemma 27, w.h.p. there is a path of length $O(k)$ in G'_{Vor} connecting the respective cells. Navigating with at most $O(k)$ hops in each traversed cell, we obtain a suitable path of length $O(k^2)$ in G'. ◁

A.5 The algorithm for general graphs

We use a combination of the algorithm in Section A with the algorithm by Baswana and Sen [7] which has the following guarantees.

▶ **Theorem 29** ([7]). *There exists a randomized k-round distributed algorithm for computing a $(2k-1)$-spanner $G' = (V, E')$ with $O(kn^{1+1/k})$ edges for an unweighted input graph $G = (V, E)$. More specifically, for every $\{u, v\} \in E'$, at the end of the k-round procedure, at least one of the endpoints u or v (but not necessarily both) has chosen to include $\{u, v\}$ in E'.*

We call a vertex v *remote* if the k-hop neighborhood of v contain less than L vertices. We denote by $\bar{R} \overset{\mathrm{def}}{=} V \setminus R$ the set of vertices which are not remote.

First Step. Run the algorithm from Section A on the subgraph induced by $\bar{R}$, i.e., $\{u, v\} \in E$ with $u, v \in \bar{R}$ is added to E' if and only if the algorithm outputs the edge.

Second Step. Run the algorithm of Baswana and Sen [7] on the subgraph $H = (V, \{\{u, v\} \in E \mid u \in R \text{ or } v \in R\})$, i.e., $\{u, v\} \in E$ with $u \in R$ or $v \in R$ is added to E' if and only if the algorithm outputs the edge.[4]

A.6 Stretch Factor

Consider any edge $e = \{u, v\} \in E \setminus E'$ we removed. If both u and v are in $\bar{R}$, then e was removed by the Algorithm from Section A, which was applied to the subgraph induced by $\bar{R}$. Applying Claim 27 to the connected component of e, we get that w.h.p. there is a path of length $O(k^2)$ from u to v in G'. If u or v are in R, by Theorem 29 there is a path of length $O(k)$ from u to v in G'.

▶ **Corollary 30.** *The above algorithm guarantees stretch $O(k^2)$ w.h.p. and satisfies that the expected number of edges in E' is $O(n^{1+1/k} \cdot k^2 \log^3 n)$*

A.7 The local implementation

In this section we prove the following theorem.

▶ **Theorem 31.** *There exists an LCA that given access to an n-vertex simple undirected graph G, with high probability constructs a $O(k^2)$-spanners with $\tilde{O}(n^{1+1/k})$ edges in expectation. The probe complexity and time complexity are $O(n^{2/3}\Delta^2)$. Moreover, the algorithm access the graph only by* `ALL_NBR` *queries (and performs $O(n^{2/3}\Delta)$ such queries).*

Proof. The local implementation of the algorithm which is described in the previous section is listed in Algorithm 3. The correctness of the algorithm follows from the previous sections. We shall prove that its complexity is as claimed.

The local implementation for remote vertices. For Step 1, we need to determine for both u and v if they are remote. Recall that a vertex u is remote if its k-hop neighborhood contains less than L vertices. Therefore, we can decide for any vertex u whether it is in R with at most L `ALL_NBR` probes. Thus the probe and time complexity is $O(L\Delta)$. If either u or v are remote then we need to determine for each vertex in their k-hop neighborhood whether it is remote or not. If either u or v are remote then the k-hop neighborhood of each of them contain at most $L\Delta$ vertices. This follows from the fact that the size of the k-hop neighborhood of v is at most Δ factor bigger from the k-hop neighborhood of u and vice versa. Thus, we need to call `ALL_NBR` at most $L^2\Delta$ times for this step. Hence, we obtain that the probe and time complexity of this step is $O(L^2\Delta^2)$, in total.

If $u, v \in \bar{R}$, namely, when both u and v are non-remote, the algorithm proceeds as in Section A.1. We next describe the local implementation of the algorithm for this case.

Finding the center and reconstructing the BFS tree. We first analyse the probe and time complexity of determining the center of a vertex. Given a vertex v we perform a BFS from v layer by layer and stop at the first layer in which we find a center or after exploring at least L vertices. Let i denote the layer in which the execution of the BFS stops. It follows that up to layer $i - 1$ we explored strictly less than L vertices. Thus this step can be implemented by $O(L)$ calls to `ALL_NBR`. In particular, the probe and time complexity of finding the center is $O(L\Delta)$.

[4] The algorithm is described for connected graphs; we simply apply it to each connected component of H.

Algorithm 3 LCA for constructing $O(k^2)$-spanners.

```
Input: $\{u, v\} \in E$
Output: whether $\{u, v\}$ is in $E'$ or not.
1. If $u$ or $v$ are in $R$, simulate the algorithm of Baswana and Sen at $u$ and $v$ when running it on the connected component of $u$ and $v$ in the subgraph $H$ (see Section A.5). Return YES if either $u$ or $v$ has chosen to include $\{u, v\}$ and NO otherwise.
2. Otherwise, $u, v \in \bar{R}$ and we proceed according to Section A.1, where all nodes in $R$ are ignored:
    a. If $\text{Vor}(u) = \text{Vor}(v)$, return YES if $\{u, v\}$ is in the BFS tree of $\text{Vor}(u)$ and NO otherwise.
    b. Otherwise, let $Q$ and $W$ denote the clusters of $u$ and $v$, respectively. Return YES if at least one of the following conditions hold for $A = Q$ and $B = W$, or symmetrically, for $A = W$ and $B = Q$, and NO otherwise.
        i. $A$ is a marked cluster and $\{u, v\}$ has minimum rank amongst the edges in $E(A, B)$.
        ii. $A$ is not engaged with any marked cluster. Namely, all the clusters which are adjacent to $A$ are not marked. In this case, we take $\{u, v\}$ if it has minimum rank amongst the edges in $E(A, \text{Vor}(B))$.
        iii. There exists a marked cluster $C$ such that $B$ is engaged with $C$, and the following holds:
            - $\{u, v\}$ has minimum rank amongst the edges in $E(A, \text{Vor}(B))$.
            - The cell $\text{Vor}(B)$ is amongst the $n^{1/k} \log n$ minimum ranked cells in $\text{Cen}(\partial A) \cap \text{Cen}(\partial C)$
```

We observe that at the same cost we also determine the path from $c(v)$ to v in the BFS tree rooted at $c(v)$ as follows. The parent of v in the tree is the neighbour of v that has minimum id amongst all neighbour of v that are closer than v to $c(v)$. Similarly, we can determine the parent of the parent of v and so on until we reach $c(v)$.

Determining if a vertex is heavy. In order to reconstruct the clusters we need to be able to determine if a vertex is heavy or not. Recall that a vertex v is heavy if $|T(v)| > L$. We explore $T(v)$ by performing a find center procedure on all the neighbours of v and then continuing recursively on all the neighbours of v that belong to $\text{Vor}(v)$. Since finding the center takes $O(L)$ calls to `ALL_NBR`, we conclude that we can determine whether v is heavy or light by using $O(L^2\Delta)$ calls to `ALL_NBR`. This follows from the fact that when we partially or completely reveal $T(v)$, we need to find the center of at most $L\Delta$ vertices. Thus, the overall probe and time complexity for this step is $O(L^2\Delta^2)$.

Reconstructing the clusters. Given a vertex v we reconstruct its cluster as follows. First, perform a find-center operation on v and let $v := u_0, u_1, \ldots, u_d$ be the path to the center. We then determine if v is heavy using the prior procedure (and if so we are done). Otherwise, iteratively find $T(u_i)$ for $i \in [d]$ (where we do not search down the path that we have already explored), terminating the search when $T(u_i) > L$. In this case, construct the tree of special vertices below u_i and again find the first ancestor of u_{i-1} in this tree that is heavy, and let the cluster be the children of the predecessor special vertex. As we ultimately explore only $O(L\Delta)$ vertices, this results in $O(L\Delta)$ calls to find-center, which results in at most $O(L^2\Delta)$ calls to `ALL_NBR`.

Determining the cells adjacent to clusters. For Step 2 we need to reconstruct the cluster of u, the cluster of v, and the clusters that u and v are engaged with; this takes $O(L^2\Delta)$ calls to `ALL_NBR`. In addition, for each of these clusters C, we need to determine the center of each vertex adjacent to C. Since the size of the clusters is bounded by L, the number of vertices adjacent to C is at most $L\Delta$. Therefore the number of calls to find-center is at most $L\Delta$. This likewise requires $O(L^2\Delta)$ calls to `ALL_NBR` and overall $O(L^2\Delta^2)$ probes and time.

We conclude that we can perform all necessary checks to decide whether $\{u,v\} \in E'$ or not using $O(L^2\Delta)$ calls to `ALL_NBR` which invokes $O(L^2\Delta^2)$ neighbour probes. By the analysis above, the time complexity is $O(L^2\Delta^2)$ as well. ◀

Classical Simulation of One-Query Quantum Distinguishers

Andrej Bogdanov
School of EECS, University of Ottawa, Canada

Tsun Ming Cheung
School of Computer Science, McGill University, Montreal, Canada

Krishnamoorthy Dinesh
Dept. of Computer Science and Engineering, Indian Institute of Technology, Palakkad, India

John C. S. Lui
Dept. of Computer Science and Engineering, Chinese University of Hong Kong, China

Abstract

We study the relative advantage of classical and quantum distinguishers of bounded query complexity over n-bit strings, focusing on the case of a single quantum query. A construction of Aaronson and Ambainis (STOC 2015) yields a pair of distributions that is ε-distinguishable by a one-query quantum algorithm, but $O(\varepsilon k/\sqrt{n})$-indistinguishable by any non-adaptive k-query classical algorithm.

We show that every pair of distributions that is ε-distinguishable by a one-query quantum algorithm is distinguishable with k classical queries and (1) advantage $\min\{\Omega(\varepsilon\sqrt{k/n})), \Omega(\varepsilon^2 k^2/n)\}$ non-adaptively (i.e., in one round), and (2) advantage $\Omega(\varepsilon^2 k/\sqrt{n \log n})$ in two rounds.

As part of our analysis, we introduce a general method for converting unbiased estimators into distinguishers.

2012 ACM Subject Classification Theory of computation → Probabilistic computation

Keywords and phrases Query complexity, quantum algorithms, hypothesis testing, Grothendieck's inequality

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.43

Category RANDOM

Funding This work is supported by RGC GRF grant CUHK 14207721 and NSERC Discovery grant RGPIN-2023-05006.

1 Introduction

A distinguisher is an algorithm for hypothesis testing. Its purpose is to tell whether its input was sampled from one distribution or from another. In algorithmic contexts including much of cryptography, pseudorandomness, and statistical inference, the computational complexity of distinguishers plays a crucial role.

In this work, we initiate the study of the classical simulation of quantum distinguishers. Quantum algorithms promise algorithmic speedups, but the realization of fully capable quantum computers is still a distant goal. It is thus important to investigate the capabilities of quantum devices of limited computational power. Our focus here is on devices of bounded *query complexity*, a fundamental efficiency measure in complexity theory and cryptographic analysis.

We are interested in the best possible advantage of simulating a quantum distinguisher of bounded query complexity by a classical distinguisher of bounded but possibly larger query complexity k. We focus on quantum distinguishers that make a single query to an n-bit Boolean-valued oracle. Although this model appears restrictive, we find it interesting for the following reasons.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 43; pp. 43:1–43:17
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

First, Aaronson and Ambainis [1] showed that in the *constant* advantage regime, one-query quantum distinguishers already require $\Omega(\sqrt{n})$ non-adaptive classical queries to simulate with the same advantage. Subsequent works [4, 11] showed a rapid deterioration as more queries are added: In general, q classical queries to a Boolean oracle require $\Omega_q(n^{1-1/2q})$ classical queries to simulate. Beyond one or a handful of quantum queries, the improvement over brute-force classical simulation becomes marginal. Moreover, addressing the case of one quantum query already brings up interesting technical challenges and reveals connections to statistical estimation and random matrix theory.

Second, a one-query quantum algorithm can be viewed as a sensible model of a noise-prone quantum device. Motivated by the challenges quantum computers pose to cryptography, it is of interest to study the power of such devices as cryptographic adversaries. In this context, the best classical simulation of a quantum adversary tells us to what extent our confidence in cryptographic security of existing constructions carries over to the quantum setting. While quantum security analyses have been successfully carried out for specific constructions, e.g., [15, 14, 8], our work provides general black-box "transfer theorems" that yield quantum security directly from sufficiently strong classical security at a bounded cost in parameters.

Our results

Our starting point is the separation between quantum and classical query complexity of Aaronson and Ambainis [1]. In response to a question of Buhrman et al. [6], they constructed a random variable F (for "Forrelation") over $\{\pm1\}^{2n}$ (where n is a power of two) for which

- There exists a one query quantum algorithm that distinguishes F from a uniformly random input in $\{\pm1\}^{2n}$ with constant advantage.
- Every classical algorithm that makes $o(\sqrt{n})$ queries fails to distinguish F from random with constant advantage.

Moreover, their example is tight [5, 2]: Every one-query quantum distinguisher with constant advantage can be simulated by a $O(\sqrt{n})$-query non-adaptive classical distinguisher with constant advantage.

Here, as in the rest of the paper, a *distinguisher* is an algorithm that produces outputs in the range $[-1, 1]$. The output of a quantum algorithm is taken to be its probability that it collapses to an accepting state. The *advantage* of D on the pair of distributions $(\mathbf{A}, \mathbf{B})$ is $\frac{1}{2}|\mathsf{E}[D(\mathbf{A})] - \mathsf{E}[D(\mathbf{B})]|$.

We are interested in the best possible advantage that a classical distinguisher with $k \ll \sqrt{n}$ queries can attain for a pair of distributions that are ε-distinguishable by a one-query quantum algorithm. The example of Aaronson and Ambainis yields the following generalization:

▶ **Proposition 1.** *For every $\varepsilon \in (0, 1)$, $k \in \mathbb{N}$, and $n = 2^m$ for some $m \in \mathbb{N}$, there exists a $\{\pm1\}^{2n}$-valued random variable F_ε that is ε-distinguishable from a uniform random $2n$-bit string by 1 quantum query, but $O(\varepsilon k/\sqrt{n})$-indistinguishable by any non-adaptive $2k$-query classical algorithm.*

The random variable F_ε is a mixture of Forrelation F and a uniform random variable U. We believe that the bound $O(\varepsilon k/\sqrt{n})$ is the best possible gap in the advantage of k-query non-adaptive classical versus one-query quantum distinguisher. Our first result is the following lower bound on the classical advantage:

▶ **Theorem 2.** *Let $\varepsilon \in (0,1)$, $k, n \in \mathbb{N}$. Suppose $(\mathbf{A}, \mathbf{B})$ is a pair of random variables over $\{\pm 1\}^n$ that is ε-distinguishable by a one-query quantum algorithm. There exist $2k$-query non-adaptive classical algorithms P_{2a} and P_{2b}, such that*

a. *P_{2a} distinguishes $(\mathbf{A}, \mathbf{B})$ with advantage $\Omega(\varepsilon\sqrt{k/n})$,*

b. *P_{2b} distinguishes $(\mathbf{A}, \mathbf{B})$ with advantage $\Omega(\varepsilon^2 k^2/n)$, assuming $k = O(\sqrt{n})$.*

Assuming ε is constant, distinguisher P_{2a} works better when k is small but does not reach constant advantage as k approaches $\sqrt{n}$. In contrast, distinguisher P_{2b} has a constant advantage when $k = \Theta(\sqrt{n})$; but for the case of $k = 1$, the advantage is worse than the upper bound $O(\varepsilon/\sqrt{n})$ as given in Proposition 1 by a factor of $1/\sqrt{n}$.

We also show that for constant ε, an advantage of $\Omega(k/\sqrt{n \log n})$ can be achieved with two rounds of queries.

▶ **Theorem 3.** *For every $\varepsilon \in (0,1), n \in \mathbb{N}$, and $k \leq \sqrt{n \log n}/\varepsilon$, every pair of random variables over $\{\pm 1\}^n$ that is ε-distinguishable by a one-query quantum algorithm is also $\Omega(\varepsilon^2 k/\sqrt{n \log n})$-distinguishable by a $2k$-query two-round adaptive classical algorithm.*

Bansal and Sinha [4] showed that no k-query adaptive classical algorithm can distinguish F from random with advantage better than $O(\varepsilon k^{1/2}(\log n)^{1/4}/n^{1/4})$. Sherstov, Storozhenko, and Wu [11] proved the same bound with different distributions.

We prove that their bound can be improved to $\tilde{O}(\varepsilon k/\sqrt{n})$ for two-round algorithms, thereby showing that the simulation in Theorem 3 is optimal in k and n up to log factors.

▶ **Theorem 4.** *For every $\varepsilon > 0$, $k \in \mathbb{N}$, and $n = 2^m$ for some $m \in \mathbb{N}$, there exists a random variable F_ε on $\{\pm 1\}^n$, such that it is $O(\varepsilon k\sqrt{\log n}/\sqrt{n})$-indistinguishable from random by any two-round classical algorithm that makes k queries per round.*

Up to the factor of $\sqrt{\log n}$, Theorem 4 generalizes Proposition 1 to adaptive two-round algorithms. The results are summarized in Table 1.

Table 1 Bounds on the best possible advantage of a k-query classical simulation of a one-query quantum distinguisher with advantage ε for distributions over the n-dimensional Boolean cube.

Type	Upper bound	Ref.	Lower bound	Ref.
Non-adaptive	$O(\varepsilon k/\sqrt{n})$	Proposition 1	$\Omega(\varepsilon\sqrt{k/n})$	Theorem 2a
			$\Omega(\varepsilon^2 k^2/n)$	Theorem 2b
Two-round	$O(\varepsilon k\sqrt{\log n}/\sqrt{n})$	Theorem 4	$\Omega(\varepsilon^2 k/\sqrt{n \log n})$	Theorem 3
Adaptive	$O(\varepsilon k^{1/2}(\log n)^{1/4}/n^{1/4})$	[4, 11]		

While it is worth mentioning that the lower bound on advantage in Theorem 3 never exceeds the upper bound from Proposition 1, it remains open whether adaptivity helps in classical simulations of one-query quantum distinguishers.

Our techniques

The acceptance probability of a quantum algorithm that makes one query to an n-bit oracle can be represented by a *bounded* $(n+1) \times (n+1)$ *bilinear form*, that is a function of the form $p(x,y) = \sum A_{ij} x_i y_j$ for some matrix $A \in \mathbb{R}^{(n+1)\times(n+1)}$ with bounded ∞-to-1 norm (see Proposition 6). It suffices to prove our results under the assumption that the two distributions are distinguishable by such bilinear forms. Aaronson et al. [3] showed that this representation fully characterizes one-query quantum algorithms: every bilinear form of bounded ∞-to-1 norm represents the acceptance probability of some one-query quantum algorithm up to constant scaling.

The general problem of identifying the optimal distinguisher in a class of algorithm $\mathcal{A}$ against a class of distribution pairs $\mathcal{B}$ can be modeled as a zero-sum game between distinguishers in $\mathcal{A}$ and distribution pairs in $\mathcal{B}$ whose payoff is the advantage. In this setting, we take $\mathcal{A}$ to be the classical algorithms that make k queries to x and k queries to y, and $\mathcal{B}$ to be the distribution pairs ε-distinguishable by some one-query quantum algorithm.

By Yao's minimax theorem, a given distinguishing advantage is achievable against any given pair in $\mathcal{B}$ if and only if there exists a mixture of distinguishers that has the same expected advantage against all pairs in $\mathcal{B}$. Hence it is sufficient (and necessary) to construct a probabilistic distinguisher that is oblivious to the actual distributions. Such a distinguisher can be obtained from an unbiased estimator for some multiple of p: if $\mathsf{E}[D(x,y)] = \frac{1}{Z}p(x,y)$ for all inputs x, y then the distinguishing advantage of D is at most Z times smaller than that of p.

In the proof of Theorem 2a, we construct an unbiased estimator P_{2a} for p/Z with $Z = O(\sqrt{n/k})$ that is a mixture of $2k$-juntas. Each junta is a homogeneous quadratic function on k bits of x-input and k bits of y-input.

The approximation factor Z is derived based on an additional assumption of boundedness of the juntas, which we explain in detail in Proposition 6. The proposition states that a one-query quantum algorithm is fully characterized by a bilinear form with *∞-to-1 norm* bounded by 1 (we say this bilinear form is 1-bounded). It can be shown that Z is the best possible within this class of unbiased estimators:

▶ **Proposition 5.** *There exists a 1-bounded $n \times n$ bilinear form p such that if p/Z is represented as a mixture of 1-bounded $k \times k$ bilinear forms, then $Z = \Omega(\sqrt{n/k})$.*

In Theorem 2b, we bypass the limitation by truncating a different *unbounded* unbiased estimator P_{2b}. Corollary 13 lower bounds the advantage of the distinguisher obtained by truncating a scaled unbiased estimator in terms of its variance. The relevant estimator is obtained from independently sampled indices $i_1, \dots, i_k, j_1, \dots, j_k$ of x and y input coordinates, respectively, where each coordinate is chosen with probability weighted by Grothendieck's factorization (see Section 2 for the definition). Proposition 16, which is also implicit in the more general analysis of [5], shows that this estimator has variance $O(n/k^2)$.

One weakness of estimator P_{2b} is that it samples the bits of the inputs x and y independently and fails to detect relevant correlations between them. In contrast, the estimator P_3 in Theorem 3 computes the distribution on y-queries adaptively depending on the answers to the x-queries. Viewing the bilinear form $p(x,y)$ as a linear function of the y-inputs, the sample of x-inputs is used to estimate the coefficient $\sum_i A_{ij}x_i$ of y_j for every j. In the second round, the y-inputs are sampled with probabilities proportional to these estimates. Using Grothendieck's factorization and exponential tail bounds, in Proposition 17 we show that this improves the effective variance of P_{2b} by a factor of $\tilde{O}(\sqrt{k})$.

2 Quantum algorithms, bilinear forms, and norms

Notations

We write $[n]$ for the set $\{1, \dots, n\}$. We use the standard computer science asymptotic notations, and the tilde notations hide logarithmic factors. We write $\mathcal{N}(\mu, \Sigma)$ for a multivariate Gaussian with mean μ and covariance matrix Σ, sd for statistical distance, and kl for KL-divergence.

For $A \in \mathbb{R}^{m\times n}$, we use $A^{i,:}$ to denote the i-th row of A and $A^{:,j}$ for the j-th column of A. We write e_i the i-th standard basis vector, and Id_k the $k \times k$ identity matrix. For a symmetric matrix A, we denote the minimum and maximum eigenvalues (which are guaranteed to be real) by $\lambda_{\min}(A)$ and $\lambda_{\max}(A)$ respectively.

Norms

For a vector, we denote $\|v\|_p$ the p-norm of v. The 1-norm, 2-norm and ∞-norm will be relevant in this work. We drop the subscript for Euclidean norm (2-norm) of a vector. The Cauchy-Schwarz inequality says that $\sum_i u_i v_i \leq \|u\|\|v\|$ and in particular $\|v\|_1 \leq \sqrt{n}\|v\|$ for $v \in \mathbb{R}^n$.

For $A \in \mathbb{R}^{m\times n}$, the *spectral norm* $\|A\|$, *Frobenius norm* $\|A\|_F$, and ∞*-to-*1*-norm* $\|A\|_{\infty\to 1}$ are defined to be

$$\|A\| := \max_{u\in\mathbb{R}^n:\|u\|=1} \|Au\| = \max_{\substack{u\in\mathbb{R}^m, v\in\mathbb{R}^n\\ \|u\|=\|v\|=1}} u^\top Av$$

$$\|A\|_F := \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}^2}$$

$$\|A\|_{\infty\to 1} := \max_{\substack{x\in\{\pm 1\}^m\\ y\in\{\pm 1\}^n}} x^\top Ay = \max_{\substack{x\in[-1,1]^m\\ y\in[-1,1]^n}} x^\top Ay = \max_{x\in\mathbb{R}^n:\|x\|_\infty=1} \|Ax\|_1$$

The relevance of the ∞-to-1 norm stems from the following connection to one-query quantum algorithms:

▶ **Proposition 6** ([3]). *For every quantum algorithm Q making one query to some oracle in $\{\pm 1\}^n$, there exists a bilinear form $p(x,y) = \sum_{i,j=1}^{n+1} A_{ij}x_iy_j$, $A_{ij} \in \mathbb{R}$, such that for every $x \in \{\pm 1\}^n$, the probability that Q accepts x equals $p((x_1,\ldots,x_n,1),(x_1,\ldots,x_n,1))$.*

We refer to p as the *advantage polynomial*, and by abuse of notation, we refer $\|p\|_\#$ to be $\|A\|_\#$ for any norm $\|\cdot\|_\#$. Clearly, the matrix defining any advantage polynomial must have ∞-to-1 norm at most 1 (hence every advantage polynomial is *1-bounded*). In general, this does not imply a constant upper bound on the spectral norm. However, the dual form of Grothendieck's inequality, also known as the *factorization Grothendieck's inequality* [9, P.239], shows that such a bound holds up to factorization.

Grothendieck's factorization

▶ **Proposition 7** ([9]). *There is a universal constant K_G such that if $A \in \mathbb{R}^{n\times n}$ satisfies that $\|A\|_{\infty\to 1} \leq 1$, then there exists $\alpha, \beta \in \mathbb{R}^n_{\geq 0}$ with $\|\alpha\| = \|\beta\| = 1$, such that A can be factored as $A_{ij} = \alpha_i \tilde{A}_{ij} \beta_j$ with $\|\tilde{A}\| \leq K_G$.*

If the matrix A has no all-zero row or all-zero column, then we can further assume that α and β are strictly positive, which is an assumption we can make for advantage polynomials.

If p is the advantage polynomial of a one-query quantum algorithm, the stronger conclusion $\|\tilde{A}\| \leq 1$ can be obtained in Proposition 6 without using Grothendieck's inequality but we will not rely on this fact (at the expense of constant factors in some proofs). We also remark that this factorization can be efficiently found by a semidefinite program.

3 Non-adaptive estimators: Proof of Theorem 2

3.1 Proof of Theorem 2a

Suppose $p(x,y) = \sum_{i,j=1}^{n} A_{ij}x_iy_j$ is the advantage polynomial of a quantum algorithm, so in particular $\|A\|_{\infty\to1} \le 1$. For $I, J \subseteq [n]$, we write A_{IJ} the submatrix of A restricted to rows indexed in I and columns indexed in J, and

$$p_{IJ}(x,y) = \sum_{i\in I, j\in J} A_{ij}x_iy_j.$$

We analyze the following $2k$-query classical distinguisher $P_{2a}(x,y)$:

1. Pick a pair of index sets $I, J \subseteq [n]$, $|I| = |J| = k$, with probability proportional to $\|p_{IJ}\|_{\infty\to1}$.
2. Query all x_i with $i \in I$ and all y_j with $j \in J$.
3. Output $p_{IJ}(x,y)/\|p_{IJ}\|_{\infty\to1}$.

As x and y take ±1 values, the step 3 above, always outputs a value $D(x,y) \in [-1,1]$ as required for a distinguisher. We first show that $D(x,y)$ is an unbiased estimator of $p(x,y)$ up to a scalar.

▷ Claim 8. $Z \cdot D(x,y)$ is an unbiased estimator of $p(x,y)$, where

$$Z = \sum_{I,J:|I|=|J|=k} \|p_{IJ}\|_{\infty\to1} \Big/ \binom{n-1}{k-1}^2.$$

Proof. The probability for choosing the index pair (I,J) in step 1 is given by

$$\frac{\|p_{IJ}\|_{\infty\to1}}{\sum_{I',J':|I'|=|J'|=k} \|p_{I'J'}\|_{\infty\to1}} = \frac{\|p_{IJ}\|_{\infty\to1}}{Z\binom{n-1}{k-1}^2}.$$

Therefore

$$\begin{aligned}
\mathsf{E}[Z \cdot D(x,y)] &= Z \sum_{|I|=|J|=k} \frac{\|p_{IJ}\|_{\infty\to1}}{Z\binom{n-1}{k-1}^2} \cdot \frac{p_{IJ}(x,y)}{\|p_{IJ}\|_{\infty\to1}} \\
&= \frac{1}{\binom{n-1}{k-1}^2} \sum_{|I|=|J|=k} \sum_{i\in I, j\in J} A_{ij}x_iy_j \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}x_iy_j \\
&= p(x,y).
\end{aligned}$$

The second-to-last line uses the fact that each index i appears in exactly $\binom{n-1}{k-1}$ sets I and likewise for j and J. ◁

To complete the analysis we use the following inequality.

▶ **Proposition 9.** *There is a constant C so that for any $n \times n$ matrix A,*

$$\frac{1}{\binom{n}{k}^2} \sum_{|I|=|J|=k} \|A_{IJ}\|_{\infty\to1} \le C\left(\frac{k}{n}\right)^{3/2} \|A\|_{\infty\to1}. \qquad (1)$$

Proof of Theorem 2a. Let p be the advantage polynomial of the one-query quantum algorithm with advantage ε on $(\mathbf{A}, \mathbf{B})$, and A be the matrix defining p. Without loss of generality, we assume

$$\mathsf{E}[p(\mathbf{A})] - \mathsf{E}[p(\mathbf{B})] \geq \varepsilon.$$

From Claim 8, we obtain

$$\mathsf{E}[D(\mathbf{A})] - \mathsf{E}[D(\mathbf{B})] = \frac{1}{Z}(\mathsf{E}[p(\mathbf{A})] - \mathsf{E}[p(\mathbf{B})]) \geq \frac{\varepsilon}{Z}.$$

It remains to upper bound the value of Z. Using Proposition 9 we get

$$Z = \frac{\binom{n}{k}^2}{\binom{n-1}{k-1}^2} \cdot \frac{1}{\binom{n}{k}^2} \sum_{|I|=|J|=k} \|A_{IJ}\|_{\infty\to 1} \leq \frac{n^2}{k^2} \cdot C\left(\frac{k}{n}\right)^{3/2} \|A\|_{\infty\to 1} \leq C\sqrt{\frac{n}{k}}.$$

This concludes the desired advantage bound of $\Omega(\varepsilon\sqrt{k/n})$. ◀

It follows from Proposition 5 that our analysis of D is tight up to constant factor.

Proposition 9 is similar to the following inequality proved by Rudelson and Vershynin [10, Equation (4.1)] who showed that for subsets I_ρ and J_ρ sampled by including each index independently with probability $\rho = k/n$,

$$\mathsf{E}[\|A_{I_\rho J_\rho}\|_{\infty\to 1}] \leq C'\rho^{3/2}\big(\|A\|_{\text{row}} + \|A\|_{\text{col}}\big) + C'\rho^2\|A\|_{\infty\to 1}, \tag{2}$$

for some constant C'. Here, $\|A\|_{\text{row}} = \sum_i \|A^{i,:}\|$ and $\|A\|_{\text{col}} = \sum_j \|A^{:,j}\|$ denote the sum of the 2-norms of its rows and columns, respectively. For completeness, we present the derivation Proposition 9 from (2) using Poissonization [13] (see also [12]).

Proof of Proposition 9. Tropp [13] showed that

$$\frac{1}{\binom{n}{k}^2} \sum_{|I|=|J|=k} \|A_{IJ}\|_{\#} \leq 4\mathsf{E}[\|A_{I_\rho J_\rho}\|_{\#}],$$

for every matrix norm $\|\cdot\|_{\#}$ that satisfies $\|A'\|_{\#} \leq \|A\|_{\#}$ for every submatrix A' of A. This is in particular true for the ∞-to-1 norm: if $\|A'\|_{\infty\to 1} = x'^{\top} A' y'$ for $x', y' \in [-1,1]^m$, then $\|A\|_{\infty\to 1} \geq x^{\top} A y = x'^{\top} A' y'$ where x and y are extended from x' and y' with zeros padded in the remaining entries. It remains to prove that $\|A\|_{\text{row}}, \|A\|_{\text{col}} \leq K_G \|A\|_{\infty\to 1}$. ◀

▷ Claim 10. For any $M \in \mathbb{R}^{m\times n}$, $\|M^{i,:}\| \leq \|M\|$ and $\|M^{:,j}\| \leq \|M\|$ for any $i \in [m]$, $j \in [n]$.

Proof. We prove the case of $\|M^{i,:}\|$ and the other case follows the same proof:

$$\|M^{i,:}\|^2 = e_i^{\top} M (M^{i,:}) \leq \|M\| \cdot \|M^{i,:}\| \implies \|M^{i,:}\| \leq \|M\|.$$ ◁

▷ Claim 11. $\|A\|_{\text{row}} \leq K_G \|A\|_{\infty\to 1}$.

Proof. By re-scaling, we assume $\|A\|_{\infty\to 1} = 1$ without loss of generality. Let $A_{ij} = \alpha_i \tilde{A}_{ij} \beta_j$be the Grothendieck's factorization of A (Proposition 7), by Cauchy-Schwarz inequality,

$$\|A\|_{\text{row}} = \sum_i \sqrt{\sum_j A_{ij}^2} = \sum_i \alpha_i \cdot \sqrt{\sum_j \beta_j^2 \tilde{A}_{ij}^2} \leq \sqrt{\sum_i \alpha_i^2}\sqrt{\sum_{ij} \beta_j^2 \tilde{A}_{ij}^2} = \sqrt{\sum_j \beta_j^2 \|\tilde{A}^{:,j}\|^2}.$$

By Claim 10 and the bound $\|\tilde{A}\| \leq K_G$, we conclude that $\|A\|_{\text{row}} \leq \sqrt{\sum_j \beta_j^2 \cdot K_G^2} = K_G$. ◁

3.2 The bias of truncated unbiased estimators

In preparation for the proof of Theorem 2b, we prove a general bound of the bias arising from truncating an unbiased estimator of low variance. Denote trunc: $\mathbb{R} \to [-1, 1]$ the truncation function

$$\mathrm{trunc}(t) = \begin{cases} t, & \text{if } |t| \leq 1, \\ \mathrm{sign}(t), & \text{if } |t| > 1. \end{cases}$$

▶ **Proposition 12.** *Assume $\|f\|_\infty = 1$, $Z \geq 1$, and $F_\mathbf{r}$ is a random function such that $\mathsf{E}[F_\mathbf{r}(x)] = f(x)$ for all x (here r denotes the randomness). The distinguisher $D_\mathbf{r}(x) = \mathrm{trunc}(F_\mathbf{r}(x)/Z)$ has advantage at least*

$$\frac{\varepsilon}{Z} - 2\max_x \int_{1-1/Z}^{\infty} \mathsf{Pr}_r\big(|F_\mathbf{r}(x) - f(x)| \geq Zt\big)dt.$$

for any pair of random variables that are ε-distinguishable by f.

▶ **Corollary 13.** *Under the assumptions of Proposition 12, D has advantage at least*

$$\frac{\varepsilon}{Z} - \frac{2}{Z(Z-1)} \cdot \max_x \mathsf{Var}\, F_\mathbf{r}(x).$$

Proof. Using Chebyshev's inequality, the integrand appearing in Proposition 12 is at most $(\mathsf{Var}\, F_\mathbf{r}(x))/Z^2t^2$ and so the integral is at most $(\mathsf{Var}\, F_\mathbf{r}(x))/Z(Z-1)$. ◀

The proposition is derived from the following claim:

▷ Claim 14. Let Y be a random variable with $|\mathsf{E}[Y]| \leq 1$, then

$$\big|\mathsf{E}[\mathrm{trunc}(Y)] - \mathsf{E}[Y]\big| \leq \int_{1-|\mathsf{E}[Y]|}^{\infty} \mathsf{Pr}(|Y - \mu| \geq t)dt.$$

Proof of Proposition 12. We apply Claim 14 to the random variable $F_\mathbf{r}(x)/Z$ to obtain

$$\begin{aligned} |\mathsf{E}[D_\mathbf{r}(x)] - f(x)/Z| &\leq \int_{1-|f(x)/Z|}^{\infty} \mathsf{Pr}_r\big(|F_\mathbf{r}(x)/Z - f(x)/Z| \geq t\big)dt \\ &\leq \int_{1-1/Z}^{\infty} \mathsf{Pr}_r\big(|F_\mathbf{r}(x) - f(x)| \geq Zt\big)dt. \end{aligned}$$

Suppose $(\mathbf{A}, \mathbf{B})$ is ε-distinguishable by f. By the triangle inequality, $|\mathsf{E}[D_\mathbf{r}(\mathbf{A})] - \mathsf{E}[f(\mathbf{A})]/Z|$ is at most the maximum of the integral over x, and the same bound holds for replacing $\mathbf{A}$ by $\mathbf{B}$. Now the bound on advantage follows from triangle inequality. ◀

In the proof of Claim 14 we use the following fact:

▶ **Fact 15.** $|\mathrm{trunc}(t) - t| = \max\{0, |t| - 1\}$.

Proof of Claim 14. Let $\mu = \mathsf{E}[Y]$.

$$\begin{aligned}
|\mathsf{E}[\mathrm{trunc}(Y)] - \mathsf{E}[Y]| &\le \mathsf{E}[|\mathrm{trunc}(Y) - Y|] \\
&= \int_0^\infty \Pr(|\mathrm{trunc}(Y) - Y| \ge t)dt \\
&= \int_0^\infty \Pr(|Y| - 1 \ge t)dt && \text{(Fact 15)} \\
&= \int_0^\infty \Pr(|Y| \ge t + 1)dt \\
&\le \int_0^\infty \Pr(|Y - \mu| \ge t + 1 - |\mu|)dt && \text{(triangle inequality)} \\
&= \int_{1-|\mu|}^\infty \Pr(|Y - \mu| \ge t)dt && \text{(change of variables)} \qquad \triangleleft
\end{aligned}$$

3.3 Proof of Theorem 2b

Let $p(x,y) = \sum_{ij} A_{ij}x_iy_j$ be the advantage polynomial so that $\|A\|_{\infty\to 1} = 1$. We let $A_{ij} = \alpha_i\tilde{A}_{ij}\beta_j$ to be the Grothendieck's factorization of A. We analyze the following $2k$-query estimator:

1. Sample a sequence $I = (I(1), \dots, I(k))$ of k i.i.d indices by picking each $i \in [n]$ with probability $p_i := \alpha_i/\|\alpha\|_1$. Query the inputs $x_{I(u)}$ for $u \in [k]$.
2. Sample a sequence $J = (J(1), \dots, J(k))$ of k i.i.d indices by picking each $j \in [n]$ with probability $q_j := \beta_j/\|\beta\|_1$. Query the inputs $y_{J(v)}$ for $v \in [k]$.
3. Output the empirical average

$$P(x,y) = \mathsf{E}_{i\sim I, j\sim J}\left[A_{ij}\frac{x_iy_j}{p_iq_j}\right].$$

Clearly this estimator makes at most $2k$ queries. Now we show that this is an unbiased estimator of bounded variance.

▶ **Proposition 16.** *$P(x,y)$ is an unbiased estimator of $p(x,y)$ of variance at most $O(n/k^2)$.*

Proof. Unbiasedness follows from linearity of expectation:

$$\mathsf{E}[P(x,y)] = \mathsf{E}\Big[\mathsf{E}\Big[A_{ij}\frac{x_iy_j}{p_iq_j} \Bigm| I, J\Big]\Big] = \mathsf{E}\Big[A_{ij}\frac{x_iy_j}{p_iq_j}\Big] = \sum_{i,j} A_{ij}\frac{x_iy_j}{p_iq_j}\cdot p_iq_j = p(x,y).$$

In preparation for calculating the variance, let $B_{uv} = A_{I(u)J(v)}/p_{I(u)}q_{J(v)}$. By independence and the fact that $x_i^2 = y_j^2 = 1$,

$$\mathsf{Cov}(B_{uv}, B_{u'v'}) = \begin{cases} \mathsf{E}\left[\frac{A_{ij}^2}{p_i^2q_j^2}\right] - p(x,y)^2, & \text{if } u = u' \text{ and } v = v' \\ \mathsf{E}\left[\frac{A_{ij}A_{ij'}y_jy_{j'}}{p_i^2q_jq_{j'}}\right] - p(x,y)^2, & \text{if } u = u' \text{ and } v \ne v' \\ \mathsf{E}\left[\frac{A_{ij}A_{i'j}x_ix_{i'}}{p_ip_{i'}q_j^2}\right] - p(x,y)^2, & \text{if } u \ne u' \text{ and } v = v' \\ 0, & \text{otherwise.} \end{cases}$$

Here i, i' and j, j' denote random indices chosen independently. Decomposing $\mathsf{Var}[P(x,y)]$ as an average of covariances, we obtain

$$\begin{aligned}\mathsf{Var}[P(x,y)] &= \frac{1}{k^4}\sum_{u,v,u',v'}\mathsf{Cov}\big(B_{uv},B_{u'v'}\big)\\ &\le \frac{1}{k^2}\mathsf{E}\left[\frac{A_{ij}^2}{p_i^2q_j^2}\right]+\frac{k-1}{k^2}\left(\mathsf{E}\left[\frac{A_{ij}A_{ij'}y_jy_{j'}}{p_i^2q_jq_{j'}}\right]+\mathsf{E}\left[\frac{A_{ij}A_{i'j}x_ix_{i'}}{p_ip_{i'}q_j^2}\right]\right).\end{aligned}\tag{3}$$

We bound the three types of terms using Grothendieck's factorization of A.

$$\begin{aligned}\mathsf{E}\left[\frac{A_{ij}^2}{p_i^2q_j^2}\right] &= \sum_{i,j}\frac{A_{ij}^2}{p_iq_j}\\ &= \|\alpha\|_1\cdot\|\beta\|_1\cdot\sum_{i,j}\alpha_i\beta_j\tilde{A}_{ij}^2 && \text{(Grothendieck's factorization)}\\ &\le \|\alpha\|_1\cdot\|\beta\|_1\cdot\sqrt{\sum_{i,j}\alpha_i^2\tilde{A}_{ij}^2}\cdot\sqrt{\sum_{i,j}\beta_j^2\tilde{A}_{ij}^2} && \text{(Cauchy-Schwarz inequality)}\\ &= \|\alpha\|_1\cdot\|\beta\|_1\cdot\sqrt{\sum_i\alpha_i^2\|\tilde{A}^{i,:}\|^2}\cdot\sqrt{\sum_j\beta_j^2\|\tilde{A}^{:,j}\|^2}\\ &\le \|\alpha\|_1\cdot\|\beta\|_1\cdot\|\tilde{A}\|\cdot\|\tilde{A}\| && \text{(Claim 10)}\\ &\le n\cdot K_G^2. && \text{(Cauchy-Schwarz inequality)}\end{aligned}$$

$$\begin{aligned}\mathsf{E}\left[\frac{A_{ij}A_{ij'}y_jy_{j'}}{p_i^2q_jq_{j'}}\right] &= \sum_{i,j,j'}\frac{A_{ij}A_{ij'}y_jy_{j'}}{p_i}\\ &= \sum_i\frac{1}{p_i}\left(\sum_j A_{ij}y_j\right)^2\\ &\le \|\alpha\|_1\sum_i\alpha_i\left(\sum_j\tilde{A}_{ij}y_j\beta_j\right)^2 && \text{(Grothendieck's factorization)}\\ &\le \|\alpha\|_1\sum_i(\tilde{A}\beta^y)_i^2 && \text{(Define } (\beta^y)_j:=y_j\beta_j\text{; and } \alpha_i\in[0,1])\\ &= \|\alpha\|_1\|\tilde{A}\beta^y\|^2\\ &\le \|\alpha\|_1\|\tilde{A}\|^2 && (\|\beta^y\|=\|\beta\|=1)\\ &\le \sqrt{n}\cdot K_G^2. && \text{(Cauchy-Schwarz inequality)}\end{aligned}$$

By symmetry the third term is also at most $\sqrt{n}K_G^2$. Plugging into (3), we obtain

$$\mathsf{Var}[P(x,y)] = O(n/k^2+\sqrt{n}/k) = O(n/k^2).$$ ◀

Proof of Theorem 2b. Unbiasedness follows from linearity of expectation. Let V be the variance bound from Proposition 16. We instantiate Corollary 13 with this P and $Z = 1+4V/\varepsilon$. This is at most 1 provided $k^2 = o(n)$. The resulting distinguishing advantage is $\Omega(\varepsilon^2/V)$. ◀

The advantage of any distinguisher with the same distribution over samples cannot be better than $\varepsilon k^2/n$. Therefore our analysis is optimal in terms of k and n. To see this, consider the distribution in which the bit-pairs (x_i,y_i) are unbiased, ε-correlated, and mutually independent. The resultant bilinear form $(\sum x_iy_i)/n$ is 1-bounded, which corresponds to a one-query quantum distinguisher; and it distinguishes this distribution from random with advantage ε. In contrast, the advantage of a classical distinguisher is at most ε times the expected number of collisions $i=j$ with $i\in I$ and $j\in J$, which is at most $\varepsilon k^2/n$.

4 An adaptive estimator: Proof of Theorem 3

We modify the estimator of Section 3.3 so that the values $\{x_i : i \in I\}$ adaptively affect the probabilities for index sampling of J. Again we assume $\|A\|_{\infty\to 1} = 1$ and let $A_{ij} = \alpha_i \tilde{A}_{ij}\beta_j$ to be the Grothendieck's factorization of A.

1. Choose a sample I of k i.i.d indices by picking each $i \in [n]$ with probability $p_i = \alpha_i/\|\alpha\|_1$. Query the inputs x_i for $i \in I$. Let $a_I^x \in \mathbb{R}^n$ be defined by $[a_I^x]_j := \mathsf{E}_{i\sim I}[A_{ij}x_i/p_i]$.
2. Choose a sample J of k i.i.d indices by picking each $j \in [n]$ with probability $q_j = |[a_I^x]_j|/\|a_I^x\|_1$. Query the inputs y_j for $j \in J$.
3. Output the empirical average $P(x,y) = \mathsf{E}_{j\sim J}[[a_I^x]_j y_j/q_j]$.

This estimator is unbiased by linearity of expectation. The main technical result of this section is the following deviation bound:

▶ **Proposition 17.** *There is a constant C such that for all x, y, $\varepsilon > 0$, and $t > 0$,*

$$\mathsf{Pr}\left(|P(x,y) - p(x,y)| \geq \frac{C\sqrt{n\log n}}{k}\frac{t}{\varepsilon}\right) \leq \frac{k}{\sqrt{n}\log n}\left(\frac{\varepsilon}{t}\right)^2 + 2n^{-(t/\varepsilon)^2}.$$

Proof of Theorem 3. With a (possible) change in the constant factor in the lower bound, we may assume that $\varepsilon \leq \varepsilon_0$ for a sufficiently small constant ε_0 and $Z := C\sqrt{n\log n}/k\varepsilon \geq 2$. We apply Proposition 17 to bound the integral in Proposition 12 by

$$\begin{aligned}
&\int_{1-1/Z}^{\infty} \mathsf{Pr}\big(|P(x,y) - p(x,y)| \geq Zt\big) \\
&\quad\leq \frac{C}{\sqrt{\log n}} \cdot \frac{\varepsilon}{Z} \int_{1-1/Z}^{\infty} \frac{dt}{t^2} + 2\int_{1-1/Z}^{\infty} n^{-(t/\varepsilon)^2} dt \\
&\quad= \frac{C}{\sqrt{\log n}} \cdot \frac{\varepsilon}{Z} \cdot \frac{1}{1-1/Z} + \sqrt{\frac{4\pi\varepsilon^2}{\log n}} \cdot \mathsf{Pr}\big(\mathcal{N}(0, \varepsilon^2/2\log n) \geq 1 - 1/Z\big) \\
&\quad\leq \frac{2C}{\sqrt{\log n}} \cdot \frac{\varepsilon}{Z} + \sqrt{\frac{4\pi\varepsilon^2}{\log n}} \cdot \mathsf{Pr}\big(\mathcal{N}(0,1) \geq \sqrt{\log n/2\varepsilon^2}\big) \\
&\quad\leq \frac{2C}{\sqrt{\log n}} \cdot \frac{\varepsilon}{Z} + \sqrt{\frac{4\pi\varepsilon^2}{\log n}} \cdot n^{-1/\varepsilon^2} \\
&\quad\leq \frac{\varepsilon}{6Z} + \frac{\varepsilon}{6Z}.
\end{aligned}$$

The second to last inequality is the Gaussian tail bound. The last inequality holds for sufficiently large n using the assumption that $\varepsilon \leq \varepsilon_0$. By Proposition 12, D has advantage at least $\varepsilon/3Z$. ◀

To prove Proposition 17, we split the difference between P and p via the "hybrid" $P'(x,y) = (a_I^x)^\top y = \sum_j [a_I^x]_j y_j$. Claims 18 show that P' has small variance and is therefore close to P. Claim 19 shows that P' is typically close to P.

▷ Claim 18. $\mathsf{Var}[P'(x,y)] \leq K_G^2\sqrt{n}/k$.

▷ Claim 19. $\mathsf{Pr}[|P(x,y) - P'(x,y)| \geq t \cdot K_G\sqrt{n}/k \mid I] \leq 2\exp(-t^2/2)$ for every $t > 0$.

Proof of Proposition 17. By Claim 18 and Chebyshev's inequality, for every $t_1 > 0$,

$$\mathsf{Pr}\big[|P'(x,y) - p(x,y)| \geq t_1 \cdot K_G(\sqrt{n}/k)^{1/2}\big] \leq \frac{1}{t_1^2}.$$

Using Claim 19 together with a union bound and the triangle inequality, it follows that

$$\mathsf{Pr}\big[|P(x,y)-p(x,y)| \geq t_1 \cdot K_G(\sqrt{n}/k)^{1/2} + t_2 \cdot K_G\sqrt{n}/k\big] \leq \frac{1}{t_1^2} + 2\exp(-t_2^2/2)$$

for every $t_1 > 0$ and $t_2 > 0$. Plugging in $t_1 = (t/\varepsilon)\cdot(\sqrt{n}\log n/k)^{1/2}$ and $t_2 = (t/\varepsilon)\cdot(2\log n)^{1/2}$ gives the desired inequality. ◀

Proof of Claim 18. As the samples in I are independent,

$$\mathsf{Var}[P'(x,y)] = \frac{1}{k}\mathsf{Var}_i\left[\sum_j \frac{A_{ij}x_iy_j}{p_i}\right] = \frac{1}{k}\mathsf{Var}_i\left[\sum_j \frac{A_{ij}y_j}{p_i}\right]$$

because $x_i^2 = 1$. As i is sampled with probability $p_i = \alpha_i/\|\alpha\|_1$, we get

$$\begin{aligned}
\mathsf{Var}\left[\sum_j \frac{A_{ij}y_j}{p_i}\right] &\leq \sum_i p_i\Big(\sum_j \frac{A_{ij}y_j}{p_i}\Big)^2 \\
&\leq \|\alpha\|_1 \sum_i \alpha_i\Big(\sum_j \tilde{A}_{ij}y_j\beta_j\Big)^2 && \text{(Grothendieck's factorization)} \\
&\leq \|\alpha\|_1 \sum_i (\tilde{A}\beta^y)_i^2 && \text{(Define } (\beta^y)_j := y_j\beta_j\text{; and } \alpha_i \in [0,1]) \\
&\leq \|\alpha\|_1 \cdot \|\tilde{A}\|^2 && (\|\beta^y\| = \|\beta\| = 1) \\
&\leq \sqrt{n}\cdot K_G^2. && \text{(Cauchy-Schwarz inequality)}
\end{aligned}$$

◀

Proof of Claim 19. Since $[a_I^x]_j y_j/q_j = \|a_I^x\|_1 y_j$, conditioned on I, $P(x,y)$ is an average of k independent random variables taking values either $-\|a_I^x\|_1$ or $\|a_I^x\|_1$ with mean $P'(x,y)$. Applying the Chernoff-Hoeffding bound to $kP/\|a_I^x\|_1$, we obtain

$$\mathsf{Pr}\big[|P(x,y)-P'(x,y)| \geq t\|a_I^x\|_1/\sqrt{k} \;\big|\; I\big] \leq 2\exp(-t^2/2).$$

It remains to show that $\|a_I^x\|_1 \leq K_G\sqrt{n/k}$ for every choice of I:

$$\begin{aligned}
\|a_I^x\|_1 &= \sum_j \left|\frac{1}{k}\sum_{i\in I}\frac{x_iA_{ij}}{p_i}\right| \\
&= \frac{\|\alpha\|_1}{k}\sum_j \beta_j\left|\sum_{i\in I} x_i\tilde{A}_{ij}\right| && \text{(Grothendieck's factorization)} \\
&\leq \frac{\|\alpha\|_1}{k}\sqrt{\sum_j\Big(\sum_{i\in I}x_i\tilde{A}_{ij}\Big)^2} && \text{(Cauchy-Schwarz inequality)} \\
&= \frac{\|\alpha\|_1}{k}\sqrt{\sum_j (x_I^\top\tilde{A})_j^2} && \text{(Define } (x_I)_i := x_i\cdot\mathbf{1}(i\in I)) \\
&= \frac{\|\alpha\|_1}{k}\|x_I^\top\tilde{A}\| \\
&\leq \frac{\|\alpha\|_1\cdot\|x_I\|\cdot\|\tilde{A}\|}{k} \\
&\leq \frac{\sqrt{n}\cdot\sqrt{k}\cdot K_G}{k}. && \text{(Cauchy-Schwarz inequality)}
\end{aligned}$$

◀

As mentioned, Theorem 4 shows that the distinguisher in Theorem 3 is best possible up to a factor of $\log n$.

5 Classical advantage upper bounds: Proofs of Proposition 1 and Theorem 4

To start this section, we first present the proof for the classical advantage upper bound of non-adaptive algorithms.

Proof of Proposition 1. Aaronson and Ambainis [1] show that F is $\Omega(1)$-distinguishable from the uniform random U by one quantum query. The random variable F is obtained by rounding a pair of n-dimensional Gaussians (X, Y) where X is standard Gaussian and Y is obtained by applying the Hadamard matrix to X. So the non-adaptive classical $2k$-query advantage is upper bounded by the maximum statistical distance between the projections (X_I, Y_J) over all sets I, J with $|I| + |J| = 2k$ and a standard $2k$-dimensional Gaussian.

In general, the statistical distance between centered multivariate Gaussians with covariance matrices Σ_1 and Σ_2 is $\Theta(1)\min\{1, \|\Sigma_2^{-1}\Sigma_1 - I\|_F\}$ [7]. As Σ_2 is the identity and all non-diagonal entries of Σ_1 are $\pm 1/\sqrt{n}$, it follows that $\|\Sigma_2^{-1}\Sigma_1 - Id\|_F = O(k/\sqrt{n})$.

Setting F_ε as $\varepsilon F + (1-\varepsilon)U$, the advantage of any distinguisher, classical or quantum, scales precisely by ε. ◀

As for the classical advantage upper bound of two-round algorithms, the proof of Theorem 4 bounds the statistical distance between the distinguisher's views on the two distributions via their KL-divergence. We need the following explicit formula for KL-divergence of multivariate Gaussians:

▶ **Fact 20.** $\mathsf{kl}(\mathcal{N}(\mu, \Sigma), \mathcal{N}(0, Id_k)) = \frac{1}{2}(\|\mu\|^2 + \mathsf{tr}(\Sigma - Id_k) - \log \mathsf{det}\Sigma)$.

The following consequence of this formula is implicit in [7]:

▷ Claim 21. Assuming $\lambda_{\min}(\Sigma) \geq 1/3$, $\mathsf{kl}((\mathcal{N}(\mu, \Sigma), \mathcal{N}(0, Id_k)) \leq \frac{1}{2}(\|\mu\|^2 + \|\Sigma - Id_k\|_F^2)$.

Proof. Let $\eta_1, \ldots, \eta_k$ be the eigenvalues of $\Sigma - Id_k$. By assumption $\eta_1, \ldots, \eta_k \geq -2/3$. Then

$$\mathsf{tr}(\Sigma - Id_k) - \log \mathsf{det}\Sigma = \sum_{i=1}^{k}(\eta_i - \log(1+\eta_i)) \leq \sum_{i=1}^{k}\eta_i^2 = \|\Sigma - Id_k\|_F^2,$$

where the inequality uses the fact that $\eta - \log(1+\eta) \leq \eta^2$ for all $\eta \geq -2/3$. ◁

The requirement $\lambda_{\min}(\Sigma) \geq 1/3$ is satisfied by matrices that are close to the identity in the following sense:

▶ **Fact 22.** *If $A \in \mathbb{R}^{k\times k}$ is a symmetric matrix with $|A_{ij}| \leq \varepsilon$ for all $i, j \in [k]$, then $\lambda_{\min}(Id_k + A) \geq 1 - k\varepsilon$ and $\lambda_{\max}(Id_k + A) \leq 1 + k\varepsilon$.*

In particular, as long as Σ is $2/3k$-close to the identity in (entrywise) infinity-norm, the bound in Claim 21 applies.

Another tool we need is the chain rule for KL-divergence:

▶ **Fact 23** (Chain rule for KL-divergence)**.** $\mathsf{kl}((U, V), (U', V')) = \mathsf{kl}(U, U') + \mathsf{kl}(V|U, V'|U')$, *where* $\mathsf{kl}(V|U, V'|U') = \mathsf{E}_{u\sim U}\mathsf{kl}(V|U = u, V'|U' = u)$.

In our application of Fact 23, $(U, V) = (U_1, \ldots, U_k, V_1, \ldots, V_k)$ is a multivariate Gaussian. This class of distributions is closed under conditioning. To calculate the effect of conditioning on the parameters, we identify the zero-mean (assumed without loss of generality) random variables $U_1, \ldots, U_k, V_1, \ldots, V_k$ with vectors in Hilbert space endowed with the inner product $\mathsf{E}[A \cdot B]$. The conditional means and conditional covariances of $V|U$ are given by

$$\mathsf{E}[V_j|U] = V_j^{\|} \tag{4}$$

$$\mathsf{Cov}[V_j, V_{j'}|U] = \mathsf{E}[V_j^{\perp} \cdot V_{j'}^{\perp}], \tag{5}$$

where $V_j = V_j^{\|} + V_j^{\perp}$ is the orthogonal decomposition of V_j into a parallel component $V_j^{\|} \in \mathrm{Span}(U)$ and a perpendicular component $V_j^{\perp} \in \mathrm{Span}(U)^{\perp}$. As $V_j^{\|}$ is in $\mathrm{Span}(U)$, its value is determined by $U_1, \ldots, U_k$. And as $V_j^{\perp}$ and $V_{j'}^{\perp}$ are in $\mathrm{Span}(U)^{\perp}$, their values are independent of $U_1, \ldots, U_k$.

Lastly we will use the following fact:

▶ **Fact 24.** *If M is the maximum of n standard Gaussian random variables, then $\mathsf{E}\big[M^2\big] \leq 4\log(\sqrt{2}n)$.*

Proof of Fact 24. By Jensen's inequality, for every $t \in (0, 1/2)$,

$$\exp(t\mathsf{E}[M^2]) \leq \mathsf{E}[\exp(tM^2)] \leq \mathsf{E}[n\exp(t\mathcal{N}(0,1)^2)] = \frac{n}{\sqrt{1-2t}}.$$

Here, the last equality follows from the formula of the moment-generating function of a squared Gaussian. We obtain the desired formula by setting $t = 1/4$ and taking logarithms. ◀

Proof of Theorem 4. The random variable $F = (\operatorname{sign} X, \operatorname{sign} Y)$ is the same as in Proposition 1. As in the previous proof, we first reduce to the case when ε is constant and use the same notations in that proof.

Aaronson and Ambainis showed that $\mathsf{E}[\mathrm{sign}(X)(H/n)\,\mathrm{sign}(Y)] = \Omega(1)$, where H is the $n \times n$ Hadamard matrix. As $\|H/n\|_{\infty\to 1} \leq \|H\| \leq 1$, this justifies the quantum advantage.

For the classical case, as taking signs can only decrease advantage, we upper bound the advantage of distinguishing $Z = (X, Y)$ from $\mathcal{N}(0, Id_{2n})$. The only relevant property of Z is that $\mathsf{E}[Z_i] = 1$ and $|\mathsf{E}[Z_i Z_j]| \leq 1/\sqrt{n}$ for all pairs $i \neq j$. Without loss of generality, we assume that $k \leq \sqrt{n}/4$.

The distinguisher's strategy is specified by the query sets I and J with $|I| = |J| = k$, issued in the first and second round, respectively. For the sake of upper bound, we can assume without loss of generality that J is a deterministic function of the coordinates $Z_I = (Z_i)_{i\in I}$ observed in the first round. The distinguisher's advantage is at most

$$\begin{aligned}
\varepsilon_C &= \max_{I,J} \mathsf{sd}((Z_I, Z_J), \mathcal{N}(0, Id_{2k})) \\
&\leq \sqrt{\frac{1}{2}\max_{I,J} \mathsf{kl}((Z_I, Z_J), \mathcal{N}(0, Id_{2k}))} && \text{(Pinsker's inequality)} \\
&= \sqrt{\frac{1}{2}\Big(\max_{I} \mathsf{kl}(Z_I, \mathcal{N}(0, Id_k)) + \max_{I}\max_{J} \mathsf{E}_{Z_I}\mathsf{kl}(Z_J|Z_I, \mathcal{N}(0, Id_k))\Big)} && \text{(Fact 23)} \\
&\leq \sqrt{\frac{1}{2}\Big(\max_{I} \mathsf{kl}(Z_I, \mathcal{N}(0, Id_k)) + \max_{I} \mathsf{E}_{Z_I}\max_{J} \mathsf{kl}(Z_J|Z_I, \mathcal{N}(0, Id_k))\Big)} && \text{(convexity of max)}
\end{aligned}$$

As $k \leq \frac{2}{3}\sqrt{n}$, the covariance matrix Σ_I of Z_I is $2/3k$-close to the identity in infinity norm, so using Claim 21, for every I one has

$$\mathsf{kl}(Z_I, \mathcal{N}(0, Id_k)) \leq \frac{1}{2}\|\Sigma_I - Id_k\|_F \leq \frac{k^2}{2n}. \tag{6}$$

For the second KL-divergence, let $\mu_{J|I}$ and $\Sigma_{J|I}$ denote the vector of conditional means $(\mathsf{E}[Z_j|Z_I])_{j\in J}$ and covariances $\mathsf{Cov}[Z_j, Z_{j'}|Z_I]$ for $j, j' \in J$, respectively. We will prove that for all choices of I and J, $\Sigma_{J|I}$ is $2/3k$-close to Id_k and apply Claim 21 to bound it by

$$\mathsf{E}_{Z_I} \max_J \mathsf{kl}(Z_J|Z_I, \mathcal{N}(0, Id_k)) \le \frac{1}{2} \max_J \mathsf{E}\big[\|\mu_{J|I}\|^2\big] + \frac{1}{2} \max_J \mathsf{E}\big[\|\Sigma_{J|I}\|_F^2\big]. \tag{7}$$

To bound the first term in (7), we analyze the projections $Z_j^{\|}$ of Z_j onto $\mathrm{Span}\{Z_i \colon i \in I\}$ for every $j \notin I$. Fix a basis for the vector space spanned by $Z_1, \dots, Z_{2n}$ and let z_i be the representation of Z_i under this basis. Let B be the $k \times n$ matrix whose rows are z_i for $i \in I$. The projection $z_j^{\|}$ of z_j onto the row-span of B is given by the formula

$$z_j^{\|} = B^\top (BB^\top)^{-1} B z_j.$$

The norm of this projection is at most

$$\|z_j^{\|}\| \le \lambda_{\max}(BB^\top)^{1/2} \cdot \lambda_{\min}(BB^\top)^{-1} \cdot \|Bz_j\| \le \left(1 + \frac{k}{\sqrt{n}}\right)^{1/2} \left(1 - \frac{k}{\sqrt{n}}\right)^{-1} \cdot \sqrt{\frac{k}{n}} \le 2\sqrt{\frac{k}{n}}.$$

The second inequality follows from Fact 22 as $BB^\top$ is $1/\sqrt{n}$-close to the identity and the entries of Bz_j are all bounded by $1/\sqrt{n}$. The third inequality follows from the assumption $k \le \sqrt{n}/3$.

By (4), for every j, $\mathsf{E}[Z_j|Z_I]$ is a Gaussian random variable of mean zero and standard deviation at most $2\sqrt{k/n}$. Letting $\mu_{J|I}$ denote the vector of conditional means $(\mathsf{E}[Z_j|Z_I])_{j \in J}$, by Fact 24, for any fixed I,

$$\max_J \mathsf{E}\big[\|\mu_{J|I}\|^2\big] \le k\mathsf{E} \max_{j \in [n] \setminus I} \mathsf{E}[Z_j|Z_I]^2 \le k \left(2\sqrt{\frac{k}{n}}\right)^2 \cdot 4 \log(\sqrt{2}n) = \frac{16k^2 \log(\sqrt{2}n)}{n}. \tag{8}$$

For the second term in (7), we apply (5) to obtain

$$\mathsf{Cov}[Z_j, Z_{j'}|Z_I] = \mathsf{E}[Z_j^\perp \cdot Z_{j'}^\perp] = \mathsf{E}[Z_j \cdot Z_{j'}] - \mathsf{E}[Z_j^{\|} \cdot Z_{j'}^{\|}]$$

by orthogonality, from which we have

$$\big|\mathsf{Cov}[Z_j, Z_{j'}|Z_I] - \mathsf{E}[Z_j \cdot Z_{j'}]\big| \le \|z_j^{\|}\| \cdot \|z_{j'}^{\|}\| \le \frac{4k}{n} \le \frac{1}{\sqrt{n}}. \tag{9}$$

As $\mathsf{E}[Z_j \cdot Z_{j'}]$ is $1/\sqrt{n}$ close to the identity, we conclude that $\Sigma_{J|I}$ is $2/\sqrt{n} \le 2/3k$-close to the identity. Therefore Claim 21 applies. Plugging (8) and (9) into (7) we obtain

$$\mathsf{E}_{Z_I} \max_J \mathsf{kl}(Z_J|Z_I, \mathcal{N}(0, Id_k)) \le \frac{1}{2} \cdot \frac{16k^2 \log(\sqrt{2}n)}{n} + \frac{1}{2} \cdot k^2 \cdot \frac{4}{n} = O\Big(\frac{k^2 \log n}{n}\Big).$$

Together with (6), this gives $\varepsilon_C = O(k\sqrt{\log n}/\sqrt{n})$ as desired. ◀

6 Optimality of local unbiased estimators: Proof of Proposition 5

Proof of Proposition 5. With a (possible) change in the constant factor in the lower bound, we may assume each form in the mixture depends on k bits of x and k bits of y. Let $p(x, y) = \sum A_{ij} x_i y_j$, and let B_{IJ} be a matrix supported on $I, J \subseteq [n]$ with $|I| = |J| = k$:

$$[B_{IJ}]_{ij} = \begin{cases} b_{ij} & \text{if } i \in I, j \in J \\ 0 & \text{otherwise} \end{cases}.$$

Suppose $\|A\|_{\infty\to 1} = 1$ and $A = \mathsf{E}_{I,J}[B_{IJ}]$, where the expectation is over an arbitrary distribution over pairs I, J. It is sufficient to show that $\|B_{IJ}\|_{\infty\to 1} = \Omega(\sqrt{n/k})$ for at least one choice of the index sets (I, J).

We prove the contrapositive. Suppose $\|B_{IJ}\|_{\infty\to 1} \le \varepsilon$ for all B_{IJ}. By Claim 11, $\|B_{IJ}\|_{\text{row}} \le K_G \cdot \varepsilon$, so $\sum_{i\in I, j\in J} |[B_{IJ}]_{ij}| \le K_G \cdot \varepsilon\sqrt{k}$. Therefore

$$\sum_{i,j} |A_{ij}| = \sum_{i,j} |\mathsf{E}_{I,J}\left[[B_{IJ}]_{ij}\right]| \le \mathsf{E} \sum_{i\in I, j\in J} |[B_{IJ}]_{ij}| \le K_G \cdot \varepsilon\sqrt{k}.$$

For n being a power of two, let A be the $n \times n$ Hadamard matrix scaled by $1/n$, such that $\|A\|_{\infty\to 1} = n$. Then $|A_{ij}| = n^{-3/2}$ for all i and j, thus $K_G \cdot \varepsilon\sqrt{k} \ge \sqrt{n}$ and hence $\varepsilon = \Omega(\sqrt{n/k})$ as desired. If n is not a power of two, we plant the largest possible Hadamard matrix and zero out the remaining entries, the same argument still applies. ◀

References

1 Scott Aaronson and Andris Ambainis. Forrelation: A problem that optimally separates quantum from classical computing. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 307–316, 2015. doi:10.1145/2746539.2746547.

2 Scott Aaronson, Andris Ambainis, Andrej Bogdanov, Krishnamoorthy Dinesh, and Cheung Tsun Ming. On quantum versus classical query complexity. *Electron. Colloquium Comput. Complex.*, TR21-115, 2021. arXiv:TR21-115.

3 Scott Aaronson, Andris Ambainis, Janis Iraids, Martins Kokainis, and Juris Smotrovs. Polynomials, quantum query complexity, and grothendieck's inequality. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 25:1–25:19, 2016. doi:10.4230/LIPIcs.CCC.2016.25.

4 Nikhil Bansal and Makrand Sinha. k-forrelation optimally separates quantum and classical query complexity. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1303–1316, 2021. doi:10.1145/3406325.3451040.

5 Sergey Bravyi, David Gosset, Daniel Grier, and Luke Schaeffer. Classical algorithms for Forrelation, 2021. arXiv:2102.06963.

6 Harry Buhrman, Lance Fortnow, Ilan Newman, and Hein Röhrig. Quantum property testing. *SIAM J. Comput.*, 37(5):1387–1400, 2008. doi:10.1137/S0097539704442416.

7 Luc Devroye, Abbas Mehrabian, and Tommy Reddad. The total variation distance between high-dimensional gaussians, 2020. arXiv:1810.08693.

8 Joseph Jaeger, Fang Song, and Stefano Tessaro. Quantum key-length extension. In *Theory of Cryptography: 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8–11, 2021, Proceedings, Part I*, pages 209–239, Berlin, Heidelberg, 2021. Springer-Verlag. doi:10.1007/978-3-030-90459-3_8.

9 Gilles Pisier. Grothendieck's theorem, past and present. *Bulletin of the American Mathematical Society*, 49(2):237–323, May 2012. doi:10.1090/s0273-0979-2011-01348-9.

10 Mark Rudelson and Roman Vershynin. Sampling from large matrices: an approach through geometric functional analysis, 2006. arXiv:math/0503442.

11 Alexander A. Sherstov, Andrey A. Storozhenko, and Pei Wu. An optimal separation of randomized and quantum query complexity. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1289–1302, 2021. doi:10.1145/3406325.3451019.

12 Joel A. Tropp. Column subset selection, matrix factorization, and eigenvalue optimization, 2008. arXiv:0806.4404.

13 Joel A. Tropp. On the linear independence of spikes and sines. *Journal of Fourier Analysis and Applications*, 14(5-6):838–858, September 2008. doi:10.1007/s00041-008-9042-0.

14 Henry Yuen. A quantum lower bound for distinguishing random functions from random permutations. *Quantum Info. Comput.*, 14(13–14):1089–1097, October 2014.

15 Mark Zhandry. How to construct quantum random functions. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 679–687, 2012. doi:10.1109/FOCS.2012.37.

On the Power of Regular and Permutation Branching Programs

Chin Ho Lee ✉
Harvard University, Cambridge, MA, USA

Edward Pyne ✉
MIT, Cambridge, MA, USA

Salil Vadhan ✉
Harvard University, Cambridge, MA, USA

Abstract

We give new upper and lower bounds on the power of several restricted classes of arbitrary-order read-once branching programs (ROBPs) and standard-order ROBPs (SOBPs) that have received significant attention in the literature on pseudorandomness for space-bounded computation.

- Regular SOBPs of length n and width $\lfloor w(n+1)/2 \rfloor$ can *exactly* simulate general SOBPs of length n and width w, and moreover an $n/2 - o(n)$ blow-up in width is necessary for such a simulation. Our result extends and simplifies prior average-case simulations (Reingold, Trevisan, and Vadhan (STOC 2006), Bogdanov, Hoza, Prakriya, and Pyne (CCC 2022)), in particular implying that *weighted* pseudorandom generators (Braverman, Cohen, and Garg (SICOMP 2020)) for regular SOBPs of width poly(n) or larger automatically extend to general SOBPs. Furthermore, our simulation also extends to general (even read-many) oblivious branching programs.
- There exist natural functions computable by regular SOBPs of constant width that are average-case hard for permutation SOBPs of exponential width. Indeed, we show that Inner-Product mod 2 is average-case hard for arbitrary-order permutation ROBPs of exponential width.
- There exist functions computable by constant-width arbitrary-order permutation ROBPs that are worst-case hard for exponential-width SOBPs.
- Read-twice permutation branching programs of subexponential width can simulate polynomial-width arbitrary-order ROBPs.

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

Keywords and phrases Pseudorandomness, Branching Programs

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.44

Category RANDOM

Funding *Chin Ho Lee*: Supported by Madhu Sudan's and Salil Vadhan's Simons Investigator Awards.
Edward Pyne: Supported by an Akamai Presidental Fellowship.
Salil Vadhan: Supported by a Simons Investigator Award.

Acknowledgements Chin Ho Lee thanks Sepehr Assadi for answering his questions on [1]. We thank the anonymous reviewers for helpful comments.

1 Introduction

Read-once branching programs (ROBPs) have been extensively studied over the past four decades, motivated by the fact that these programs capture how small-space machines use random coins, and hence optimal and explicit pseudorandom generators for them would imply $\mathbf{BPL} = \mathbf{L}$, showing that every randomized logspace algorithm can be simulated deterministically with only a constant factor blow-up in space. Thus, there has been several decades of research on constructing pseudorandom generators for different variants of ROBPs.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 44; pp. 44:1–44:22
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In this paper, we study how those variants compare to each other in computational power, through new simulations and separations. To describe our results, we first define the models we are studying, starting from the most general model of read-many branching programs.

▶ **Definition 1.** *An **(oblivious) branching program (BP)** B of **length** m and **width** w computes a function $B\colon \{0,1\}^n \to \{0,1\}$. On an input $x \in \{0,1\}^n$, the branching program computes as follows. It has $m+1$ layers $V_0, \dots, V_m$, each with vertices labeled $\{1, \dots, w\}$. It starts at a fixed start state $v_{st} \in V_0$. Then for each step $t = 1, \dots, m$, it reads the next symbol $x_{i(t)}$ for some $i(t) \in [n]$, and updates its state according to a transition function $B_t\colon V_{t-1} \times \{0,1\} \to V_t$ by taking $v_t = B_t[v_{t-1}, x_{i(t)}]$. For $v \in V_s$ and $u \in V_t$ for $t > s$, we write $B[v, y] = u$ if the program transitions to state u starting from state v upon reading $y = (x_{i(s+1)}, \dots, x_{i(t)})$. Moreover, there is a set of accept states $V_{acc} \subseteq V_m$. For $x \in \{0,1\}^n$, we define $B(x) = 1$ if and only if $B[v_{st}, x] \in V_{acc}$. That is, B accepts the inputs x that lead it from the start state $v_{st} \in V_0$ in the first layer to an accept state in the last layer $v_{acc} \in V_{acc} \subseteq V_m$. We write $B(v, x) = 1$ if the program transitions to an accept state from a state v on input x. We call the function $i : [m] \to [n]$ the* read order *of B.*

▶ **Definition 2.** *A **read-k branching program** is a BP where the read order i satisfies $|i^{-1}(j)| \leq k$ for every $j \in [n]$. For $k = 1$ we denote this a **read-once branching program (ROBP)**.*

▶ **Definition 3.** *A **standard-order ROBP (SOBP)** is an ROBP whose read order is the identity function (i.e. $i(t) = t$ for every $t \in [n]$).*

Note that we have the inclusions

SOBPs ⊆ ROBPs ⊆ BPs.

To emphasize the distinction between the standard-order model and general ROBPs (which have $i(t) = \pi(t)$ for some permutation π), we denote the latter as **arbitrary-order ROBPs.**

▶ Remark 4. Our choice of notation follows the recent surveys of Hatami and Hoza [26] and Hoza [28]. There have been several (inconsistent) choices of notation in prior papers. In particular, prior works have referred to standard order ROBPs as simply ROBPs, or "ordered BPs". Other works have referred to ROBPs as "unordered ROBPs".

In 1990, Nisan [38] constructed an explicit pseudorandom generator (PRG) for SOBPs with seed length $O(\log n \cdot \log(nw/\varepsilon))$ (c.f. the optimal $O(\log(nw/\varepsilon))$ achieved by the probabilistic method). Despite extensive effort, this result has not been improved when the width of the programs w is at least 4 and at most $2^{n^{o(1)}}$. Motivated by this longstanding challenge, researchers have extensively studied restricted cases of the model, known as *regular* and *permutation* SOBPs:

▶ **Definition 5.** *A **regular BP** is a branching program where for all $t \in [m]$ and $i \in [w]$, there are exactly two distinct pairs $(j_1, b_1), (j_2, b_2)$ such that $B_t[j_1, b_1] = B_t[j_2, b_2] = i$. Equivalently, the graph of transitions from V_{t-1} to V_t is 2-in-regular.*

▶ **Definition 6.** *A **permutation BP** is an branching program where for all $t \in [m]$ and $\sigma \in \{0,1\}$, $B_t[\cdot, \sigma]$ is a permutation on $[w]$.*

Note that we have the inclusions

permutation BPs ⊆ regular BPs ⊆ BPs

and the same inclusions hold when restricting BPs to ROBPs or SOBPs.

There has been extensive prior work studying pseudorandomness for regular [31, 8, 16, 5, 34, 12] and permutation [32, 44, 41, 10, 29, 39, 22] SOBPs and ROBPs over roughly the last decade.

For regular SOBPs, the PRG of Braverman, Rao, Raz, and Yehudayoff [8] improves on Nisan's (which has seed length $O(\log^2 n)$ even for $w = 4$ and $\varepsilon = 1/3$) in the regime where *both* w and $1/\varepsilon$ are subpolynomial, i.e. $n^{o(1)}$. The later work [5] obtained better seed length than $O(\log^2 n)$ when *either* w or $1/\varepsilon$ was $n^{o(1)}$ (whereas Braverman et al. required both parameters to be small relative to n), at the cost of obtaining only a hitting set generator (HSG), a weaker object than pseudorandom generators that is sufficient for most derandomization tasks. For permutation SOBPs, Pyne and Vadhan [39] achieved seed length $\tilde{O}(\log^{3/2} n)$ for an object known as a *weighted PRG*,[1] in the $w = n$ regime motivated by derandomizing logspace.

Despite this extensive prior work, and the status quo where the known pseudorandom objects for regular and permutation SOBPs are better than those known for generic SOBPs in many regimes, there was relatively little work investigating the relative power of these models. As an example, it is well known that SOBPs can be simulated by a one-way two-party communication protocol, and therefore any program of width less than $2^{\Omega(n)}$ cannot compute the Inner-Product function $\mathsf{IP}_{2n}(x) := \sum_{i=1}^{n} x_i x_{n+i} \pmod 2$ on average. However, this result does not say anything about the relative power of general SOBPs versus regular and permutation SOBPs.

1.1 Our Results

We begin a systematic study of the relative power of SOBPs, regular SOBPs, and permutation SOBPs. We first survey the landscape of known results before stating our results.

1.1.1 General vs. Regular Programs

Perhaps the best known upper bound in this regard is the work of Reingold, Trevisan, and Vadhan [42] and its recent extension of Bogdanov, Hoza, Pyne, and Prakriya [5]. They showed that regular SOBPs of width poly(nw) and length $\tilde{O}(n)$ can *approximately* simulate general SOBPs of width w and length n. This implies a "transfer result": in the width-poly(n) regime, optimal PRGs or HSGs for regular ROBPs imply the equivalent objects for general ROBPs, and hence for logspace computation. However, these results have a few limitations. First, the simulation was average-case, and due to this did not imply a transfer result for weighted PRGs, a pseudorandom object that has seen extensive recent interest [7, 11, 14, 39, 27]. Moreover, both proofs are relatively involved.

We show that this upper bound can be improved and substantially simplified, and in fact, general and regular programs of the same length m, regardless of being read-once or not, are equivalent up to a factor of m in the width.

▶ **Theorem 7** (Informal statement of Theorem 18). *Let B be an oblivious branching program of length m and width $w \geq 4$. There exists a regular oblivious branching program R of length m and width $mw/2$ such that $R(x) = B(x)$ for all x. Moreover, R has the same read order as B.*

[1] A weighted PRG is a tuple of functions $(G, \rho)\colon \{0,1\}^s \to \{0,1\}^n \times \mathbb{R}$, where the weighted expectation $\mathbb{E}_x[\rho(x) \cdot B(G(x))]$ is within ε of $\mathbb{E}[B(U_n)]$ for all B in the class.

As a consequence, weighted PRGs for regular SOBPs with seed length matching those known for permutation SOBPs [39] would imply an improved derandomization of logspace:[2]

▶ **Corollary 8.** *Suppose there is an explicit weighted PRG for regular SOBPs of length n and width w with seed length $\tilde{O}(\log n \cdot (\log n + \sqrt{\log(w/\varepsilon)}) + \log(w/\varepsilon))$. Then $\mathbf{BPL} \subset \mathbf{L}^{4/3+o(1)}$.*

This follows as a corollary of Theorem 7 and the argument of Chattopadhyay and Liao [11] that the Saks–Zhou algorithm [43] can be instantiated with a weighted PRG.

As mentioned above, Theorem 7 holds even for *non-read-once* branching programs (as defined in Definition 1), in contrast to the prior results of [42, 5]. As a corollary, we derive that $\mathbf{L}$ can be computed by polynomial width regular branching programs:

▶ **Corollary 9.** *Every language in $\mathbf{L}$ can be decided by a (read-many) regular branching program of length and width* $\mathrm{poly}(n)$*, on inputs of size n.*

In terms of separation results, some simple observations were known. The AND function, which can be shown to require width n for permutation (in fact, regular) BPs, has a trivial general BP of width 2. (See Observation 19 for a proof.) We extend this separation to larger widths. This complements our simulation result (Theorem 7) by showing that in the case of ROBPs, the loss of a factor of $n/2$ is tight up to an *additive* term of $(w \log w)/2$.

▶ **Proposition 10.** *For every $w = 2^t, n \in \mathbb{N}$, there is a function $f : \{0,1\}^n \to \{0,1\}$ computable by an general SOBP of width w such that every regular SOBP computing f has width at least $\frac{nw}{2} - w \log w$.*

It is known that general SOBPs of constant width cannot be approximated by regular SOBPs of some $\mathrm{poly}(n)$ width in the “sandwiching notion” [3]. This can be derived by combining the results of [9, 8]. Brody and Verbin [9] showed that there is an instantiation of the Impagliazzo–Nisan–Wigderson PRG [31] that does *not* fool general SOBPs of width 3, and yet Braverman et al. [8] shows that this same PRG fools regular SOBPs of width n^c for some $c > 0$.

1.1.2 Regular vs. Permutation Programs

For the relationship between permutation and regular SOBPs, the situation was even less clear. As discussed in the previous section, despite extensive work on pseudorandomness for permutation and regular SOBPs, prior work has not proven separations between the two models. In fact, as far as we know, prior work did not exhibit *any* function computable by a regular program that was not computable by a permutation program of equal width.

We develop new lower bounds that separate these models to a near-maximal extent.

▶ **Theorem 11.** *There is $c > 0$ and $w_0 \in \mathbb{N}$ such that for every $\varepsilon > 0$ and n the following holds. There exists a function $f\colon \{0,1\}^n \to \{0,1\}$ computable by a regular SOBP of width w_0 such that no permutation SOBP of width $2^{cn/\log(1/\varepsilon)}$ agrees with f on a $1/2 + \varepsilon$ fraction of the inputs. In particular, no permutation SOBP of width $2^{c\sqrt{n}}$ agrees with f on a $1/2 + 2^{-\sqrt{n}}$ fraction of inputs.*

[2] A preprint circulated by the second author claimed this as a consequence of [5]. However, it does not follow from the argument in that work.

The hard function in Theorem 11 is the Inner-Product function with a specific variable-ordering. Our techniques for proving Theorem 11 are information-theoretic, and rely on showing that the entropy of the state over the $n+1$ layers of a permutation ROBP must be non-decreasing.

Our next result shows that the Inner-Product function is in fact average-case hard for *arbitrary-order* permutation ROBPs of exponential width.

▶ **Theorem 12.** *Every arbitrary-order permutation ROBP B that computes $\mathsf{IP}^{\oplus n}(x) := \sum_{i=1}^{n} x_{2i-1}x_{2i} \pmod 2$ on more than a $3/4+\varepsilon$ fraction of inputs has width at least $2^{4\varepsilon^2 n}$. Moreover, $\mathsf{IP}^{\oplus n}$ can be computed by a regular SOBP of width 4.*

We conjecture that Theorems 11 and 12 can be strengthened to give optimal average-case hardness, namely $1/2+2^{-n}$, but we have not been able to prove such a result.

▶ **Conjecture 13.** *There exists a constant $c > 0$ such that the following holds. Every arbitrary-order permutation ROBP B that computes $\mathsf{IP}^{\oplus n}(x) := \sum_{i=1}^{n} x_{2i-1}x_{2i} \pmod 2$ on more than a $1/2+2^{-cn}$ fraction of inputs has width at least 2^{cn}.*

1.1.3 Standard-Order vs. Arbitrary-Order Programs

In the past decade, researchers have turned their attention of constructing PRGs from SOBPs to the more general model of arbitrary-order ROBPs, as a way to generate new ideas to improve the state-of-the-art PRGs for SOBPs, and to develop PRGs for several natural subclasses of circuits that are not captured by SOBPs, as circuit classes are closed under permutation of the input coordinates. This line of research has received extensive interests [30, 41, 45, 25, 36, 21, 19], and in particular has resulted near-optimal PRGs for several well-studied models of computation, including read-once formulas [6, 23, 13, 17, 19], constant-width arbitrary-order permutation ROBPs [41, 10, 34], and read-once $\mathbb{F}_2$-polynomials [35, 36, 33, 18].

While Theorem 12 shows that there are regular SOBPs which cannot be approximated by arbitrary-order permutation ROBPs of exponential width, we show that the opposite direction is also true, by giving a function that is computable by an arbitrary-order permutation ROBP of constant width that requires exponential width for (even general) SOBPs.

▶ **Proposition 14** (Informal statement of Proposition 35)**.** *For every n, there exists a function $f\colon \{0,1\}^n \to \{0,1\}$ such that f is computable by an arbitrary-order permutation ROBP of width 6, and every SOBP computing f has width at least $2^{n/2}$.*

This result uses a non-Abelian group product and an adversarial argument.

1.1.4 Read Once vs. Read Many

Given our exponential lower bounds (Theorems 11 and 12) for permutation ROBPs, it is natural to ask whether any of them extends to read-k programs.

We show that even in the read-2 setting, permutation branching programs already become substantially more powerful. Specifically, read-twice permutation branching programs of subexponential width can simulate arbitrary-order ROBPs of polynomial width:

▶ **Proposition 15.** *Let $f\colon \{0,1\}^n \to [w]$ be computable by an arbitrary-order ROBP B of width w. Then for every $k \in \mathbb{N}$, f is computable by a read-(2^k) permutation branching program B' of width $w^{(k+1)n^{1/k}}$.*

Our simulation in Proposition 15 follows directly from Bennett's work on reversible computation [4]. We complement Proposition 15 by showing that a subexponential blow-up in the width is necessary for read-twice programs: there is no *fixed* read order such that read-twice permutation BPs reading bits in that order can simulate even regular SOBPs of constant width.

▶ **Theorem 16.** *For every read-twice ordering $i : [2n] \to [n]$, there exists a function $g\colon \{0,1\}^n \to \{0,1\}$ computable by a regular ROBP of width $O(1)$, such that every read-twice permutation branching program P of width $2^{n^{1/8}}$ with read order i computes g correctly on at most $1/2 + 2^{-\Omega(n^{1/8})}$ fraction of inputs.*

1.1.5 Permutation vs. Monotone Programs

Several works [37, 19] have studied the model of monotone branching programs, which correspond to branching programs where the edges labeled 1 do not cross, and likewise for the edges labeled 0. They are considered to be the "extreme opposite" of permutation programs [19]. We provide evidence for this belief by showing that read-once DNFs, which are computable by constant-width monotone programs, are worst-case hard for permutation SOBPs of exponential width:

▶ **Proposition 17.** *Let $f(x_1, y_1, \ldots, x_n, y_n) = \bigvee_i (x_i \wedge y_i)$. Then every permutation SOBP computing f has width at least 2^n.*

2 Regular Branching Programs

We show that regular programs can exactly simulate general programs with a moderate blow-up in width. We emphasize that our simulation is not restricted to the read-once setting.

▶ **Theorem 18.** *Let $B\colon \{0,1\}^n \to \{0,1\}$ be a branching program of length m and width w. There is a regular branching program $R\colon \{0,1\}^n \to \{0,1\}$ of length m and width $w' := \max\{w, \frac{wm}{2} + w(1 - \frac{\log w}{2})\}$ such that $R(x) = B(x)$ for all $x \in \{0,1\}^n$. Moreover, R has the same variable read order as B. In particular, for $w \geq 4$, we have $w' \leq wm/2$.*

Proof. We prove by induction on length m. We show the stronger claim that R exactly computes the states of B, i.e. that there are maps $\phi_t\colon [w'] \to [w]$ such that $\phi_i(R[v_{st}, (x_{i(1)}, \ldots, x_{i(t)})]) = B[v_{st}, (x_{i(1)}, \ldots, x_{i(t)})]$ for every $x \in \{0,1\}^n$ and $t \in [m]$.

When $m \leq \log w$, we can simulate B trivially by storing the bits read in at most $2^m \leq w$ states. Now, suppose $m \geq \log w + 1$. For each state v in the $(m-1)$-th layer B_{m-1} of B, let $C_{m-1}(v) := \phi_{m-1}^{-1}(v)$. By the inductive assumption, we have $\sum_{v \in B_{m-1}} |C_{m-1}(v)| \leq w(m-1)/2 + w(1 - \log(w)/2)$.

Now, for each state u in the m-th layer B_m of B, create

$$\left\lceil \frac{1}{2} \sum_{(v,b): B[v,b]=u} |C_{m-1}(v)| \right\rceil$$

states, denoted $C_m(u)$, and define ϕ_m such that $\phi_m(C_m(u)) := u$.

Finally, for each $b \in \{0,1\}$ and $v \in B_{m-1}$ such that $B[v,b] = u$, we add a b-edge from every state in $C_{m-1}(v)$ to some state in $C_m(u)$. There are $s_u := \sum_{(v,b): B[v,b]=u} |C_{m-1}(v)|$ many such edges, and hence there are enough states in $C_m(u)$ to accommodate this (with each state having at most 2 edges). Now, summing over all $u \in B_m$, we have

$$\begin{aligned}|R_m| &= \sum_{u \in B_m} |C_m(u)| \\ &= \sum_{u \in B_m} \left[\frac{1}{2} \sum_{(v,b):B[v,b]=u} |C_{m-1}(v)| \right] \\ &\leq \sum_{u \in B_m} \left(\frac{1}{2} + \frac{1}{2} \cdot \sum_{(v,b):B[v,b]=u} |C_{m-1}(v)| \right) \\ &= \frac{|B_m|}{2} + \sum_{v \in V_{m-1}} |C_{m-1}(v)| \\ &\leq \frac{w(m-1)}{2} + w\Big(1 - \frac{\log w}{2}\Big) + \frac{w}{2} \\ &= \frac{wm}{2} + w\Big(1 - \frac{\log w}{2}\Big).\end{aligned}$$

Let $k \leq w$ be the number of u such that s_u is odd. Note that k must be even. For each such u there is a state in $C_m(u)$ such that it has in-degree one. To preserve regularity, we add $k/2 \leq w/2$ of dummy states in R_{m-1} that are not reachable from the start state and connect the k outgoing edges of these states to these u's. ◀

We now show that for general SOBPs, this loss of a factor of m is tight, and in fact the loss is even tight in the leading constant.

▶ **Proposition 10.** *For every $w = 2^t, n \in \mathbb{N}$, there is a function $f : \{0,1\}^n \to \{0,1\}$ computable by an general SOBP of width w such that every regular SOBP computing f has width at least $\frac{nw}{2} - w \log w$.*

We recall the well-known fact that AND_n can be computed by a constant-width SOBP, but requires width n for regular ROBPs. We provide a proof for completeness.

▶ **Observation 19.** *Given $n \in \mathbb{N}$, $\mathsf{AND} := \mathsf{AND}_n$ can be computed by a general SOBP of width 2. However, every regular SOBP R computing AND must have $i+1$ distinct states reachable from v_{st} in layer i.*

Proof. The fact that AND can be computed by a general SOBP of width 2 is direct. We show the lower bound by induction. It is clearly true for layer 0 as v_{st} can reach itself. Assuming it holds for layer i, we note that from correctness, $u := R[v_{st}, 1^{i+1}] \neq R[v_{st}, 1^i||0]$ and hence there are two distinct states reachable in layer $i+1$ from $R[v_{st}, 1^i]$. Let R_i be the reachable states in layer i that are not $R[v_{st}, 1^i]$. We have that there are at least $2|R_i|$ edges from R_i (and every endpoint of such an edge is reachable). Moreover, we claim that these edges cannot reach u. Otherwise there would be $\tau \neq 1^{i+1}$ such that $B[v_{st}, \tau||1^{n-i-1}] = B[v_{st}, 1^n]$ which contradicts R computing AND. Thus there are at least $|R_i| + 1$ vertices reachable in layer $i+1$ that are not u, so we conclude. ◀

We can then bootstrap this separation to work for larger widths. Essentially, we use a multiplexer to force the program to remember a large amount of information before computing AND.

▶ **Definition 20.** *Given $n, w = 2^t$, let $m = n - 2(t-1)$. Define $f \colon \{0,1\}^{t-1} \times \{0,1\}^m \times \{0,1\}^{t-1} \to \{0,1\}$ as $f(x,y,z) = \langle x, z \rangle \oplus \mathsf{AND}(y) = \sum_{i=1}^{t-1} x_i z_i + \mathsf{AND}(y) \pmod 2$.*

We first argue that f can be computed by a SOBP of width w.

▷ Claim 21. f can be computed by an SOBP of width w.

Proof. We define a program $B(x, y, z)$. In the first t layers, B stores the entire input. For each state in layer t, B uses 2 states to compute $\mathsf{AND}(y)$, and hence at layer $m+t-1$ the states are labeled $(x, \mathsf{AND}(y))$. Then the program reads in z and computes $\langle x, z\rangle$, such that the states in the final layer are labeled $(\langle x, z\rangle, \mathsf{AND}(y))$ and hence B can return the value of f. It is clear from this description that B has width $2 \cdot 2^{t-1} = w$. ◁

We then argue that no regular SOBP can do better than remembering the first $t-1$ bits, and moreover must compute AND using essentially disjoint states.

▷ Claim 22. For every regular SOBP B computing f, for every $x \neq x' \in \{0,1\}^{t-1}$ we have $B[v_{st}, x] \neq B[v_{st}, x']$. Furthermore, for every $k < m$ the states reachable in layer $t-1+k$ from $B[v_{st}, x]$ must be disjoint from those reachable from $B[v_{st}, x']$.

Proof. First assume for contradiction there are $x, x' \in \{0,1\}^{t-1}$ with $x' \neq x$ where $B[v_{st}, x] = B[v_{st}, x']$. Let i be some index where $x'_i \neq x_i$ and hence $\langle x, e_i\rangle \neq \langle x', e_i\rangle$. Thus, $f(x, 0^m, e_i) \neq f(x', 0^m, e_i)$, but

$$B\big[v_{st}, x||0^m||e_i\big] = B\big[v_{st}, x'||0^m||e_i\big]$$

which is a contradiction. For the second claim, assume for contradiction there are $\tau, \tau' \in \{0,1\}^k$ (where we do not require $\tau \neq \tau'$) such that $B[v_{st}, x||\tau] = B[v_{\mathrm{acc}}, x'||\tau']$. But then $f(x, \tau||0^{m-k}, e_i) \neq f(x', \tau'||0^{m-k}, e_i)$ from before, but

$$B\big[v_{st}, x||\tau||0^{m-k}||e_i\big] = B\big[v_{st}, x'||\tau'||0^{m-k}||e_i\big]$$

which is a contradiction. ◁

We can then prove the result.

Proof of Proposition 10. Let f be the function in Definition 20 with $n, w = 2^t$. By Claim 21, f can be computed by a general SOBP of width w.

Now let R be an arbitrary regular SOBP computing f. By Claim 22, we must have $R[v_{st}, x] \neq R[v_{st}, x']$ for every $x \neq x' \in \{0,1\}^{t-1}$. Since R must correctly compute $\mathsf{AND}(y)$ (which can be shown by a similar extension argument), we obtain that for every x, there are at least m states reachable from $R[v_{st}, x]$ in layer $t+m-1$ for every x, and all of these states are disjoint by Claim 22. Thus, it follows from $m = n - 2(t-1)$ that R has width at least

$$2^{t-1} \cdot m = \frac{w}{2} \cdot m = \frac{nw}{2} + w(1 - \log w). \qquad ◀$$

3 Permutation Read-Once Branching Programs

In this section, we give explicit functions computable by small width regular SOBPs that are average-case hard against permutation SOBPs and ROBPs of large widths. We will be working with the Inner-Product functions with their input bits ordered in a certain manner.

▶ **Definition 23.** *For integers ℓ, m, define $\mathsf{IP}_{2\ell}^{\oplus m} \colon (\{0,1\}^{2\ell})^m \to \{0,1\}$ to be*

$$\mathsf{IP}_{2\ell}^{\oplus m}(x^1, y^1, \ldots, x^m, y^m) := \bigoplus_{i=1}^{m} \langle x^i, y^i\rangle,$$

where $\langle x_1, \ldots, x_\ell, y_1, \ldots, y_\ell\rangle := \bigoplus_{j=1}^{\ell} x_j y_j$. We omit the subscript 2ℓ when $\ell = 1$.

We first show that $\mathsf{IP}_{2\ell}^{\oplus m}$ can be computed by a regular SOBP of width $2^{2\ell+1}$ via a simple argument. This follows from the fact that regular SOBPs can compute the XOR of an arbitrary function on 2ℓ bits using $2\ell+1$ bits, because we can store all the 2ℓ bits and maintaining the prefix-XOR with 1 extra bit using a regular program. The program we construct is essentially the one used in simulating high-degree regular programs by binary regular programs in [5]:

▶ **Lemma 24.** *Let $f\colon \{0,1\}^k \to \{0,1\}$ be an arbitrary function. Then $g\colon (\{0,1\}^k)^n \to \{0,1\}$ defined as*

$$g(x^1,\dots,x^n) := \bigoplus_{i\in[n]} f(x^i),$$

where $x^i \in \{0,1\}^k$ for each $i \in [n]$, can be computed by a regular SOBP of width 2^{k+1}.

Proof. Let B be a program where each state has label $(s,b) \in \{0,1\}^k \times \{0,1\}$. On reading x^i_j where $j \in [k]$, the program updates as

$$(s,b) \to \begin{cases} (s',b) & \text{if } 1 \le j \le k-1 \\ (s', b \oplus f(s)) & \text{if } j = k. \end{cases}$$

where s' is s with the j-th coordinate replaced with the bit x^i_j. The width of this program is $2^k \cdot 2$, and the fact that it computes f is direct. Finally, the program is regular as every s' has a single $b \in \{0,1\}$ and two strings $s \in \{0,1\}^k$ for which the replacement of the j-th coordinate of s with b produces s'. ◀

We recall our average-case lower bound against permutation ROBPs computing inner product.

▶ **Theorem 12.** *Every arbitrary-order permutation ROBP B that computes $\mathsf{IP}^{\oplus n}(x) := \sum_{i=1}^n x_{2i-1}x_{2i} \pmod 2$ on more than a $3/4+\varepsilon$ fraction of inputs has width at least $2^{4\varepsilon^2 n}$. Moreover, $\mathsf{IP}^{\oplus n}$ can be computed by a regular SOBP of width 4.*

For permutation SOBPs, we can strengthen this to a strong average case lower bound:

▶ **Theorem 11.** *There is $c > 0$ and $w_0 \in \mathbb{N}$ such that for every $\varepsilon > 0$ and n the following holds. There exists a function $f\colon \{0,1\}^n \to \{0,1\}$ computable by a regular SOBP of width w_0 such that no permutation SOBP of width $2^{cn/\log(1/\varepsilon)}$ agrees with f on a $1/2+\varepsilon$ fraction of the inputs. In particular, no permutation SOBP of width $2^{c\sqrt{n}}$ agrees with f on a $1/2+2^{-\sqrt{n}}$ fraction of inputs.*

Our argument relies on the fact that the entropy of the states in each layer of a permutation ROBP is non-decreasing. Before stating this property formally, we first recall some basic facts in information theory. We use capital letters to denote random variables, and lower case to denote specific assignments.

▶ **Definition 25.** *Given a joint random variable (X,Y), let*

- *$H(X) := \sum_{x\in Supp(X)} p(x)\log_2(1/p(x))$ be the* (binary) entropy *of X;*
- *$H(X \mid Y) := H(X,Y) - H(Y)$ be the* conditional entropy *of X given Y, and*
- *$I(X;Y) := H(X) - H(X \mid Y)$ be the* mutual information *of X and Y.*

Moreover, given $p \in [0,1]$, let $H(p) := p\log_2(1/p) + (1-p)\log_2(1/(1-p))$ be the entropy of a p-biased Bernoulli random variable.

We define the distributions over states of a program.

▶ **Definition 26.** *Given a ROBP B of length n, for $i \in \{0, \dots, n\}$, let S_i be the distribution over the reachable states after reading X_i of a uniformly random $X \sim \{0,1\}^n$.*

We then note the most important property of permutation SOBPs from this perspective: given the state reached after reading x_i and the value of x_i, one can exactly recover the state after reading x_{i-1}. More generally, we have the following proposition.

▶ **Proposition 27.** *Let $(X_1, \dots, X_n) \leftarrow U_n$. For every SOBP B and $i < j$, we have $H(S_j \mid S_i, X_{i+1}, \dots, X_j) = 0$. Moreover, if B is a permutation SOBP then $H(S_i \mid S_j, X_{i+1}, \dots, X_j) = 0$.*

Proof. The first claim is immediate from the fact that knowing the current state S_i and next $j - i$ bits $X_{i+1}, \dots, X_j$ to be read determines the state S_j. The second claim is likewise immediate, as for a permutation SOBP there is exactly one state S_i in layer i that reaches the state S_j in layer j after reading $X_{i+1}, \dots, X_j$. ◀

We use this property to show that for permutation SOBPs, the entropy of the state at layer i must increase by at least the mutual information between the state and the i-th input bit, and use this to conclude a lower bound on the width.

▶ **Lemma 28.** *Let $(X_1, \dots, X_n) \sim \{0,1\}^n$ be a uniform n-bit input. For a permutation SOBP B of length n and width w, let $i_1 < \dots < i_m$ be some m layers in B, and $X^{i_j} := (X_{i_{j-1}+1}, \dots, X_{i_j})$, where $i_0 := 0$. Then*

$$\log w \geq \sum_{j=1}^{m} I(X^{i_j}; S_{i_j}).$$

Proof. We first prove that for every $j \in [m]$,

$$H(S_{i_j}) = I(X^{i_j}; S_{i_j}) + H(S_{i_j - 1}). \tag{1}$$

Given this, the lemma follows from $H(S_0) = 0$ and

$$\log w = \log \operatorname{supp}(S_{i_m}) \geq H(S_{i_m}) = \sum_{j=1}^{m} H(S_{i_j}) - H(S_{i_{j-1}}) = \sum_{j=1}^{m} I(X^{i_j}; S_{i_j}).$$

We now prove Equation (1). By Proposition 27 we have

$$H(S_{i_j} \mid S_{i_{j-1}}, X^{i_j}) = 0 = H(S_{i_{j-1}} \mid S_{i_{j-1}}, X^{i_j}).$$

Applying the chain rule to both sides we obtain

$$\begin{aligned} H(S_{i_{j-1}}, X^{i_j}) &= H(S_{i_j}, S_{i_{j-1}}, X^{i_j}) - H(S_{i_j} \mid S_{i_{j-1}}, X^{i_j}) \\ &= H(S_{i_{j-1}}, S_{i_j}, X^{i_j}) - H(S_{i_{j-1}} \mid S_{i_j}, X^{i_j}) \\ &= H(S_{i_j}, X^{i_j}). \end{aligned}$$

Another chain rule to both sides gives

$$H(X^{i_j} \mid S_{i_j}) + H(S_{i_j}) = H(X^{i_j} \mid S_{i_{j-1}}) + H(S_{i_{j-1}}).$$

Thus,

$$\begin{aligned}H(S_{i_j}) &= H(X^{i_j} \mid S_{i_{j-1}}) + H(S_{i_{j-1}}) - H(X^{j_i} \mid S_{j_i})\\ &= H(S_{i_{j-1}}) + \big(H(X_{i_j}) - H(X^{i_j} \mid S_{i_j})\big) - \big(H(X^{i_j}) - H(X^{i_j} \mid S_{i_{j-1}})\big)\\ &= H(S_{i_{j-1}}) + I(S_{i_j}; X^{i_j}) - I(S_{i_{j-1}}; X^{i_j})\\ &= H(S_{i_{j-1}}) + I(S_{i_j}; X^{i_j}),\end{aligned}$$

where the final step follows from the fact that X_{i_j} is independent of all prior bits, and thus the state at layer i_j. ◀

We are now prepared to prove the lower bounds. In both cases, we require Fano's inequality. For the inner product bound, we use a simple formulation due to Regev [40]:

▶ **Lemma 29** (Claim 2.1 [40]). *Let X be uniformly distributed over $\{0,1\}$. Let S be a random variable such that there exists f such that $\Pr_{X,S}[f(S) \neq X] =: p \leq 1/2$. Then $I(X;S) \geq 1 - H(p)$.*

3.1 Mild Average-Case Lower Bounds for Arbitrary-Order Programs

We now prove the lower bound in Theorem 12: To illustrate the idea, consider a permutation SOBP B that reads its input x in the order of $x_1, \ldots, x_{2n}$. We will show that when X is uniform over $\{0,1\}^{2n}$, for every $i \in [n]$, given the state S_{2i-1} reached by B after reading $X_1, \ldots, X_{2i-1}$, we can use B to predict the value of X_{2i-1} better than random guessing, showing that there is non-trivial amount of mutual information between S_{2i-1} and X_{2i-1}. To see this, note that for every $x \in \{0,1\}^n$,

$$x_{2i-1} = \mathsf{IP}^{\oplus 2n}(x_1, \ldots, x_{2i-1}, 0, x_{2i+1}, \ldots, x_{2n}) \oplus \mathsf{IP}^{\oplus 2n}(x_1, \ldots, x_{2i-1}, 1, x_{2i+1}, \ldots, x_{2n}).$$

Moreover, given a state S_{i+1}, we can simulate the remaining program on the two inputs $(x_{2i} = 0, X_{2i+1}, \ldots, X_{2n})$ and $(x_{2i} = 1, X_{2i+1}, \ldots, X_{2n})$, for a uniform $X_{2i+1}, \ldots, X_{2n}$, to compute the right hand side, which by a union bound, is correct and thus equals X_{2i-1} with probability at least $1/2 + 2\varepsilon$.

Proof of Theorem 12. Let $X = (X_1, \ldots, X_{2n}) := (X_1, Y_1, \ldots, X_n, Y_n)$ be a uniform random input. Fix an arbitrary-order permutation ROBP B that reads x in the order of $x_{\sigma(1)}, \ldots, x_{\sigma(2n)}$ for some permutation σ. By assumption we have $\Pr[B(X) \neq f(X)] \leq 1/4 - \varepsilon$.

For every $i \in [n]$, let $r_i := \min\{\sigma^{-1}(2i-1), \sigma^{-1}(2i)\}$ be the layer reached by B after reading the first bit of x_{2i-1} and x_{2i}. Let $L \subset [2n]$ be the indices of the variables of X read up to this point (i.e., $L = \{\sigma(1), \ldots, \sigma(r_i)\}$ and let $R := [2n] \setminus L = \{\sigma(r_i + 1), \ldots, \sigma(2n)\}$.

We now show that $I(X_{r_i}; S_{r_i}) \geq 1 - H(1/2 - 2\varepsilon)$, which suffices to prove the result by Lemma 28. To do so, given the state s_{r_i} in layer r_i, we let our guess of x_{r_i} be

$$g(s_{r_i}) = B\big[v, y^0\big] \oplus B\big[v, y^1\big],$$

where $y \leftarrow U_R$ is a random suffix and y^b is y with its $(t_i := \max\{\sigma^{-1}(2i-1), \sigma^{-1}(2i)\})$-th bit replaced with $b \in \{0,1\}$. Observe that $B[S_{r_i}, Y^{b*}]$ is identical to $f(X)$ conditioned on $X_{t_i} = b$. We have

$$\begin{aligned}\Pr_X\big[g(S_{r_i}) \neq X_{r_i}\big] &= \Pr_X\Big[B\big[S_{r_i}, Y^{0*}\big] \oplus B\big[S_{r_i}, Y^{1*}\big] \neq X_{r_i}\Big] \\ &\leq \Pr_X\Big[B\big[S_{r_i}, Y^{0*}\big] \neq f(X)\Big] + \Pr_X\Big[B\big[S_{r_i}, Y^{1*}\big] \neq f(X)\Big] \\ &\leq \Pr_X\big[B(X) \neq f(X) \mid X_{t_i} = 0\big] + \Pr\big[B(X) \neq f(X) \mid X_{t_i} = 1\big] \\ &= 2\Pr\big[B(X) \neq f(X)\big] \\ &\leq 1/2 - 2\varepsilon.\end{aligned}$$

By Lemma 29 we have $I(X_{r_i}; S_{r_i}) \geq 1 - H(1/2 - 2\varepsilon) \geq 4\varepsilon^2$. Therefore by Lemma 28 we have

$$\log w \geq \sum_{i=1}^{n} I(X_{r_i}; S_{r_i}) \geq 4\varepsilon^2 n,$$

and hence $w \geq 2^{4\varepsilon^2 n}$. The "moreover" claim follows from Lemma 24. ◀

3.2 Moderate Average-Case Lower Bounds

Before proving our strong average-case lower bound (Theorem 11), we have to extend Theorem 12 to improve the correlation bound from $3/4 + \varepsilon$ to $1/2 + \varepsilon_0$ for an arbitrary constant ε_0.

▶ **Theorem 30.** *Let $\ell \geq 8\log(1/\varepsilon)$. If B is a permutation SOBP of width w and length $2\ell m$ that agrees with $\mathsf{IP}_{2\ell}^{\oplus m}$ on a $1/2 + \varepsilon$ fraction of inputs, then $w \geq 2^{\varepsilon m \ell/4}$.*

The high-level idea is to combine the idea in the previous subsection with Goldreich–Levin list-decoding. Instead of predicting 1 bit, we will divide the input into blocks and show that we can predict the whole block of X^i given the state S_i reached by B upon reading X^i. To do so, we first show that with probability at least $\varepsilon/2$ over all-but-the-Y^i-part of the input, we have the following property: Given the state S_i, we can predict $\langle X^i, Y^i\rangle$ for a random sample $Y^i \sim \{0,1\}^\ell$ correctly with probability $1/2+\varepsilon$. Then by the Goldreich–Levin theorem, we can use this predictor to narrow X^i down to a list of size $1/\varepsilon^2$, showing that there is a non-trivial amount of mutual information between X^i and S_i.

Our argument only requires the following bound on the "list size," which follows from Parseval's identity.

▷ Claim 31. For every Boolean function $f\colon \{0,1\}^\ell \to \{0,1\}$, there are at most $1/\varepsilon^2$ many $a \in \{0,1\}^\ell$ such that $\Pr[f(U) = \langle a, x\rangle] \geq 1/2 + \varepsilon/2$.

Proof. This is equivalent to $\widehat{f}(a) \geq \varepsilon$, where $\widehat{f}(a) := \mathbb{E}_x[f(x)(-1)^{\langle a,x\rangle}]$. Let L be the number of such a's. Then by Parseval's identity, we have $L\varepsilon^2 \leq \sum_{a\in\{0,1\}^\ell} \widehat{f}(a)^2 = \mathbb{E}[f(x)^2] \leq 1$. Rearranging gives $L \leq 1/\varepsilon^2$. ◁

3.2.1 Proof of Theorem 30

Proof. Let $(X^1, \ldots, Y^m)$ be a uniformly random input of $\mathsf{IP}_{2\ell}^{\oplus m}$. Let B be a width-w permutation SOBP that agrees with $\mathsf{IP}_{2\ell}^{\oplus m}$ with probability $1/2 + \varepsilon$. Our goal is to show that $w \geq 2^{\varepsilon \ell m/4}$. For $i \in [m]$, let S_i denote the state B reaches after reading X^i. We will show that $I(S_i; X^i) \geq \varepsilon\ell/4$, from which the theorem follows from Lemma 28.

To proceed, fix an $i \in [m]$. Given an input $(x^1, y^1, \dots, x^m, y^m)$ of B, let $z \in (\{0,1\}^\ell)^{2m-1}$ denote all but the y^i-th block of y, that is, $z = (x^1, \dots, y^{i-1}, x^i, x^{i+1}, \dots, y^m)$. We will use the shorthand $B(z, y^i)$ to denote $B(x^1, y^1, \dots, x^m, y^m)$. Given $z \in (\{0,1\}^\ell)^{2m-1}$ and an auxiliary bit $a \in \{0,1\}$, consider the function $B_{z,a} \colon \{0,1\}^\ell \to \{0,1\}$ defined by

$$B_{z,a}(y^i) := B(z, y^i) \oplus \bigoplus_{j>i} \langle x^i, y^i \rangle \oplus a. \tag{2}$$

(One should think of a as a guess of the bit $\bigoplus_{j<i} \langle x^j, y^j \rangle$.) Let s_i be the state reached by B upon reading the prefix $(x^1, y^1 \dots, x^i) \in (\{0,1\}^\ell)^{2i-1}$. Observe that we can compute $B_{z,a}(y^i)$ by simulating B starting from state s_i on the remaining inputs $(y^i, x^{i+1}, \dots, y^m)$ and then XORing its output with a.

We claim that with probability at least $\varepsilon/2$ over $(Z, A) \sim (\{0,1\}^\ell)^{2m-1} \times \{0,1\}$, we have

$$\Pr_{Y^i \sim \{0,1\}^n} \Big[B_{Z,A}(Y^i) = \langle X^i, Y^i \rangle \Big] \geq 1/2 + \varepsilon/2. \tag{3}$$

To see this, note that A is a correct guess of the bit $\bigoplus_{j<i} \langle x^j, y^j \rangle$ with probability $1/2$, i.e. $\Pr_{A \sim \{0,1\}}[\bigoplus_{j<i} \langle x^j, y^j \rangle = A] = 1/2$. Conditioned on A being the correct guess, it follows by an averaging argument that with probability at least $\varepsilon/2$ over $Z \sim (\{0,1\}^\ell)^{2m-1}$ we have

$$\begin{aligned} \Pr_{Y^i \sim \{0,1\}^\ell} \Big[B_{Z,A}(Y^i) = \langle X^i, Y^i \rangle \Big] &= \Pr_{Y^i \sim \{0,1\}^\ell} \Big[B(Z, Y^i) = \langle X^i, Y^i \rangle + \bigoplus_{j>i} \langle X^j, Y^j \rangle + \bigoplus_{j<i} \langle X^j, Y^j \rangle \Big] \\ &= \Pr_{Y^i \sim \{0,1\}^\ell} \Big[B(Z, Y^i) = \mathsf{IP}^{\oplus k}(Z, Y^i) \Big] \geq 1/2 + \varepsilon/2. \end{aligned}$$

Let us call the pair (z, a) *good* if it satisfies Equation (3). Note that for a good (z, a), by Claim 31, there are at most $1/\varepsilon^2$ many choices of $r \in \{0,1\}^\ell$ such that

$$\Pr_{Y^i \sim \{0,1\}^\ell} \Big[B_{z,b}(Y^i) = \langle r, Y^i \rangle \Big] \geq 1/2 + \varepsilon/2,$$

and x^i is one of them, and thus we have the following claim.

▷ Claim 32. $H(X^i \mid S_i, (Z, A) \text{ is good}) \leq \log(1/\varepsilon^2)$.

We will use the following fact behind the proof of Fano's inequality.

▷ Claim 33 (Fano's inequality). Let X, Y, G be three random variables such that $H(G \mid X, Y) = 0$. Then

$$H(X \mid Y) = H(G \mid Y) + H(X \mid G, Y).$$

For a uniform $(X^1, \dots, Y^m) \sim (\{0,1\}^\ell)^{2m}$, let $G := G(Z, A)$ be the indicator random variable of whether (Z, A) is good. Let $Z_{>i}$ denote $(X^{i+1}, \dots, Y^m)$. Since X^i is independent of $Z_{>i}$ and A,

$$H(X^i \mid S_i) = H(X^i \mid S_i, Z_{>i}, A).$$

Now, given $S_i, Z_{>i}, A$, and X^i, we can compute $B_{Z,A}$ and determine if (Z, A) is good, and thus we have $H(G \mid S_i, X^i, Z_{>i}, A) = 0$. So by Claim 33,

$$H(X^i \mid S_i, Z_{>i}, A) = H(G \mid S_i, Z_{>i}, A) + H(X^i \mid G, S_i, Z_{>i}, A). \tag{4}$$

We can bound the first term $H(G \mid S_i, Z_{>i}, A)$ by $H(G)$. For the second term, we apply Claim 32 as follows:

$$\begin{aligned} H(X^i \mid S_i, Z_{>i}, A, G) &= \Pr[G] \cdot H(X^i \mid S_i, Z_{>i}, A, G = 1) + (1 - \Pr[G]) \cdot H(X^i \mid S_i, Z_{>i}, A, G = 0) \\ &\leq \Pr[G] \cdot \log(1/\varepsilon^2) + (1 - \Pr[G]) \cdot H(X^i). \end{aligned}$$

Applying both bounds to the right hand side of Equation (4) gives

$$H(X^i \mid S_i, Z_{>i}, A) \leq H(G) + \Pr[G] \cdot \log(1/\varepsilon^2) + (1 - \Pr[G]) \cdot H(X^i).$$

As $\Pr[G] \geq \varepsilon/2$, we have $H(G) \leq 2\Pr[G] \log(1/\Pr[G]) \leq 2\Pr[G] \log(2/\varepsilon)$. Therefore,

$$\begin{aligned} I(X^i; S_i) &= H(X^i) - H(X^i \mid S_i) \\ &= H(X^i) - H(X^i \mid S_i, Z_{>i}, A) \\ &\geq \Pr[G] \cdot \big(H(X^i) - \log(1/\varepsilon^2)\big) - H(G) \\ &\geq \Pr[G] \cdot \big(H(X^i) - 4\log(1/\varepsilon)\big) \\ &\geq (\varepsilon/2) \cdot \big(\ell - 4\log(1/\varepsilon)\big), \end{aligned}$$

which is at least $\varepsilon \cdot \ell/4$ for $\ell \geq 8\log(1/\varepsilon)$. It follows from Lemma 28 that

$$\log w \geq \sum_{i=1}^{m} I(S_i; X^i) \geq \varepsilon m \ell/4. \qquad \blacktriangleleft$$

3.3 Strong Average-Case Lower Bounds

We now prove Theorem 11. Assadi and N. [1] proved the following XOR Lemma for multi-pass streaming algorithms.[3] In the case of one pass this is essentially the same as SOBPs. Moreover, we observe that their argument also applies to permutation SOBPs.

▶ **Lemma 34** ([1]). *There exists an absolute constant $\varepsilon_0 > 0$ such that the following holds. Let $f\colon \{0,1\}^m \to \{0,1\}$ be any function and let $f^{\oplus \ell}$ be the XOR of ℓ copies of f on disjoint (sequential) blocks. Suppose $\Pr[P(U) = f(U)] \leq 1/2 + \varepsilon$ for some $\varepsilon \leq \varepsilon_0$ for every permutation SOBP P of width w. Then $\Pr[P(U) = f^{\oplus \ell}(U)] \leq 1/2 + \varepsilon^{\ell/7}$ for every permutation SOBP P of width w.*

Proof of Theorem 11. By Theorem 30, for every constant $\varepsilon_0 > 0$, there exists a constant ℓ such that the function $\mathsf{IP}_{2\ell}^{\oplus m}$ on $2\ell m$ bits is $(1/2 + \varepsilon_0)$-hard for permutation SOBPs of width $2^{\varepsilon_0 m \ell/8}$. By Lemma 34, the function $\mathsf{IP}_{2\ell}^{\oplus mk}$ on $2\ell mk$ bits is $(1/2 + \varepsilon_0^{k/7})$-hard for permutation SOBPs of width $2^{\varepsilon_0 m \ell/8}$. Choosing ε_0 to be a sufficiently small constant, $k = 7\log_{\varepsilon_0}(1/\varepsilon)$, and letting $n := 2\ell mk$ gives us a hard function on n bits that is $(1/2+\varepsilon)$-hard for permutation SOBPs of width $2^{cn/\log(1/\varepsilon)}$ for a universal constant c. ◀

3.4 Worst Case Lower Bounds Against Monotone Functions

Next, we show there are monotone functions (in fact, read-once DNFs) that are worst-case hard for permutation SOBPs of exponential width.

▶ **Proposition 17.** *Let $f(x_1, y_1, \ldots, x_n, y_n) = \bigvee_i (x_i \wedge y_i)$. Then every permutation SOBP computing f has width at least 2^n.*

[3] There is a mistake in the publicly available versions which has been corrected by the authors [2].

Proof. Let B be a permutation SOBP computing f. We will show that in layer $2i$ (the state after reading both variables in the ith term), there are 2^i states reachable by strings that have not yet satisfied a term. This holds vacuously for $i = 0$. Now suppose this holds for term i and let T_i be the set of such states, and for $b \in \{0, 1\}$ define

$$T_i[b] := \{v \in V_{2i+1} : \exists u \in T_i \text{ s.t. } B[u, b] = v\}.$$

We first observe that $|T_i[1]| = |T_i[0]| = |T_i| \geq 2^i$ and $T_i[1] \cap T_i[0] = \emptyset$. The first follows since B is a permutation SOBP (and hence all states in $T_i[1]$ can have a single in-1-edge and likewise for $T_i[0]$) and the second follows via an extension argument, since otherwise B fails to compute f. This implies $|T_i[10] \cup T_i[00]| \geq 2|T_i|$ again using that B is a permutation SOBP. Finally, we observe that $T_{i+1} \supseteq T_i[10] \cup T_i[00]$ and hence $|T_{i+1}| \geq 2^{i+1}$ as claimed, which completes the induction.

The "moreover" claim follows from inspection. ◀

3.5 Separating General SOBPs From Permutation ROBPs

We now give a function f that is computable by an arbitrary-order permutation ROBP of constant width but is hard for any SOBP of exponential width.

▶ **Proposition 35.** *Let D_3 be the Dihedral group of order 6 with identity element e and fix two reflections r, s such that $r^2 = s^2 = e$ and $rs \neq sr$. Let $S = \{r, s, sr\}$. Consider $f \colon \{0, 1\}^{2n} \to \{0, 1\}$ defined by*

$$f(x, y) := \mathbb{1}(r^{x_1} s^{y_1} \cdots r^{x_n} s^{y_n} \in S).$$

Then every general SOBP computing f has width at least 2^n. Moreover, f can be computed by a arbitrary-order permutation ROBP of width 6.

Proof. The "moreover" claim follows from the fact that any group product can be simulated by a permutation ROBP of width equal to the group's order. We now claim that for every $x \neq x' \in \{0, 1\}^n$, there is a $y \in \{0, 1\}^n$ such that $f(x, y) \neq f(x', y)$, and therefore any SOBP must use 2^n states to remember x after reading it.

First, consider the case where the Hamming weights of x and x' have different parities. Then by taking y to be the all-zero string 0^n and using $r^2 = e$, we have $f(x, y) = r^b$ and $f(x', y) = r^{1-b}$ for some $b \in \{0, 1\}$.

Now, suppose their parities are the same. Let $i \in [n]$ be the first position where x and x' differ, and without loss of generality assume $x_i = 1$ (and so $x'_i = 0$). Let $y = e_i$. Then $f(x, y) = rsr^b$ and $f(x', y) = sr^{1-b}$ for some $b \in \{0, 1\}$, but $s, sr \in S$ and $rsr, rs \notin S$. So in either case we have $f(x, y) \neq f(x', y)$. ◀

References

1 Sepehr Assadi and Vishvajeet N. Graph streaming lower bounds for parameter estimation and property testing via a streaming XOR lemma. In *STOC '21 – Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 612–625. ACM, New York, 2021. doi:10.1145/3406325.3451110.

2 Sepehr Assadi and Vishvajeet N. Personal communication, 2022.

3 Louay M. J. Bazzi. Polylogarithmic independence can fool DNF formulas. *SIAM J. Comput.*, 38(6):2220–2272, 2009. doi:10.1137/070691954.

4 C. H. Bennett. Logical reversibility of computation. *IBM J. Res. Develop.*, 17:525–532, 1973. doi:10.1147/rd.176.0525.

5 Andrej Bogdanov, William M. Hoza, Gautam Prakriya, and Edward Pyne. Hitting sets for regular branching programs. In Shachar Lovett, editor, *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, volume 234 of *LIPIcs*, pages 3:1–3:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CCC.2022.3.

6 Andrej Bogdanov, Periklis A. Papakonstantinou, and Andrew Wan. Pseudorandomness for read-once formulas. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science – FOCS 2011*, pages 240–246. IEEE Computer Soc., Los Alamitos, CA, 2011. doi:10.1109/FOCS.2011.57.

7 Mark Braverman, Gil Cohen, and Sumegha Garg. Pseudorandom pseudo-distributions with near-optimal error for read-once branching programs. *SIAM J. Comput.*, 49(5), 2020. doi:10.1137/18M1197734.

8 Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom generators for regular branching programs. *SIAM J. Comput.*, 43(3):973–986, 2014. doi:10.1137/120875673.

9 Joshua Brody and Elad Verbin. The coin problem, and pseudorandomness for branching programs. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science – FOCS 2010*, pages 30–39. IEEE Computer Soc., Los Alamitos, CA, 2010.

10 Eshan Chattopadhyay, Pooya Hatami, Kaave Hosseini, and Shachar Lovett. Pseudorandom generators from polarizing random walks. *Theory Comput.*, 15:Paper No. 10, 26, 2019. doi:10.4086/toc.2019.v015a010.

11 Eshan Chattopadhyay and Jyun-Jie Liao. Optimal error pseudodistributions for read-once branching programs. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 25:1–25:27. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CCC.2020.25.

12 Lijie Chen, Xin Lyu, Avishay Tal, and Hongxun Wu. New PRGs for Unbounded-Width/Adaptive-Order Read-Once Branching Programs. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, volume 261 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 39:1–39:20, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2023.39.

13 Sitan Chen, Thomas Steinke, and Salil P. Vadhan. Pseudorandomness for read-once, constant-depth circuits. *CoRR*, abs/1504.04675, 2015. arXiv:1504.04675.

14 Gil Cohen, Dean Doron, Oren Renard, Ori Sberlo, and Amnon Ta-Shma. Error Reduction for Weighted PRGs Against Read Once Branching Programs. In *Proceedings of the 36th Computational Complexity Conference (CCC)*, pages 22:1–22:17, 2021.

15 Matei David, Periklis A. Papakonstantinou, and Anastasios Sidiropoulos. How strong is nisan's pseudo-random generator? *Inf. Process. Lett.*, 111(16):804–808, 2011. doi:10.1016/j.ipl.2011.04.013.

16 Anindya De. Pseudorandomness for permutation and regular branching programs. In *26th Annual IEEE Conference on Computational Complexity*, pages 221–231. IEEE Computer Soc., Los Alamitos, CA, 2011.

17 Dean Doron, Pooya Hatami, and William M. Hoza. Near-optimal pseudorandom generators for constant-depth read-once formulas. In *34th Computational Complexity Conference*, volume 137 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 16, 34. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2019.

18 Dean Doron, Pooya Hatami, and William M. Hoza. Log-seed pseudorandom generators via iterated restrictions. In *35th computational complexity conference*, volume 169 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 6, 36, 2020.

19 Dean Doron, Raghu Meka, Omer Reingold, Avishay Tal, and Salil Vadhan. Pseudorandom Generators for Read-Once Monotone Branching Programs. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*, volume 207 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 58:1–58:21, 2021. doi:10.4230/LIPIcs.APPROX/RANDOM.2021.58.

20 P. Erdös and G. Szekeres. A combinatorial problem in geometry. *Compositio Math.*, 2:463–470, 1935. URL: http://www.numdam.org/item?id=CM_1935__2__463_0.

21 Michael A. Forbes and Zander Kelley. Pseudorandom generators for read-once branching programs, in any order. In *59th Annual IEEE Symposium on Foundations of Computer Science – FOCS 2018*, pages 946–955. IEEE Computer Soc., Los Alamitos, CA, 2018. doi:10.1109/FOCS.2018.00093.

22 Louis Golowich and Salil P. Vadhan. Pseudorandomness of expander random walks for symmetric functions and permutation branching programs. In Shachar Lovett, editor, *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, volume 234 of *LIPIcs*, pages 27:1–27:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CCC.2022.27.

23 Parikshit Gopalan, Raghu Meka, Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Better pseudorandom generators from milder pseudorandom restrictions. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 120–129. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.77.

24 Rohit Gurjar and Ben Lee Volk. Pseudorandom bits for oblivious branching programs. *ACM Trans. Comput. Theory*, 12(2):Art. 8, 12, 2020. doi:10.1145/3378663.

25 Elad Haramaty, Chin Ho Lee, and Emanuele Viola. Bounded independence plus noise fools products. *SIAM J. Comput.*, 47(2):493–523, 2018. doi:10.1137/17M1129088.

26 Pooya Hatami and William Hoza. Theory of unconditional pseudorandom generators. *Electron. Colloquium Comput. Complex.*, TR23-019, 2023. arXiv:TR23-019.

27 William M. Hoza. Better pseudodistributions and derandomization for space-bounded computation. In *Proceedings of the 25th International Conference on Randomization and Computation (RANDOM)*, pages 28:1–28:23, 2021.

28 William M. Hoza. Recent progress on derandomizing space-bounded computation. *Electron. Colloquium Comput. Complex.*, TR22-121, 2022. arXiv:TR22-121.

29 William M. Hoza, Edward Pyne, and Salil Vadhan. Pseudorandom generators for unbounded-width permutation branching programs. In *12th Innovations in Theoretical Computer Science Conference*, volume 185 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 7, 20, 2021.

30 Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from shrinkage. *J. ACM*, 66(2):Art. 11, 16, 2019. doi:10.1145/3230630.

31 Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 356–364, Montréal, Québec, Canada, 23–25 May 1994.

32 Michal Koucký, Prajakta Nimbhorkar, and Pavel Pudlák. Pseudorandom generators for group products: extended abstract. In *STOC*, pages 263–272, 2011. doi:10.1145/1993636.1993672.

33 Chin Ho Lee. Fourier bounds and pseudorandom generators for product tests. In *34th Computational Complexity Conference*, volume 137, 2019.

34 Chin Ho Lee, Edward Pyne, and Salil P. Vadhan. Fourier growth of regular branching programs. In Amit Chakrabarti and Chaitanya Swamy, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022, September 19-21, 2022, University of Illinois, Urbana-Champaign, USA (Virtual Conference)*, volume 245 of *LIPIcs*, pages 2:1–2:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.APPROX/RANDOM.2022.2.

35 Chin Ho Lee and Emanuele Viola. More on bounded independence plus noise: pseudorandom generators for read-once polynomials. *Theory Comput.*, 16:Paper No. 7, 50, 2020. doi:10.4086/toc.2020.v016a007.

36 Raghu Meka, Omer Reingold, and Avishay Tal. Pseudorandom generators for width-3 branching programs. In *STOC'19 – Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 626–637. ACM, New York, 2019. doi:10.1145/3313276.3316319.

37 Raghu Meka and David Zuckerman. Small-bias spaces for group products. In *Approximation, randomization, and combinatorial optimization*, volume 5687 of *Lecture Notes in Comput. Sci.*, pages 658–672. Springer, Berlin, 2009. doi:10.1007/978-3-642-03685-9_49.

38 Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992. doi:10.1007/BF01305237.

39 Edward Pyne and Salil Vadhan. Pseudodistributions that beat all pseudorandom generators (extended abstract). In *36th Computational Complexity Conference*, volume 200 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 33, 15, 2021.

40 Oded Regev. Entropy-based bounds on dimension reduction in L_1. *Israel Journal of Mathematics*, 195(2):825–832, 2013.

41 Omer Reingold, Thomas Steinke, and Salil Vadhan. Pseudorandomness for regular branching programs via Fourier analysis. In *Approximation, randomization, and combinatorial optimization*, volume 8096 of *Lecture Notes in Comput. Sci.*, pages 655–670. Springer, Heidelberg, 2013. doi:10.1007/978-3-642-40328-6_45.

42 Omer Reingold, Luca Trevisan, and Salil Vadhan. Pseudorandom walks on regular digraphs and the **RL** vs. **L** problem. In *STOC'06: Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 457–466. ACM, New York, 2006. doi:10.1145/1132516.1132583.

43 Michael E. Saks and Shiyu Zhou. BP $_{\mathrm{h}}$space(s) subseteq dspace(s$^{3/2}$). *J. Comput. Syst. Sci.*, 58(2):376–403, 1999. doi:10.1006/jcss.1998.1616.

44 Thomas Steinke. Pseudorandomness for permutation branching programs without the group theory. *Electron. Colloquium Comput. Complex.*, 2012. URL: http://eccc.hpi-web.de/report/2012/083.

45 Thomas Steinke, Salil Vadhan, and Andrew Wan. Pseudorandomness and Fourier-growth bounds for width-3 branching programs. *Theory Comput.*, 13:Paper No. 12, 50, 2017. doi:10.4086/toc.2017.v013a012.

A Read-Twice Permutation Branching Programs

We first show that read-*twice* permutation branching programs can compute polynomial width arbitrary-order ROBPs in subexponential width. This follows from Bennett's simulation in reversible computation [4], but here we have to take into account the number of reads of the program. We remark that the proof of this result implicitly uses permutation branching programs where the bound on the number of accept states is much smaller than the width bound [29].

▶ **Proposition 15.** *Let $f\colon \{0,1\}^n \to [w]$ be computable by an arbitrary-order ROBP B of width w. Then for every $k \in \mathbb{N}$, f is computable by a read-(2^k) permutation branching program B' of width $w^{(k+1)n^{1/k}}$.*

Proof. As both programs can read inputs in arbitrary order, by permuting the indices of x we can without loss of generality assume B is a SOBP. Given a SOBP O of width w, we first how to simulate it using a permutation read-2 BP P of width w^m for some $m \leq \sqrt{2n}$, in which every state is represented by an m-tuple in $[w]^m$. We will show that every step in P is *reversible*, that is, for every state $s^p_{t-1} \in [w]^m$ in P, not only can we compute $s^p_t := B_t(s^p_{t-1}, x_{i(t)})$ given s^p_{t-1}, but we can also compute s^p_{t-1} given s^p_t and $x_{i(t)}$, where $i(t) \in [n]$ is the coordinate of x read by P at the t-th step. One can verify that this is equivalent to the condition that P is a permutation program.

We construct P as follows. At each step, P remembers the at most m out of the n states reached by O. Let $s_i \in [w]$ be the state reached by O after reading $x_1, \ldots, x_i$. The program P first reads $x_1, \ldots, x_m$ to compute and store the m states $(s_1, \ldots, s_m) \in [w]^m$ reached by O in the first m steps. Knowing s_{m-2}, we can read x_{m-1} again to erase s_{m-1} from the memory, reaching the state $(s_1, \ldots, s_{m-2}, 0, s_m)$. Knowing s_{m-3}, we can read x_{m-2} again to erase s_{m-2}, reaching $(s_1, \ldots, s_{m-3}, 0, 0, s_m)$. More generally, after reading $x_{m-2}, \ldots, x_1$ the second time [4], we can erase $s_{m-1}, \ldots, s_1$ from memory and we are left with

$$(0, \ldots, 0, s_m).$$

Now, given s_m, we read the next $m-1$ bits $x_{m+1}, \ldots, x_{2m-1}$ to compute and store $(s_{m+1}, \ldots, s_{2m-1}, s_m)$. Using a similar strategy, we can read $x_{2m-2}, \ldots, x_{m+1}$ again to erase $s_{2m-2}, \ldots, s_{m+1}$ from memory, giving us

$$(0, \ldots, 0, s_{2m-1}, s_m).$$

Continuing, we can compute $(s_{\sum_{i=1}^m i}, s_{\sum_{i=2}^m i}, \ldots, s_{2m-1}, s_m)$ reversibly with w^m states. Thus we can compute s_n as long as

$$\sum_{i=1}^{m} i = \frac{m(m+1)}{2} \ge n.$$

which holds when $m = \sqrt{2n}$.

We just showed how to compute the m-tuple of states $(s_{\sum_{i=1}^m i}, s_{\sum_{i=2}^m i}, \ldots, s_{2m-1}, s_m)$ reversibly by reading the input twice. By reading the input another two times, from $(s_{\sum_{i=1}^m i}, s_{\sum_{i=2}^m i}, \ldots, s_{2m-1}, s_m)$ we can erase everything but $s_{\sum_{i=1}^m i} =: s_{f(m)}$ to compute $(s_{f(m)}, 0, \ldots, 0)$ reversibly. We now repeat the above strategy recursively to compute $(s_{\sum_{i=1}^m f(i)}, s_{\sum_{i=2}^m f(i)}, \ldots, s_{2f(m)-1}, s_{f(m)})$ with a read-4 permutation program of width $\sum_{i=1}^m f(i)$.

By an inductive argument, we can compute the state s_n of the read-once branching program reversibly with a read-(2^k) permutation program of width w^m, whenever

$$n \le \sum_{i_k=1}^{m} \sum_{i_{k-1}=1}^{i_k} \cdots \sum_{i_1=1}^{i_2} i_1 = \binom{m+k}{k+1}.$$

Choosing $m \ge (k+1)n^{1/(k+1)}$ completes the proof. ◀

A.1 Hardness for Read-Twice Permutation Programs

We now show that an exponential blow-up in the width in Proposition 15 is necessary. We first restate the theorem.

▶ **Theorem 16.** *For every read-twice ordering $i : [2n] \to [n]$, there exists a function $g\colon \{0,1\}^n \to \{0,1\}$ computable by a regular ROBP of width $O(1)$, such that every read-twice permutation branching program P of width $2^{n^{1/8}}$ with read order i computes g correctly on at most $1/2 + 2^{-\Omega(n^{1/8})}$ fraction of inputs.*

[4] Note that the order of how the bits are read matters. For example, without knowing s_{m-2} we cannot compute s_{m-1} using x_{m-1}.

A 2-pass BP is a read-2 BP where the first read of all its n-bit input come before the second read of any bit. We first prove an average-case lower bound against 2-pass permutation programs of width $2^{\sqrt{n}}$, where the second pass of the n-bit input is read in the same or the reverse order as the first pass. We show that given any read-twice ordering of the input bits, either $\sqrt{n}$ of the bits can be read in a read-once manner, or $n^{1/4}$ of the bits can be read in the 2-pass manner described above. In either case, we can define our hard function on at least $n^{1/4}$ bits and apply our average-case lower bounds. As both programs in the theorem can read input bits in arbitrary order, by permuting the indices of the input we can assume the indices in the first pass of the read are in increasing order.

A.1.1 From 2-pass lower bound to read-once lower bound

We obtain our 2-pass lower bound by a reduction to our read-once lower bound (Theorem 11) based on an idea by David, Papakonstantinou, and Sidiropoulos [15].

▶ **Proposition 36.** *Let P be a 2-pass permutation BP of width w that reads its first pass of the input in the standard order, and its second pass in the same or reverse order as the first pass. If $\Pr[P(U) = f(U)] \geq 1/2 + \varepsilon$ for some $f\colon \{0,1\}^n \to \{0,1\}$, then there exists a permutation SOBP permutation program P' of width w^2 such that $\Pr[P'(U) = f(U)] \geq 1/2 + \varepsilon/w$.*

▶ **Corollary 37.** *There exists a function $f\colon \{0,1\}^n \to \{0,1\}$ computable by a regular SOBP of constant width that is $(1/2 + 2^{-\sqrt{n}})$-hard against 2-pass permutation BPs of width $2^{c\sqrt{n}}$ that reads its second pass of the input in the same or reverse order as the first pass for a universal constant c.*

Proof. Let f be the function in Theorem 11 with $\varepsilon = 2^{-(1+c)\sqrt{n}}$, which is hard against permutation SOBPs of width $2^{(c'/(1+c))\sqrt{n}}$ for some universal constant c. Suppose f is not $(1/2 + 2^{-\sqrt{n}})$-hard against a 2-pass permutation SOBP of width $w = 2^{c\sqrt{n}}$. Then by Proposition 36, f is not $(1/2 + 2^{-(1+c)\sqrt{n}})$-hard against a permutation SOBP of width $2^{2c\sqrt{n}}$. Choosing c such that $2c < c'/(1 + c)$, we get a contradiction. ◀

Proof of Proposition 36. We first handle the case where the second pass is in the same order as the first pass. Suppose

$$\Pr[P(U) = f(U)] - \Pr[P(U) \neq f(U)] > \varepsilon.$$

Let V_n be the layer P reaches after making its first pass on x. By an averaging argument, there must be a state $v^* \in V_n$ such that

$$\begin{aligned}
&\Pr\big[\big(P(U) = f(U)\big) \wedge P_{\to v^*}(U)\big] - \Pr\big[\big(P(U) \neq f(U)\big) \wedge P_{\to v^*}(U)\big] \\
&\quad \geq \frac{1}{|V_n|} \sum_{v \in V_n} \Big(\Pr\big[\big(P(U) = f(U)\big) \wedge P_{\to v}(U)\big] - \Pr\big[\big(P(U) \neq f(U)\big) \wedge P_{\to v}(U)\big]\Big) \\
&\quad \geq \frac{1}{|V_n|} \Big(\Pr\big[P(U) = f(U)\big] - \Pr\big[P(U) \neq f(U)\big]\Big) \\
&\quad \geq \frac{\varepsilon}{|V_n|}.
\end{aligned}$$

For $b \in \{0,1\}$, consider the new function P'_b that outputs $P(x)$ if $P_{v^*}(x) = 1$ and outputs b otherwise. Note that $\mathsf{adv}(P'_b, f) \geq \varepsilon/|V_n|$ for one of the $b \in \{0,1\}$. Assume $b = 0$ without loss of generality.

We now show that the function P'_0 can be computed by a permutation SOBP of width w^2 as follows. Its i-th layer V'_i is $V_i \times V_{n+i}$. Its start state is (v_0, v^*). Its accept states are $V'_{acc} := \{(v_1, v_2) : (v_1 = v^* \wedge v_2 \in V_{acc})\}$.

To handle the case where the second pass is in the reverse order, we use a similar idea. Suppose

$$\Pr[P(U) = f(U)] - \Pr[P(U) \neq f(U)] > \varepsilon.$$

Let $V_{acc} \subseteq V_{2n}$ be the set of accept states in the final layer. By an averaging argument, there must be a state $v^* \in V_{acc}$ such that

$$\Pr\big[P_{\to v^*}(U) = f(U)\big] - \Pr\big[P_{\to v^*}(U) \neq f(U)\big] \geq \frac{\varepsilon}{|V_{acc}|}.$$

We now show that the function $P_{\to v^*}$ can be computed by a permutation SOBP program of width w^2. Here we use the fact that P is a permutation BP, where we can reverse the transitions in the program as follows. Define the reversed transition $P_r^{-1}\colon V_r \times \{0,1\} \to V_{r-1}$ to be $P_r^{-1}[v_r, x_r] := v_{r-1}$, where v_{r-1} is the unique state $v \in V_{r-1}$ such that $P_r[v_{r-1}, x_r] = v_r$.

To implement $P_{\to v^*}$, its i-th layer V'_i is $V_i \times V_{2n-i}$. Its start state is $(v_0, v^*) \in V_0 \times V_{2n}$. Its transition $P'_i\colon V'_{i-1} \to V'_i$ is $P'_i((v_1, v_2), x_i) = (P_i(v_1, x_i), P^{-1}_{n-i+1}(v_2, x_i))$. Its accept states are $V'_{acc} := \{(v_1, v_2) : v_1, v_2 \in V_n : v_1 = v_2)\}$. ◀

A.1.2 From 2-pass lower bound to read-2 lower bound

We follow a similar idea that is used in [24]. Given a read-2 sequence, by permuting the indices of the input bits, we may assume the first pass is in increasing order. We will show that it contains a subsequence of the form $i_1 i_1 i_2 i_2 \cdots i_{\sqrt{n}} i_{\sqrt{n}}$, in which case we can define the hard function on $x_{i_1}, \ldots, x_{i_{\sqrt{n}}}$ and applying our read-once lower bound on $\sqrt{n}$ bits, or it contains a 2-pass subsequence of the form $i_1 \cdots i_{\sqrt{n}} i_{\sigma(1)} \cdots i_{\sigma(\sqrt{n})}$ for some permutation $\sigma\colon [\sqrt{n}] \to [\sqrt{n}]$, in which case by the Erdős–Szekeres theorem (Theorem 38 below), the sequence $i_{\sigma(1)} \cdots i_{\sigma(\sqrt{n})}$ must contain a monotone subsequence $i_{j_1} \cdots i_{j_{n^{1/4}-1}}$ of length $n^{1/4} - 1$, and so the read-2 sequence contains a 2-pass sequence on $n^{1/4} - 1$ bits where the second pass is in the same or reverse order as the first pass. So we can apply our 2-pass lower bound.

Before proving Theorem 16, we first state the Erdős–Szekeres theorem, which will be used in our proof.

▶ **Theorem 38** (Erdős–Szekeres [20]). *For any integers s and r, any sequence of distinct real numbers of length $sr + 1$ contains a monotonically increasing subsequence of length $s + 1$ or a monotonically decreasing subsequence of length $r + 1$.*

Proof of Theorem 16. Let $s \in [n]^{2n}$ be a read-2 sequence. For $i \in [n]$, let $\mathsf{pos}^1(i)$ and $\mathsf{pos}^2(i)$ be the locations of the first and second occurrence of i, respectively. Partition the $2n$ indices of the sequence into $\sqrt{n}$ blocks $B_k : k \in [\sqrt{n}]$, where $B_k := [\mathsf{pos}^1((k-1)\sqrt{n}+1) : \mathsf{pos}^1(k\sqrt{n})-1]$. We consider two cases.

Suppose each block contains both occurrences of some element. That is, for each block $B_k : k \in [\sqrt{n}]$, we have $\mathsf{pos}^1(i_k), \mathsf{pos}^2(i_k) \in B_k$ for some $i_k \in [n]$. Then s contains the subsequence

$$i_1 i_1 \cdots i_{\sqrt{n}} i_{\sqrt{n}}.$$

We define $g(x) := f(x_{i_1}, \ldots, x_{i_{\sqrt{n}}})$, where f is the hard function defined in Theorem 11 (but on $\sqrt{n}$ bits). Let P be a read-2 permutation BP of width $2n^{1/8} \leq 2^{n^{1/4}}$ which reads its input in the order given by s. By Theorem 11, we have that $\Pr[P(U) = g(U)] \leq 1/2 + 2^{-\Omega(n^{1/4})}$ and g is computable by a regular ROBP of constant width.

Otherwise, some block does not contain both occurrences of any element. In other words, there exists a block B_k such that none of $\mathsf{pos}^2((k-1)\sqrt{n}+1), \ldots, \mathsf{pos}^2(k\sqrt{n})$ lies in B_k. In this case, s contains the 2-pass subsequence

$$\big((k-1)\sqrt{n}+1\big) \cdots \big(k\sqrt{n}\big) \cdot \sigma\big((k-1)\sqrt{n}+1\big) \cdots \sigma\big(k\sqrt{n}\big)$$

for some permutation σ on the $\sqrt{n}$ elements in the subsequence. Applying the Erdős–Szekeres theorem to the second half of the subsequence, we obtain a 2-pass subsequence on $n^{1/4} - 1$ elements from s where the second pass in the same or reverse order as the first pass. As in the previous case, by defining g to be the hard function f in Theorem 11 on these $n^{1/4} - 1$ bits, we conclude that $\Pr[P(U) = g(U)] \leq 1/2 + 2^{-\Omega(n^{1/8})}$ for any read-twice permutation program reading its input in the order given by s, and g is computable by a regular ROBP of constant width. ◀

Private Data Stream Analysis for Universal Symmetric Norm Estimation

Vladimir Braverman
Rice University, Houston, TX, USA

Joel Manning
Carnegie Mellon University, Pittsburgh, PA, USA

Zhiwei Steven Wu
Carnegie Mellon University, Pittsburgh, PA, USA

Samson Zhou
University of California Berkeley, CA, USA
Rice University, Houston, TX, USA

Abstract

We study how to release summary statistics on a data stream subject to the constraint of differential privacy. In particular, we focus on releasing the family of *symmetric norms*, which are invariant under sign-flips and coordinate-wise permutations on an input data stream and include L_p norms, k-support norms, top-k norms, and the box norm as special cases. Although it may be possible to design and analyze a separate mechanism for each symmetric norm, we propose a general parametrizable framework that differentially privately releases a number of sufficient statistics from which the approximation of all symmetric norms can be simultaneously computed. Our framework partitions the coordinates of the underlying frequency vector into different levels based on their magnitude and releases approximate frequencies for the "heavy" coordinates in important levels and releases approximate level sizes for the "light" coordinates in important levels. Surprisingly, our mechanism allows for the release of an *arbitrary* number of symmetric norm approximations without any overhead or additional loss in privacy. Moreover, our mechanism permits $(1+\alpha)$-approximation to each of the symmetric norms and can be implemented using sublinear space in the streaming model for many regimes of the accuracy and privacy parameters.

2012 ACM Subject Classification Security and privacy → Usability in security and privacy

Keywords and phrases Differential privacy, norm estimation

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.45

Category RANDOM

Related Version *Full Version*: https://arxiv.org/pdf/2307.04249.pdf

1 Introduction

The family of L_p norms represent important statistics on an underlying dataset, where the L_p norm[1] of an n-dimensional frequency vector x is defined as the number of nonzero coordinates of x for $p=0$ and $L_p(x)=(x_1^p+\ldots+x_n^p)^{1/p}$ for $p>0$. Thus, the L_0 norm counts the number of distinct elements in the dataset and, e.g., is used to detect denial of service or port scan attacks in network monitoring [3, 32], to understand the magnitude of quantities such as search engine queries or internet graph connectivity in data mining [55], to manage workload in database design [33], and to select a minimum-cost query plan in

[1] L_p for $p\in(0,1)$ does not satisfy the triangle inequality and therefore is not a norm, but is still well-defined/well-motivated and can be computed

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 45; pp. 45:1–45:24
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

query optimization [57]. The L_1 norm computes the total number of elements in the dataset and, e.g., is used for data mining [26] and hypothesis testing [39], while the L_2 norm, e.g., is used for training random forests in machine learning [20], computing the Gini index in statistics [50, 36], and network anomaly detection in traffic monitoring [44, 62], in particular in the context of heavy-hitters, e.g., [24, 16, 15, 17, 49, 14]. More generally, L_p norms for $p \in (0,2)$ have been used for entropy estimation [37]. Consequently, L_p estimation has been extensively studied in the data stream model [4, 40, 38, 45, 41, 5, 18, 35, 65, 66]. The simplest streaming model is perhaps the insertion-only model, in which a sequence of m updates increments coordinates of an n-dimensional frequency vector x and the goal is to compute or approximate some statistic of x in space that is sublinear in both m and n. For a more formal introduction to the streaming model, see Section 2.1.

In many cases, the underlying dataset contains sensitive information that should not be leaked. Hence, an active line of work has focused on estimating L_p norms for various values of p, while preserving differential privacy [53, 12, 59, 21, 63].

▶ **Definition 1** (Differential privacy, [29]). *Given $\varepsilon > 0$ and $\delta \in (0,1)$, a randomized algorithm $\mathcal{A} : \mathfrak{U}^* \to \mathcal{Y}$ is (ε, δ)-differentially private if, for every neighboring streams $\mathfrak{S}$ and $\mathfrak{S}'$ and for all $E \subseteq \mathcal{Y}$,*

$$\mathbf{Pr}\left[\mathcal{A}(\mathfrak{S}) \in E\right] \le e^{\varepsilon} \cdot \mathbf{Pr}\left[\mathcal{A}(\mathfrak{S}') \in E\right] + \delta.$$

For example, [12] showed that the Johnson-Lindenstrauss transformation preserves differential privacy (DP), thereby showing one of the main techniques in the streaming model for L_2 estimation already guarantees DP. Similarly, [59] showed that the Flajolet-Martin sketch, which is one of the main approaches for L_0 estimation in the streaming model, also preserves DP. However, algorithmic designs for L_p estimation in the streaming model differ greatly and require individual analysis to ensure DP, especially because it is known that for some problems, guaranteeing DP provably requires more space [28]. Unfortunately, the privacy and utility analysis can be quite difficult due to the complexity of the various techniques. This is especially pronounced in the work of [63], who studied the p-stable sketch [38], which estimates the L_p norm for $p \in (0,2]$. [63] showed that for $p \in (0,1]$, the p-stable sketch preserves DP, but was unable to show DP for $p \in (1,2]$, even though the general algorithmic approach remains the same. Thus the natural question is whether differential privacy can be guaranteed for an approach that simultaneously estimates the L_p norm in the streaming model, for all p. More generally, the family of L_p norms are all symmetric norms, which are invariant under sign-flips and coordinate-wise permutations on an input data stream. Symmetric norms thus also include other important families of norms such as the k-support norms and the top-k norms.

1.1 Our Contributions

In this paper, we show that not only does there exist a differentially private algorithm for the estimation of symmetric norms in the streaming model, but also that there exists an algorithm that privately releases a set of statistics, from which estimates of all (properly parametrized) symmetric norms can be simultaneously computed. To illustrate the difference, suppose we wanted to release approximations of the L_p norm of the stream for k different values of p. To guarantee (ε, δ)-DP for the set of k statistics, we would need, by advanced composition, to demand $\left(\mathcal{O}\left(\frac{\varepsilon}{\sqrt{k}}\right), \mathcal{O}\left(\frac{\delta}{k}\right)\right)$-DP from k instances of a single differentially private L_p-estimation algorithm, corresponding to the k different values of p. Due to accuracy-privacy tradeoffs, the quality of the estimation will degrade severely as k increases. For an extreme example, consider when k is some large polynomial of n and m so that the added noise will also be polynomial in n and m, and then there is no utility at all – the private algorithm might as well just release 0 for all queries!

In contrast, our algorithm releases a single set C of private statistics. By post-processing, we can then estimate the L_p norms for k different values of p while only requiring (ε, δ)-DP from C. Hence, our algorithm can simultaneously handle any large number of estimations of symmetric norms without compromising the quality of approximation.

We first informally introduce the definition of the maximum modulus of concentration of a norm, which measures the worst-case ratio of the maximum value of a norm on the L_2-unit sphere to the median value of a norm on the L_2-unit sphere, where the median can be taken over any restriction of the coordinates. Intuitively, maximum modulus of concentration of a norm quantifies the complexity of computing a norm. For example, the L_1 norm is generally "easy" to compute and has maximum modulus of concentration $\mathcal{O}(\log n)$. See Definition 18 for a more formal definition. Then our main result can informally be stated as follows:

▶ **Theorem 2** (Informal). *There exists a (ε, δ)-differentially private algorithm that outputs a set C, from which the $(1+\alpha)$-approximation to any norm, with* maximum modulus of concentration *at most M of a vector $x \in \mathbb{R}^n$ induced by a stream of length* $\text{poly}(n)$ *can be computed, with probability at least $1-\delta$. The algorithm uses $M^2 \cdot \text{poly}\left(\frac{1}{\alpha}, \frac{1}{\varepsilon}, \log n, \log \frac{1}{\delta}\right)$ bits of space.*

We remark that as is standard in differential privacy on data streams, both the privacy parameter ε and the accuracy parameter α cannot be too small or the additive noise will be too large and cannot be absorbed into the $(1+\alpha)$-multiplicative bounds. See Theorem 33 for the formal statement of Theorem 2 describing these bounds.

We also remark that in the statement of Theorem 2, the δ failure parameter of approximate DP is equal to the failure parameter δ of the utility guarantees of the algorithm. More generally, if the desired failure probability δ' of the utility guarantee is not equal to the privacy parameter δ, then the dependencies will change from $\log \frac{1}{\delta}$ to $\log \frac{1}{\delta\delta'}$.

We emphasize that prior to our work, there is no algorithm that can handle private symmetric norm estimation for arbitrary symmetric norms, much less simultaneously for all parametrized symmetric norms. Although there is specific analysis for various norm estimation algorithms, e.g., see the discussion on related work in Section 1.3, these algorithms require a specific predetermined norm for their input. Thus a separate private algorithm must be run for each estimation, which increases the overall space. Moreover, for a large number of queries, the privacy parameter will need to be much smaller due to the composition of privacy, and thus to ensure privacy, the utility of each algorithm is provably poor. Our algorithm sidesteps both the space and accuracy problems and is the first and only work to do so, as of yet.

Applications. We briefly describe a number of specific symmetric norms that are handled by Theorem 2 and commonly used across various applications in machine learning. We first note the following parameterization of the previously discussed L_p norms.

▶ **Lemma 3** ([52, 43]). *For L_p norms, we have that* $\text{mmc}(L) = \mathcal{O}(\log n)$ *for $p \in [1, 2]$ and* $\text{mmc}(L) = \mathcal{O}\left(n^{1/2-1/p}\right)$ *for $p > 2$.*

Thus our algorithm immediately introduces a differentially private mechanism for the approximation of L_p norms that unlike previous work, e.g., [12, 58, 25, 59, 21, 63], does not need to provide separate analysis for specific values of p. Moreover for constant-factor approximation, the space complexity is tight with the *optimal* L_p-approximation algorithms that do not consider privacy, up to polylogarithmic factors [42, 46, 34, 65] in the universe size n.

▶ **Definition 4** (Q-norm and Q'-norm). *We call a norm L a Q-norm if there exists a symmetric norm L' such that $L(x) = L'(x^2)^{1/2}$ for all $x \in \mathbb{R}^n$. Here, we use x^2 to denote the coordinate-wise square power of x. We also call a norm L' a Q'-norm if its dual norm is a Q-norm.*

The family of Q'-norms includes the L_p norms for $1 \le p \le 2$, the k-support norm, and the box norm [10] and thus Q'-norms have been proposed to regularize sparse recovery problems in machine learning. For instance, [7] showed that Q' norms have tighter relaxations than elastic nets and can thus be more effective for sparse prediction. Similarly, [51] used Q' norms to optimize sparse prediction algorithms for multitask clustering.

▶ **Lemma 5** ([11]). $\text{mmc}(L) = \mathcal{O}(\log n)$ *for every Q'-norm L.*

Theorem 2 and Lemma 5 thus present a differentially private algorithm for Q'-norm approximation that uses polylogarithmic space.

▶ **Definition 6** (Top-k norm). *The top-k norm for a vector $x \in \mathbb{R}^n$ is the sum of the largest k coordinates of $|x|$, where we use $|x|$ to denote the vector whose entries are the coordinate-wise absolute value of x.*

The top-k norm is frequently used to understand the more general Ky Fan k-norm [67], which is used to regularize optimization problems in numerical linear algebra. Whereas the Ky Fan k norm is defined as the sum of the k largest singular values of a matrix, the top-k norm is equivalent to the Ky Fan k norm when the input vector x represents the vector of the singular values of the matrix.

▶ **Lemma 7** ([11]). $\text{mmc}(L) = \tilde{\mathcal{O}}\left(\sqrt{\frac{n}{k}}\right)$ *for the top-k norm L.*

In particular, the top-k norm for a vector of singular values when $k = n$ is equivalent to the Schatten-1 norm of a matrix, which is a common metric for matrix fitting problems such as low-rank approximation [47].

▶ **Definition 8** (Shannon entropy). *For a frequency vector $v \in \mathbb{R}^n$, we define the Shannon entropy by $H(v) = -\sum_{i=1}^n v_i \log v_i$.*

To achieve an additive approximation to the Shannon entropy, we instead compute a multiplicative approximation to the exponential form, as follows:

▶ **Observation 9.** *A $(1+\alpha)$-multiplicative approximation of the function $h(v) := 2^{H(v)}$ corresponds to an α-additive approximation of the Shannon Entropy $H(v)$ (and vice versa).*

Moreover, computing a $(1+\alpha)$-approximation to $2^{H(v)}$ can be achieved through computing a $(1+\alpha)$-approximation to various L_p norms for $p \in (0, 2)$.

▶ **Lemma 10** (Section 3.3 in [37]). *Let $k = \log\frac{1}{\alpha} + \log\log m$ and $\alpha' = \frac{\alpha}{12(k+1)^3 \log m}$. There exists an explicit set $\{y_0, \ldots, y_k\}$ with $y_i \in (0, 2)$ for all i and a post-processing function that takes $(1+\alpha')$-approximations to $F_{y_i}(x)$, i.e., the (y_i)-th frequency moment of x, and outputs a $(1+\alpha)$-approximation to $h(v) = 2^{H(x)}$. Furthermore, the set $\{y_0, \ldots, y_k\}$ and post-processing function are both efficiently computable, i.e., polynomial runtime.*

Since our mechanism releases a private set of statistics from which $(1+\alpha)$-approximations to L_p norms can be computed for any $p \in (0, 2)$, then our mechanism also privately achieves an additive α-approximation to Shannon entropy.

1.2 Algorithmic Intuition and Overview

Our starting point is the L_p estimation algorithm of [40], which was parametrized by [11] to handle symmetric norms. For a $(1+\alpha)$-approximation, the algorithm partitions the n coordinates of the frequency vector x into powers of ξ-based on their magnitudes, where $\xi > 1$ is a fixed function of α. Each partition forms a level set, so that the i-th level set consists of the coordinates of x with frequency $[\xi^i, \xi^{i+1})$, but [40, 11] showed that it suffices to accurately count the size of each *important* level set and zero out to the other level sets, where a level set is considered important if its size is large enough to contribute an $\frac{\alpha^2}{\log m}$ fraction of the symmetric norm. In other words, if $\tilde{x}$ is a vector whose coordinates match those of x in important levels sets and are 0 elsewhere, then $(1-\alpha)L(x) \le L(\tilde{x}) \le (1+\alpha)L(x)$. We formalize the definition of importance in Section 2.2.

Private symmetric norm estimation in the centralized setting. To preserve (ε, δ)-differential privacy, one initial approach would be to view the frequency vector as a histogram and add Laplacian noise with scale $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$ to the frequency of each element. However, the level sets consisting of elements with frequencies between $[\xi^i, \xi^{i+1})$ for small i, say $i = 0$, could be largely perturbed by such Laplacian noise. For example, it is possible that for some coordinate j in an important level set, we have $x_j = 1$, in which case adding Laplacian noise with scale $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$ to x_j will heavily distort the coordinate. This can happen to all coordinates in the important level set, which results in an inaccurate estimation of the norm.

Fortunately, if i is small, the corresponding level set must contain a large number of elements if it is important, so it seems possible to privately release the size Γ_i of the level set. Indeed, we can show that the L_1 sensitivity of the vector corresponding to level set sizes is small and so we can add Laplacian noise with scale $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$ to each level set size. Hence if the level set has size Γ_i roughly $\Omega\left(\frac{1}{\alpha\varepsilon}\right)$, then the Laplacian noise will affect Γ_i by a $(1+\alpha)$-factor.

Unfortunately, there can be level sets that are both important and small in size. For example, if there is a single element with frequency m, then the size of the corresponding level set is just one. Then adding Laplacian noise with scale $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$ will severely affect the size of the level set and thus the estimation of the symmetric norm. On the other hand, for $m > \frac{1}{\alpha\varepsilon}$, the frequency of the coordinate is quite large so again it seems like we can just add Laplacian noise with scale $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$ and output the noisy frequency of the coordinate.

New approach: classifying and separately handling high, medium, and low frequency levels. The main takeaway from these challenges is that we should handle different level sets separately. For the level sets of small coordinates, the important level sets must have large size and thus we would like to release noisy sizes. For the important level sets of large coordinates, we would like to release noisy frequencies of the coordinates.

In that vein, we partition the levels into three groups after defining thresholds T_1 and T_2, with $T_1 > T_2$. We define the “high frequency levels” as the levels whose coordinates exceed T_1 in frequency. The intuition is that because the high frequency levels have such large magnitude, their frequencies can be well-approximated by running an L_2-heavy hitters algorithm on the stream S.

We define the “medium frequency levels” as the levels whose coordinates are between T_1 and T_2 in frequency. These coordinates are not large enough to be detected by running an L_2-heavy hitters algorithm on the stream S. However, the sizes of these level sets must be large if the level set is important. Thus there exists a substream S_j for which a large number of these coordinates are subsampled and their frequencies can be well-approximated by running an L_2-heavy hitters algorithm on the substream S_j.

Finally, we define the "low frequency levels" as the levels whose coordinates are less than T_2 in frequency. These coordinates are small enough that we cannot add Laplacian noise to their frequencies without affecting the level sets they are mapped to. Instead, we show that the L_1 sensitivity for the level set estimations is particularly small for the low frequency levels. Thus, for these frequency levels, we report the size of the frequency levels rather than the approximate frequencies of the heavy-hitters. We remark that if our goal was to just approximate the symmetric norms without preserving differential privacy, then it would suffice to just consider the high and medium frequency levels, since the low frequency levels are particularly problematic when Laplacian noise is added to the frequency vector. We also remark that we only use the thresholds T_1 and T_2 for the purposes of describing our algorithm – in the actual implementation of the algorithm, the thresholds T_1 and T_2 will be implicitly defined by each of the substreams.

Private symmetric norm estimation in the streaming model. Although the previously discussed intuition builds towards a working algorithm, the main caveat is that so far, we have mainly discussed the centralized model, where space is not restricted and so each coordinate and thus each level set size can be counted exactly. In the streaming model, we cannot explicitly track the frequency vector, or even the frequencies of a constant fraction of coordinates. Instead, to estimate the sizes of each level set, [40, 11] take the stream S and form $s = \mathcal{O}(\log n)$ substreams $S_1, \ldots, S_s$, where the j-th substream is created by sampling the universe of size n at a rate of $\frac{1}{2^{j-1}}$. Then S_j will only consist of the stream updates to the particular coordinates of x that are sampled. Thus in expectation, the frequency vector induced by S_j will have sparsity $\frac{\|x\|_0}{2^{j-1}}$. Similarly, if a level set i has size Γ_i, then $\frac{\Gamma_i}{2^{j-1}}$ of its members will be sampled in S_j in expectation. It can then be shown through a variance argument that if level set i is important, then there exists an explicit substream j from which Γ_i can be well-approximated using the L_2-heavy hitter algorithm CountSketch and as a result, the symmetric norm of x can be well-approximated. The main point of the subsampling approach is that if there exists a level set with large size consisting of small coordinates, then the coordinates will not be detected by the CountSketch on S, but because S_j has significantly smaller L_2 norm, then the coordinates will be detected by CountSketch on S_j.

However, adapting the subsampling and heavy-hitter approach introduces additional challenges for privacy. For instance, we can analyze the L_2-heavy hitter algorithm CountSketch and show that although the L_1 sensitivity of the estimated frequency for a single coordinate is small, the L_1 sensitivity of the estimated frequency vector for all the coordinates may be large. Instead, we use the view that CountSketch is a composition function that first only estimates frequencies for the top poly $\left(\frac{1}{\alpha}, \frac{1}{\varepsilon}, \log n\right)$ and then outputs only those estimates that are above a certain threshold. Similarly, the Laplacian noise added to privately use CountSketch can alter the sizes of a significant number of level sets for small coordinates. Thus for the small coordinates (corresponding to the substreams S_j with large j), we invoke CountSketch with much higher accuracy, so that with high probability, it will return *exactly* the frequencies for the small coordinates. For example, note that if the frequency x_k of a coordinate $k \in [n]$ is at most $\frac{1}{2\alpha^2\varepsilon}$, then any $(1 + \alpha^2\varepsilon)$-approximation to x_k can be rounded to exactly recover x_k. This decreases the L_1 sensitivity of the vector of estimated level set sizes, therefore allowing us to add Laplacian noise without greatly affecting the quality of approximation.

1.3 Related Work

Non-private L_p norm estimation is one of the fundamental problems in the streaming model, beginning with [4]'s seminal work that tracks the inner product of the frequency vector with a random sign vector for L_2 estimation (as well as a telescoping argument for integer $p > 0$). [38, 45] later showed that this approach could be generalized for $p \in (0, 2]$ by tracking the inner product of the frequency vector with a vector with randomly generated p-stable variables, which only exist for $p \in (0, 2]$. For $p > 2$, [5] gave an L_p estimation algorithm using the max-stability property of exponential random variables. More generally, [40] introduced the framework of subsampling and using heavy-hitters for L_p estimation, which [11] parametrized to all symmetric norms. It should be emphasized that these techniques all handle the more general turnstile model, in which ± 1 updates are allowed to each coordinate, rather than single positive increments. Hence our techniques also extend to the turnstile model with a minor change on the conditions.

More recently, [13, 61] given a general framework for converting non-private approximation algorithms into private approximation algorithms, provided that the accuracy of these algorithms could be tuned with an input parameter $\varepsilon > 0$, i.e., the algorithms can achieve $(1 + \varepsilon)$-approximation for a wide range of $\varepsilon > 0$. Their results presented a solution that addresses the difficulty of adapting privacy specifically to each non-private algorithm separately. However, their framework only applies to problems with scalar outputs and thus do not handle synthetic data release. Therefore, privately answering multiple norm queries while circumventing composition bounds is still a challenge that their results cannot handle.

Symmetric norms have also recently received attention in other big data models as well. [6] studied approximate near neighbors for general symmetric norms while [48] studied symmetric norm estimation for network monitoring. [60] considered Orlicz norm regression and other loss functions where the penalty is a symmetric norm. [19] gave an algorithm to approximate the symmetric norm in the sliding window model, where updates in the data stream implicitly expire after a fixed amount of time.

Specific cases of private L_p estimation in the streaming model have also been previously well-studied. [25, 59] studied private L_0 estimation using the Flajolet-Martin sketch, while [63] studied private L_p estimation for $p \in (0, 1]$ using the p-stable sketch and [12, 58, 25, 21] studied private L_2 estimation using the Johnson-Lindenstrauss projection. Specifically, [12] gave an (ε, δ)-DP algorithm for L_2 estimation that achieves a $(1 + \varepsilon)$-approximation while using $\mathcal{O}\left(\frac{1}{\varepsilon^2} \log n \log \frac{1}{\delta}\right)$ bits of space and [63] gave an (ε, δ)-DP algorithm for L_p estimation that achieves a $(1 + \alpha)$-approximation while using $\mathcal{O}\left(\frac{1}{\alpha^2} \log n \log \frac{1}{\delta}\right)$ bits of space for constant ε and $p \in (0, 1)$. For fractional $p > 1$, private distribution estimation algorithms [2, 68, 9, 64] can be used to approximate the L_p norm, but since the algorithms provide information over a much larger distribution, e.g., much larger histograms of frequencies, the privacy-accuracy trade-off is sub-optimal and the space complexity is exponentially worse.

The related problem of privately releasing heavy-hitters in big data models has also been well-studied. [23] studied the problem of continually releasing L_1-heavy hitters in a stream, while [30] studied L_1-heavy hitters and other problems in the pan-private streaming model. The heavy-hitter problem has also received significant attention in the local model, e.g., [9, 27, 1, 22, 8], where individual users should locally randomize their data before sending differentially private information to an untrusted server that aggregates the statistics across all users.

2 Preliminaries

In this section, we introduce definitions and simple or well-known results from differential privacy, sketching algorithms, and symmetric norms. For notation, we use $[n]$ for an integer $n > 0$ to denote the set $\{1, \ldots, n\}$. We also use the notation $\text{poly}(n)$ to represent a constant degree polynomial in n and we say an event occurs *with high probability* if the event holds with probability $1 - \frac{1}{\text{poly}(n)}$. Similarly, we use $\text{polylog}(n)$ to denote $\text{poly}(\log n)$. Given a vector $x \in \mathbb{R}^n$, we define its second frequency moment $F_2(x) = x_1^2 + \ldots + x_n^2$. Finally, for a parameter $c \geq 1$, we say that X provides a C-approximation to a quantity Y if $\frac{X}{C} \leq Y \leq C \cdot X$.

Privately releasing multiple statistics that are individually differentially private can also be done, but comes at a slight cost.

▶ **Theorem 11** (Composition and post-processing of differential privacy, [31]). *Let $\mathcal{A}_i : \mathfrak{U}_i \to X_i$ be an $(\varepsilon_i, \delta_i)$-differential private algorithm for $i \in [k]$. Then $\mathcal{A}_{[k]}(x) = (\mathcal{A}_1(x), \ldots, \mathcal{A}_k(x))$ is $\left(\sum_{i=1}^k \varepsilon_i, \sum_{i=1}^k \delta_i\right)$-differentially private. Furthermore, if $g_i : X_i \to X_i'$ is an arbitrary random mapping, then $g_i(\mathcal{M}_i(x))$ is $(\varepsilon_i, \delta_i)$-differentially private.*

Although there exists more sophisticated approaches for composition, such as advanced composition, we do not need them for our purposes.

2.1 Streaming and Sketching Algorithms

In the streaming model, a frequency vector $x \in \mathbb{R}^n$ is induced by a sequence of updates. In the insertion-only streaming model, x is defined through a stream of m updates $u_1, \ldots, u_m$, where $u_t \in [n]$ for each $t \in [m]$ so that $x_i = |\{t \in [m] \,|\, u_t = i\}|$ for all $i \in [n]$. In other words, x_i is the number of times that $i \in [n]$ appears in the stream. We remark that our techniques generalize to some degree to turnstile streams, where each update is an ordered pair $u_t = (\Delta_t, c_t)$, so that the t-th update changes the c_t-th coordinate by Δ_t, i.e., $c_t \in [n]$ is a coordinate and $\Delta_t \in [-M, M]$ for some parameter $M > 0$. In this turnstile model, the vector x is defined so that $x_i = \sum_{t:c_t=i} \Delta_t$ for all $i \in [n]$. Although our techniques can apply to the general turnstile model with a minor change on the conditions and assumptions, we shall work with the insertion-only streaming model throughout the remainder of the paper.

Given a frequency vector $x \in \mathbb{R}^n$ on a data stream, the AMS algorithm for L_2-estimation first generates a sign vector $\sigma \in \{-1, +1\}^n$ and sets $S_1 = (\langle \sigma, x \rangle)^2$. We remark that to maintain σ in small space, it suffices for the coordinates of the sign vector σ to be 4-wise independent and therefore it suffices to randomly generate and store a 4-wise independent hash function. The AMS algorithm then repeats this process $b = \frac{6}{\alpha^2}$ independent times to obtain dot products $S_1, \ldots, S_b$, sets Z^2 to be the arithmetic mean of $S_1, \ldots, S_b$, and reports Z. We define the L_2 norm of a vector $x \in \mathbb{R}^n$ by $L_2(x) = \sqrt{x_1^2 + \ldots + x_n^2}$.

▶ **Definition 12** (ν-approximate η L_2-heavy hitters problem). *Given an accuracy parameter $\nu \in (0, 1)$, a threshold parameter η, and a frequency vector $x \in \mathbb{R}^n$, compute a set $H \subseteq [n]$ and a set of approximations $\widehat{x_k}$ for all $k \in H$ such that:*

(1) *If $x_k \geq \eta L_2(x)$ for any $k \in [n]$, then $k \in H$, so that H contains all η L_2-heavy hitters of x.*

(2) *There exists a universal constant $C \in (0, 1)$ so that if $x_k \leq \frac{C\eta}{2} L_2(x)$ for any $k \in [n]$, then $k \notin H$, so that H does not contain any index that is not an $\frac{C\eta}{2}$ L_2-heavy hitter of x.*

(3) *If $k \in H$ for any $k \in [n]$, then compute $(1 \pm \nu)$-approximation to the frequency x_k, i.e., a value $\widehat{x_k}$ such that $(1 - \nu)x_k \leq \widehat{x_k} \leq (1 + \nu)x_k$.*

The well-known CountSketch algorithm can be parametrized to provide an estimated frequency to each item and then releases the approximate frequencies of each item that surpasses a threshold proportional to the output of AMS:

▶ **Theorem 13** (CountSketch for ν-approximate η L_2-heavy hitters, [24]). *There exists a one-pass streaming algorithm* CountSketch *that takes an accuracy parameter $\nu \in (0,1)$ and a threshold parameter η^2 and outputs a list H that contains all indices $k \in [n]$ of an underlying frequency vector x with $x_k \ge \eta\, L_2(x)$ and no index $k \in [n]$ with $x_k \le \eta(1-\nu)\, L_2(x)$. For each $k \in H$,* CountSketch *also reports a estimated frequency $\widehat{x_k}$ such that $(1-\nu)x_k \le \widehat{x_k} \le (1+\nu)x_k$. The algorithm uses $\mathcal{O}\left(\frac{1}{\eta^2\nu^2}\log^2 n\right)$ bits of space and succeeds with probability $1-\frac{1}{\mathrm{poly}(m)}$.*

Algorithm 1 Heavy-hitter algorithm CountSketch.

```
Input: Stream $\mathfrak{S}$ inducing frequency vector $x \in \mathbb{R}^n$, accuracy parameter $\nu \in (0,1)$, and
    threshold parameter $\eta \in (0,1)$
Output: $L_2$ Heavy-hitter algorithm
$r \leftarrow \mathcal{O}(\log n)$, $b \leftarrow \mathcal{O}\left(\frac{1}{\eta^2\nu^2}\right)$
Pick hash functions $h^{(1)}, \ldots, h^{(r)} : [n] \to [b]$ and $s^{(1)}, \ldots, s^{(r)} : [n] \to \{-1,+1\}$
$S_{i,j} \leftarrow 0$ for $(i,j) \in [r] \times [b]$
for each update $u_i \in [n]$, $i \in [m]$ do
    for each $j \in [r]$ do
        $b_{i,j} \leftarrow h^{(j)}(u_i)$ and $s_{i,j} \leftarrow s^{(j)}(u_i)$
        $S_{j,b_{i,j}} \leftarrow S_{j,b_{i,j}} + s_{i,j}$
for each $i \in [n]$ do
    $b_{i,j} \leftarrow h^{(j)}(u_i)$ for each $j \in [r]$
    return $\mathrm{median}_{j\in[r]} |S_{j,b_{i,j}}|$ as the estimated frequency for $x_i$
```

We recall the following sensitivity analysis of CountSketch.

▶ **Lemma 14** (Sensitivity of CountSketch). *Let $x, x' \in \mathbb{R}^n$ with* $\max(\|x-x'\|_0, \|x-x'\|_1) \le 2$.

There exists a private variant PrivCountSketch of CountSketch that adds noise to each coordinate and then uses a standard private threshold routine to ensure differential privacy, giving the following guarantees:

▶ **Lemma 15.** *There exists a one-pass streaming algorithm* PrivCountSketch *that takes an accuracy parameter $\nu \in (0,1)$ and a threshold parameter η^2 and outputs a list H that contains all indices $k \in [n]$ of an underlying frequency vector x with $x_k \ge \eta\, L_2(x)$ and no index $k \in [n]$ with $x_k \le \eta(1-\nu)\, L_2(x)$. For each $k \in H$,* PrivCountSketch *also reports a estimated frequency $\widehat{x_k}$ such that $(1-\nu)x_k - \mathcal{O}\left(\frac{\log m}{\eta\nu}\right) \le \widehat{x_k} \le (1+\nu)x_k + \mathcal{O}\left(\frac{\log m}{\eta\nu}\right)$. The algorithm uses $\mathcal{O}\left(\frac{1}{\eta^2\nu^2}\log^2 n\right)$ bits of space and succeeds with probability $1-\frac{1}{\mathrm{poly}(m)}$.*

2.2 Symmetric Norms

In this section, we provide necessary preliminaries for symmetric norm estimation.

▶ **Definition 16** (Symmetric norm). *A function $L : \mathbb{R}^n \to \mathbb{R}$ is a* symmetric norm *if L is a norm and for all $x \in \mathbb{R}^n$ and any vector $y \in \mathbb{R}^n$ that is a permutation of the coordinates of x, we have $L(x) = L(y)$. Moreover, we have $L(x) = L(|x|)$, where $|x|$ is the coordinate-wise absolute value of x.*

▶ **Definition 17** (Modulus of concentration). *Let $x \in \mathbb{R}^n$ be a random variable drawn from the uniform distribution on the L_2-unit sphere S^{n-1} and let b_L denote the maximum value of $L(x)$ over S^{n-1}. The median of a symmetric norm L is the unique value M_L such that $\mathbf{Pr}\left[L(x) \geq M_L\right] \geq \frac{1}{2}$ and $\mathbf{Pr}\left[L(x) \leq M_L\right] \geq \frac{1}{2}$. Then the ratio $\mathrm{mc}(L) := \frac{b_L}{M_L}$ is the* modulus of concentration *of the norm L.*

Although the modulus of concentration quantifies the "average" behavior of the norm L on $\mathbb{R}^n$, norms with challenging behavior can still be embedded in lower-dimensional subspaces. For instance, the L_1 norm satisfies $\mathrm{mc}(L) = \mathcal{O}(1)$, but when $x \in \mathbb{R}^n$ has fewer than $\sqrt{n}$ nonzero coordinates, the norm $\max(L_\infty(x), L_1(x)/\sqrt{n})$ on the unit ball becomes identically $L_\infty(x)$ [11], which requires $\Omega(\sqrt{n})$ space [4] to estimate. Hence, we further quantify the behavior of a norm L by examining its behavior on all lower dimensions.

▶ **Definition 18** (Maximum modulus of concentration). *For a norm $L : \mathbb{R}^n \to \mathbb{R}$ and every $k \leq n$, define the norm $L^{(k)} : \mathbb{R}^k \to \mathbb{R}$ by $L^{(k)}((x_1, \ldots, x_k)) := L((x_1, \ldots, x_k, 0, \ldots, 0))$. Then the* maximum modulus of concentration *of the norm L is $\mathrm{mmc}(L) := \max_{k \leq n} \mathrm{mc}(L^{(k)}) = \max_{k \leq n} \frac{b_{L^{(k)}}}{M_{L^{(k)}}}$.*

▶ **Definition 19** (Important Levels). *For $x \in \mathbb{R}^n$ and $\xi > 1$, we define the level i as the set $B_i = \{k \in [n] \,:\, \xi^{i-1} \leq |x_k| \leq \xi^i\}$. We define $b_i := |B_i|$ as the size of level i. For $\beta \in (0, 1]$, we say level i is β-*important *if*

$$b_i > \beta \sum_{j > i} b_j, \qquad b_i \xi^{2i} \geq \beta \sum_{j \leq i} b_j \xi^{2j}.$$

Informally, level i is β-important if (1) its size is at least a β-fraction of the total sizes of the higher levels and (2) its contribution is roughly a β-fraction of the total contribution of all the lower levels. We would like to show that to approximate a symmetric norm $L(x)$, it suffices to identify the β-important levels and their sizes for a fixed base $\xi > 1$.

▶ **Definition 20** (Level Vectors and Buckets). *For $x \in \mathbb{R}^n$ and $\xi > 1$, the* level vector *for x is*

$$V(x) := (\underbrace{\xi^1, \ldots, \xi^1}_{b_1 \text{ times}}, \underbrace{\xi^2, \ldots, \xi^2}_{b_2 \text{ times}}, \ldots, \underbrace{\xi^k, \ldots, \xi^k}_{b_k \text{ times}}, 0, \ldots, 0) \in \mathbb{R}^n,$$

where each b_i is the size of level i. The i-th bucket of $V(x)$ is

$$V_i(x) := (\underbrace{0, \ldots, 0,}_{b_1 + \ldots + b_{i-1} \text{ times}} \underbrace{\xi^i, \ldots, \xi^i}_{b_i \text{ times}}, \ldots, \underbrace{0, \ldots, 0}_{b_{i+1} + \ldots + b_k \text{ times}}, 0, \ldots, 0) \in \mathbb{R}^n.$$

We similarly define the approximate level vectors $\widehat{V(x)}$ and $\widehat{V_i(x)}$ using approximations $\widehat{b_1}, \ldots, \widehat{b_k}$ for $b_1, \ldots, b_k$. We write $V(x) \setminus V_i(x)$ to denote the vector that replaces the i-th bucket in $V(x)$ with all zeros and we write $V(x) \setminus V_i(x) \cup \widehat{V_i(x)}$ to denote the vector that replaces the i-th bucket in $V(x)$ with $\widehat{b_i}$ instances of ξ^i.

Rather than directly handle the important levels, we define the β-contributing levels and instead work toward estimating the contribution of the β-contributing levels.

▶ **Definition 21** (Contributing Levels). *Given $x \in \mathbb{R}^n$, a level i defined by base $\xi > 1$ is β-*contributing *if $L(V_i(x)) \geq \beta L(V(x))$.*

[11] showed that even if all levels that are not β-contributing are removed, the contribution of the remaining levels forms a good approximation to $L(x)$.

▶ **Lemma 22** ([11]). *Given $x \in \mathbb{R}^n$ and levels defined by a base $\xi > 1$, let $V'(x)$ be the vector obtained by removing all levels that are not β-contributing from $V(x)$. Then $(1 - \mathcal{O}\left(\log_\xi n\right) \cdot \beta) L(V(x)) \le L(V'(x)) \le L(V(x))$.*

Hence for appropriate $\xi > 1$ and $\beta \in (0, 1]$, it suffices to identify the β-contributing levels, zero out the remaining levels, and determine the contribution of the resulting vector to approximate the symmetric norm $L(x)$.

▶ **Lemma 23** ([11]). *Given an accuracy parameter $\alpha \in (0, 1]$, let base $\xi = (1 + \mathcal{O}(\alpha))$, importance parameter $\beta = \mathcal{O}\left(\frac{\alpha^5}{\mathrm{mmc}(\ell)^2 \cdot \log^5 m}\right)$, and $\alpha' = \mathcal{O}\left(\frac{\alpha^2}{\log n}\right)$. Let $\widehat{b_i} \le b_i$ for all i and $\widehat{b_i} \ge (1 - \alpha') b_i$ for all β-important levels. Let $\widehat{V}$ be the level vector constructed using the estimates $\widehat{b_1}, \widehat{b_2}, \ldots$ and let V' be the level vector constructed by removing all the buckets that are not β-contributing in $\widehat{V}$. Then $(1 - \alpha) L(V(x)) \le L(V'(x)) \le L(V(x))$.*

To identify the β-contributing levels, [11] first notes that the size of the level must be at least a significant fraction of the total size of the higher levels.

▶ **Lemma 24** ([11]). *Given $x \in \mathbb{R}^n$, let the level sets be defined by a base $\xi > 1$. If level i is β-contributing, then there exists some fixed constant $\lambda > 0$ such that $b_i \ge \frac{\lambda \beta^2}{\mathrm{mmc}(\ell)^2 \log^2 n} \cdot \sum_{j > i} b_j$.*

Moreover, [11] observes that the squared mass of a β-contributing level must be at least a significant fraction of the total squared mass of the lower levels.

▶ **Lemma 25** ([11]). *Given $x \in \mathbb{R}^n$, let the level sets be defined by a base $\xi > 1$. If level i is β-contributing, then there exists some fixed constant $\lambda > 0$ such that $b_i \xi^{2i} \ge \frac{\lambda \beta^2}{\mathrm{mmc}(\ell)^2 (\log_\xi n) \log^2 n} \cdot \sum_{j \le i} b_j \xi^{2j}$.*

Observe that together, Lemma 24 and Lemma 25 imply that a β-contributing level i must also be an important level as defined in Definition 19. Crucially, since Lemma 25 states that the squared mass (or the F_2 frequency moment) of the β-contributing levels must be a significant fraction of the total squared mass of the lower levels, then it suggests we might be able to identify the β-contributing levels through an L_2-heavy hitters algorithm after removing the higher levels. Indeed, [11] show that the problem of identifying the size (and thus the contribution) of the β-contributing levels can be reduced to the task of finding ν-approximate η-heavy hitters for specific parameters of ν and η.

▶ **Lemma 26** ([11]). *Let $s = \mathcal{O}(\log n)$. If a level i is β-important, then either $\xi^{2i} \ge \frac{\alpha^2 \beta \varepsilon^2}{\log^2 m} F_2(x)$ or there exists $j \in [s]$ such that $b_i \ge \frac{2^j \log^2 m}{\alpha^2 \varepsilon^2}$ and $\xi^{2i} \in \left[\frac{\alpha^2 \beta \varepsilon^2}{\log^2 m} \cdot \frac{F_2(x)}{2^j}, \frac{\alpha^2 \beta \varepsilon^2}{\log^2 m} \cdot \frac{F_2(x)}{2^{j-1}}\right]$.*

Lemma 26 implies that if level i is β-important, then either (1) it will be identified by using PrivCountSketch, i.e., Lemma 15, with threshold $\frac{\alpha^2 \beta}{\log^2 m}$ on the stream or (2) its contribution can be well-approximated by using PrivCountSketch with threshold $\frac{\alpha^2 \beta \varepsilon^2}{\log^2 m}$ on a substream formed by sampling coordinates of the universe with probability $\frac{1}{2^j}$. We thus split our algorithm and analysis to handle these cases. In particular, we call a frequency level i "high" if $\xi^{2i} \ge \frac{\alpha^2 \beta \varepsilon^2}{\log^2 m} F_2(x)$. We call a frequency level i "medium" if $\xi^{2i} \ge \frac{\alpha^2 \beta' \varepsilon^2}{2^j} F_2(x) > T$ and $b_i \ge \mathcal{O}\left(\frac{2^j \log^2 m}{\alpha^2 \varepsilon^2}\right)$ for a certain $\beta' > 0$ and a threshold T. We call a frequency level i "low" if $\xi^{2i} \ge \frac{\alpha^2 \beta' \varepsilon^2}{2^j} F_2(x)$ and $b_i \ge \mathcal{O}\left(\frac{2^j \log^2 m}{\alpha^2 \varepsilon^2}\right)$, but $T \ge \frac{\alpha^2 \beta' \varepsilon^2}{2^j} F_2(x)$.

3 Private Symmetric Norm Estimation Algorithm

In this section, we give our algorithm that releases a set of private statistics from which an arbitrary number of symmetric norms can be well-approximated. In particular, recall that Lemma 23 suggests that it suffices to approximate the sizes of the important levels and identity the non-important levels, so that the contributions of the non-important levels can be set to zero. We partition the levels into three groups after defining explicit thresholds T_1 and T_2, with $T_1 > T_2$. Recall that we define the "high frequency levels" as the levels whose coordinates exceed T_1 in frequency, the "medium frequency levels" as the levels whose coordinates are between T_1 and T_2 in frequency, and the "low frequency levels" as the levels whose coordinates are less than T_2 in frequency.

The intuition is that because the high frequency levels have such large magnitude, their frequencies can be well-approximated by running an L_2-heavy hitters algorithm on the stream S. On the other hand, the medium frequency level coordinates are not large enough to be detected by running an L_2-heavy hitters algorithm on the stream S, but the sizes of these level sets must be large if the level set is important and therefore, there exists a substream S_j for which a large number of these coordinates are subsampled and their frequencies can be well-approximated by running an L_2-heavy hitters algorithm on the substream S_j. Here we form substreams $S_0, S_1, \ldots$ so that S_j first samples elements of the universe $[n]$ at a rate $\frac{1}{2^j}$ and then only contains the stream updates that are relevant to the sampled elements. Finally, the low frequency level coordinates are small enough that we cannot add Laplacian noise to their frequencies without affecting the level sets they are mapped to. We instead show that L_1 sensitivity for the level set estimations is particularly small for the low frequency levels and thus, we report the size of the level sets of the low frequency levels rather than the approximate frequencies of the heavy-hitters.

We emphasize that we only use the thresholds T_1 and T_2 for the purposes of describing our algorithm – in the actual implementation of the algorithm, the thresholds T_1 and T_2 will be implicitly defined by each of the substreams. For example, the items with threshold larger than T_1 will automatically be revealed through the stream S, while the items with thresholds between T_1 and T_2 will be revealed through the substreams S_j with $2^j > \frac{\log n}{\beta' \alpha \varepsilon}$ for explicit parameters α, β', and ε. More specifically, note that Algorithm 2 sets $\beta' = \mathcal{O}\left(\frac{\alpha^2 \beta \varepsilon^2}{\log^2 m}\right)$ or more specifically $\beta' = \frac{\alpha^2 \beta \varepsilon^2}{2 \log^2 m}$. Then $\beta' \cdot F_2(x)$ corresponds to the threshold T_1, which is utilized in the proofs of Section 3.1. Similarly, Algorithm 3 leverages the quantity $\frac{\log n}{\beta' \alpha \varepsilon}$ to define the threshold T_2, which is then utilized in the proofs of Section 3.2.

3.1 Recovery of High Frequency Levels

In this section, we describe our algorithm for recovering the high frequency levels, whose coordinates have sufficiently large magnitude and thus their frequencies can be well-approximated by running an L_2-heavy hitters algorithm on the stream S. Moreover, with high probability, adding Laplacian noise will not affect the level sets because the frequencies are so large. Thus it simply suffices to return the noisy estimated frequencies of each of the elements in the high frequency levels. This algorithm is the simplest of our cases and we give the algorithm in full in Algorithm 2.

We first show that coordinates in high frequency levels are identified and their frequencies are accurately estimated. Similarly, we show that if a coordinate does not have high frequency, it will not be output by Algorithm 2.

Algorithm 2 Algorithm to privately estimate the high levels.

```
Input: Privacy parameter $\varepsilon > 0$, accuracy parameter $\alpha \in (0,1)$
Output: Private estimation of the frequencies of the coordinates of the high frequency levels
$\beta \leftarrow \mathcal{O}\left(\frac{\alpha^5}{\mathrm{mmc}(L)^2 \log^5 m}\right)$, $\beta' \leftarrow \mathcal{O}\left(\frac{\alpha^2\beta\varepsilon^2}{\log^2 m}\right)$
Run PrivCountSketch on the stream $S$ with threshold $\alpha^2\beta'$ and failure probability $\frac{1}{\mathrm{poly}(m)}$
for each heavy-hitter $k \in [n]$ reported by PrivCountSketch do
    Let $\widetilde{x_k}$ be the frequency estimated by PrivCountSketch
    $\widehat{x_k} \leftarrow \widetilde{x_k} + \mathsf{Lap}\left(\frac{8}{\beta'\varepsilon}\right)$
    return $\widehat{x_k}$
```

▶ **Lemma 27.** *Suppose $m = \frac{\Omega(\log^5 m)}{\alpha^5\beta^2\varepsilon^5}$. Then with high probability Algorithm 2 outputs $\widehat{x_k}$ such that if $x_k^2 \geq \frac{\alpha^2\beta\varepsilon^2}{\log^2 m} F_2(x)$, then $(1-\alpha^2)x_k \leq \widehat{x_k} \leq x_k$ and if $x_k^2 < \frac{\alpha^2\beta\varepsilon^2}{2\log^2 m} F_2(x)$, then $\widehat{x_k} < \frac{3\alpha^2\beta\varepsilon^2}{4\log^2 m} F_2(x)$.*

We then show that Algorithm 2 preserves differential privacy and analyze its space complexity.

▶ **Lemma 28.** *Algorithm 2 is $\left(\frac{\varepsilon}{4}, \frac{\delta}{4}\right)$-differentially private for $\delta = \frac{1}{\mathrm{poly}(m)}$ and uses space* $\mathrm{mmc}(L)^2 \cdot \mathrm{poly}\left(\frac{1}{\alpha}, \frac{1}{\varepsilon}, \log m\right)$.

3.2 Recovery of Medium Frequency Levels

In this section, we describe our algorithm for recovering the medium frequency levels, whose coordinates do not have sufficiently large magnitude to be detected by running an L_2-heavy hitters algorithm on the stream S, but have sufficiently large size, so that there exists some $j \in [s]$ across the s subsampling levels such that the coordinates can be detected by running an L_2-heavy hitters algorithm on the stream S_j. On the other hand, their magnitudes are sufficiently large so that with high probability, adding Laplacian noise will not affect the level sets. We give the algorithm in full in Algorithm 3.

We first upper bound the second frequency moment (and hence the L_2 norm) of each substream. This is necessary because we want to detect the coordinates of the medium frequency levels as L_2-heavy hitters for each substream, but if the substream has overwhelmingly large L_2 norm, then we will not be able to find coordinates of the medium frequency levels. However, it may not be true that $F_2(S_j)$ is significantly smaller than $F_2(S)$ with high probability. For example, if there were a single large element, then the probability it is sampled at level s is $\frac{1}{2^s}$, which is roughly $\frac{1}{n} > \frac{1}{\mathrm{poly}(m)}$. Instead, we note that PrivCountSketch benefits from the stronger *tail guarantee*, which states that not only does PrivCountSketch with threshold $\eta < 1$ detect the elements k such that $(x_k)^2 \geq \eta F_2(S)$, but it also detects the elements k such that $(x_k)^2 \geq \eta F_2(S_{\mathrm{tail}(1/\eta)})$, where $S_{\mathrm{tail}(1/\eta)}$ is the frequency vector x induced by S, with the largest $\frac{1}{\eta}$ entries instead set to zero [15, 17].

▶ **Lemma 29.** *Consider a β-important level i with $\xi^{2i} \in \left[\frac{\beta\alpha^2\varepsilon^2}{\log^2 m} \cdot \frac{F_2(x)}{2^j}, \frac{\beta\alpha^2\varepsilon^2}{\log^2 m} \cdot \frac{F_2(x)}{2^{j-1}}\right]$ for some integer $j > 0$ and $\xi^i > \frac{\log n}{\beta'\alpha\varepsilon}$. If $F_2((S_j)_{1/(\alpha^2\beta'\varepsilon^2)}) \leq \frac{200\log m}{2^j} F_2(x)$ for all $j \in [s]$, then with high probability, Algorithm 3 outputs $\widehat{b_i}$ such that $(1-\mathcal{O}(\alpha))b_i \leq \widehat{b_i} \leq b_i$, where b_i is the size of level i.*

Algorithm 3 Algorithm to privately estimate the medium levels.

```
Input: Privacy parameter $\varepsilon > 0$, accuracy parameter $\alpha \in (0,1)$
Output: Private estimations of the sizes of the medium frequency levels
$\beta \leftarrow \mathcal{O}\left(\frac{\alpha^5}{\mathrm{mmc}(L)^2 \log^5 m}\right)$, $\beta' \leftarrow \mathcal{O}\left(\frac{\alpha^3 \beta \varepsilon^2}{\log^2 m}\right)$, $\xi \leftarrow (1 + \mathcal{O}(\varepsilon))$
$\gamma \leftarrow (1/2, 1)$ uniformly at random, $\ell \leftarrow \lceil \log_{\xi}(2m) \rceil$, $s \leftarrow \mathcal{O}(\log n)$
for $j \in [s]$ with $2^j > \frac{\log n}{\beta' \alpha \varepsilon}$ do
    Form stream $S_j$ by sampling elements of $[n]$ with probability $\frac{1}{2^j}$
    Run PrivCountSketch$_j$ on stream $S_j$ with threshold $\alpha^2 \beta' \varepsilon^2$ and failure probability $\frac{1}{\mathrm{poly}(m)}$
    for each heavy-hitter $k \in [n]$ reported by PrivCountSketch$_j$ do
        Let $\widehat{x_k}$ be the frequency estimated by PrivCountSketch$_j$
        if $\widehat{x_k} > \frac{\log n}{\beta' \alpha \varepsilon}$ then
            $\widetilde{x_k} \leftarrow \widehat{x_k} + \mathsf{Lap}\left(\frac{8}{\beta' \varepsilon}\right)$
    for $i \in [\ell]$ with $\frac{m^2}{2^{j+1}} > \gamma \xi^{2i} \geq 2^j > \mathcal{O}\left(\frac{\log n}{\beta' \alpha^2 \varepsilon}\right)$ do
        Let $\widetilde{b_i}$ be the number of indices $k \in [n]$ such that $\gamma \xi^{2i} \leq \widetilde{x_k} < \gamma \xi^{2i+2}$
        $\widehat{b_i} \leftarrow \frac{2^j}{(1+\mathcal{O}(\alpha))} \widetilde{b_i}$
        return $\widehat{b_i}$
```

We now show that Algorithm 3 preserves differential privacy and analyze its space complexity.

▶ **Lemma 30.** *Algorithm 3 is $\left(\frac{\varepsilon}{4}, \frac{\delta}{4}\right)$-differentially private for $\delta = \frac{1}{\mathrm{poly}(m)}$ and uses space* $\mathrm{mmc}(L)^2 \cdot \mathrm{poly}\left(\frac{1}{\alpha}, \frac{1}{\varepsilon}, \log m\right)$.

3.3 Recovery of Low Frequency Levels

In this section, we describe our algorithm for recovering the low frequency levels, whose coordinates have magnitude small enough that we cannot add Laplacian noise to their frequencies without affecting the corresponding level set sizes. We instead report the sizes of the level sets for the low frequency levels rather than the approximate frequencies of the heavy-hitters. Thus we must add Laplacian noise to the sizes of the level sets; we show that the L_1 sensitivity for the level set estimations is particularly small for the low frequency levels and thus the Laplacian noise does not greatly affect the estimates of the level set sizes. We note that this approach does not work for the high frequency levels because the high frequency levels may have small level set sizes, so that adding Laplacian noise to the sizes can significantly affect the resulting estimates of the level set sizes. Similarly, it is more challenging to argue the low L_1 sensitivity for the level set estimations for the medium frequency levels. Hence, both the algorithm and analysis are especially well-catered to the low frequency levels. We give the algorithm in full in Algorithm 4.

We first show that the estimates of the level set sizes for the low frequency levels are accurate.

▶ **Lemma 31.** *Consider a β-important level i with $\xi^{2i} \in \left[\frac{\beta\alpha^2\varepsilon^2}{\log^2 m} \cdot \frac{F_2(x)}{2^j}, \frac{\beta\alpha^2\varepsilon^2}{\log^2 m} \cdot \frac{F_2(x)}{2^{j-1}}\right]$ for some integer $j > 0$ and $\xi^i \leq \frac{\log n}{\beta' \alpha \varepsilon}$. If $F_2((S_j)_{1/(\alpha^2\beta'\varepsilon^2)}) \leq \frac{200 \log m}{2^j} F_2(x)$ for all $j \in [s]$, then with high probability, Algorithm 4 outputs $\widehat{b_i}$ such that*

$$(1 - \mathcal{O}(\alpha)) b_i \leq \widehat{b_i} \leq b_i,$$

where b_i is the size of level set i.

Algorithm 4 Algorithm to privately estimate the low levels.

```
Input: Privacy parameter $\varepsilon > 0$, accuracy parameter $\alpha \in (0,1)$
Output: Private estimations of the sizes of the low frequency levels
$\beta \leftarrow \mathcal{O}\left(\frac{\alpha^5}{\mathrm{mmc}(L)^2 \log^5 m}\right)$, $\beta' \leftarrow \mathcal{O}\left(\frac{\alpha^2\beta\varepsilon}{\log n}\right)$, $\xi \leftarrow (1+\mathcal{O}(\varepsilon))$
$\gamma \leftarrow (1/2,1)$ uniformly at random, $\ell \leftarrow \lceil \log_\xi(2m) \rceil$, $s \leftarrow \mathcal{O}(\log n)$
for $j \in [s]$ with $2^j \le \frac{\log n}{\beta'\alpha\varepsilon}$ do
    Form stream $S_j$ by sampling elements of $[n]$ with probability $\frac{1}{2^j}$
    Run PrivCountSketch$_j$ on stream $S_j$ with threshold $\beta'' := \mathcal{O}\left(\frac{\beta'\alpha^2\varepsilon^3}{\log^2 n}\right)$
    for each heavy-hitter $k \in [n]$ reported by PrivCountSketch$_j$ do
        Let $\widehat{x_k}$ be the frequency estimated by PrivCountSketch$_j$
    for $i \in [\ell]$ with $\mathcal{O}\left(\frac{\log n}{\beta'\alpha^2\varepsilon}\right) \ge 2^{j+1} > \gamma\xi^{2i} \ge 2^j$ do
        Let $\widetilde{b_i}$ be the number of indices $k \in [n]$ such that $\gamma\xi^{2i} \le \widehat{x_k} < \gamma\xi^{2i+2}$
        $\widehat{b_i} \leftarrow \frac{2^j}{(1+\mathcal{O}(\alpha))}\left(\widetilde{b_i} + \mathsf{Lap}\left(\frac{8}{\varepsilon}\right)\right)$
        return $\widehat{b_i}$
```

We then show that Algorithm 4 is differentially private and analyze its space complexity.

▶ **Lemma 32.** *Algorithm 4 is $\left(\frac{\varepsilon}{4}, \frac{\delta}{4}\right)$-differentially private for $\delta = \frac{1}{\mathrm{poly}(m)}$ and uses space* $\mathrm{mmc}(L)^2 \cdot \mathrm{poly}\left(\frac{1}{\alpha}, \frac{1}{\varepsilon}, \log m\right)$.

3.4 Putting Things Together

We would like to combine the subroutines from the previous sections to output a private dataset for symmetric norm estimation. Thus it remains to describe how to privately partition the coordinates into the high, medium, and low frequency levels. To that end, we remark that by Lemma 14, the sensitivity of PrivCountSketch in Algorithm 1 is at most 2. Moreover, although PrivCountSketch actually provides an estimated frequency for each coordinate, for our purposes, we only need estimated frequencies for the L_2-heavy hitters and there are at most $K := \mathcal{O}\left(\frac{1}{\eta^2}\right)$ possible L_2-heavy hitters with whichever threshold η that we choose, e.g., $\eta = \alpha^2\beta'$ in Algorithm 2. Thus it suffices to observe that we can privately partition the coordinates into the high, medium, and low frequency levels by first privately outputting the top K estimated frequencies and then partitioning the coordinates according to their noisy estimated frequencies, which can be viewed as post-processing. In particular, [56] observes that it suffices to add Laplacian noise with scale $\frac{8}{\eta\varepsilon}$ to each of the frequencies and then outputting the top K noisy estimated frequencies to achieve $\frac{\varepsilon}{4}$-differential privacy.

We now finally put together the results from the previous sections to show the following result. We remark that we set $\varepsilon, \alpha = \tilde{\Omega}\left(\left(\frac{M^2}{m}\right)^{\frac{1}{30}}\right)$ so that along with the assumption that $m \ge n$, the conditions of the previous statements are satisfied, e.g., Lemma 34, we obtain the following formalization of Theorem 2.

▶ **Theorem 33.** *Given a parameter $M > 1$, let $\varepsilon, \alpha = \tilde{\Omega}\left(\left(\frac{M^2}{m}\right)^{\frac{1}{30}}\right)$. There exists a (ε, δ)-differentially private algorithm that outputs a set C, from which the $(1+\alpha)$-approximation to any norm , with* maximum modulus of concentration *at most M of a vector $x \in \mathbb{R}^n$ induced by a stream of length* $\mathrm{poly}(n)$ *can be computed, with probability at least $1-\delta$. The algorithm uses* $M^2 \cdot \mathrm{poly}\left(\frac{1}{\alpha}, \frac{1}{\varepsilon}, \log n, \log \frac{1}{\delta}\right)$ *bits of space.*

References

1 Jayadev Acharya and Ziteng Sun. Communication complexity in locally private distribution estimation and heavy hitters. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, pages 51–60, 2019.

2 Gergely Ács, Claude Castelluccia, and Rui Chen. Differentially private histogram publishing through lossy compression. In *12th IEEE International Conference on Data Mining, ICDM*, pages 1–10, 2012.

3 Aditya Akella, Ashwin Bharambe, Mike Reiter, and Srinivasan Seshan. Detecting ddos attacks on isp networks. In *Proceedings of the Twenty-Second ACM SIGMOD/PODS Workshop on Management and Processing of Data Streams*, pages 1–3, 2003.

4 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

5 Alexandr Andoni. High frequency moments via max-stability. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 6364–6368, 2017.

6 Alexandr Andoni, Huy L. Nguyen, Aleksandar Nikolov, Ilya P. Razenshteyn, and Erik Waingarten. Approximate near neighbors for general symmetric norms. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theoryof Computing, STOC*, pages 902–913, 2017.

7 Andreas Argyriou, Rina Foygel, and Nathan Srebro. Sparse prediction with the k-support norm. In *Advances in Neural Information Processing Systems 25: Annual Conference on Neural Information Processing Systems*, pages 1466–1474, 2012.

8 Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Thakurta. Practical locally private heavy hitters. *J. Mach. Learn. Res.*, 21:16:1–16:42, 2020.

9 Raef Bassily and Adam D. Smith. Local, private, efficient protocols for succinct histograms. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC*, pages 127–135, 2015.

10 Rajendra Bhatia. *Matrix analysis*, volume 169. Springer Science & Business Media, 2013.

11 Jaroslaw Blasiok, Vladimir Braverman, Stephen R. Chestnut, Robert Krauthgamer, and Lin F. Yang. Streaming symmetric norms via measure concentration. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 716–729, 2017.

12 Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. The johnson-lindenstrauss transform itself preserves differential privacy. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 410–419, 2012.

13 Jeremiah Blocki, Elena Grigorescu, Tamalika Mukherjee, and Samson Zhou. How to make your approximation algorithm private: A black-box differentially-private transformation for tunable approximation algorithms of functions with low sensitivity. *CoRR*, abs/2210.03831, 2022.

14 Jeremiah Blocki, Seunghoon Lee, Tamalika Mukherjee, and Samson Zhou. Differentially private L_2-heavy hitters in the sliding window model. In *The Eleventh International Conference on Learning Representations, ICLR*, 2023.

15 Vladimir Braverman, Stephen R. Chestnut, Nikita Ivkin, Jelani Nelson, Zhengyu Wang, and David P. Woodruff. Bptree: An l_2 heavy hitters algorithm using constant memory. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS*, pages 361–376, 2017.

16 Vladimir Braverman, Stephen R. Chestnut, Nikita Ivkin, and David P. Woodruff. Beating countsketch for heavy hitters in insertion streams. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 740–753, 2016.

17 Vladimir Braverman, Elena Grigorescu, Harry Lang, David P. Woodruff, and Samson Zhou. Nearly optimal distinct elements and heavy hitters on sliding windows. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 7:1–7:22, 2018.

18 Vladimir Braverman, Emanuele Viola, David P. Woodruff, and Lin F. Yang. Revisiting frequency moment estimation in random order streams. In *45th International Colloquium on Automata, Languages, and Programming, ICALP*, pages 25:1–25:14, 2018.

19 Vladimir Braverman, Viska Wei, and Samson Zhou. Symmetric norm estimation and regression on sliding windows. In *Computing and Combinatorics – 27th International Conference, COCOON, Proceedings*, pages 528–539, 2021.

20 Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

21 Zhiqi Bu, Sivakanth Gopi, Janardhan Kulkarni, Yin Tat Lee, Judy Hanwen Shen, and Uthaipon Tantipongpipat. Fast and memory efficient differentially private-sgd via JL projections. *CoRR*, abs/2102.03013, 2021.

22 Mark Bun, Jelani Nelson, and Uri Stemmer. Heavy hitters and the structure of local privacy. *ACM Trans. Algorithms*, 15(4):51:1–51:40, 2019.

23 T.-H. Hubert Chan, Mingfei Li, Elaine Shi, and Wenchang Xu. Differentially private continual monitoring of heavy hitters from distributed streams. In *Privacy Enhancing Technologies – 12th International Symposium, PETS Proceedings*, pages 140–159, 2012.

24 Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312(1):3–15, 2004.

25 Seung Geol Choi, Dana Dachman-Soled, Mukul Kulkarni, and Arkady Yerukhimovich. Differentially-private multi-party sketching for large-scale statistics. *Proc. Priv. Enhancing Technol.*, 2020(3):153–174, 2020.

26 Graham Cormode, S. Muthukrishnan, and Irina Rozenbaum. Summarizing and mining inverse distributions on data streams via dynamic inverse sampling. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 25–36, 2005.

27 Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pages 3571–3580, 2017.

28 Itai Dinur, Uri Stemmer, David P. Woodruff, and Samson Zhou. On differential privacy and adaptive data analysis with bounded space. In *Advances in Cryptology – EUROCRYPT – 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part III*, pages 35–65, 2023.

29 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC, Proceedings*, pages 265–284, 2006.

30 Cynthia Dwork, Moni Naor, Toniann Pitassi, Guy N. Rothblum, and Sergey Yekhanin. Pan-private streaming algorithms. In *Innovations in Computer Science – ICS. Proceedings*, pages 66–80, 2010.

31 Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

32 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, 2003.

33 Sheldon J. Finkelstein, Mario Schkolnick, and Paolo Tiberio. Physical database design for relational databases. *ACM Trans. Database Syst.*, 13(1):91–128, 1988.

34 Sumit Ganguly. Taylor polynomial estimator for estimating frequency moments. In *Automata, Languages, and Programming – 42nd International Colloquium, ICALP Proceedings, Part I*, pages 542–553, 2015.

35 Sumit Ganguly and David P. Woodruff. High probability frequency moment sketches. In *45th International Colloquium on Automata, Languages, and Programming, ICALP*, pages 58:1–58:15, 2018.

36 Corrado Gini. Variabilità e mutabilità. *Reprinted in Memorie di metodologica statistica*, 1912.

37 Nicholas J. A. Harvey, Jelani Nelson, and Krzysztof Onak. Sketching and streaming entropy via approximation theory. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 489–498, 2008.

38 Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, 2006.

39 Piotr Indyk and Andrew McGregor. Declaring independence via the sketching of sketches. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 737–745, 2008.

40 Piotr Indyk and David P. Woodruff. Optimal approximations of the frequency moments of data streams. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 202–208, 2005.

41 Daniel M. Kane, Jelani Nelson, Ely Porat, and David P. Woodruff. Fast moment estimation in data streams in optimal space. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC*, pages 745–754, 2011.

42 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, SODA*, pages 1161–1178, 2010.

43 Bo'az Klartag and Roman Vershynin. Small ball probability and dvoretzky's theorem. *Israel Journal of Mathematics*, 157(1):193–207, 2007.

44 Balachander Krishnamurthy, Subhabrata Sen, Yin Zhang, and Yan Chen. Sketch-based change detection: Methods, evaluation, and applications. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 234–247, 2003.

45 Ping Li. Estimators and tail bounds for dimension reduction in l_α $(0 < \alpha \leq 2)$ using stable random projections. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 10–19, 2008.

46 Yi Li and David P. Woodruff. A tight lower bound for high frequency moment estimation with small error. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques – 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM. Proceedings*, pages 623–638, 2013.

47 Yi Li and David P. Woodruff. Input-sparsity low rank approximation in schatten norm. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, pages 6001–6009, 2020.

48 Zaoxing Liu, Antonis Manousis, Gregory Vorsanger, Vyas Sekar, and Vladimir Braverman. One sketch to rule them all: Rethinking network flow monitoring with univmon. In *Proceedings of the ACM SIGCOMM 2016 Conference*, pages 101–114, 2016.

49 Zaoxing Liu, Samson Zhou, Ori Rottenstreich, Vladimir Braverman, and Jennifer Rexford. Memory-efficient performance monitoring on programmable switches with lean algorithms. In *1st Symposium on Algorithmic Principles of Computer Systems, APOCS*, pages 31–44, 2020.

50 Max O Lorenz. Methods of measuring the concentration of wealth. *Publications of the American statistical association*, 9(70):209–219, 1905.

51 Andrew M. McDonald, Massimiliano Pontil, and Dimitris Stamos. Spectral k-support norm regularization. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems*, pages 3644–3652, 2014.

52 Vitali D Milman and Gideon Schechtman. *Asymptotic theory of finite dimensional normed spaces: Isoperimetric inequalities in riemannian manifolds*, volume 1200. Springer, 2009.

53 Darakhshan J. Mir, S. Muthukrishnan, Aleksandar Nikolov, and Rebecca N. Wright. Pan-private algorithms via statistics on sketches. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*, pages 37–48, 2011.

54 Noam Nisan. Pseudorandom generators for space-bounded computation. *Comb.*, 12(4):449–461, 1992.

55 Christopher R Palmer, Georgos Siganos, Michalis Faloutsos, Christos Faloutsos, and Phillip B Gibbons. The connectivity and fault-tolerance of the internet topology, 2001.

56 Gang Qiao, Weijie J. Su, and Li Zhang. Oneshot differentially private top-k selection. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, pages 8672–8681, 2021.

57 Patricia G. Selinger, Morton M. Astrahan, Donald D. Chamberlin, Raymond A. Lorie, and Thomas G. Price. Access path selection in a relational database management system. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*, pages 23–34, 1979.

58 Or Sheffet. Differentially private ordinary least squares. *J. Priv. Confidentiality*, 9(1), 2019.

59 Adam D. Smith, Shuang Song, and Abhradeep Thakurta. The flajolet-martin sketch itself preserves differential privacy: Private counting with minimal space. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020.

60 Zhao Song, Ruosong Wang, Lin F. Yang, Hongyang Zhang, and Peilin Zhong. Efficient symmetric norm regression via linear sketching. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 828–838, 2019.

61 Jakub Tetek. Additive noise mechanisms for making randomized approximation algorithms differentially private. *CoRR*, abs/2211.03695, 2022.

62 Mikkel Thorup and Yin Zhang. Tabulation based 4-universal hashing with applications to second moment estimation. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 615–624, 2004.

63 Lun Wang, Iosif Pinelis, and Dawn Song. Differentially private fractional frequency moments estimation with polylogarithmic space. In *The Tenth International Conference on Learning Representations, ICLR*, 2022.

64 Tianhao Wang, Milan Lopuhaä-Zwakenberg, Zitao Li, Boris Skoric, and Ninghui Li. Locally differentially private frequency estimation with consistency. In *27th Annual Network and Distributed System Security Symposium, NDSS*, 2020.

65 David P. Woodruff and Samson Zhou. Separations for estimating large frequency moments on data streams. In *48th International Colloquium on Automata, Languages, and Programming, ICALP*, volume 198, pages 112:1–112:21, 2021.

66 David P. Woodruff and Samson Zhou. Tight bounds for adversarially robust streams and sliding windows via difference estimators. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 1183–1196, 2021.

67 Bin Wu, Chao Ding, Defeng Sun, and Kim-Chuan Toh. On the moreau-yosida regularization of the vector k-norm related functions. *SIAM J. Optim.*, 24(2):766–794, 2014.

68 Jia Xu, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, Ge Yu, and Marianne Winslett. Differentially private histogram publication. *VLDB J.*, 22(6):797–822, 2013.

A Missing Proofs

We first show that coordinates in high frequency levels are identified and their frequencies are accurately estimated.

▶ **Lemma 34.** *Suppose* $x_k^2 \geq \frac{\alpha^2\beta\varepsilon^2}{\log^2 m} F_2(x)$ *and* $m = \frac{\Omega\left(\log^5 m\right)}{\alpha^5\beta^2\varepsilon^5}$. *Then with high probability, Algorithm 2 outputs* $\widehat{x_k}$ *such that*

$$(1-\alpha^2)x_k \leq \widehat{x_k} \leq x_k.$$

Proof. Consider Algorithm 2. Since $x_k^2 \geq \frac{\alpha^2\beta\varepsilon^2}{2\log^2 m} F_2(x)$ and we call PrivCountSketch with threshold $\alpha^2\beta'$ with $\beta' := \mathcal{O}\left(\frac{\alpha^2\beta\varepsilon^2}{\log^2 m}\right)$, then with high probability, the output $\widetilde{x_k}$ satisfies

$$(1-\mathcal{O}\left(\alpha^2\right))x_k \leq \widetilde{x_k} \leq x_k.$$

We then add Laplacian noise $\mathsf{Lap}\left(\frac{8}{\beta'\varepsilon}\right)$ to $\widetilde{x_k}$ to form $\widehat{x_k}$. Since $x_k^2 \geq \frac{\alpha^2\beta\varepsilon^2}{2\log^2 m}\, F_2(x) = \beta'\, F_2(x)$ and $F_2(x) \geq m$, then with high probability, the Laplacian noise is at most an α^2 fraction of $\widehat{x_k}$ for $\frac{\mathcal{O}(\log m)}{\beta'\varepsilon} \leq \alpha^2 m$ or equivalently, $m \geq \frac{\Omega(\log m)}{\alpha(\beta')^2\varepsilon} \geq \frac{\Omega\left(\log^5 m\right)}{\alpha^5\beta^2\varepsilon^5}$. Hence with high probability,

$$(1-\alpha^2)x_k \leq \widehat{x_k} \leq x_k. \qquad ◀$$

Similarly, we show that if a coordinate does not have high frequency, it will not be output by Algorithm 2.

► **Lemma 35.** *Suppose $x_k^2 < \frac{\alpha^2\beta\varepsilon^2}{2\log^2 m}\, F_2(x)$ and $m = \frac{\Omega\left(\log^5 m\right)}{\alpha^5\beta^2\varepsilon^5}$. Then with high probability, Algorithm 2 outputs $\widehat{x_k}$ such that*

$$\widehat{x_k} < \frac{3\alpha^2\beta\varepsilon^2}{4\log^2 m}\, F_2(x).$$

Proof. Since $x_k^2 < \frac{\alpha^2\beta\varepsilon^2}{2\log^2 m}\, F_2(x)$ and we call PrivCountSketch with threshold $\alpha^2\beta'$ with $\beta' := \mathcal{O}\left(\frac{\alpha^2\beta\varepsilon^2}{\log^2 m}\right)$, then the output $\widetilde{x_k}$ satisfies

$$|(\widetilde{x_k})^2 - (x_k)^2| \leq 2\alpha^2\beta'\, F_2(x).$$

We then add Laplacian noise $\mathsf{Lap}\left(\frac{8}{\beta'\varepsilon}\right)$ to $\widetilde{x_k}$ to form $\widehat{x_k}$. Since $F_2(x) \geq m$, then with high probability, the Laplacian noise is at most an $\alpha^2\beta'$ fraction of $F_2(x)$ for $\frac{\mathcal{O}(\log m)}{\beta'\varepsilon} \leq \alpha^2 m$ or equivalently, $m \geq \frac{\Omega(\log m)}{\alpha(\beta')^2\varepsilon} \geq \frac{\Omega\left(\log^5 m\right)}{\alpha^5\beta^2\varepsilon^5}$. Hence with high probability,

$$|(\widetilde{x_k})^2 - (x_k)^2| \leq \frac{\alpha^2\beta\varepsilon^2}{4\log^2 m}\, F_2(x).$$

Since $x_k^2 < \frac{\alpha^2\beta\varepsilon^2}{2\log^2 m}\, F_2(x)$, then it follows that

$$\widehat{x_k} < \frac{3\alpha^2\beta\varepsilon^2}{4\log^2 m}\, F_2(x). \qquad ◀$$

We now show that Algorithm 2 preserves differential privacy.

► **Lemma 36.** *Algorithm 2 is $\left(\frac{\varepsilon}{4}, \frac{\delta}{4}\right)$-differentially private for $\delta = \frac{1}{\mathrm{poly}(m)}$. Algorithm 2 uses space $\mathrm{mmc}(L)^2 \cdot \mathrm{poly}\left(\frac{1}{\alpha}, \frac{1}{\varepsilon}, \log m\right)$.*

Proof. By Lemma 14, the sensitivity of PrivCountSketch is at most 2 and the failure probability is $\frac{1}{\mathrm{poly}(m)}$. Thus by adding Laplacian noise $\mathsf{Lap}\left(\frac{8}{\beta'\varepsilon}\right)$ to $\widetilde{x_k}$, each estimated frequency is $\left(\frac{\beta'\varepsilon}{4}, \frac{\delta}{4\beta}\right)$-differentially private for $\delta = \frac{1}{\mathrm{poly}(m)}$. Since PrivCountSketch with threshold β' can release at most $\frac{1}{\beta}$ estimated frequencies and post-processing does not cause loss in privacy, then by Theorem 11, Algorithm 2 is $\left(\frac{\varepsilon}{4}, \frac{\delta}{4}\right)$. ◀

Finally, we analyze the space complexity of Algorithm 2.

► **Lemma 37.** *Algorithm 2 uses space $\mathrm{mmc}(L)^2 \cdot \mathrm{poly}\left(\frac{1}{\alpha}, \frac{1}{\varepsilon}, \log m\right)$.*

Proof. The space complexity follows from running a single instance of PrivCountSketch with threshold $\alpha^2\beta'$ and failure probability $\frac{1}{\mathrm{poly}(m)}$, where $\beta' = \mathcal{O}\left(\frac{\alpha^2\beta\varepsilon^2}{\log^2 m}\right)$ and $\beta = \mathcal{O}\left(\frac{\alpha^5}{\mathrm{mmc}(L)^2\log^5 m}\right)$. ◀

▶ **Lemma 38.** *With high probability, we have that* $F_2((S_j)_{1/(\alpha^2\beta'\varepsilon^2)}) \le \frac{200\log m}{2^j}\,F_2(x)$ *for all* $j \in [s]$.

Proof. For each $j \in [s]$, we have that $\mathbb{E}\left[F_2(S_j)\right] = \frac{F_2(x)}{2^j}$. By Chernoff bounds with $\mathcal{O}\left(\log n\right)$-wise limited independence, we have that

$$\mathbf{Pr}\left[F_2((S_j)_{1/(\alpha^2\beta'\varepsilon^2)}) > \frac{200\log m}{2^j}\,F_2(x)\right] \le \frac{1}{\mathrm{poly}(m)}.$$

Since $s \le 2\log m$, then by a union bound over all $j \in [s]$, we have that $F_2(S_j) \le (200\log m)F_2(x)$ for all $j \in [s]$. ◀

We now show that conditioned on the event that the L_2 norm of the subsampled streams are not too large, then we can well-approximate the frequency of any coordinate of the medium frequency levels, provided that they are sampled in the substream.

▶ **Lemma 39.** *Suppose* i *is a* β*-important level and* $k \in [n]$ *is in level* i*, so that* $x_k \in [\xi^i, \xi^{i+1})$. *If* $F_2((S_j)_{1/(\alpha^2\beta'\varepsilon^2)}) \le \frac{200\log m}{2^j}\,F_2(x)$ *for all* $j \in [s]$ *and* k *is sampled in stream* S_j *with* $2^j > \frac{\log n}{\beta'\alpha\varepsilon}$*, then with high probability, Algorithm 3 outputs* $\widehat{x_k}$ *such that*

$$(1-\alpha^2)x_k \le \widehat{x_k} \le x_k.$$

Proof. Consider Algorithm 3. By Lemma 26, $x_2^2 \in \left[\frac{\alpha^2\beta\varepsilon^2}{\log^2 m}\cdot\frac{F_2(x)}{2^j}, \frac{\alpha^2\beta\varepsilon^2}{\log^2 m}\cdot\frac{F_2(x)}{2^{j-1}}\right]$. Conditioned on the event that $F_2((S_j)_{1/(\alpha^2\beta'\varepsilon^2)}) \le \frac{200\log m}{2^j}\,F_2(x)$ for all $j \in [s]$, then $x_k^2 \ge \frac{\alpha^2\beta\varepsilon^2}{200\log m}\,F_2(S_j)$. We call PrivCountSketch with threshold $\alpha^2\beta'\varepsilon^2 = \mathcal{O}\left(\frac{\alpha^4\beta\varepsilon^3}{\log^2 m}\right)$. Thus with high probability, the output $\widetilde{x_k}$ satisfies

$$(1-\mathcal{O}\left(\alpha^2\right))x_k \le \widetilde{x_k} \le x_k.$$

We then add Laplacian noise $\mathsf{Lap}\left(\frac{8}{\beta'\varepsilon}\right)$ to $\widetilde{x_k}$ to form $\widehat{x_k}$. Since $x_k^2 \ge \mathcal{O}\left(\frac{\log n}{\beta'\alpha^2\varepsilon}\right)$, then with high probability, the Laplacian noise is at most an α^2 fraction of $\widehat{x_k}$. Hence with high probability,

$$(1-\alpha^2)x_k \le \widehat{x_k} \le x_k.$$ ◀

Unfortunately, Lemma 39 only provides guarantees for the coordinates of the medium frequency levels that are sampled. Thus, we still need to use Lemma 39 to show that a good estimator to the sizes of the medium frequency levels can be obtained from the estimates of the coordinates of the medium frequency levels that are sampled. In particular, we show that rescaling the empirical sizes of the medium frequency levels forms a good estimator to the actual sizes of the medium frequency levels.

Proof of Lemma 29. Suppose i is a β-important level. Then by Lemma 26 and a shifting of the index j, $b_i \ge \mathcal{O}\left(\frac{2^j\log^2 m}{\alpha^2\varepsilon^2}\right)$. Thus in S_j, the expected number of items E_j from level i is at least $\frac{\log^2 m}{\alpha^2\varepsilon^2}$ and the variance V_j is at most E_j. Hence by Chernoff bounds with $\mathcal{O}\left(\log n\right)$-wise limited independence, we have that the number of items N_j from level i satisfies

$$(1-\mathcal{O}\left(\alpha\right))b_i \le 2^j\cdot N_j \le (1+\mathcal{O}\left(\alpha\right))b_i,$$

with high probability. [11] show that due to the uniformly random chosen $\gamma \in (1/2, 1)$, we further have

$$(1-\mathcal{O}\left(\alpha\right))N_j \le (1+\mathcal{O}\left(\alpha\right))\widehat{b_i} \le (1+\mathcal{O}\left(\alpha\right))N_j,$$

with high probability. Since $s \le 2\log m$, then by a union bound over all $j \in [s]$, we have that with high probability, Algorithm 3 outputs $\widehat{b_i}$ such that

$$(1 - \mathcal{O}(\alpha))b_i \le \widehat{b_i} \le b_i. \qquad ◀$$

We now show that Algorithm 3 preserves differential privacy.

▶ **Lemma 40.** *Algorithm 3 is $\left(\frac{\varepsilon}{4}, \frac{\delta}{4}\right)$-differentially private for $\delta = \frac{1}{\text{poly}(m)}$.*

Proof. By Lemma 14, the sensitivity of PrivCountSketch is at most 2 and the failure probability is $\frac{1}{\text{poly}(m)}$. Thus by adding Laplacian noise $\mathsf{Lap}\left(\frac{8}{\beta'\varepsilon}\right)$ to $\widetilde{x_k}$, each estimated frequency is $\left(\frac{\beta'\varepsilon}{4}, \frac{\delta}{4\beta}\right)$-differentially private for $\delta = \frac{1}{\text{poly}(m)}$. Since PrivCountSketch with threshold β' can release at most $\frac{1}{\beta}$ estimated frequencies, then by Theorem 11, Algorithm 3 is $\left(\frac{\varepsilon}{4}, \frac{\delta}{4}\right)$. ◀

It remains to analyze the space complexity of Algorithm 3.

▶ **Lemma 41.** *Algorithm 3 uses space* $\text{mmc}(L)^2 \cdot \text{poly}\left(\frac{1}{\alpha}, \frac{1}{\varepsilon}, \log m\right)$.

Proof. The space complexity follows from running s instances of PrivCountSketch with threshold $\alpha^2\beta'$ and failure probability $\frac{1}{\text{poly}(m)}$, where $\beta' = \mathcal{O}\left(\frac{\alpha^2\beta\varepsilon^2}{\log^2 m}\right)$ and $\beta = \mathcal{O}\left(\frac{\alpha^5}{\text{mmc}(L)^2\log^5 m}\right)$. Since $s = \mathcal{O}(\log n)$ and we assume $n \le m$ so that $\mathcal{O}(\log n) = \mathcal{O}(\log m)$, then the space complexity follows. ◀

Proof of Lemma 31. Suppose i is a β-important level. Hence by a shifting of the index j in Lemma 26, we have that $b_i \ge \mathcal{O}\left(\frac{2^j\log^2 m}{\alpha^2\varepsilon^2}\right)$. Therefore, the expected number of items E_j from level i sampled in the substream S_j is at least $\frac{\log^2 m}{\alpha^2\varepsilon^2}$ and the variance V_j is at most E_j. Thus by Chernoff bounds with $\mathcal{O}(\log n)$-wise limited independence, the number of items N_j from level i satisfies

$$(1 - \mathcal{O}(\alpha))b_i \le 2^j \cdot N_j \le (1 + \mathcal{O}(\alpha))b_i,$$

with high probability. [11] show that due to the uniformly random chosen $\gamma \in (1/2, 1)$, we further have

$$(1 - \mathcal{O}(\alpha))N_j \le (1 + \mathcal{O}(\alpha))\widehat{b_i} \le (1 + \mathcal{O}(\alpha))N_j,$$

with high probability. Since $s \le 2\log m$ and $\mathsf{Lap}\left(\frac{8}{\varepsilon}\right)$ is at most an ε-fraction of $b_i \ge \mathcal{O}\left(\frac{2^j\log^2 m}{\alpha^2\varepsilon^2}\right)$ with high probability, then by a union bound over all $j \in [s]$, we have that with high probability, Algorithm 3 outputs $\widehat{b_i}$ such that

$$(1 - \mathcal{O}(\alpha))b_i \le \widehat{b_i} \le b_i. \qquad ◀$$

We then show that Algorithm 4 is differentially private.

▶ **Lemma 42.** *Algorithm 4 is $\left(\frac{\varepsilon}{4}, \frac{\delta}{4}\right)$-differentially private for $\delta = \frac{1}{\text{poly}(m)}$.*

Proof. Note that since each instance of PrivCountSketch$_j$ uses threshold $\beta'' := \mathcal{O}\left(\frac{\beta'\alpha^2\varepsilon^3}{\log^2 n}\right)$ on a stream S_j with $F_2(S_j) \le \frac{200\log m}{2^j}F_2(x)$, then for any $k \in [n]$ with $x_k \le \mathcal{O}\left(\frac{\log n}{\beta'\alpha^2\varepsilon}\right)$, we have that PrivCountSketch$_j$ outputs x_k exactly. Hence, at most two estimates of the sizes of the level sets $\widehat{b_i}$ can change, and then can change by at most one. Thus the sensitivity is at most 2, so it suffices to add Laplcian noise $\mathsf{Lap}\left(\frac{8}{\varepsilon}\right)$ to each estimate $\widehat{b_i}$ to obtain $\left(\frac{\varepsilon}{4}, \frac{\delta}{4}\right)$-differentially private for $\delta = \frac{1}{\text{poly}(m)}$. ◀

Finally, we argue the space complexity of Algorithm 4.

▶ **Lemma 43.** *Algorithm 4 uses space* $\mathrm{mmc}(L)^2 \cdot \mathrm{poly}\left(\frac{1}{\alpha}, \frac{1}{\varepsilon}, \log m\right)$.

Proof. Similar to Algorithm 3, the space complexity follows as a result of running s instances of PrivCountSketch with threshold $\alpha^2\beta'$ and failure probability $\frac{1}{\mathrm{poly}(m)}$, where $\beta' = \mathcal{O}\left(\frac{\alpha^2\beta\varepsilon^2}{\log^2 m}\right)$ and $\beta = \mathcal{O}\left(\frac{\alpha^5}{\mathrm{mmc}(L)^2 \log^5 m}\right)$. Since $s = \mathcal{O}(\log n)$ and we assume $n \le m$ so that $\mathcal{O}(\log n) = \mathcal{O}(\log m)$, then the space complexity follows. ◀

▶ **Theorem 44.** *Given a parameter* $M > 1$*, let* $\varepsilon, \alpha = \tilde{\Omega}\left(\left(\frac{M^2}{m}\right)^{\frac{1}{30}}\right)$. *There exists a* (ε, δ)*-differentially private algorithm that outputs a set* C*, for* $\delta = \frac{1}{\mathrm{poly}(m)}$. *From* C*, the* $(1+\alpha)$*-approximation to any norm with* maximum modulus of concentration *at most* M *can be computed, with probability at least* $1-\delta$. *The algorithm uses* $M^2 \cdot \mathrm{poly}\left(\frac{1}{\alpha}, \frac{1}{\varepsilon}, \log m\right)$ *bits of space.*

Proof. Note that from Lemma 34 and Lemma 35, the frequencies of the coordinates in the high frequency levels are well-approximated with high probability. Similarly, from Lemma 29 and Lemma 31, the sizes of the level sets of the medium and low frequency levels are well-approximated with high probability. Moreover, all the level sets are partitioned into the high, medium, or low frequency levels. We would like to say that by Lemma 23, these statistics are sufficient to recover a $(1+\alpha)$-approximation to any norm with *maximum modulus of concentration* at most M and so we achieve a $(1+\alpha)$-approximation to any norm with maximum modulus of concentration at most M that with high probability. Indeed, in an idealized process where $\xi^i \le \widehat{x_k} \le \xi^{i+1}$ if and only if k is sampled by the substream j assigned to level i and $\xi^i \le x_k < \xi^{i+1}$, Lemma 23 would show that we achieve a $(1+\alpha)$-approximation to any norm with maximum modulus of concentration at most M that with high probability. However, this may not always be the case because the frequency x_k may lie near the boundary of the interval $[\xi^i, \xi^{i+1})$ and the estimate $\widehat{x_k}$ may lie outside of the interval, in which case $\widehat{x_k}$ is used toward the estimation of some other level set. Thus, our algorithm randomizes the boundaries of the level sets by instead defining the level sets as $[\gamma\xi^i, \gamma\xi^{i+1})$ for some $\gamma \in (1/2, 1)$ chosen uniformly at random. Since we call PrivCountSketch with threshold at most $\alpha^2\beta'$, then the probability that item $k \in [n]$ is misclassified over the choice of γ is at most $\mathcal{O}\left(\alpha^2\beta'\right)$. Furthermore, if k in level set i is misclassified, it can only be classified into level set $i-1$ or $i+1$, causing at most an incorrect multiplicative factor of two. Then in expectation across all $k \in [n]$, the error due to the misclassification is at most an $\mathcal{O}\left(\alpha^2\beta'\right)$ fraction of the symmetric norm. Hence by Markov's inequality, the error due to the misclassification is at most an additive $\frac{\alpha}{2}$ fraction of the symmetric norm with probability at least 0.99. To obtain high probability of success, it then suffices to take the median across $\mathcal{O}(\log m)$ independent instances, finally showing correctness of our algorithm.

The private partitioning of the coordinates into the high, medium, and low frequency levels is $\frac{\varepsilon}{4}$-differentially private. Each of the three sets of statistics released by the high, medium, and low frequency levels are $\left(\frac{\varepsilon}{4}, \frac{\delta}{4}\right)$-differentially private, by Lemma 36, Lemma 40, and Lemma 42. Then (ε, δ)-differential privacy follows from the composition of differential privacy, i.e., Theorem 11.

Finally, the space complexity follows from Lemma 37, Lemma 41, and Lemma 43. ◀

We remark that our algorithm is presented as having unlimited access to random bits but is analyzed using $\mathcal{O}\left(\log m\right)$-wise independence, so it can be properly derandomized to provide the space guarantees without needing to store a large number of random bits. Alternatively, our algorithm can also be derandomized using Nisan's pseudorandom generator, which induces an extra multiplicative factor of $\mathcal{O}\left(\log m\right)$ in the space overhead [54].

Finally, we remark that the failure probability can be raised from $\delta=\frac{1}{\mathrm{poly}(m)}$ to arbitrarily $\delta>0$ using additional space overhead $\mathrm{polylog}\,\frac{1}{\delta}$, since the space dependency in each subroutine on the failure probability δ is $\mathrm{polylog}\,\frac{1}{\delta}$.

Testing Versus Estimation of Graph Properties, Revisited

Lior Gishboliner ✉
ETH Zürich, Switzerland

Nick Kushnir ✉
School of Mathematics, Tel Aviv University, Israel

Asaf Shapira ✉
School of Mathematics, Tel Aviv University, Israel

Abstract

A graph G on n vertices is ε-far from property $\mathcal{P}$ if one should add/delete at least εn^2 edges to turn G into a graph satisfying $\mathcal{P}$. A *distance estimator* for $\mathcal{P}$ is an algorithm that given G and $\alpha, \varepsilon > 0$ distinguishes between the case that G is $(\alpha - \varepsilon)$-close to $\mathcal{P}$ and the case that G is α-far from $\mathcal{P}$. If $\mathcal{P}$ has a distance estimator whose query complexity depends only on ε, then $\mathcal{P}$ is said to be *estimable*.

Every estimable property is clearly also testable, since testing corresponds to estimating with $\alpha = \varepsilon$. A central result in the area of property testing is the Fischer–Newman theorem, stating that an inverse statement also holds, that is, that every testable property is in fact estimable. The proof of Fischer and Newmann was highly ineffective, since it incurred a tower-type loss when transforming a testing algorithm for $\mathcal{P}$ into a distance estimator. This raised the natural problem, studied recently by Fiat–Ron and by Hoppen–Kohayakawa–Lang–Lefmann–Stagni, whether one can find a transformation with a polynomial loss. We obtain the following results.

- We show that if $\mathcal{P}$ is hereditary, then one can turn a tester for $\mathcal{P}$ into a distance estimator with an exponential loss. This is an exponential improvement over the result of Hoppen et. al., who obtained a transformation with a double exponential loss.
- We show that for every $\mathcal{P}$, one can turn a testing algorithm for $\mathcal{P}$ into a distance estimator with a double exponential loss. This improves over the transformation of Fischer–Newman that incurred a tower-type loss.

Our main conceptual contribution in this work is that we manage to turn the approach of Fischer–Newman, which was inherently ineffective, into an efficient one. On the technical level, our main contribution is in establishing certain properties of Frieze–Kannan Weak Regular partitions that are of independent interest.

2012 ACM Subject Classification Mathematics of computing → Approximation algorithms

Keywords and phrases Testing, estimation, weak regularity, randomized algorithms, graph theory, Frieze-Kannan Regularity

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.46

Category RANDOM

Related Version *Full Version*: `https://arxiv.org/abs/2305.05487`

Funding *Lior Gishboliner*: Supported by SNSF grant 200021_196965.
Asaf Shapira: Supported in part by ERC Consolidator Grant 863438 and NSF-BSF Grant 20196.

1 Introduction

1.1 Background on graph property testing

Property testers are fast randomized algorithms that can distinguish between objects satisfying some predetermined property $\mathcal{P}$ and those that are ε-far from satisfying $\mathcal{P}$. In most cases, ε-far means that an ε-proportion of the object's representation needs to be changed in order

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 46; pp. 46:1–46:18
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

to obtain a new object satisfying $\mathcal{P}$. Hence, testing for $\mathcal{P}$ is a relaxed version of the classical decision problem which asks to decide whether an object satisfies $\mathcal{P}$. In this paper we study properties of graphs in the so called *adjacency matrix model* (which is also sometimes referred to as the *dense graph model*). This is arguably one of the most well studied models in the area of property testing. The reader is referred to [20] for more background and references on property testing.

We now introduce the model of testing graph properties in the adjacency matrix model. A graph property $\mathcal{P}$ is a family of graphs closed under isomorphism. A graph G on n vertices is ε-far from $\mathcal{P}$ if one should add/delete at least εn^2 edges to turn G into a graph satisfying $\mathcal{P}$. If G is not ε-far from $\mathcal{P}$ then it is ε-close to $\mathcal{P}$. A *tester* for $\mathcal{P}$ is a randomized algorithm that given $\varepsilon > 0$ distinguishes with high probability (say, $2/3$) between graphs satisfying $\mathcal{P}$ and those that are ε-far from $\mathcal{P}$. We assume the algorithm can query for each $1 \leq i, j \leq n$ whether the input G contains the edge (i, j). The *edge query complexity*, denoted $Q(\varepsilon)$, of a tester is the number of edge queries it performs. If $\mathcal{P}$ has a tester whose edge query complexity depends only on ε (and is independent of n) then $\mathcal{P}$ is called *testable*. In what follows we will mainly work with *vertex query complexity* which is the smallest $q = q(\varepsilon)$ so that we can ε-test $\mathcal{P}$ by inspecting a subgraph of the input graph G, induced by a set of q randomly selected vertices. By a theorem of Goldreich and Trevisan [22] we know that $q(\varepsilon) \leq 2Q(\varepsilon) \leq q^2(\varepsilon)$. In most (but not all) discussions below we will not care much about these quadratic factors. In such cases we might use the term *query complexity* without mentioning if this is vertex or edge query complexity.

Property testing in the adjacency matrix model was first introduced by Goldreich, Goldwasser and Ron [21], who proved that every *partition property* (e.g. k-colorability and MAX-CUT) is testable. There are several general results guaranteeing that a graph property is testable [3, 10]. A result of this nature was obtained by Alon and Shapira [5] who proved that every hereditary[1] graph property is testable. Their proof applied Szemerédi's regularity lemma [35] (see also [33]), which is one of the most useful tools when studying properties of dense graphs. Using this tool comes with a hefty price, since the bounds one obtains when using the regularity lemma are of tower-type[2].

One of the central open (meta) problems related to testing graph properties is when can one turn an ineffective (e.g. one with tower-type bounds) result into an efficient one, preferably with polynomial bounds. While this is a quantitative question, what lies beneath it is in fact the following qualitative problem; when can we prove a testability result while avoiding Szemerédi's regularity lemma, either by giving a direct combinatorial argument or by using a weaker variant of the regularity lemma (e.g. the Frieze–Kannan regularity lemma [18] which we discuss below). For example, Rödl and Duke [31] used the regularity lemma in order to (implicitly) prove that k-colorability is testable. The tower-type bounds obtained in [31] were improved to polynomial in [21] using a direct argument which avoided the use of the regularity lemma. A specific central open problem, due to Alon and Fox [4], concerns hereditary properties, and asks which hereditary properties are testable with query complexity $\text{poly}(1/\varepsilon)$. A systematic investigation of this problem was carried out in [19].

1.2 Distance estimation

In the dense graph model we say that a graph's distance from $\mathcal{P}$ is α, if α is the smallest real so that G is α-close to $\mathcal{P}$. In other words, this is the minimum number of edges one should add/delete in order to obtain a graph satisfying $\mathcal{P}$, normalised by n^2. We denote

[1] A graph property is hereditary if it is closed under vertex removal. Some examples are being 3-colorable, being triangle-free and being induced H-free, for some fixed H.

[2] The tower function $\text{tower}(x)$ is a tower of exponents of height x.

this quantity by $\mathrm{dist}_{\mathcal{P}}(G)$. A *distance estimator* for $\mathcal{P}$ is a randomized algorithm that given $\alpha, \varepsilon > 0$ distinguishes with high probability (say, 2/3) between graphs that are $(\alpha - \varepsilon)$-close to $\mathcal{P}$ and those that are α-far from $\mathcal{P}$. If for every α, ε there is a distance estimator for $\mathcal{P}$ whose query complexity depends only on ε, then $\mathcal{P}$ is said to be *estimable*. Note that testing $\mathcal{P}$ is equivalent to distance estimation with $\alpha = \varepsilon$, hence this notion is at least as strong as testability.

Distance estimation was first studied in [30] and has since been studied in various other settings such as distributions [7], strings [6], sparse graphs [11, 13, 28], boolean functions [1, 9], error correcting codes [23, 26] and image processing [8]. It is known that in certain settings, there are testable properties which are not estimable [15]. One of the central and most unexpected results in the area of graph property testing is the Fischer–Newman theorem [16], which states that in the setting of graphs, every testable property is also estimable. As with several of the main results in this area, the proof in [16] relied on Szemerédi's regularity lemma [35] and thus resulted in a tower-type loss when transforming a tester for $\mathcal{P}$ into a distance estimator for $\mathcal{P}$. Returning to the discussion in the last paragraph of the previous subsection, it is natural to ask if one can improve the transformation of [16] and turn a tester for $\mathcal{P}$ into a distance estimator with a polynomial loss.

1.3 New results concerning hereditary graph properties

As we mentioned in the previous subsection, the family of hereditary graph properties has been extensively studied within the setting of graph property testing. The fact that every hereditary property is testable follows from the following statement, where we use $ind(F, G)$ to denote the probability that a random mapping $\varphi : V(F) \to V(G)$ is an injective induced homomorphism. [3]

▶ **Lemma 1** (Induced Removal Lemma, [5]). *For every $\varepsilon > 0$ and every hereditary $\mathcal{P}$, there exists $M = M_1(\varepsilon, \mathcal{P})$, $\delta = \delta_1(\varepsilon, \mathcal{P}) > 0$ and $n_0 = n_1(\varepsilon, \mathcal{P})$ such that if a graph G on $n \geq n_0$ vertices is ε-far from $\mathcal{P}$ then there is a graph $F \notin \mathcal{P}$ with $|V(F)| \leq M$ such that $ind(F, G) \geq \delta$.*

The first version of the above lemma was obtained by Alon, Fischer, Krivelevich and Szegedy [2] who proved it when $\mathcal{P}$ can be characterized using a finite number of forbidden induced subgraphs. The lemma was proved in full generality by Alon and Shapira [5]. Alternative proofs were later obtained by Lovász and Szegedy [27], Conlon and Fox [12] and Borgs et al. [10]. It was also extended to the setting of hypergraphs by Rödl and Schacht [32].

Note that it follows immediately from Lemma 1 that every hereditary property is testable with vertex query complexity

$$q(\varepsilon) = \max\{n_0, M/\delta\} \ . \tag{1}$$

Indeed, the algorithm samples a set X of q vertices, queries about all pairs within X, and then accepts if and only if the graph on X satisfies $\mathcal{P}$. If G satisfies $\mathcal{P}$ then the algorithm clearly answers correctly (with probability 1). If G is ε-far from $\mathcal{P}$, then by Lemma 1 a random M-tuple of vertices spans an induced copy of a graph $F \notin \mathcal{P}$ with probability at least δ. Hence, a sample of size M/δ contains an induced copy of F with probability at least 2/3, thus guaranteeing that the sample of vertices does not satisfy $\mathcal{P}$ (since $\mathcal{P}$ is hereditary). Recall that [22] proved that if $\mathcal{P}$ is testable, then it is testable using an algorithm as above. Hence, the bounds in Lemma 1 more or less determine the query complexity of testing a

[3] A mapping $\varphi : V(F) \to V(G)$ is an induced homomorphism if $uv \in E(F)$ if and only if $\varphi(u)\varphi(v) \in E(G)$.

hereditary $\mathcal{P}$. This raises the following natural problem, introduced by Hoppen et al. [25, 24] and by Fiat and Ron [14], asking if it is possible to estimate every hereditary $\mathcal{P}$ with (roughly) the same query complexity with which it can be tested as in (1).

▶ **Problem 2.** *Determine if every hereditary graph property $\mathcal{P}$ is estimable with query complexity*

$$n_0 \cdot M/\delta \ ,$$

where $M = M_1(\varepsilon', \mathcal{P})$, $\delta = \delta_1(\varepsilon', \mathcal{P})$, $n_0 = n_1(\varepsilon', \mathcal{P})$ are given by Lemma 1 with $\varepsilon' = \mathrm{poly}(\varepsilon)$.

▶ Remark 3. There are hereditary graph properties (e.g. triangle-freeness) for which the best known bounds for M and δ in Lemma 1 are of tower-type. One can argue that in such cases there is little difference between the tower(M/δ) bounds given by [16] and those suggested by Problem 2. However, we should emphasize that for many of these properties (e.g. triangle-freeness) the tower-type bounds are not known to be tight (indeed, the best known lower bounds are just slightly super polynomial). Perhaps more importantly, there are numerous hereditary graph properties for which it is known that both M and δ in Lemma 1 are polynomial in ε (e.g. k-colorability, being an interval graph or being a line graph; see the detailed discussion in [19]). For all these properties, Problem 2 suggests a poly$(1/\varepsilon)$ bound, versus the tower$(1/\varepsilon)$ bound given by [16].

Problem 2 was studied by Hoppen et al. [25, 24]. Their main result was that every hereditary $\mathcal{P}$ is estimable with query complexity $2^{\mathrm{poly}((1/\delta)^{M^2}, \log n_0)}$. Our first main result is the following exponential improvement of this result, making a significant step towards resolving Problem 2.

▶ **Theorem 4.** *Every hereditary $\mathcal{P}$ is estimable with query complexity*

$$2^{\mathrm{poly}(M/\delta, \log n_0)} \ ,$$

where $M = M_1(\varepsilon/2, \mathcal{P})$, $\delta = \delta_1(\varepsilon/2, \mathcal{P})$ and $n_0 = n_1(\varepsilon/2, \mathcal{P})$ are the parameters of Lemma 1.

▶ Remark 5. In all known cases, the best bounds in Lemma 1 are such that $\log n_0 \ll 1/\delta$, hence the upper bound of [25] is $2^{(1/\delta)^{O(M^2)}}$ while the one in Theorem 4 is $2^{\mathrm{poly}(M/\delta)}$.

In almost all cases, results concerning testing of dense graphs rely on combinatorial statements which imply trivial algorithms. For example, the algorithm for testing a hereditary property $\mathcal{P}$ is trivial once we have Lemma 1 at our disposal. In sharp contrast, many estimation results involve sampling a set of vertices and then carrying out a highly non-trivial computation over this sample. This is certainly the case in the present paper, see the proofs of Lemmas 14 and 15. However, thanks to a well known sampling trick [21], one can transfer any estimation result into a combinatorial statement. For example, this trick gives the following corollary of Theorem 4.

▶ **Corollary 6.** *Set $q = 2^{\mathrm{poly}(M/\delta, \log n_0)}$ as in Theorem 4. Then*

$$\Pr_X \left[|\mathrm{dist}_{\mathcal{P}}(G[X]) - \mathrm{dist}_{\mathcal{P}}(G)| \le \varepsilon \right] \ge 2/3 \ ,$$

where the probability is over randomly selected subsets X of q vertices from G, and $G[X]$ is the graph induced by G on X.

It is interesting to note that with Corollary 6 at hand, we can now go back and reprove Theorem 4 using the "trivial/natural" algorithm which samples a set of q vertices X, computes $\text{dist}_{\mathcal{P}}(G[X])$, and then states that G is $(\alpha - \varepsilon)$-close to $\mathcal{P}$ if $\text{dist}_{\mathcal{P}}(G[X]) \leq \alpha - \varepsilon/2$ and is otherwise α-far from $\mathcal{P}$.

Our proof of Theorem 4 actually gives the bound $2^{\text{poly}(M/\varepsilon\delta, \log n_0)}$. One can speculate that $\text{poly}(M/\varepsilon\delta) = \text{poly}(M/\delta)$ since in all known cases δ is at best polynomial in ε, and in many cases much smaller. In order to formally be able to remove the dependence on ε from our bound, we prove the following proposition, where $\mathcal{P}$ is *trivial* if either $\mathcal{P}$ contains all graphs or if it contains finitely many graphs. The proof of this proposition relies on a subtle application of Ramsey's theorem.

▶ **Proposition 7.** *The following holds for every non-trivial hereditary property $\mathcal{P}$. If $q(\varepsilon)$ denotes the vertex query complexity of $\mathcal{P}$ then for every small enough ε, we have*

$$M/\delta \geq q(\varepsilon) \geq \Omega(1/\varepsilon) , \tag{2}$$

where $M = M_1(\varepsilon, \mathcal{P})$ and $\delta = \delta_1(\varepsilon, \mathcal{P})$ are the constants of Lemma 1.

The left inequality above follows from (1). Observe that the lower bound on $q(\varepsilon)$ is best possible since it is tight when $\mathcal{P}$ is the property of having no edges (in which case $q(\varepsilon) = O(1/\varepsilon)$). The proof of the proposition will appear in the journal version of the paper.

It is of course natural to study Problem 2 also for specific hereditary properties. A natural problem of this type is whether every hereditary $\mathcal{P}$ that is testable with query complexity $\text{poly}(1/\varepsilon)$ is also estimable with query complexity $\text{poly}(1/\varepsilon)$. Such an investigation was initiated recently by Fiat and Ron [14] who proved such a statement for many natural hereditary properties such as Chordality and not containing an induced path on 4 vertices.

1.4 New results concerning general graph properties

Given the discussion above, the following problem seems natural.

▶ **Problem 8.** *Determine if every property $\mathcal{P}$ that is testable with vertex query complexity $q(\varepsilon)$, is estimable with query complexity $q(\varepsilon')$ for some $\varepsilon' = \text{poly}(\varepsilon)$.*

Prior to this work, the only result concerning general graph properties $\mathcal{P}$ was the transformation of Fischer and Newman [16] which turns a testing algorithm for a graph property $\mathcal{P}$ with query complexity $q(\varepsilon)$ into a distance estimator with query complexity $\text{tower}(q(\varepsilon/2))$. Using the tools we develop in order to obtain Theorem 4, we also obtain the following improved bound.

▶ **Theorem 9.** *If $\mathcal{P}$ is testable with query complexity $q(\varepsilon)$ then it is estimable with query complexity $2^{\text{poly}(1/\varepsilon)\cdot 2^{q(\varepsilon/2)}}$.*

We would like to argue at this point that since any "natural" property satisfies $q(\varepsilon) \geq \log(1/\varepsilon)$ the above bound can be written as $\exp(\exp(\text{poly}(q(\varepsilon/2))))$. In order to formally make such a claim, we prove the following variant of Proposition 7, in which $\mathcal{P}$ is *unnatural* if there is ε_0 so that the following holds for every $0 < \varepsilon < \varepsilon_0$ and $n \geq n_0(\varepsilon)$: either every n-vertex graphs is ε-close to $\mathcal{P}$, or every n-vertex graph does not belong to $\mathcal{P}$. If $\mathcal{P}$ is not unnatural then it is (naturally) *natural.*

▶ **Proposition 10.** *Let $\mathcal{P}$ be a natural property and let $q(\varepsilon)$ be its vertex query complexity, and $Q(\varepsilon)$ be its edge query complexity. Then*

$$Q(\varepsilon) = \Omega(1/\varepsilon) . \tag{3}$$

In particular, $q(\varepsilon) = \Omega(\sqrt{1/\varepsilon})$.

The "in particular" part above follows directly from the Goldreich–Trevisan [22] theorem mentioned earlier. Observe that the general lower bound given in (3) is best possible since it is tight when $\mathcal{P}$ is the property of having no edges, where $Q(\varepsilon) = O(1/\varepsilon)$. The proof of the proposition will appear in the journal version of the paper.

1.5 Main technical contributions and comparison to previous approaches

Summary of previous approaches

The main reason why Szemerédi's regularity lemma is so useful when studying testing/estimation problems is that an ε-regular partition of a graph G determines (approximately) the values of $ind(F, G)$ for all small F. Hence, on a very high level, the way one can estimate a graph's distance to a hereditary property $\mathcal{P}$ is to take a *single* ε-regular partition of G (one such exists by the regularity lemma) and then try to modify this partition using the smallest possible number of edge modifications, so that the new partition "predicts" that there are no induced copies of graphs $F \notin \mathcal{P}$ in the new graph G'. A key "continuity" feature one has to use at this stage is that if G has a regular partition with certain edge densities between the clusters of the partition, and one would like to modify G so that in the new graph G' one has a regular partition where the edge densities between the clusters will change on average by γ, then one can achieve this by modifying $(\gamma + o(1))n^2$ edges of G. Fischer and Newman [16] critically relied on the fact that regular partitions in the sense of Szemerédi have this continuity property. The approach of [16] was ineffective since although a regular partition has constant size (i.e., depending only on ε), this constant has tower-type dependence on ε. We should point that one of the key novel ideas of [16] was a method for obtaining the densities of a single Szemerédi partition of the input G.

The way Hoppen et al. [25, 24] managed to improve upon [16] (for hereditary $\mathcal{P}$) was by first observing that in order to estimate $ind(F, G)$ for all small F, one does not need the full power of Szemerédi's regularity lemma. Instead, one can use the weak regularity lemma of Frieze and Kannan [17] which involves constants that are only exponential in ε. The main reason why their proof gave a doubly exponential bound is that Frieze–Kannan regular partitions do not (seem to) have the same continuity feature we mentioned in the previous paragraph with respect to Szemerédi partitions. To overcome this, Hoppen et al. [25, 24] introduced a sophisticated method that somehow combines working with Frieze–Kannan regular partitions in some parts of the proof, together with vertex partitions that have no regularity[4] features at all (these are sometimes called GGR partitions, after [21]) in other parts of the proof.

Our main technical contribution

Our main technical contribution in this paper establishes that Frieze-Kannan weak regular partitions "almost" satisfy the same continuity feature we mentioned above with respect to Szemerédi partitions. What we show is that one can indeed efficiently modify a Frieze–Kannan partition if one starts with a partition with guarantees slightly stronger than those of Frieze–Kannan, and one is content with ending with a usual Frieze–Kannan partition.

[4] Working with partitions that have no regularity requirements has the advantage that they trivially have the continuity property. Indeed, if we want to change the edge density between two sets A, B by γ we just add/remove $\gamma|A||B|$ edges. Needless to say that working with such partitions has various disadvantages resulting from their lack of regularity features.

See Lemma 28 for the precise statement, whose proof relies on a randomized-rounding-type argument. With the above continuity feature at hand, we can now go back to the Fischer–Newman approach and turn it into an effective one, by taking full advantage of the Frieze–Kannan lemma. One additional hurdle we need to overcome in order to make sure we only incur an exponential loss in our proof, is a method for finding a Frieze–Kannan partition of a graph using a constant number of queries. Here we introduce a variant of the method of Fischer–Newman tailored for Frieze–Kannan partitions, see Lemma 14. The main tools we develop for proving Theorem 4 turn out to be also applicable for proving Theorem 9. The reason why in Theorem 9 we have a double exponential loss is that it is not enough to estimate $ind(F, G)$ for a single F (as in Theorem 4 thanks to Lemma 1) but we instead need to control $ind(F, G)$ for all graphs F of order $q(\varepsilon)$. We expect Lemmas 14 and 28 to be applicable in future studies related to efficient testing and estimation of graph properties.

Paper overview

In Section 2 we introduce the two main lemmas in the paper, and show how they imply Theorem 4. These lemmas are proved in Sections 3 and 4. In Section 5 we prove Theorem 9. We prove Proposition 7 at the end of Section 2 and Proposition 10 at the end of Section 5. We use $a = \text{poly}(x)$ to denote the fact that a is bounded from above (or below, when $0 < x < 1$) by x^d for some fixed d, which is independent of n or ε. Also, when we say that "for every $a = \text{poly}(x)$ there is $b = \text{poly}(x)$" we mean that for every d there is d' so that if $a \le x^d$ then there is a $b \le x^{d'}$.

2 The Key Lemmas and Proof of Theorem 4

Our goal in this section is to state Lemmas 14 and 15 and then use them to derive Theorem 4. We prove these lemmas in Sections 3 and 4. At the end of this section we also prove Proposition 7.

To state Lemmas 14 and 15 we need some definitions. We first recall that given a graph $G = (V, E)$, an equipartition $A = \{V_1, \dots, V_k\}$ of $V(G)$ is a partition satisfying $||V_i| - |V_j|| \le 1$. Given a graph G and subsets $X, Y \subseteq V(G)$, we use $e(X, Y)$ to denote the number of edges between X and Y, and $d(X, Y) = e(X, Y)/|X||Y|$ to denote the *density* between them.

▶ **Definition 11** (Signature). *For an equipartition $A = \{V_1, \dots, V_t\}$ of $V(G)$, a (γ, ε)-signature of A is a sequence of reals $S = (\eta_{i,j})_{1 \le i < j \le t}$, such that $|d(V_i, V_j) - \eta_{i,j}| \le \gamma$ for all but at most $\varepsilon\binom{t}{2}$ of the pairs $i < j$. A (γ, γ)-signature is referred to as γ-signature.*

▶ **Definition 12** (Index of a partition). *For an equipartition A of a graph $V(G)$ into t sets, we define the* index *of A to be*

$$ind(A) = \frac{1}{t^2} \sum_{1 \le i < j \le t} d^2(V_i, V_j) .$$

▶ **Definition 13** (Final partition). *For a function $f : \mathbb{N} \to \mathbb{N}$ and $\gamma > 0$, we say that an equipartition A of G consisting of t sets is (f, γ)-final if there exists no equipartition B of $V(G)$ with at least t and up to $f(t)$ sets for which $ind(B) \ge ind(A) + \gamma$.*

The above notion of a final partition is useful since (as we show later) every graph has such a partition and furthermore, we can design an algorithm for finding a signature of one such partition of an input G. The first key lemma leading to the proof of Theorem 4 does exactly that.

▶ **Lemma 14.** *For every $k, \zeta > 0$, and every $\gamma = \mathrm{poly}(\zeta)$ and $f_\zeta(x) = x \cdot 2^{\mathrm{poly}(1/\zeta)}$, there are $q = q_{14}(\zeta, k)$, $N = N_{14}(\zeta, k)$ and $T = T_{14}(\zeta, k)$ so that*

$$q, N, T \leq \mathrm{poly}(k) \cdot 2^{\mathrm{poly}(1/\zeta)}$$

and such that the following holds. If G is a graph on at least N vertices then there is an algorithm making at most q queries to G, computing with probability at least $\frac{2}{3}$ a γ-signature of an (f_ζ, γ)-final partition of G into at least k and at most T sets.

We prove the above lemma is Section 3. The following is the second key lemma, which we prove in Section 4. In its statement we use the notion *ind*(F, G) which we defined before the statement of Lemma 1. What it roughly states, is that having a signature of G (with good parameters) is enough for estimating G's distance to satisfying $\mathcal{P}$.

▶ **Lemma 15.** *For every $h, \varepsilon, \delta > 0$, there are $\gamma = \gamma_{15}(h, \varepsilon, \delta)$, $s = s_{15}(h, \varepsilon, \delta)$ and $f_{15}^{(h,\varepsilon,\delta)} : \mathbb{N} \to \mathbb{N}$ so that*

$$\gamma = \mathrm{poly}(\varepsilon\delta/h), \quad s = \mathrm{poly}(h/\varepsilon\delta), \quad f_{15}(x) = x \cdot 2^{\mathrm{poly}(h/\varepsilon\delta)}$$

and the following holds. For every family $\mathcal{H}$ of graphs, each on at most h vertices, there exists a deterministic algorithm, that receives as an input a γ-signature S of an (f_{15}, γ)-final partition A into $t \geq s$ sets of a graph G with $n \geq N_{15}(h, \varepsilon, \delta, t) = \mathrm{poly}(t) \cdot 2^{\mathrm{poly}(h/\varepsilon\delta)}$ vertices, and distinguishes given any α between the following two cases:

(i) G is $(\alpha - \varepsilon)$ close to some graph G' for which $ind(H, G') = 0$ for every $H \in \mathcal{H}$.

(ii) G is α-far from every G' for which $ind(H, G') < \delta$ for every $H \in \mathcal{H}$.

Proof (of Theorem 4). Suppose $\mathcal{P}$ is a hereditary graph property, and let $\alpha, \varepsilon > 0$. Lemma 1 with inputs $\varepsilon/2$ and $\mathcal{P}$ asserts that there are

$$h = M_1(\varepsilon/2), \quad \delta = \delta_1(\varepsilon/2), \quad n_0 = n_1(\varepsilon/2)\ ,$$

so that if a graph G on at least n_0 vertices is $\varepsilon/2$-far from $\mathcal{P}$, then $ind(H, G) \geq \delta$ for some $H \notin \mathcal{P}$ with $|V(H)| \leq h$. We need to describe an algorithm making $2^{\mathrm{poly}(h/\delta, \log n_0)}$ queries to G and distinguishes with probability at least $2/3$ between the case that G is $(\alpha - \varepsilon)$-close to $\mathcal{P}$ and the case that G is α-far from $\mathcal{P}$. Set

$$\gamma = \gamma_{15}(h, \varepsilon/2, \delta), \quad s = s_{15}(h, \varepsilon/2, \delta), \quad f = f_{15}^{(h,\varepsilon/2,\delta)}\ .$$

Finally, set $\zeta = \delta\varepsilon/h$ and observe that

$$\gamma = \gamma_{15}(h, \varepsilon/2, \delta) = \mathrm{poly}(\varepsilon\delta/2h) = \mathrm{poly}(\zeta)\ ,$$

that

$$f(x) = f_{15}^{(h,\varepsilon/2,\delta)}(x) = x \cdot 2^{\mathrm{poly}(2h/\varepsilon\delta)} = x \cdot 2^{\mathrm{poly}(1/\zeta)}\ ,$$

that

$$s = s_{15}(h, \varepsilon/2, \delta) = \mathrm{poly}(2h/\varepsilon\delta) = \mathrm{poly}(1/\zeta)\ .$$

Also, note that by Proposition 7 we have $\mathrm{poly}(1/\zeta) = \mathrm{poly}(h/\delta)$. Let q, N, T be the parameters given by Lemma 14 when applied with $k = s$, and ζ, γ, f defined above. (note that γ and f satisfy the assumptions of the lemma). Lemma 14 then guarantees that $q, N, T \leq 2^{\mathrm{poly}(1/\zeta)} \leq 2^{\mathrm{poly}(h/\delta)}$.

If G has less than N vertices then we can just ask about all the edges of G and answer correctly with probability 1. The number of queries is then at most $N^2 \leq 2^{\mathrm{poly}(h/\delta)}$ as needed. If G has more than N vertices then we can use the algorithm of Lemma 14 with the parameters k, ζ, γ, f defined above. The algorithm makes at most $q \leq 2^{\mathrm{poly}(h/\delta)}$ queries and with probability at least $2/3$ returns a γ-signature S of an equipartition of G into $s \leq t \leq T$ sets that is (f, γ)-final. Let

$$N' = N_{15}(h, \varepsilon/2, \delta, T) = \mathrm{poly}(T) \cdot 2^{\mathrm{poly}(h/\varepsilon\delta)} = 2^{\mathrm{poly}(h/\delta)} .$$

Again, if G has less than $N_1 = \max\{N', n_0\}$ vertices then we can just ask about all the edges of G and answer correctly with probability 1. The number of queries is then at most $(N_1)^2 \leq 2^{\mathrm{poly}(h/\delta, \log n_0)}$ as needed.

Suppose then that G has at least $\max\{N, N_1\}$ vertices. Let $\mathcal{H}$ be the family of graph on at most h vertices which do not satisfy $\mathcal{P}$. Then we can now run the algorithm of Lemma 15 on the signature S, with respect to $\mathcal{H}$, with $\alpha' = \alpha - \varepsilon/2$ and with $\varepsilon/2$ instead of ε (note that we chose the parameters with $\varepsilon/2$). If the algorithm says that case (i) holds (namely that G is $(\alpha' - \varepsilon/2)$-close to some G' with $ind(H, G') = 0$ for every $H \in \mathcal{H}$) then we declare that G is $(\alpha - \varepsilon)$-close to $\mathcal{P}$, and if the algorithm says that case (ii) holds (namely that G is α'-far from every G' with $ind(H, G') < \delta$ for every $H \in \mathcal{H}$) then we declare that G is α-far from $\mathcal{P}$.

Let us prove the correctness of the above algorithm. If G is $(\alpha - \varepsilon)$-close to $\mathcal{P}$ then it is $(\alpha - \varepsilon)$-close to a graph G' satisfying $ind(H, G') = 0$ for every $H \in \mathcal{H}$. Since $\alpha - \varepsilon = \alpha' - \varepsilon/2$ the algorithm will say that case (i) holds, hence the algorithm answers correctly in this case. Suppose now that G is α-far from $\mathcal{P}$. Then any G' that is α'-close to G must be $\varepsilon/2$-far from $\mathcal{P}$. Hence, by Lemma 1 in any such G' we have $ind(H, G') \geq \delta$ for at least one $H \in \mathcal{H}$. We conclude that G is α'-far from every G' satisfying $ind(H, G') < \delta$ for every $H \in \mathcal{H}$. Hence, the algorithm of Lemma 15 will say that case (ii) holds , so our algorithm will answer correctly in this case as well. ◀

3 Proof of Lemma 14

The proof is similar to one in [16]. What they have shown is that for every f, γ, one can find an (f, γ)-final partition with a constant, albeit huge tower-type, query complexity. What we do here is show that for restricted types of f, one can get a much better bound. To do this we also need to rely on a recent result of [34].

3.1 Preliminary lemmas

In this subsection we describe some preliminary lemmas that will be used in the next subsection in which we prove Lemma 14. We will need the following Chernoff-type large deviation inequality.

▶ **Lemma 16.** *Suppose $X_1, \ldots, X_m$ are m independent Boolean random variables, so that for every $1 \leq i \leq m$ we have $\Pr[X_i = 1] = p_i$. Let $E = \sum_{i=1}^m p_i$. Then, $\Pr[|\sum_{i=1}^m X_i - E| \geq \theta m] \leq 2e^{-2\theta^2 m}$.*

▶ **Definition 17** (Partition Properties). *A partition property is a triple $\pi = (s, \ell, u)$ where s is an integer (the size of the partition property), ℓ is a vector of $\binom{s}{2}$ reals $0 \leq \alpha_{i,j} \leq 1$ for each $1 \leq i < j \leq s$, and u is a vector of $\binom{s}{2}$ reals $0 \leq \beta_{i,j} \leq 1$ for each $1 \leq i < j \leq s$. We say that a graph G satisfies π if there is an equipartition $\{V_1, \ldots, V_s\}$ of $V(G)$, such that $\alpha_{ij} \leq d(V_i, V_j) \leq \beta_{ij}$ for every $1 \leq i < j \leq s$.*

Given s and μ we use $\pi(s,\mu)$ to denote the family of partition properties π of size s in which every $\alpha_{i,j}$ and $\beta_{i,j}$ is an integer multiple of μ (so $\pi(s,\mu)$ contains $\{0,\mu,2\mu,\dots,1\}^{2\binom{s}{2}}$ partition properties). Finally, define $\Pi(t,\mu)=\bigcup_{s\le t}\pi(s,\mu)$.

Note that each π as above is one of the partition properties studied in [21], where it was shown that they are μ-testable with query complexity $(1/\mu)^{\mathrm{poly}(s)}$. This was improved recently to $\mathrm{poly}(s/\mu)$ in [34]. The next lemma states that with (roughly) the same query complexity we can in fact *simultaneously* test all properties in $\Pi(t,\mu)$.

The proof of the next lemma will appear in the journal version of the paper.

▶ **Lemma 18.** *For every t and $\mu>0$ there is $q=q_{18}(t,\mu)=\mathrm{poly}(t/\mu)$ satisfying the following. There is a randomized algorithm, that given a graph G, makes q queries to G and with probability at least $2/3$, for every $\pi\in\Pi(t,\mu)$, distinguishes between the case that G satisfies π and the case that G is μ-far from π.*

Proof (of Lemma 14): Given k,ζ,γ and f_ζ as in the statement of the lemma, we define $T_0=k$ and for $i\ge 1$ define $T_i=f_\zeta(T_{i-1})$. Now set the following parameters.

$$N=N_{14}(k,\zeta)=T_{2/\gamma}=k\cdot 2^{\mathrm{poly}(1/\zeta)},\quad T=T_{14}(k,\zeta)=T_{2/\gamma}=k\cdot 2^{\mathrm{poly}(1/\zeta)}\ ,$$

and

$$t=f_\zeta(T)=k\cdot 2^{\mathrm{poly}(1/\zeta)},\quad \mu=\frac{\gamma}{48(f_\zeta(T))^2}=\frac{1}{\mathrm{poly}(k)\cdot 2^{\mathrm{poly}(1/\zeta)}}\ .$$

We now describe the algorithm for finding a signature S satisfying the requirement of the lemma. For what follows let $\pi'(s,\mu)$ be the partition properties in which $\beta_{i,j}=\alpha_{i,j}+\mu$ for every $1\le i<j\le s$. Also for each $\pi\in\pi'(s,\mu)$ define the index of π to be $ind(\pi)=\frac{1}{t^2}\sum_{1\le i<j\le t}\alpha_{ij}^2$. In the Step-1 we run the algorithm of Lemma 18 with the parameters t,μ defined above. This is the only randomized part of the algorithm. In the Step-2 of the algorithm we do the following.

(i) For each $k\le s\le t$ set $M(s)=\max_\pi ind(\pi)$ where the maximum is taken over all $\pi\in\pi'(s,\mu)$ which the algorithm of Step-1 accepted.

(ii) Let $s^\star$ be the smallest number in $\{k,\dots,T\}$ such that $M(s')\le M(s^\star)+\frac{3}{4}\gamma$ for every $s'\in\{s^\star+1,\dots,f_\zeta(s^\star)\}$. If there exists such an $s^\star$, output the signature $S^\star$ that achieves the maximum over $s^\star$. Otherwise, the algorithm fails.

Note that the query complexity of the algorithm is $q=q_{18}(t,\mu)=\mathrm{poly}(t/\mu)=\mathrm{poly}(k)\cdot 2^{\mathrm{poly}(1/\zeta)}$, as needed. Also, Lemma 18 guarantees that Step-1 of the above described algorithm succeeds with probability at least $2/3$. It thus remains to show that assuming this event holds, Step-2 of the algorithm will return an (f_ζ,γ)-final partition. First of all note that if it succeeds then it returns a partition of size at least k and at most T, as required.

The proof that if Step-1 succeeded, then Step-2 returns an (f_ζ,γ)-final partition is identical to the proof of Claim 5.5 in [16], so we give a sketch of the proof. First, the reader might be wondering why every graph necessarily has an (f_ζ,γ)-final partition as in the statement of the lemma. Let us actually explain why every G has an $(f_\zeta,\gamma/2)$-final partition, while using the definitions we introduced above. Start from an arbitrary equipartition A_0 of G into $T_0=k$ sets, and let $ind_0=ind(A_0)$ denote the index of A_0 as in Definition 12. If A_0 is $(f_\zeta,\gamma/2)$-final then we are done. If not, then there must be another partition A_1 of G with at least T_0 and at most $f(T_0)=T_1$ parts, with index $ind(A_1)\ge ind(A_0)+\gamma/2$. Since $0\le ind(A)\le 1$ for every equipartition, we see that this process will eventually end up with a partition A of size $k \leq s \leq T$ so that all partitions of G into at least s and at most $f(s)$ parts have index less than $ind(A) + \gamma/2$. But this means that A is $(f_\zeta, \gamma/2)$-final. Note that we thus get that G has a $(f_\zeta, \gamma/2)$-final partition A of size $s \leq T$.

Let us now explain how to turn the above existential proof into a proof of correctness of the algorithm describe earlier. Let $M_G(s)$ denote the largest index of an equipartition of G of size s. First we claim that for every $k \leq s \leq t$,

$$M(s) - \gamma/8 \leq M_G(s) \leq M(s) + \gamma/8. \tag{4}$$

For the second inequality in (4), let A be an equipartition with s parts such that $M_G(s) = ind(A)$. Let $\pi \in \pi'(s, \mu)$ be the partition property obtained from A by rounding down the densities to the closest integer multiple of μ. Then we have $|ind(A) - ind(\pi)| \leq 3\mu \leq \gamma/8$. Hence, $M(s) \geq ind(\pi) \geq ind(A) - \gamma/8 = M_G(s) - \gamma/8$.

For the first inequality in (4), let $\pi \in \pi'(s, \mu)$ be a partition property which the algorithm accepted and such that $M(s) = ind(\pi)$. Then G must be μ-close to π (as otherwise π should have been rejected). Let G' be a graph μ-close to G that satisfies π, and let A be the vertex partition of G' witnessing that G' satisfies π. Note that when turning G into G', for each pair of parts of A, we change the density between this pair by at most μs^2. Hence, in G, the partition property π is a $2\mu s^2$-signature of A (here and in what follows, we view π as a signature). So $|ind(A) - ind(\pi)| \leq 6\mu s^2 \leq \gamma/8$, using our choice of μ. Now, $M_G(s) \geq ind(A) \geq ind(\pi) - \gamma/8 = M(s) - \gamma/8$. This proves (4).

It follows from the existential proof above that there is $k \leq s^\star \leq T$ and an equipartition A of G into $s^\star$ parts which is $(f_\zeta, \gamma/2)$-final. We can assume that $M_G(s^\star) = ind(A)$, because the equipartition satisfying this must also be final. We have $M_G(s') \leq M_G(s^\star) + \gamma/2$ for every $s^\star \leq s' \leq f_\zeta(s^\star)$. By (4), this implies that $M(s') \leq M(s^\star) + 3\gamma/4$ for every $s^\star \leq s' \leq f_\zeta(s^\star)$. So the algorithm will return a partition.

Note that the algorithm does not necessarily return the same signature/partition-property as above π that is μ-close to the above partition A. The reason for the algorithm to choose a different partition is that there might be another partition of size s with a larger index (which is of course also (f_ζ, γ)-final) or there might be an $s^* < s$ with the same properties, or there might be other partitions with the same index. However, one can invert the reasoning in the previous paragraph and show that if a π is returned then it must be the γ-signature of an (f_ζ, γ)-final partition. ◀

4 Proof of Lemma 15

4.1 Preliminary lemmas

In this subsection we describe some preliminary lemmas that will be used in the next subsection in which we prove Lemma 15. We start with introducing the Frieze–Kannan regularity lemma [17, 18]. We first state their notion of γ-regularity.

▶ **Definition 19** (Frieze–Kannan Regularity [18]). *Let $G = (V, E)$ be a graph and $A = \{V_1 \ldots, V_k\}$ be an equipartition of $V(G)$. For a subset $X \subseteq V$ and $1 \leq i \leq k$ denote $X_i = X \cap V_i$. We say that A is γ-Frieze–Kannan-regular if:*

$$d^A_\square(G) := \max_{S,T \subseteq V} \frac{1}{n^2} \left| \sum_{i,j \in [k]^2} \Big(d(S_i, T_j) - d_{ij} \Big) |S_i||T_j| \right| < \gamma \tag{5}$$

Roughly speaking, a partition A is γ-Frieze–Kannan-regular, or γ-FK-regular for short, if we can estimate the number of edges between large sets S, T from the intersection sizes $S \cap V_i$ and $T \cap V_i$. We will also need the following slightly stronger notion of weak regularity that was introduced in [29].

▶ **Definition 20** (Frieze–Kannan Regularity* [29]). *In the setting of Definition 19, we say that A is γ-Frieze–Kannan Regular* if:*

$$d_\square^{\star A}(G) := \max_{S,T\subseteq V} \frac{1}{n^2} \sum_{i,j\in[k]^2} \left| d(S_i, T_j) - d_{ij} \right| |S_i||T_j| < \gamma \tag{6}$$

The translation between these two notions will be crucial in Lemma 28 below. Suppose $A = \{V_1, \dots, V_k\}$ is an equipartition of $V(G)$. Then an equipartition $B = \{W_1, ..., W_\ell\}$ of $V(G)$ is said to *refine* A if each $W_i \in B$ is contained in some $V_j \in A$. The following lemma is proved in [29] using a simple variant of the original proof of Frieze and Kannan [18].

▶ **Lemma 21** (Frieze–Kannan Weak Regularity Lemma [18, 29]). *For every k_0 and $\gamma > 0$ there is $T = T_{21}(k_0, \gamma) = k_0 \cdot 2^{\mathrm{poly}(1/\gamma)}$ so that the following holds for every graph G on at least T vertices. If A is an equipartition of $V(G)$ into at most k_0 sets, then there is a refinement B of A into at most T sets such that $d_\square^{\star B}(G) < \gamma$.*

Let us now extend the definition of $d_\square$ to distance between pairs of weighted graph, where a weighted graph R is a complete graph, so that every edge (i, j) is assigned a weight $0 \le R(i, j) \le 1$.

If R, R' are two weighted graphs on n vertices then we define

$$d_1(R, R') = \frac{1}{n^2} \sum_{i<j} |R(i, j) - R'(i, j)| \,, \tag{7}$$

and

$$d_\square(R, R') = \max_{\alpha,\beta} \frac{1}{n^2} \left| \sum_{i<j} \alpha(i)\beta(j)(R(i, j) - R'(i, j)) \right| \,, \tag{8}$$

where the maximum is taken over all functions $\alpha, \beta : [n] \to [0, 1]$.

▶ **Definition 22** ($ind(F, R)$). *Let R be a weighted graph on $[k]$ and let φ be an injective function $\varphi : V(F) \to [k]$. We set*

$$ind_\varphi(F, R) = \prod_{i<j\in E(F)} R(\varphi(i), \varphi(j)) \prod_{i<j\notin E(F)} (1 - R(\varphi(i), \varphi(j)))$$

In the case of φ not being injective, we define

$$ind_\varphi(F, R) = 0$$

Denoting by Φ the set of functions from $V(F)$ to $[k]$, we define

$$ind(F, R) = \frac{1}{|\Phi|} \sum_{\varphi\in\Phi} ind_\varphi(F, R) \,. \tag{9}$$

Note that we can think of a signature $S = (\eta_{i,j})_{1\le i<j\le t}$ as a weighted graph on t vertices. This means that for a pair of signatures S, S' we can define $d_1(S, S')$ and $d_\square(S, S')$ as in (7) and (8) respectively, and we can also define $ind(F, S)$ as in (9). We will need the following lemmas from [25]. The proof of the next two lemmas will appear in the journal version of the paper.

▶ **Lemma 23.** *Suppose R, R' are two weighted graphs on n vertices, and H is a graph on h vertices. Then for any $\gamma \geq d_{\square}(R, R')$ and $n \geq \frac{2}{\gamma}$, we have $|ind(H, R) - ind(H, R')| \leq 2h^2 \cdot \gamma$*

Given a graph G on n vertices, and an equipartition $A = \{V_1, \dots, V_k\}$, we define the graph G_A on $V(G)$ to be the weighted graph with weights $G_A(u, v) = d(V_i, V_j)$ for every $u \in V_i$ and $v \in V_j$. Let S_A be the 0-signature of A, that is, the weighted graph on k vertices with $S(i, j) = d(V_i, V_j)$. Observe that if k divides n (so all sets of A are of equal size) then $ind(H, G_A)$ is almost the same as $ind(H, S_A)$. It is not hard to see that for general equipartitions these quantities do not differ my much.

▶ **Lemma 24.** *Given a graph G on n vertices, and an equipartition $A = \{V_1, \dots, V_k\}$, let G_A and S_A be defined as above. Then $|ind(H, G_A) - ind(H, S_A)| \leq \frac{2h^2}{k} + \frac{2kh}{n}$ for every graph H on h vertices.*

We now combine the above facts to conclude that a signature of a γ-FK-partition of a graph gives a good approximation of $ind(H, G)$. The proof of the next lemma will appear in the journal version of the paper.

▶ **Lemma 25.** *For every h, k and $\delta > 0$ there are*

$$\gamma = \gamma_{25}(h, \delta) = \text{poly}(\delta/h), \quad r = r_{25}(h, \delta) = \text{poly}(h/\delta), \quad N = N_{25}(h, k, \delta) = \text{poly}(hk/\delta) ,$$

so that if G is a graph on at least N vertices, and A is a γ-FK-regular partition of G with at least r and up to k parts, then for every γ-signature S of A, we have $|ind(H, G) - ind(H, S)| \leq \delta$ for every H on h vertices.

▶ **Definition 26** (Extension)**.** *Given a signature $S = (\eta_{ij})_{1 \leq i < j \leq t}$ of an equipartition A, and a refinement $B = \{W_1, \dots, W_s\}$ of A, the extension of S to B is the sequence $S' = (\eta'_{ij})_{1 \leq i < j \leq s}$ defined as $\eta'_{i,j} = \eta_{k,l}$ if there exist $k \neq l$ such that $W_i \subseteq V_k$ and $W_j \subseteq V_l$, and setting $\eta'_{i,j} = 0$ if W_i and W_j are both subsets of the same V_k.*

The proof of the next claim will appear in the journal version of the paper.

▷ Claim 27. For every ε and s there exists $r = r_{27}(\varepsilon) = \text{poly}(1/\varepsilon)$ and $N = N_{27}(\varepsilon, s) = \text{poly}(s/\varepsilon)$ so that the following holds for every pair of graphs G, G' on the same set of $n \geq N$ vertices. If G, G' are α-close and S, S' are γ, γ'-signatures of G, G' respectively, of the same equipartition A of the vertex set of G, G' into $s \geq r$ sets, then $d_1(S, S') \leq \alpha + \varepsilon + 2(\gamma + \gamma')$.

The proof of the next lemma will appear in the journal version of the paper.

▶ **Lemma 28.** *For every ε and t there exists $\gamma = \gamma_{28}(\varepsilon) = \text{poly}(\varepsilon)$ and $N = N_{28}(t, \varepsilon) = \text{poly}(t/\varepsilon)$, so that for every graph G on $n \geq N$ vertices, if S is a γ-signature of a γ-FK-regular* partition A of G with t sets, then for every signature S' satisfying $d_1(S, S') \leq \delta$ for some δ, there is a graph G' that is $(\delta + \varepsilon)$-close to G, so that A is an ε-FK-regular partition of G', and S' is an ε-signature of A.*

We will also need the following lemmas.

▶ **Lemma 29** ([2] Lemma 3.7)**.** *For every ε, t there exists $\gamma = \gamma_{29}(\varepsilon) = \text{poly}(\varepsilon)$ and $N = N_{29}(t, \varepsilon) = \text{poly}(t/\varepsilon)$ satisfying the following. Assume A is an equipartition into s sets of a graph G with $n \geq N$ vertices, and that B is a refinement of A into at most t sets. Assume further that S is any γ-signature of A, and that T is its extension to B. If B satisfies $ind(B) \leq ind(A) + \gamma$, then T is an ε-signature for B.*

▶ **Lemma 30** ([16] Lemma 6.6). *For every ε, t there exists $N = N_{30}(t, \varepsilon) = \mathrm{poly}(t/\varepsilon)$ so that for every equipartition A of G with $n \geq N$ vertices into s sets, and every refinement B of A into at most t sets, $ind(B) \geq ind(A) - \varepsilon$.*

The next observation is implicit in the proof of the Frieze–Kannan Regularity Lemma (i.e. Lemma 21). The main step of the proof involves showing that if A is an equipartition of G into t parts and A is not ε-FK-regular*, then A has a refinement B into $k \leq 16t/\varepsilon^4$ sets so that $ind(B) \geq ind(A) + \frac{\varepsilon^4}{2}$ (see, e.g., the proof of Theorem 1.1 in [33] and the proof of Theorem 6 in [29]).

▶ **Lemma 31.** *For every $\varepsilon > 0$ there exists $\gamma = \gamma_{31}(\varepsilon) = \mathrm{poly}(\varepsilon)$ and $f = f_{31}^{(\varepsilon)} : \mathbb{N} \to \mathbb{N}$ satisfying $f(x) = \mathrm{poly}(1/\varepsilon) \cdot x$ and such that every (f, γ)-final partition of a graph is also ε-FK-regular*.*

The proof of the next lemma will appear in the journal version of the paper.

▶ **Lemma 32.** *For every s and $\varepsilon > 0$ there are $\gamma = \gamma_{32}(\varepsilon)$, $T = T_{32}(s, \varepsilon)$, $f = f_{32}^{(\varepsilon)}$ and $N = N_{32}(\varepsilon, s)$ so that*

$$\gamma = \mathrm{poly}(\varepsilon), \quad T = s \cdot 2^{\mathrm{poly}(1/\varepsilon)}, \quad f(x) = x \cdot 2^{\mathrm{poly}(1/\varepsilon)}, \quad N = \mathrm{poly}(s) \cdot 2^{\mathrm{poly}(1/\varepsilon)}$$

and the following holds. Suppose G has at least N vertices and A is an (f, γ)-final partition of G into at most s sets and that S is a γ-signature of A. Then for every G' on the same vertex set of G, there exists a refinement A' of A into $t \leq T$ sets so that

(i) *A' is an ε-FK-regular* partition of G'.*

(ii) *Every refinement A'' of A with $t \leq T$ sets (and in particular A'), is an ε-FK-regular* partition of G.*

(iii) *For every refinement A'' of A with $t \leq T$ sets, the extension S'' of S (in the sense of Definition 26) with respect to A'' is an ε-signature of A'' with respect to G (note that A' is such an A'').*

4.2 Proof of Lemma 15

Given h, ε and δ we first choose

$$\gamma_0 = \min\{\varepsilon/10, \gamma_{25}(h, \delta/6), \gamma_{28}(\min\{\varepsilon/2, \gamma_{25}(h, \delta/6)\})\} = \mathrm{poly}(\varepsilon\delta/h) ,$$

and then define

$$\gamma = \gamma_{32}(\gamma_0) = \mathrm{poly}(\varepsilon\delta/h), \quad s = \max\{r_{25}(h, \delta/6), r_{27}(\varepsilon/10), 20h^2/\delta\} = \mathrm{poly}(h/\varepsilon\delta) ,$$

$$f(x) = f_{32}^{(\gamma_0)}(x) = x \cdot 2^{\mathrm{poly}(1/\gamma_0)} = x \cdot 2^{\mathrm{poly}(h/\varepsilon\delta)} ,$$

to be the constants and function in the statement of Lemma 15, noting that they satisfy the guarantees of that lemma. Given t as in the statement of Lemma 15, we set

$$T = T_{32}(t, \gamma_0)$$

and define

$$N = \max\{N_{25}(h, T, \delta/6), N_{27}(\varepsilon/10, T), N_{32}(\gamma_0, s), N_{28}(t, \gamma_0)\} = \mathrm{poly}(t) \cdot 2^{\mathrm{poly}(h/\varepsilon\delta)} ,$$

to be the constant in Lemma 15.

Given a family of graphs $\mathcal{H}$ on at most h vertices, we define a family of signatures as follows

$$\mathcal{C}_{\delta,\mathcal{H},T} = \{C : |C| \leq T \text{ and } ind(H,C) \leq \delta/2 \text{ for every } H \in \mathcal{H}\}\ .$$

In order for $\mathcal{C}_{\delta,\mathcal{H},T}$ to be finite, we only put in it signatures C with edge weights $\eta_{i,j}$ that are integer multiples of $\beta = \min\{\varepsilon/10, \delta/10h^2\}$. Intuitively, this is the set of signatures "certifying" (hence C) that a graph with that signature is close to being induced $\mathcal{H}$-free. We also define $\mathcal{S}_T$ to be the set of all signatures on up to T parts, that are extensions[5] of S. Intuitively, these are the signatures one can obtain by refining A into at most T sets (recall that the crucial point is that the algorithm only has access to S and not to G).

Suppose now that we are given a γ-signature S of some (f,γ)-final (with the above defined f,γ) partition A of a graph G, so that S has $t \geq s$ parts and G has at least N vertices. The algorithm checks if there are $S' \in \mathcal{S}_T$ and $C \in \mathcal{C}_{\delta,\mathcal{H},T}$ satisfying $d_1(S',C) \leq \alpha - \frac{\varepsilon}{2}$. If there is such a pair, the algorithm says that case (i) holds, otherwise it says that case (ii) holds. We now prove the correctness of the algorithm.

Proof of first direction

Suppose there is a graph G' which is $(\alpha - \varepsilon)$-close to G, and satisfies $ind(H,G') = 0$ for every $H \in \mathcal{H}$. We will show that the algorithm will declare that case (i) holds.

Recall that A is an (f,γ)-final partition of G into $t \geq s$ sets and that S is a γ-signature of A. By Lemma 32, there exists a refinement A' of A into at most T sets so that A' is γ_0-FK-regular* for both G **and** G'. Moreover, denoting by S' the corresponding extension of S to A', we have that S' is a γ_0-signature of A' with respect to G. Note that $S' \in \mathcal{S}_T$. By the choice of γ_0, this implies that A' is $\gamma_{25}(h,\delta/6)$-FK-regular* for both G and G', and that S' is a $\frac{1}{10}\varepsilon$-signature of A' with respect to G. Let C' be the 0-signature of A' over G'. Lemma 25 (using A' and G') implies that $|ind(H,G') - ind(H,C')| \leq \delta/6$ for all $H \in \mathcal{H}$. Thus $ind(H,C') \leq \delta/6$ for all $H \in \mathcal{H}$. Clearly there is a signature C of size C' so that all of C's weights are constant multiples of β and $d_1(C',C) \leq \beta$. Since $d_\square(C',C) \leq d_1(C',C) \leq \delta/10h^2$ we infer from Lemma 23 (applied on $\frac{\delta}{10h^2}$, as $s \geq \frac{20h^2}{\delta}$) that $ind(H,C) \leq \delta/6 + \delta/5 < \delta/2$ for all $H \in \mathcal{H}$, so $C \in \mathcal{C}_{\delta,\mathcal{H},T}$. In addition, by Claim 27 (since A' has at least $r_{27}(\varepsilon/10)$ parts and assuming that n is large enough), we infer that $d_1(S',C) \leq \alpha - \frac{\varepsilon}{2}$ (since G and G' are $(\alpha - \varepsilon)$-close and $d_1(C,C') \leq \varepsilon/10$). Thus, S' and C provide a witness that the algorithm will indeed declare that case (i) holds.

Proof of second direction

Suppose the algorithm declares that case (i) holds. We show that in this case there is a graph G', which is α-close to G, and satisfies $ind(H,G') < \delta$ for all $H \in \mathcal{H}$.

Indeed, if the algorithm declared that case (i) holds then there are signatures $S' \in \mathcal{S}_T$ and $C \in \mathcal{C}_{\delta,\mathcal{H},T}$ satisfying $d_1(S',C) \leq \alpha - \frac{\varepsilon}{2}$. As $S' \in \mathcal{S}_T$, there is a refinement A' of A, so that S' is the extension of S according to A'. Lemma 32 (regarding A' as a possible

[5] Note that strictly speaking, an extension per Definition 26 must be relative to a partition A and its refinement B, while here we only have the signature S. So what we mean here is that if one takes some graph that has a partition A whose 0-signature is S, then $\mathcal{S}_T$ is the family of all signatures that one obtains by taking all refinements of A into at most T sets, and then taking the extension of S to these refinements. Of course we do not need any graph in order to produce $\mathcal{S}_T$; we just break the "parts" of S into a total of at most T new "parts", and then define the densities $\eta'_{i,j}$ between the new vertices as in Definition 26.

refinement of A with respect to G) asserts that S' is a γ_0-signature of A' (with respect to G), which by the choice of γ_0 means that it is a $\gamma_{28}(\min\{\frac{\varepsilon}{2}, \gamma_{25}(h, \delta/6)\})$-signature for A' with respect to G. Now, Lemma 28 (applied with A' as the γ_0-FK-regular* partition of G, and with S' as S and C as S') implies that there is a graph G' that is $(\alpha - \frac{\varepsilon}{2} + \frac{\varepsilon}{2})$-close to G, namely α-close to G, and for which C is a $\gamma_{25}(h, \delta/6)$-signature of A', which in turn is $\gamma_{25}(h, \delta/6)$-FK-regular over G' . Lemma 25 implies that $|ind(H, G') - ind(H, C)| \le \delta/6$ for all $H \in \mathcal{H}$. Thus, $ind(H, G') < \delta/2 + \delta/6 < \delta$ for all $H \in \mathcal{H}$ as required. Hence we have found the required G'.

5 Proof of Theorem 9

The proof of Theorem 9 is very similar to that of Theorem 4. In order to assist the reader who is already familiar with the proof of Theorem 4, we mention in several places where certain lemmas are analogous to lemmas we introduced in one of the previous sections. The idea is the following: by a theorem of Goldreich and Trevisan [22], every testable property is testable by a canonical tester, which samples a set of vertices of size $q = q_{\mathcal{P}}(\varepsilon)$ and accepts/rejects based on the graph induced by these q vertices. Hence the acceptance/rejection of the algorithm only depends on the number of induced copies in G of graphs on q vertices. Hence, turning a graph into a graph satisfying $\mathcal{P}$ is equivalent to turning it into a graph with a certain number of copies of certain graphs on q vertices. As evident, this is very similar to the case of Theorem 4 where we wanted to have a very small number of copies of graphs not in $\mathcal{P}$. The reason why there is an additional exponential factor is that we need to control the number of induced copies of *all* graphs on q vertices.

We now state the key lemmas, which are variants of lemmas we used in the proof of Theorem 4.

▶ **Definition 33.** *Given two distributions μ and ν over a finite family $\mathcal{H}$ of combinatorial structures, their variation distance is defined as:* $|\mu - \nu| = \frac{1}{2}\sum_{H\in\mathcal{H}} |\Pr_\mu(H) - \Pr_\nu(H)|$

▶ **Lemma 34.** *If two distributions μ and ν over a finite family $\mathcal{H}$ of combinatorial structures satisfy $|\mu - \nu| \le \delta$, then for any set $A \subset \mathcal{H}$ we have* $|\Pr_\mu(A) - \Pr_\nu(A)| \le \delta$

▶ **Lemma 35.** *Suppose that μ and ν are two probability distributions over graphs with set of vertices $\{v_1, \dots, v_q\}$, where each edge v_iv_j is independently chosen to be an edge with probability $\mu_{i,j}$ and $\nu_{i,j}$ respectively. If $|\mu_{i,j} - \nu_{i,j}| \le \varepsilon/\binom{q}{2}$ for every $1 \le i < j \le q$, then the variation distance between μ and ν is bounded by ε.*

▶ **Definition 36** (q-statistic)**.** *The q-statistic of a graph G is the probability distribution over all (labeled) graphs with q vertices that result from picking at random q distinct vertices of G and considering the induced subgraph. For a given graph H we denote the probability for obtaining H when drawing a graph according to the q-statistic by* $\Pr_G(H)$*.*

▶ **Definition 37.** *For an equipartition $A = \{V_1, \dots, V_t\}$ of G, and a signature $S = (\eta_{i,j})_{1\le i<j\le t}$ of A, the perceived q-statistic according to S is the following distribution* $\Pr_S$ *over labeled graphs with q vertices $v_1, \dots, v_q$. Start by choosing a uniformly random sequence without repetitions of indices $i_1, \dots, i_q$ from $1, \dots, t$. Then, independently, take every v_kv_l for $k < l$ to be an edge with probability η_{i_k,i_l}if $i_k < i_l$ and with probability η_{i_l,i_k}if $i_l < i_k$. Then* $\Pr_S(H)$ *is defined as the probability that the resulting labeled graph equals H.*

The following lemma will replace Lemma 1 in the proof of Theorem 9.

▶ **Lemma 38** (see [22])**.** *If there is an ε-test for a graph property $\mathcal{P}$ that makes $Q = Q(\varepsilon)$ edge queries, then there exists an appropriate family $\mathcal{H}$ of labeled graphs on $q = 2Q$ vertices such that any graph G which satisfies $\mathcal{P}$, satisfies also $\Pr_G(\mathcal{H}) \geq \frac{2}{3}$, and any graph G that is ε-far from satisfying $\mathcal{P}$, satisfies also $\Pr_G(\mathcal{H}) < \frac{1}{3}$.*

We now introduce a variant of Lemma 25 that is suited for the proof of Theorem 9. The proof of the lemma will appear in the journal version of the paper.

▶ **Lemma 39.** *For every q, ε there are $\gamma = \gamma_{39}(q,\varepsilon)$, $r = r_{39}(q,\varepsilon)$ so that*

$$\gamma = \operatorname{poly}(\varepsilon \cdot 2^{-q^2}), \quad r = \operatorname{poly}(1/\varepsilon \cdot 2^{q^2})$$

and for every γ-signature S of a γ-FK-regular equipartition A into $t \geq r$ sets, of a graph G on $n \geq N_{39}(q,\varepsilon,t) = \operatorname{poly}(t/\varepsilon)2^{\operatorname{poly}(q)}$ vertices, we have $|\Pr_S - \Pr_G| \leq \varepsilon$, where $\Pr_G$ is the q-statistic and $\Pr_S$ is the perceived q-statistic according to S.

We now introduce a variant of Lemma 15 that is suited for the proof of Theorem 9. The proof of the lemma will appear in the journal version of the paper.

▶ **Lemma 40.** *For every q and ε there exist $\gamma = \gamma_{40}(q,\varepsilon)$, $s = s_{40}(q,\varepsilon)$ and $f_{40}^{(q,\varepsilon)} : \mathbb{N} \to \mathbb{N}$, such that*

$$\gamma = \operatorname{poly}(\varepsilon \cdot 2^{-q^2}), \quad s = \operatorname{poly}\left(\frac{2^{q^2}}{\varepsilon}\right), \quad f_{40}^{(q,\varepsilon)}(x) = x \cdot 2^{\operatorname{poly}\left(\frac{2^{q^2}}{\varepsilon}\right)}$$

with the following property. For every family $\mathcal{H}$ of graphs with q vertices, there exists a deterministic algorithm, that receives as an input a γ-signature S of an (f,γ)-final partition A into $t \geq s$ sets of a graph G with $n \geq N_{40}(q,\varepsilon,t) = t \cdot 2^{\operatorname{poly}(1/\varepsilon)\cdot 2^{\operatorname{poly}(q)}}$ vertices and distinguishes given any α between the following two cases:

(i) *G is $(\alpha - \varepsilon)$-close to some graph G' for which $\Pr_{G'}(\mathcal{H}) \geq \frac{2}{3}$.*
(ii) *G is α-far from every G' for which $\Pr_{G'}(\mathcal{H}) \geq \frac{1}{3}$.*

Theorem 9 is derived from Lemmas 14 and 40, similarly to how Theorem 4 is derived from Lemmas 14 and 15. This will appear in the journal version of the paper.

References

1 N. Alon, B. Chazelle, S. Comandur, and D. Liue. Estimating the distance to a monotone function. *Random Struct Algorithms*, 31:371–383, 2007.

2 N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy. Efficient testing of large graphs. *Combinatorica*, 20:451–476, 2000.

3 N. Alon, E. Fischer, I. Newman, and A. Shapira. A combinatorial characterization of the testable graph properties: it's all about regularity,. *SIAM J Comput*, 39:143–167, 2009.

4 N. Alon and J. Fox. Easily testable graph properties. *Combin Probab. Comput*, 24:646–657, 2015.

5 N. Alon and A. Shapira. A characterization of the (natural) graph properties testable with one-sided error. *SIAM J Comput.*, 37:1703–1727, 2008.

6 T. Batu, F. Ergun, J. Kilian, A. Magen, S. Raskhodnikova, R. Rubinfeld, and R. Sami. A sublinear algorithm for weakly approximating edit distance. *ACM Comput Surv.*, 35:316–324, 2003.

7 T. Batu, L. Fortnow, R. Rubinfeld, W. Smith, and P. White. Testing closeness of discrete distributions. *Journal of the ACM*, 60:1–25, 2013.

8 P. Berman, M. Murzabulatov, and S. Raskhodnikova. Tolerant testers of image properties. *Proc. of ICALP*, pages 1–14, 2016.

9 E. Blais, C. Canonne, T. Eden, A. Levi, and D. Ron. Tolerant junta testing and the connection to submodular optimization and function isomorphism. *ACM Trans Comput. Theory*, 11, 2019.

10 C. Borgs, J. Chayes, L. Lovász, V. T. Sós, B. Szegedy, and K. Vesztergombi. Graph limits and parameter testing. *Proc. of STOC*, pages 261–270, 2006.

11 A. Campagna, A. Guo, and R. Rubinfeld. Local reconstructors and tolerant testers for connectivity and diameter. *Proc. of APPROX*, pages 411–424, 2013.

12 D. Conlon and J. Fox. Bounds for graph regularity and removal lemmas. *Geom Funct. Anal*, 22:1191–1256, 2012.

13 T. Eden, R. Levi, and D. Ron. Testing bounded arboricity. *Proc. of SODA*, pages 2081–2092, 2018.

14 N. Fiat and D. Ron. On efficient distance approximation for graph properties. *Proc. of SODA*, pages 1618–1637, 2021.

15 E. Fischer and L. Fortnow. Tolerant versus intolerant testing for boolean properties. *Theory Comput.*, 2:173–183, 2006.

16 E. Fischer and I. Newman. Testing versus estimation of graph properties. *SIAM J Comput.*, 37:482–501, 2007.

17 A. Frieze and R. Kannan. The regularity lemma and approximation schemes for dense problems. *Proc. of FOCS*, pages 12–20, 1996.

18 A. Frieze and R. Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19:175–220, 1999.

19 L. Gishboliner and A. Shapira. Removal lemmas with polynomial bounds. *Proc. of STOC*, pages 510–522, 2017.

20 O. Goldreich. *Introduction to Property Testing.* Cambridge University Press, 2017.

21 O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45:653–750, 1998.

22 O. Goldreich and L. Trevisan. Three theorems regarding testing graph properties. *Random Struct Algorithms*, 23:23–57, 2003.

23 V. Guruswami and A. Rudra. Tolerant locally testable codes. *Proc. of RANDOM*, pages 306–317, 2005.

24 C. Hoppen, Y. Kohayakawa, R. Lang, H. Lefmann, and H. Stagni. Estimating parameters associated with monotone properties,. *Combin. Probab. Comput.*, 29(2020):616–632, 2016.

25 C. Hoppen, Y. Kohayakawa, R. Lang, H. Lefmann, and H. Stagni. On the query complexity of estimating the distance to hereditary graph properties. *SIAM J Discret. Math.*, 35:1238–1251, 2021.

26 S. Kopparty and S. Saraf. Tolerant linearity testing and locally testable codes. *Proc. of RANDOM*, pages 601–614, 2009.

27 L. Lovász and B. Szegedy. Szemerédi's lemma for the analyst. *Geom. Funct. Anal*, 17:252–270, 2007.

28 S. Marko and D. Ron. Distance approximation in bounded-degree and general sparse graphs. *ACM Trans. Algorithms*, 5:22:1–22:28, 2009.

29 G. Moshkovitz and A. Shapira. A sparse regular approximation lemma. *Trans. Amer. Math. Soc.*, 371:6779–6814, 2019.

30 M. Parnas, D. Ron, and R. Rubinfeld. Tolerant property testing and distance approximation. *J. Comput. Syst. Sci.*, 72:1012–1042, 2006.

31 V. Rödl and R. Duke. On graphs with small subgraphs of large chromatic number. *Graphs and Combinatorics*, 1:91–96, 1985.

32 V. Rödl and M. Schacht. Generalizations of the removal lemma. *Combinatorica*, 29:467–501, 2009.

33 V. Rödl and M. Schacht. Regularity lemmas for graphs. *Fete of Combinatorics and Computer Science, vol. 20 series, Bolyai Soc Math. Stud*, pages 287–325, 2010.

34 A. Shapira and H. Stagni. A tight bound for testing partition properties. 2023.

35 E. Szemerédi. Regular partitions of graphs, 1978. In *Proc. Colloque Inter CNRS (J. C. Bermond, J. C. Fournier, M. Las Vergnas and D. Sotteau, eds.).*

Efficient Interactive Proofs for Non-Deterministic Bounded Space

Joshua Cook
Department of Computer Science, University of Texas Austin, TX, USA

Ron D. Rothblum
Faculty of Computer Science, Technion, Haifa, Israel

Abstract

The celebrated **IP** = **PSPACE** Theorem gives an efficient interactive proof for any bounded-space algorithm. In this work we study interactive proofs for *non-deterministic* bounded space computations. While Savitch's Theorem shows that nondeterministic bounded-space algorithms can be simulated by deterministic bounded-space algorithms, this simulation has a quadratic overhead. We give interactive protocols for nondeterministic algorithms directly to get faster verifiers.

More specifically, for any non-deterministic space S algorithm, we construct an interactive proof in which the verifier runs in time $\tilde{O}(n+S^2)$. This improves on the best previous bound of $\tilde{O}(n+S^3)$ and matches the result for *deterministic* space bounded algorithms, up to polylog(S) factors.

We further generalize to *alternating* bounded space algorithms. For any language L decided by a time T, space S algorithm that uses d alternations, we construct an interactive proof in which the verifier runs in time $\tilde{O}(n+S\log(T)+Sd)$ and the prover runs in time $2^{O(S)}$. For $d=O(\log(T))$, this matches the best known interactive proofs for deterministic algorithms, up to polylog(S) factors, and improves on the previous best verifier time for nondeterministic algorithms by a factor of $\log(T)$. We also improve the best prior verifier time for *unbounded* alternations by a factor of S.

Using known connections of bounded alternation algorithms to bounded depth circuits, we also obtain faster verifiers for bounded depth circuits with *unbounded* fan-in.

2012 ACM Subject Classification Theory of computation → Interactive proof systems

Keywords and phrases Interactive Proofs, Alternating Algorithms, AC0[2], Doubly Efficient Proofs

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.47

Category RANDOM

Related Version *Full Version*: https://eccc.weizmann.ac.il/report/2023/097/

Funding *Joshua Cook*: Supported by the National Science Foundation under grant number 2200956.
Ron D. Rothblum: Funded by the European Union (ERC, FASTPROOF, 101041208). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

Acknowledgements Thanks to Dana Moshkovitz for introducing the authors of this paper, and Anna Gál for finding the references relating low depth circuits to alternating algorithms.

1 Introduction

Interactive proofs, introduced by Goldwasser Micali and Rackoff [22], are proof systems that enable a prover to convince a verifier of the truth of a given statement. The interaction proceeds in rounds where in each round the prover sends a message and the verifier responds. Crucially, in every round the verifier uses randomness that the prover cannot predict. At the end of the interaction the verifier either accepts or rejects the statement. We require that the honest prover convinces the verifier to accept true statements with high probability (and

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 47; pp. 47:1–47:22
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

in fact, in most[1] protocols with probability 1) and that no prover, even a computationally unbounded one, can convince the verifier to accept a false statement other than with some small probability.

One of the most celebrated results in complexity theory is that **IP** = **PSPACE** [26, 34]. That is, the set of languages with polynomial space algorithms is exactly the set of languages with interactive protocols whose verifiers run in polynomial time. Interactive proofs have been prolific throughout other areas of complexity theory, including circuit lower bounds [33, 28], pseudorandomness from uniform assumptions [42], and has also been very influential in other proof systems, such as MIPs [5], PCPs [6, 14, 4, 3], and IOPs [7, 32].

The **IP** = **PSPACE** result can be generalized to any deterministic bounded space computation. For a space S deterministic algorithm, the interactive protocols with the fastest verifiers [9, 40] have a time $\tilde{O}(n + S^2)$ verifier and time $2^{O(S)}$ prover, where $\tilde{O}$ hides polylog(S) factors.[2]

In this work we study interactive proofs for more general forms of bounded space computations: *non-deterministic bounded space* and *alternating bounded space*. Recall that a non-deterministic space S algorithm is a space S Turing machine that gets in addition *read-once* access to a witness (which can be as long as 2^S). For example, the complexity class **NL** refers to non-deterministic logarithmic-space algorithms. Alternating algorithms are a generalization of nondeterministic algorithms that can "alternate" quantifiers. The prior best protocols [9] for space S nondeterministic algorithms have verifier time $\Omega(n + S^3)$, which is an S factor slower than the best verifiers for deterministic algorithms. See Table 1 for a more complete comparison with prior works.

1.1 Our Results

Our main result is an improved interactive proof for alternating algorithms. We start by highlighting a special-case of this result for nondeterministic bounded-space algorithms. We construct interactive proofs for space S nondeterministic algorithms whose verifier runs in time $\tilde{O}(n + S^2)$, matching the time bound for deterministic verifiers (up to polylog(S) factors). Broadly, our techniques combine the recent verifier efficient interactive proofs for bounded space by Cook [9], with an efficient interactive proof for $\mathbf{AC}^0_{\oplus}$ circuits of Goldreich and Rothblum [20], and an improved derandomization through random walks on expander graphs.

The new interactive proof for non-deterministic bounded space is a special case of a more general result that we show for alternating bounded space algorithms. To state the result precisely, we first set up the notation. Let $\mathbf{ATISP}_d[T, S]$ be the set of languages decided by a simultaneous time T, space S and d alternation algorithm. Alternating algorithms have 3 tapes, a read only input tape containing the input, a read once input containing a witness, and a work tape. Only the work tape is limited to have space S. The input tape is read only, but can be read many times. The witness tape can have T symbols on it, but must be read sequentially and each symbol can only be read once. The witness can be thought of as being separated into d segments, each with a different quantifier. The change of quantifier is called an alternation. For example, nondeterministic algorithms have $d = 1$ since they only use existential quantifiers.

[1] By [17], probability 1 can always be achieved, but that reduction has a significant cost to the prover's runtime.

[2] Throughout this work we mostly optimize for verification time and leave the proving time as a secondary consideration. This is in contrast to *doubly efficient interactive proofs* (see [19]) in which we insist on a polynomial-time prover. In this "doubly-efficient" regime, interactive proofs with a polynomial-time prover and almost linear time verifier are known for linear depth, poly-size, uniform circuits [21] and poly-time and bounded-poly space computation [32].

Let $\mathbf{ITIME}[T_V, T_P]$ be the set of languages with an interactive proof whose verifier runs in time T_V and whose prover runs in time T_P. If T_P is omitted, we assume it is the trivial bound of $T_P = O(2^{T_V})$. In this paper, all our protocols are public-coin and have perfect completeness.

Our most general result is an interactive protocol for alternating bounded-space algorithms.

▶ **Theorem 1** (Interactive Proof For Alternating Space). *For any T, S, and d constructible in time $O(S\log(T))$ and space $O(S)$:*

$$\mathbf{ATISP}_d[T,S] \subseteq \mathbf{ITIME}\left[\tilde{O}\left(n + S\log(T) + Sd\right), 2^{O(S)}\right].$$

Further, the verifier runs in space $O(S\log(d+S))$, the protocol is public coin, has $O(S\log(S)(\log(T)+d))$ rounds, $O(S\log(S)(\log(T)+d)\log(d+S))$ bits of communication, and perfect completeness.

For $d = O(\log(T))$, up to small polylog(S) factors, our protocol has the same verifier time and prover time as the best known protocol for deterministic bounded space algorithms [9]: verifier time $\tilde{O}(n + S\log(T))$ and prover time $2^{O(S)}$. As a special case for nondeterministic algorithms, this gives an interactive protocol with verifier time $\tilde{O}(n + S\log(T))$, improving upon the nondeterministic algorithms in [9], whose verifiers required time $\tilde{O}(n + S\log(T)^2)$, by a factor of $\log(T)$. We note $\log(T)$ may be as large as S.

In a limited sense, these results could be seen as tight, as they match, up to polylog(S) factors, the best known results for simulating deterministic algorithms by alternating ones. Chandra, Kozen, and Stockmeyer [8] show that any deterministic algorithm running in time T and space S has an alternating algorithm running in time $S\log(T)$. Specifically, $\mathbf{TISP}[T,S] \subseteq \mathbf{ATISP}_{\log(T)}[O(S\log(T)), O(S)]$. If we improved our verifier time dependence on $S\log(T)$ or Sd, this would improve the time of alternating algorithms simulating deterministic ones.

For $d = T$, Theorem 1 improves over the best known interactive proofs for alternating algorithms, with unbounded alternations, by Fortnow and Lund [16], which have verifier time $\tilde{O}(n + S^2T)$ and verifier space $O(S\log(T))$. Our protocol's verifier is at least a factor S faster (when $ST = \Omega(n)$).

See Table 1 for a comparison of how our protocol compares to prior protocols for nondeterministic algorithms, and Table 2 for a comparison of how our protocol compares to prior protocols for alternating algorithms.

The best verifiers [9, 40] for deterministic algorithms have verifier time $\tilde{O}(S\log(T) + n)$, verifier space $O(S\log(S))$, and provers with time $2^{O(S)}$. The best provers [32] for deterministic algorithms have prover time $T^{1+o(1)}\mathrm{poly}(S)$, but require verifier time $T^{o(1)}\mathrm{poly}(S) + n\mathrm{polylog}(T)$. These protocols are incomparable for T much larger than S, but much smaller than 2^S. For a more comprehensive summary, see the full version [11].

Table 1 Comparison of different protocols for $\mathbf{NTISP}[T,S]$ with polylog(S) factors omitted.

$\mathbf{NTISP}[T,S]$	Verifier Time	Verifier Space	Prover Time
[34]	$(n+S)\log(T)^2$	$(n+S)\log(n+T)$	$2^{\mathrm{poly}(S,n)}$
[16]	$n + S^3\log(T)$	$S\log(S)$	$2^{\mathrm{poly}(S,n)}$
[21]	$n + S^2\log(T)$	$S\log(S)$	$2^{O(S)}$
[9]	$n + S\log(T)^2$	$S\log(T)$	$2^{O(S)}$
This Work	$n + S\log(T)$	$S\log(S)$	$2^{O(S)}$

Table 2 Comparison of different protocols for $\mathbf{ATISP}_d[T, S]$ with polylog(S) factors omitted.

$\mathbf{ATISP}_d[T, S]$	Verifier Time	Verifier Space	Prover Time
[34]	$(n + S(\log(T) + d))S(\log(T) + d)$	$n + S\log(T) + Sd$	$2^{\text{poly}(S,n)}$
[16]	$n + S^2T$	$S\log(T)$	$2^{\text{poly}(S,n)}$
[21]	$n + S^2\log(T) + S^2d$	$S\log(S + d)$	$2^{O(S)}$
[9]	$n + (S\log(T) + Sd)^2$	$S\log(T) + Sd$	$2^{O(S\log(T)+Sd)}$
This Work	$n + S\log(T) + Sd$	$S\log(S + d)$	$2^{O(S)}$

When $S = O(\log(n))$, our prover runs in polynomial-time. This gives us doubly efficient proofs for alternating algorithms with few alternations and logarithmic space. As a special case, we give doubly efficient interactive proofs for **NL** where the number of bits communicated is $\tilde{O}(\log(n)^2)$. This improves on the amount of communication achieved by GKR (specialized for **NL**), which uses $\tilde{\Omega}(\log(n)^3)$ bits of communication.

▶ **Corollary 2** (Doubly Efficient Interactive Proofs for **NL**). **NL** *has interactive protocols whose provers run in polynomial time, verifiers run in quasilinear time, verifiers use* $\tilde{O}(\log(n))$ *space, the protocol uses* $\tilde{O}(\log(n)^2)$ *rounds,* $\tilde{O}(\log(n)^2)$ *bits of communication, is public coin and has perfect completeness.*

More generally, our protocols for nondeterministic algorithms use a factor $\log(T)$ less communication then the previous best protocols by Cook, and match the best prior protocols for deterministic algorithms, up to polylogarithmic factors.

1.1.1 Unbounded Fan in Circuit Results

Let $\mathbf{SIZE}-\mathbf{DEPTH}[2^S, d]$ be the set of space $O(S)$ uniform circuits of size 2^S and depth d with *unbounded* fan in AND and OR gates. Let T-uniform $\mathbf{SIZE}-\mathbf{DEPTH}[2^S, d]$ be the set of time T uniform, space S circuits of size 2^S and depth d with *unbounded* fan in AND and OR gates. Then due to a close relationship between alternating circuits and low depth circuits by Ruzzo and Tompa [37] (see the full version [11]), we have

▶ **Theorem 3** (Uniform Shallow Circuits Have Fast Interactive Proofs). *For any d, T, S constructible in time $O(S\log(T))$ and space $O(S)$, we have*

$$T\text{-uniform } \mathbf{SIZE}-\mathbf{DEPTH}\left[2^S, d\right] \subseteq \mathbf{ITIME}\left[\tilde{O}(n + S\log(T) + Sd), 2^{O(S)}\right]$$
$$\mathbf{SIZE}-\mathbf{DEPTH}\left[2^S, d\right] \subseteq \mathbf{ITIME}\left[\tilde{O}(n + S^2 + Sd), 2^{O(S)}\right].$$

Further, the verifier runs in space $O(S\log(d + S))$ and the protocol is public coin and has perfect completeness.

For fan in 2 circuits, this matches the verifier time of GKR [21], while the prover time remains polynomial in the circuit size[3]. For unbounded fan in circuits, or for alternating algorithms, our verifier is a factor of S faster than GKR.

[3] Note however that recent improvements [12, 39, 44, 45] of GKR have a (close to) linear prover, whereas our prover is only polynomial in the circuit size.

1.2 Proof Overview

We start by reviewing our efficient interactive proofs for deterministic algorithms. Then we explain the difficulty of extending this to nondeterministic algorithms, and how to overcome these problems. Finally we show how to extend this technique to alternating algorithms. We assume familiarity with the sumcheck protocol [26]. For a more detailed explanation of our interactive proofs for deterministic algorithms, see [9] or the nearly identical protocol by Thaler [41, Section 4.5.5] (see also [23, 40]).

1.2.1 Deterministic Algorithms

For a deterministic algorithm A, we first reduce the problem to repeated matrix squaring, then give an interactive protocol for that. Suppose A runs in time T on some input x and has unique start state a and accept state b. Let M be the adjacency matrix of A's computation graph on input x. Then A accepts x if and only if $(M^T)_{a,b} = 1$ (where M^T is M raised to the Tth power, *not* M transposed). For notation, we write $M_{a,b}$ as $M(a,b)$. At a high level, the idea is that if we have an interactive protocol that can reduce a claim that $M^{2i}(u,v) = \alpha$ to the claim that $M^i(u',v') = \alpha'$, then by applying this protocol $\log(T)$ times, we can verify the value of $M^T(a,b)$. We give such a reduction, but on the multilinear extensions of M^{2i} and M^i.

Like [9, 40], we reduce to matrix exponentiation and give an interactive protocol for that, instead of reducing to a quantified Boolean formula [34], or to a uniform circuit [21]. This both simplifies the protocol somewhat and makes it more efficient to compute the prover. The idea is to arithmetize these adjacency matrices, and then use a sum check [26] to reduce the statement about M^{2i} to the statement about M^i. In particular, we use the sum check for matrix exponentiation given by Thaler [39], details follow.

For a finite field $\mathbb{F}$, for any i define $\widehat{M^i} : \mathbb{F}^S \times \mathbb{F}^S \to \mathbb{F}$ as the multilinear extension of M^i. That is, $\widehat{M^i}$ is multilinear and for each $u, v \in \{0,1\}^S$ we have $\widehat{M^i}(u,v) = M^i(u,v)$. Then observe that for any i, and $u, v \in \mathbb{F}^S$ we have

$$\widehat{M^{2i}}(u,v) = \sum_{w \in \{0,1\}^S} \widehat{M^i}(u,w)\widehat{M^i}(w,v).$$

To see that this formula is correct, first observe that it is correct for *Boolean* values as it precisely corresponds to the definition of matrix multiplication. So the formula is correct on Boolean values. Since both sides of the equation are multi-linear[4], it follows that the formula holds for all values.

Then, we can use the sum check of [26] to reduce this to a claim that for some $w' \in \mathbb{F}^S$ and some $\beta \in \mathbb{F}$ we have $\beta = \widehat{M^i}(u,w')\widehat{M^i}(w',v)$. Then using a multi-point reduction, as was done in GKR [21], we reduce this to a claim that for some $u', v' \in \mathbb{F}^S$ and $\alpha' \in \mathbb{F}$ we have that $\alpha' = \widehat{M^i}(u',v')$.

Finally running this $\log(T)$ times gives the interactive protocol for deterministic algorithms, since the verifier can efficiently calculate $\widehat{M}$ itself.

[4] To show it is multilinear, we take any variable, say u_i, and show the formula is linear in u_i. For any w see that $\widehat{M^i}(u,w)$ is linear in u_i since $\widehat{M^i}$ is multilinear, and $\widehat{M^i}(w,v)$ is constant in u_i. Thus $\sum_{w \in \{0,1\}^S} \widehat{M^i}(u,w)\widehat{M^i}(w,v)$ is linear in u_i.

We remark that, using linearization type ideas (as in [35]), the above can be extended from the task of deciding whether a deterministic algorithm accepts, to verifying the multilinear extension of a function that the algorithm computes. This will be important for us later on when we use the above interactive proof as a subroutine in the protocol for nondeterministic algorithms.

1.2.2 Nondeterministic Algorithms and Changing Arithmetization

To try to apply this technique to a nondeterministic algorithm, A, we immediately encounter an issue with how to formulate the problem. Namely, if we are doing arithmetic over $\mathbb{Z}$, if the underlying matrix M corresponds to a non-deterministic computation, then the matrix $M^T_{a,b}$ is no longer 1 if and only if A accepts x. Rather, $M^T_{a,b}$ specifies the number of length T paths from a to b. This might be as large as $2^{\Omega(T)}$. If we do arithmetic over a field of characteristic q, then $M^T_{a,b}$ is the number of paths mod q. If the number of paths is some adversarial product of many small primes, we may need $q = \Omega(T)$ for the number of accepting paths to be non zero, mod q. This gave the less efficient verifier time for nondeterministic algorithms in [9].

We will still solve this problem by arithmetization, but we need to change our matrix multiplication from a field matrix multiplication to a binary multiplication, then arithmetize that. We define the matrix multiplication with binary operations where multiplication is AND and addition is OR. So let $M^{(2)} : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ denote this binary matrix multiplication, squaring, so that for any $u, v \in \{0,1\}^S$ we have

$$M^{(2i)}(u,v) = \bigvee_{w\in\{0,1\}^S} M^{(i)}(u,w)M^{(i)}(w,v).$$

With this form of matrix exponentiation, it suffices to check if $M^{(T)}_{a,b} = 1$. To do so, we convert binary matrix multiplication into an algebraic circuit. The obvious approach is to use a formula like

$$\tilde{M^{(2i)}}(u,v) = 1 - \prod_{w\in\{0,1\}^S} \left(1 - \widehat{M^{(i)}}(u,w)\cdot\widehat{M^{(i)}}(w,v)\right).$$

Unfortunately, this has too high of individual degree: 2^S. One can insert some linearization operations between the multiplications to reduce the degree, like those used by Shen [35]. But then for each of the S variables in w, one would need to add $O(S)$ linearization operations, giving a size $O(S^2)$ algebraic circuit, which we cannot afford.

Instead, we use an idea of Goldreich and Rothblum [20] to probabilistically reduce the degree of these large conjunctions by leveraging the Razborov-Smolensky [31, 36] approximation of large disjunctions as low degree polynomials. Razborov-Smolensky give a reduction from a large disjunction to a random parity check that succeeds with high probability:

$$\forall g \in \{0,1\}^n : \Pr_{r\in\{0,1\}^n}\left[\bigvee_{i\in[n]} g_i = \sum_{i\in[n]} g_i r_i \pmod 2\right] \geq \frac{1}{2}.$$

We note that if $g = 0^n$, then for any r, we have $\sum_{i\in[n]} g_i r_i \pmod 2 = 0$. That is, the error is one sided. The formula $\sum_{i\in[n]} g_i r_i \pmod 2$ is a linear polynomial in a field of characteristic 2. As this is useful for us, we shall only work with fields of characteristic 2 in this paper.

Then, taking an OR of k independent choices of randomness, we get an individual degree k polynomial that succeeds with probability $1 - \frac{1}{2^k}$. If $n = 2^S$ and $k = S$, this gives us a degree $\log(n)$ polynomial for the disjunction that is only wrong with probability $\frac{1}{n}$.

The idea is to replace our boolean formula with a low degree polynomial through Razborov-Smolensky. So let $D_r : \{0,1\}^\ell \times \{0,1\}^S \to \{0,1\}$ be a function outputting our random bits. Here $2^\ell = k = O(S)$ is the number of choices of random bits. Then our new approximation for $M^{(2i)}$ is

$$\widetilde{M^{(2i)}}(u,v) = 1 - \prod_{j\in\{0,1\}^\ell} \left(1 - \sum_{w\in\{0,1\}^S} D_r(j,w)\widehat{M^{(i)}}(u,w)\widehat{M^{(i)}}(w,v) \right). \tag{1}$$

Now we only need to insert $\ell = \log(S)$ levels of linearizations. In the technical details of the paper, we will not actually use algebraic circuits with linearization operations, but will work with these polynomials directly with an "unlinearization" procedure, to avoid discussing circuit uniformity.

1.2.3 Efficient Randomness

At this point we encounter a problem - Equation (1) calls for sampling $2^{\ell+S}$ random bits for D_r, which we cannot afford (since we want our verifier to run in time $\tilde{O}(S)$). So as in GR, we need to sample these using an ϵ biased set. For our ϵ biased set, we use the same one as GR, described in [1] (which is based on a Reed-Solomon code concatenated with a Hadamard code).

Thus, for every value of $j \in \{0,1\}^\ell$, we would like to set $D_r(j,\cdot)$ to be an ϵ-biased set. As $\ell = \log(S)$, if we were to sample these independently, as in GR (i.e., the protocol given in [20]), our verifier would require $O(S^2)$ bits of randomness. Instead, we sample these small bias sets in a correlated manner - via a random walk on an expander (each node in the expander specifies a seed for a small bias set). We use the Margulis [27] expander since it is a constant degree, constant spectral expander with extremely simple edge descriptions: simple additions and subtractions. This makes it very easy to take a start vertex and a (specification of a) random walk and compute any given step on that walk in both small space and small time.

Thus, we only require $R = O(S)$ truly random bits to describe a length $O(S)$ random walk on the ϵ biased sets described by a Reed-Solomon code concatenated with a Hadamard code. So let $D : \{0,1\}^R \times \{0,1\}^\ell \times \{0,1\}^S \to \{0,1\}$ be a function that generates our pseudorandomness, given R bits of true randomness. The verifier first chooses that randomness r, and then $D_r(j,w) = D(r,j,w)$.

Since D is both space and time efficient, we can have the prover compute its value for the verifier, and then have the verifier run the *deterministic* interactive protocol to confirm its value. In contrast, the GR verifier must calculate some low degree extension of D_r directly to use a constant number of rounds. This saves us time over GR.

Finally, as in GR, there is a chance that our pseudorandom bits give a polynomial that fails to compute the disjunctions correctly. In this case, to get perfect completeness we need to prove that the pseudorandom bits are incorrect. To do this, the prover just finds a disjunction closest to the input where the low degree approximation fails and tells the verifier where it fails. This would be a gate where its value in the low degree polynomial is one thing, but one of its input gates should force it to be something else. For instance, an OR gate with a value of 0, and an input to it with a value of 1. Then the verifier can run the interactive protocol to confirm that the low degree polynomial indeed says the gate's value conflicts with its input gate value, showing the pseudorandom bits were bad.

1.2.4 Alternating Algorithms In Terms Of Nondeterministic Algorithms

To use our protocol with alternating algorithms, we want to reduce the alternating algorithm to one with a few large disjunctions or conjunctions over a nondeterministic algorithm. This is similar to what is done when converting alternating algorithms to alternating circuits. Once we have few conjunctions and disjunctions over a nondeterministic algorithm, we can do the same low degree approximations again.

The idea is to, instead of quantifying over the symbols in the read once proof, quantify over the potential states the algorithm could be in when the quantifier changes. Then a nondeterministic algorithm describes if a proof could cause the state to change from one intermediate state to the next when the quantifier changes.

For example, suppose A is an algorithm with $d = 2$ alternations and running in space S recognizing language L. Think of A as a deterministic algorithm taking a proof and outputting true or false. Then since A is an alternating algorithm, $x \in L$ if and only if

$$\forall \text{BigProof1} : \exists \text{BigProof2} : A(x, (\text{BigProof1}, \text{BigProof2})).$$

We can instead be more fine grain with A and talk about its states. Let a be the start state of A and b be its unique accept state. Let B be the algorithm which takes an initial state u a final state v and a proof p, then checks if A starting at u is at state v when given the proof p after time $|p|$. Then our algorithm accepts x if and only if

$$\forall w \in \{0,1\}^S : ((\exists \text{Proof1} : B(x, a, w, \text{Proof1})) \implies (\exists \text{Proof2} : B(x, w, b, \text{Proof2}))) .$$

If we know how long Proof1 is supposed to be, we can replace

$$\exists \text{Proof1} : B(x, a, w, \text{Proof1})$$

with a nondeterministic algorithm C. Then our alternating algorithm becomes

$$\forall w \in \{0,1\}^S : C(x, a, w) \implies C(x, w, b).$$

Now, beside our nondeterministic algorithm, we are only quantifying over a variable of size $O(S)$, whereas Proof1 has size $O(T)$.

For a more general example, we can replace

$$\forall \pi_1 : \exists \pi_2 : \forall \pi_3 : \exists \pi_4 : A(x, (\pi_1, \pi_2, \pi_3, \pi_4))$$

with

$$\forall w_1 : C(x, a, w_1) \implies (\exists w_2 : C(x, w_1, w_2) \wedge (\forall w_3 : C(x, w_2, w_3) \implies C(x, w_3, b))).$$

1.2.5 Protocols for Alternating Algorithms

At this point, each quantification is now only over S variables, so we can use the same trick as before to replace these quantifications with low degree polynomials. Each of the universal quantifiers gets replaced with a large conjunction, and each of the existential quantifiers gets replaced with a large disjunction. Then we use Razborov-Smolensky to replace these conjunctions and disjunctions with low degree polynomials and use an interactive proof to remove the quantifiers one by one.

A few subtleties show up when doing this. One subtlety of this process is that in a straightforward reduction, a d alternation algorithm would give our verifier a claim about C at d different places. Running an interactive protocol d times to confirm each of these d

claims independently would require time $dS\log(T)$, which is too much for us. Instead, we need to use a multi point reduction again to reduce this to a claim about C at one location before running an interactive protocol to confirm that value.

Another subtlety is that it is not convenient to represent C as a nondeterministic algorithm taking two states as an input and checking if there is a computation path from one to the other. It is more convenient to describe C directly with the computation graph of A (now viewing A as a nondeterministic algorithm). For this to work, we need to make sure each alternation takes the same amount of time, say T. Then we write $C(x, u, v) = \widehat{M_x^{(T)}}(u, v)$.

So for example, consider the simple case of a 2 alternation algorithm. That is, suppose we want to verify that

$$\forall w \in \{0,1\}^S : C(x, a, w) \implies C(x, w, b).$$

As described before, we replace C with $\widehat{M^{(T)}}$. So we want to verify

$$\forall w \in \{0,1\}^S : \widehat{M_x^{(T)}}(a, w) \implies \widehat{M_x^{(T)}}(w, b).$$

Now we need to arithmetize the formula being quantified. So let

$$\widetilde{E}(w) = 1 - \widehat{M_x^{(T)}}(a, w)(1 - \widehat{M_x^{(T)}}(w, b)).$$

See that E is low degree and agrees with the predicate $\widehat{M_x^{(T)}}(a, w) \implies \widehat{M_x^{(T)}}(w, b)$ on binary inputs. Of course, $\tilde{E}$ is not multilinear, it has individual degree 2. Luckily, if we let $\widehat{E}$ be the multilinear function consistent with D on binary inputs, then one can use an unlinearization operation (similar to those used by Shen [35]) to reduce from a statement about $\widehat{E}$ to a statement about $\widetilde{E}$. So we need to verify that

$$\forall w \in \{0,1\}^S : \widehat{E}(w).$$

Using our low degree approximation, our verifier first chooses D_r, then wants to check if

$$1 = \prod_{j \in \{0,1\}^\ell} \left(1 - \sum_{w \in \{0,1\}^S} D_r(j, w)(1 - \widehat{E}(w)) \right). \tag{2}$$

Then we can reduce this to a statement about $\widehat{D_r}$ at a random location, and $\widehat{E}$ at a random location by using $\ell = O(\log(S))$ product reductions. We can unlinearize the statement about $\widehat{E}$ to get a claim about $\widetilde{E}$, or equivalently, about $\widehat{M_x^{(T)}}$ at two locations. Now we can verify the value of $\widehat{M_x^{(T)}}$ by using our protocol for nondeterministic algorithms. But to avoid doing this twice, we first run a multi-point reduction to reduce this to a statement about $\widehat{M_x^{(T)}}$ at one location first.

We can do a similar thing d times for an alternating algorithm. One more subtlety is that for $d > 2$, we need to make $\widehat{E}$ a function of a and b. This is so that we can view the formula in Equation (2) as a function of a and b so we can properly linearize and unlinearize it with respect to a and b. See the full proof for details.

▶ Remark 4 (Proof For Unbounded Fan in Depth Circuits Directly). We could have made an interactive protocol for unbounded fan-in circuits directly. After all, we start with a formula that is essentially the low depth, unbounded fan in circuit for an alternating algorithm, if we view C as a low depth circuit. We can think of our alternating algorithms as a particular kind of very uniform circuit. We don't give an interactive proof for circuits directly to avoid handling uniformity.

One reason we chose not to just provide an interactive protocol for circuits directly is that we need a faster interactive protocol for deterministic algorithms as a subroutine to verify our pseudorandom bits. Since we view this interactive protocol as a problem for bounded space, we find it natural to present the rest of the results in this framework.

An overview of the protocol for alternating algorithms are in Appendix A, but a full proof is given in the full version [11].

1.2.6 Extensions

We note that our paper focuses on verifier time, so we have not optimized other parameters, like verifier space. There are also some other straightforward extensions to further generalizations of alternations we don't prove here.

▶ Remark 5 (Parity Gates and Parity Quantifiers). Like GR, our techniques can also be used on alternating circuits with parity gates, or bounded space algorithms with parity quantifiers. This is clear since a parity gate is an addition gate over fields of characteristic 2, so is of low degree already.

We emphasize that our protocol is different from GR since we need more randomness efficient pseudorandom bits, efficient computation of those pseudorandom bits, more rounds of interaction to keep our degree (and thus verifier time) lower, use of an interactive protocol for deterministic algorithms as a subroutine, and by using connections between low space algorithms, low alternation algorithms, and uniform, low depth circuits.

▶ Remark 6 (Space Efficiency of Our Verifier). While we only achieve a verifier running in space $O(S\log(S))$, for any $\epsilon > 0$ we should be able to get verifier space $O(S/\epsilon)$ using standard techniques, at the cost of increasing verifier time by a factor of S^{ϵ}. Specifically, instead of using multilinear polynomials, we would use individual degree S^{ϵ} polynomials. Since we are focused on improving verifier time, we do not prove this result.

This technique was used by Shamir, Fortnow and Lund, and GKR [34, 16, 21] to give the space efficiency claimed in those papers. We state the special case where $\epsilon = \frac{1}{\log(S)}$ in our results since we want to compare verifier time.

2 Preliminaries

We assume the reader is familiar with basic complexity concepts like circuits, Turing machines, and big O notation. See [2] for a reference. For notation, we define $\tilde{O}$ to hide polylogarithmic factors in whatever is inside it in general, not specifically polylog(T) or polylog(n). That is:

▶ **Definition 7** (Big Tilde O). *For functions $f, g : \mathbb{N} \to \mathbb{N}$, we define $f(n) = \tilde{O}(g(n))$ if and only if there exists some constant c such that $f(n) = O(g(n)\log(g(n))^c)$.*

2.1 Bounded Space and Alternating Algorithms

We denote by $\mathbf{TISP}[T, S]$ languages that are computable by a Turing Machine running in time T and space S.

▶ **Definition 8** (TISP). *For functions $T, S : \mathbb{N} \to \mathbb{N}$, we say language L is in $\mathbf{TISP}[T, S]$ if there is an Turing Machine, A, running in time T and space S that decides L.*

We want interactive proofs for a generalization of nondeterministic algorithms called alternating algorithms, as was formally defined in [8]. Like how a nondeterministic algorithms have existential states where the algorithm accepts if any transition from that state accepts, alternating algorithms get both existential and accept states.

We further parameterize our alternating algorithms by the number of alternations. The number of alternations is the number of times it switches between existential and universal states, plus one. For instance, nondeterministic algorithms can be viewed as having one alternation, the second level of the polynomial hierarchy has two, and so on.

▶ **Definition 9** (**ATISP**). *For functions $T, S, d : \mathbb{N} \to \mathbb{N}$, we say a language L is in $\mathbf{ATISP}_d[T, S]$ if there is an alternating Turing Machine, A, running in time T and space S that recognizes L such that on any input x, our algorithm A only changes from no quantification to one, from existential to universal quantifiers, or from universal to existential quantifiers d times.*

So for instance, nondeterministic time T and space S algorithms would be contained in $\mathbf{ATISP}_1[T, S]$.

2.2 Interactive Proofs

An interactive proof informally is a proof system where a verifier with access to unpredictable randomness can verify a result such that if the statement is true, an honest prover can convince the verifier with high probability. And if the statement is false, no prover, no matter how powerful, can convince the verifier the statement is true above some constant probability.

For our definition of interactive time, let $\mathrm{Int}(V, P', x)$ denote the random variable that V outputs on input x when interacting with prover P'. See the full version for a more detailed definition [11]. Now we define interactive time. We note that in all our protocols, we achieve perfect completeness. That is, $c = 1$.

▶ **Definition 10** (Interactive Time ($\mathbf{ITIME}$)). *If for any language L, soundness $s \in [0, 1]$, completeness $c \in [0, 1]$, verifier V and prover P we have*
Completeness: *If $x \in L$, then $\Pr[\mathrm{Int}(V, P, x) = 1] \geq c$, and*
Soundness: *if $x \notin L$, then for any function P' we have $\Pr[\mathrm{Int}(V, P', x) = 1] \leq s$,*
then we say V and P are an interactive protocol for L with soundness s and completeness c.

If in addition verifier V runs in time T_V, soundness $s < \frac{1}{3}$, and completeness $c > \frac{2}{3}$, then

$$L \in \mathbf{ITIME}[T_V].$$

If P is also computable by an algorithm running in time T_P, we say

$$L \in \mathbf{ITIME}[T_V, T_P].$$

2.3 Expander Graphs

We will use expander graphs to create hitting samplers through the hitting properties of expanders [24]. See [43] for a more detailed review of expander graphs. We will assume some basic familiarity with graphs here.

We use an expander graph given by Margulis [27] proven by Gabber and Galil [18]. We use this expander because it's simple structure makes it very clear that we can compute random walks on it in very little time and linear space.

▶ **Lemma 11** (Efficient Expander Graphs). *For any square $n = m^2$, there exists an expander graph G with constant degree d and constant spectral expansion $\lambda < 1$.*

Let V be the vertex set of G, and $E : V \times [d] \to V$ be the edge function taking in a vertex, v, and the index of an edge, e, out of v, and outputting the other vertex incident to e. Then E can be computed in space $O(\log(n))$ and time $O(\log(n))$.

2.4 Arithmetization

A core technique of standard interactive proofs is called "arithmetization". Arithmetization is the process of converting some Boolean function, f, to a low degree formula over a larger field, $\mathbb{F}$, which agrees with f on Boolean inputs. The main function we will be arithmetizing is the state transition of Turing Machines. So for a given algorithm, on a length n input x and two states s_0 and s_1, let the state transition function $M_n(x, s_0, s_1)$ be one if and only if the algorithm on input x starting in state s_0 can transition to state s_1 in one step. See the full version for a more detailed definition [11].

We use [10, Lemma 36] for our arithmetization of the Turing Machine State transition.

▶ **Theorem 12** (Arithmetization of State Transition). *Suppose A is a space S nondeterministic algorithm with transition matrix $M_n : \{0,1\}^n \times \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ as above.*

Then we can compute the multilinear extension of M_n, denoted $\widehat{M_n} : \mathbb{F}^n \times \mathbb{F}^S \times \mathbb{F}^S \to \mathbb{F}$, in time $(n+S)\tilde{O}(\log(|\mathbb{F}|))$ and space $O((\log(S) + \log(n))\log(|\mathbb{F}|))$.

2.5 Standard Algebraic, Interactive Proof Tools

We use a few standard tools in interactive proofs. Like [21, 9, 40, 26, 34, 35], perhaps the most important tool is the original sum check from [26]. We also use an unlinearization protocol, like the one used by Shen [35]. And a multi-point reduction, like those used in [21, 9]. Similar query reductions have a long history in PCP literature [3, 15, 13, 30, 25]. To see statements of these lemmas, see the full version [11].

3 Interactive Proof For Deterministic Algorithms

Internally, our proof will need interactive proofs for deterministic algorithms. We use a variation of the deterministic protocols from [9, 40]. A full proof can be found in our full version [11], here is just an overview.

The idea of the algorithm is that for a time T algorithm A, if on an input x algorithm A has computation graph G with adjacency matrix M, then for unique start state a and end state b, algorithm A accepts A if and only if $M^T_{a,b} = 1$. Then by using a matrix square reduction repeatedly, this can be reduced to a statement about the value of $\widehat{M}$, the multilinear extension of M, at a random point. And $\widehat{M}$ can be calculated quickly using Theorem 12.

Our matrix square reduction is very similar to the matrix reduction by Thaler [39], except generalized to the case where the matrix is also a multilinear extension of a third input. The main difference with Thaler's is that we need to perform a few unlinearizations, similar to Shen's [35].

▶ **Lemma 13** (Matrix Square To Matrix Reduction). *Given a function $M : \{0,1\}^n \times \{0,1\}^S \times \{0,1\}^S \to \mathbb{F}$, denote for any $x \in \{0,1\}^n$ the matrix M_x such that $(M_x)_{u,v} = M(x, u, v)$. Then M_x^2 is defined in the usual way: $(M_x^2)_{u,v} = \sum_{w \in \{0,1\}^S} M_x(u,w)M_x(w,v)$. Now define $M^2 : \{0,1\}^n \times \{0,1\}^S \times \{0,1\}^S \to \mathbb{F}$ by $M_x^2(u,v) = (M_x^2)_{u,v}$. Let $\widehat{M}$ be the multilinear extension of M and $\widehat{M^2}$ be the multilinear extension of M^2.*

Then there is an $S+n+2$ round interactive protocol with $O((S+n)\log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $(S+n)\tilde{O}(\log(|\mathbb{F}|))$ and space $O((S+n)\log(|\mathbb{F}|))$, and a prover P that runs in time $2^{S+n}\tilde{O}(\log(|\mathbb{F}|))$ with $O(2^{S+n})$ oracle queries to $\widehat{M}$. The protocol takes as input $\alpha \in \mathbb{F}$, $u, v \in \mathbb{F}^S$, and $x \in \mathbb{F}^n$ and acts such that

Completeness: *If $\alpha = \widehat{M^2}(x,u,v)$, then when V interacts with P, V outputs a $u',v' \in \mathbb{F}^S$, $x' \in \mathbb{F}^n$, and $\alpha' \in \mathbb{F}$ such that $\alpha' = \widehat{M}(x',u',v')$.*

Soundness: *If $\alpha \neq \widehat{M^2}(x,u,v)$, then for any prover P' with probability at most $\frac{4S+3n}{|\mathbb{F}|}$ will V output a $u',v' \in \mathbb{F}^S$, $x' \in \mathbb{F}^n$, and $\alpha' \in \mathbb{F}$ such that $\alpha' = \widehat{M}(x',u',v')$.*

Now applying this square reduction $\log(T)$ times gives our interactive proof for the multilinear extension of a space efficient function. The proof is essentially many calls to Lemma 13, along with a final check by computing $\hat{M}_n$ directly at the final point using Theorem 12.

▶ **Theorem 14** (Interactive Proof For Multilinear Extension of Bounded Space)**.** *For any function $D : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$, for any $x \in \{0,1\}^n$, denote $D_x : \{0,1\}^m \to \{0,1\}$ by $D_x(y) = D(x,y)$. Let $\widehat{D_x}$ be the multilinear extension of D_x. If D is computed by a space S time T deterministic algorithm, then there is a $(m+S+2)\log(T)$ round interactive protocol with $O((m+S)\log(T)\log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $(n+(m+S)\log(T))\tilde{O}(\log(|\mathbb{F}|))$ and space $O((\log(n)+m+S)\log(|\mathbb{F}|))$, and a prover P that runs in time $2^{2m+2S}\log(T)\tilde{O}(\log(|\mathbb{F}|))$ which takes as input an $x \in \{0,1\}^n$, $w \in \mathbb{F}^S$ and $\alpha \in \mathbb{F}$ such that*

Completeness: *If $\widehat{D_x}(w) = \alpha$, then when V interacts with P, V accepts.*

Soundness: *If $\widehat{D_x}(w) \neq \alpha$, then for any prover P' with probability at most $\frac{(4S+3m)\log(T)}{|\mathbb{F}|}$ will V accept.*

4 Interactive Proofs For Nondeterministic Algorithms

Now we describe an interactive proof for nondeterministic algorithms because it is an interesting special case in its own right, it develops the tools needed for the more general alternating algorithm, and gives a good warm up for the general case. Here we give an outline of the proof, a full proof is in the full version [11].

But before we start, we quickly make a detour to explain that the matrix “multiplication” used here for nondeterministic algorithms is different than the one used for deterministic algorithms. For deterministic algorithms, we used standard multiplication and addition in some field. But for nondeterministic algorithms, we do binary matrix multiplication, with multiplication replaced with AND, and addition replaced with OR. To emphasize the difference, we use parentheses around the exponent to indicate we are performing binary matrix multiplication.

▶ **Definition 15** (Binary Matrix Multiplication)**.** *Let $M : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ be any function. Then by induction, define $M^{(1)} = M$ and for any i, define*

$$M^{(i+1)}(u,v) = \bigvee_w M^{(i)}(u,w)M(w,v).$$

See that if M is an adjacency matrix of a graph, then $M^{(i)}(s,t) = 1$ if and only if there is a path from s to t of length i.

▶ Remark 16. M^i is different from $M^{(i)}$ in that $M^{(i+1)}$ uses an OR function, whereas M^{i+1} uses a plus function. These are equivalent for the adjacency matrix of a deterministic algorithm, but crucially differ for a nondeterministic algorithm.

We emphasize that these binary matrix multiplications algebraically act very similarly to integer matrix multiplication. Specifically, $M^{(T)}$ can still be calculated with $\log(T)$ repeated binary squaring.

Now our goal is to replace the matrix sum check used for deterministic algorithms, with a new efficient reduction for nondeterministic algorithms. It is not clear how to do this directly, so we use a Razborov-Smolensky style low degree approximation, and give a reduction for that instead.

4.1 Extended Product Reduction

The main tool for this new reduction is this extended product reduction. This reduces a statement about the *multilinear extension* of a large product of terms to a statement about the *multilinear extension* of one term. This product reduction could be used to give a square reduction for nondeterministic algorithms directly, but is much more efficient if the number of multiplications is smaller. This is why we use Razborov-Smolensky.

The idea is to just apply many unlinearizations and product reductions, to one variable the product ranges over at a time. See the full version for a proof [11].

▶ **Lemma 17** (Extended Product Reduction). *Suppose $\widehat{f} : \mathbb{F}^\ell \times \mathbb{F}^S \to \mathbb{F}$ is multilinear. Let $g : \{0,1\}^S \to \mathbb{F}$ be defined by $g(v) = \prod_{u \in \{0,1\}^\ell} \widehat{f}(u,v)$ and let $\widehat{g}$ be the multilinear extension of g.*

Then there is an $\ell(S+1)$ round interactive protocol with $O(\ell S \log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $\ell S \tilde{O}(\log(|\mathbb{F}|))$ and space $O((\ell + S)\log(|\mathbb{F}|))$, and a prover P that runs in time $2^{\ell+S}\tilde{O}(\log(|\mathbb{F}|))$ which takes as input $w \in \mathbb{F}^S$, and $\alpha \in \mathbb{F}$ such that

Completeness: *If $\widehat{g}(w) = \alpha$, then when V interacts with P, V outputs a $u' \in \mathbb{F}^\ell$, $v' \in \mathbb{F}^S$, and $\alpha' \in \mathbb{F}$ such that $\widehat{f}(u',v') = \alpha'$.*

Soundness: *If $\widehat{g}(w) \neq \alpha$, then for any prover P' with probability at most $\frac{l(3S+1)}{|\mathbb{F}|}$ will V output a $u' \in \mathbb{F}^\ell$, $v' \in \mathbb{F}^S$, and $\alpha' \in \mathbb{F}$ such that $\widehat{f}(u',v') = \alpha'$.*

4.2 Low degree Approximations

To use Razborov-Smolenski efficiently, we need to be able to sample and calculate our ϵ biased sets, D_r, so they work with high probability and can be calculated efficiently.

Construction of efficient ϵ-biased sets is well researched and very efficient constructions are known [29, 38] and is equivalent to constructing good, linear codes. We use the third construction in [1] as our ϵ biased sets. This is the same ϵ biased set used in [20], except that we need to sample them more efficiently.

▶ **Lemma 18** (ϵ-Biased Set). *For any S, there is an $m = O(S)$ and a function $D' : \{0,1\}^m \times \{0,1\}^S \to \{0,1\}$ such that for any $X \subseteq \{0,1\}^S \setminus \emptyset$*

$$\Pr_{r \in \{0,1\}^m}\left[\sum_{x \in X} D'(r,x) = 1 \pmod 2\right] \geq \frac{1}{4}$$

such that D' runs in $\mathrm{poly}(S)$ time and $O(S)$ space.

Now we have a D' which with constant probability correctly converts an OR to a parity. Now we need to sample enough of these so that with constant probability, we convert 2^S ORs into parities. If we take $O(S)$ independent samples of D', then with probability less than 2^{-2S} will any of these ORs fail to be converted into a parity, so by a union bound with probability at most 2^{-S} will any of them fail to be converted into parity. Of course, we can not afford to take $O(S)$ samples of a string of length $O(S)$. So we take correlated samples using random walks on an expander graph.

Then by using a random walk on the Margulis expander, Lemma 11, with our ϵ biased sets as vertices, we can sample a good choice of ϵ biased sets with high probability.

▶ **Lemma 19** (Sampling a Good D for Many ORs). *Suppose for $n = 2^S$ and $m = 2^{S'}$, for each $i \in [m]$ there is an $f_i \in \{0,1\}$ and $u^i \in \{0,1\}^i$ such that $f_i = \bigvee_{j\in[n]} u^i_j$.*

Then for any ϵ, for $R = O(S + S' + \log(1/\epsilon))$, $L = 2^\ell = O(S' + \log(1/\epsilon))$, there is a space $O(S + \log(1/\epsilon))$, time $\mathrm{poly}(S + \log(1/\epsilon))$ algorithm D such that for each $i \in [m]$ define

$$F_i^r = 1 + \prod_{k\in\{0,1\}^\ell} (1 + \sum_{j\in[n]} D_r(k,j)u^i_j) \mod 2.$$

If $\forall i \in [m] : F_i^r = f_i$, then we say D_r is good for each f. Then $\Pr_r[D_r$ is not good for $f] < \epsilon$.

4.3 Interactive Proofs For Nondeterministic Algorithms

Now we give a summary of our proof, full version available at [11]. We start by defining our interactive proof we wish to run, assuming we got a good D_r. This is a square reduction, assuming we can use the Razborov-Smolensky formula to describe $M^{(2)}$. We call the Razborov-Smolensky style polynomial given by our pseudorandomness D as "M relative to D". We say that our D_r is good if $M^{(T)}$ relative to D is $M^{(T)}$. If D_r is good, we are done. Otherwise, D_r is bad, and makes some mistake first. Then we give an interactive proof to show where it is bad.

First, we formally define M relative to D.

▶ **Definition 20** (M Relative to D). *For any $M : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$, and $D : \{0,1\}^\ell \times \{0,1\}^S \to \mathbb{F}$, we define M relative to D as the functions, for $k = 1$, $M_D^{(1)} = M_D = M$, and for any $k > 1$ the function $M_D^{(2^k)} : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ is*

$$M_D^{(2^k)}(u,v) = 1 + \prod_{j\in\{0,1\}^\ell} (1 + \sum_{w\in\{0,1\}^S} D(j,w)M_D^{(2^{k-1})}(u,w)M_D^{(2^{k-1})}(w,v)).$$

Similar to the deterministic case Lemma 13, we have a repeated square reduction for M relative to D. This is based on an extended product reduction, Lemma 17, and a sum check.

▶ **Lemma 21** (Repeated Square Reduction For M relative to D). *For some $M : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$, let $\widehat{M}$ be the multilinear extension of M and $\widehat{M^{(2)}}$ be the multilinear extension of $M^{(2)}$. For some $D : \{0,1\}^\ell \times \{0,1\}^S \to \mathbb{F}$ and $T = 2^t$ let $\widehat{M_D^{(T)}}$ be the multilinear extension of M relative to D given by Definition 20.*

Then there is an $O(\ell S \log(T))$ round interactive protocol with $O(\ell S \log(T) \log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $\ell S \log(T)\tilde{O}(\log(|\mathbb{F}|))$ and space $O((S + \ell)\log(|\mathbb{F}|))$, and a prover P (with access to the truth table of M and D) that runs in time $2^{O(\ell+S)}\tilde{O}(\log(|\mathbb{F}|))$ which takes as input $u, v \in \mathbb{F}^S$, and $\alpha \in \mathbb{F}$ such that

Completeness: *If $\widehat{M_D^{(T)}}(u,v) = \alpha$, then when V interacts with P, V outputs a $u', v', w' \in \mathbb{F}^S$, $j' \in \mathbb{F}^\ell$, and $\alpha', \beta' \in \mathbb{F}$ such that $\widehat{M}(u', v') = \alpha'$ and $D(j', w') = \beta'$.*

Soundness: *If $\widehat{M_D^{(T)}}(u,v) \neq \alpha$, then for any prover P' with at most $\frac{\log(T)(\ell+2)(6S+2)}{|\mathbb{F}|}$ probability will V output a $u', v', w' \in \mathbb{F}^S$, $j' \in \mathbb{F}^\ell$, and $\alpha', \beta' \in \mathbb{F}$ such that $\widehat{M}(u', v') = \alpha'$ and $D(j', w') = \beta'$.*

If D is good, this gives our interactive protocol for nondeterministic algorithms. Unfortunately, D is not always good. But if it is not good, then a prover can show the verifier where it is bad, giving us perfect completeness. See [11] for full details.

We note here that the field size in our final protocol is $|\mathbb{F}| = \text{poly}(S)$ and $l = O(\log(S))$. So the specific polylog(S) hidden by $\tilde{O}$ in our main result is $O(\log(S)^2 \text{polylog}(\log(S)))$. This is worse than the polylog(S) overhead for deterministic algorithms given in [9], which was $O(\log(S)\text{polylog}(\log(S)))$. This is because our extended product reduction is slower than a sum check, but only a $\log(S)$ factor slower.

An overview of the protocol for alternating algorithms are in Appendix A, but a full proof is given in the full version [11].

5 Open Problems

While this mostly closes the gap between the best verifier for deterministic and nondeterministic algorithms, many interesting open problems remain, including:

1. Finding a stronger relationship between verifier time (or even alternating time) and bounded space. We know, for $S \geq n$, that

   $$\mathbf{TISP}[T, S] \subseteq \mathbf{ITIME}[\tilde{O}(S \log(T))].$$

   But it is unknown whether, even with the stronger class of alternating algorithms, if

   $$\mathbf{TISP}[T, S] \subseteq \mathbf{ATIME}[o(S \log(T))].$$

2. Finding a stronger relationship between verifier time and alternating time. We know, for $T \geq n$, that

   $$\mathbf{ATISP}[T, S] \subseteq \mathbf{ITIME}[\tilde{O}(ST)].$$

   Can this factor of S in verifier time be removed?
3. Find interactive protocols for $\mathbf{BPTISP}[T, S]$ with simultaneous verifier time $\tilde{O}(n + S\log(T))$, prover time $2^{O(S)}$ *and* perfect completeness.
   Cook [9] gave a protocol with that verifier and prover time, but with imperfect completeness. Perfect completeness can be achieved in a black box way [17], but these black box reductions do not preserve the prover time.
4. Better doubly efficient proofs. In our special case of alternating algorithms, we can not get provers who run in less than exponential time, without giving sub-exponential time deterministic algorithms for nondeterministic problems.
   But even in the deterministic time and space bounded setting, for $S \geq n$, a major open problem is whether

   $$\mathbf{TISP}[T, S] \subseteq \mathbf{ITIME}[\text{poly}(S), \text{poly}(T)].$$

   We do know from [21] that

   $$\mathbf{TISP}[T, S] \subseteq \mathbf{ITIME}[\text{poly}(S), 2^{O(S)}],$$

   and from [32] that

   $$\mathbf{TISP}[T, S] \subseteq \mathbf{ITIME}[T^{o(1)}, \text{poly}(T)],$$

   but it is unknown if both the fast verifier time and prover time can be achieved simultaneously.

5. Similar verifier time for algorithms with more general kinds of quantifiers. For instance, threshold quantifiers.
Currently the most verifier efficient known interactive protocol for threshold circuits is to use shallow circuits to compute threshold and run GKR. In this paper, we showed one can do better for unbounded fan-in AND and OR gates. Can this also be done for unbounded fan-in threshold gates? This would be interesting because threshold gates seem much more powerful than AND, OR, or parity gates.

References

1 N. Alon, O. Goldreich, J. Hastad, and R. Peralta. Simple construction of almost k-wise independent random variables. In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, pages 544–553 vol.2, 1990. doi:10.1109/FSCS.1990.89575.

2 Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, USA, 1st edition, 2009.

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, May 1998. doi:10.1145/278298.278306.

4 Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of np. *J. ACM*, 45(1):70–122, January 1998. doi:10.1145/273865.273901.

5 L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, pages 16–25 vol.1, 1990. doi:10.1109/FSCS.1990.89520.

6 László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, STOC '91, pages 21–32. Association for Computing Machinery, 1991. doi:10.1145/103418.103428.

7 Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *Theory of Cryptography Conference*, 2016.

8 Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981. doi:10.1145/322234.322243.

9 Joshua Cook. More Verifier Efficient Interactive Protocols for Bounded Space. In Anuj Dawar and Venkatesan Guruswami, editors, *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2022)*, volume 250 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:18, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.FSTTCS.2022.14.

10 Joshua Cook. More verifier efficient interactive protocols for bounded space, 2022. URL: https://eccc.weizmann.ac.il/report/2022/093/.

11 Joshua Cook and Ron Rothblum. Efficient interactive proofs for non-deterministic bounded space, 2023. URL: https://eccc.weizmann.ac.il/report/2023/097/.

12 Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 90–112. Association for Computing Machinery, 2012. doi:10.1145/2090236.2090245.

13 Irit Dinur, Eldar Fischer, Guy Kindler, Ran Raz, and Shmuel Safra. Pcp characterizations of np: Towards a polynomially-small error-probability. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, STOC '99, pages 29–40. Association for Computing Machinery, 1999. doi:10.1145/301250.301265.

14 Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost np-complete (preliminary version). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 2–12. IEEE Computer Society, 1991. doi:10.1109/SFCS.1991.185341.

15 Uriel Feige and László Lovász. Two-prover one-round proof systems: Their power and their problems (extended abstract). In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '92, pages 733–744. Association for Computing Machinery, 1992. doi:10.1145/129712.129783.

16 Lance Fortnow and Carsten Lund. Interactive proof systems and alternating time-space complexity. *Theor. Comput. Sci.*, 113(1):55–73, 1993. doi:10.1016/0304-3975(93)90210-K.

17 Martin Fürer, Oded Goldreich, Y. Mansour, Michael Sipser, and Stathis Zachos. On completeness and soundness in interactive proof systems. *Adv. Comput. Res.*, 5:429–442, 1989.

18 Ofer Gabber and Zvi Galil. Explicit constructions of linear size superconcentrators. In *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pages 364–370, 1979. doi:10.1109/SFCS.1979.16.

19 Oded Goldreich. On doubly-efficient interactive proof systems, 2018. URL: https://www.wisdom.weizmann.ac.il/~oded/de-ip.html.

20 Oded Goldreich and Guy N. Rothblum. *Constant-Round Interactive Proof Systems for AC0[2] and NC1*, pages 326–351. Springer International Publishing, Cham, 2020. doi:10.1007/978-3-030-43662-9_18.

21 Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. *J. ACM*, 62(4), September 2015. doi:10.1145/2699436.

22 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989. doi:10.1137/0218012.

23 Edward Hirsch, Dieter van Melkebeek, and Alexander Smal. Succinct interactive proofs for quantified boolean formulas, comment 2. Electronic Colloquium on Computational Complexity (ECCC), 2013. URL: https://eccc.weizmann.ac.il/report/2012/077/comment/2/download/.

24 Nabil Kahale. Eigenvalues and expansion of regular graphs. *Journal of the ACM*, 42(5):1091–1106, 1995.

25 Yael Tauman Kalai and Ran Raz. Interactive pcp. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming, Part II*, ICALP '08, pages 536–547. Springer-Verlag, 2008. doi:10.1007/978-3-540-70583-3_44.

26 C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, pages 2–10 vol.1, 1990. doi:10.1109/FSCS.1990.89518.

27 Grigorii Aleksandrovich Margulis. Explicit construction of concentrators. *Problemy Peredachi Informatsii*, 9:325–332, 1973. URL: https://cir.nii.ac.jp/crid/1572824500389149056.

28 Cody Murray and Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime: An easy witness lemma for np and nqp. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 890–901. Association for Computing Machinery, 2018. doi:10.1145/3188745.3188910.

29 Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, 1993. doi:10.1137/0222053.

30 Ran Raz. Quantum information and the pcp theorem. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '05, pages 459–468, USA, 2005. IEEE Computer Society. doi:10.1109/SFCS.2005.62.

31 Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical notes of the Academy of Sciences of the USSR*, 41:333–338, 1987.

32 Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 49–62. Association for Computing Machinery, 2016. doi:10.1145/2897518.2897652.

33 Rahul Santhanam. Circuit lower bounds for merlin-arthur classes. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 275–283. Association for Computing Machinery, 2007. doi:10.1145/1250790.1250832.

34 Adi Shamir. Ip = pspace. *J. ACM*, 39(4):869–877, October 1992. doi:10.1145/146585.146609.

35 A. Shen. Ip = space: Simplified proof. *J. ACM*, 39(4):878–880, 1992. doi:10.1145/146585.146613.

36 R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 77–82. Association for Computing Machinery, 1987. doi:10.1145/28395.28404.

37 Larry Stockmeyer and Uzi Vishkin. Simulation of parallel random access machines by circuits. *SIAM Journal on Computing*, 13(2):409–422, 1984. doi:10.1137/0213027.

38 Amnon Ta-Shma. Explicit, almost optimal, epsilon-balanced codes. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 238–251. Association for Computing Machinery, 2017. doi:10.1145/3055399.3055408.

39 Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 71–89, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

40 Justin Thaler. The unreasonable power of the sum-check protocol, March 2020. URL: https://zkproof.org/2020/03/16/sum-checkprotocol/.

41 Justin Thaler. Proofs, arguments, and zero-knowledge. *Found. Trends Priv. Secur.*, 4(2-4):117–660, 2022. doi:10.1561/3300000030.

42 L. Trevisan and S. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. In *Proceedings 17th IEEE Annual Conference on Computational Complexity*, pages 129–138, 2002. doi:10.1109/CCC.2002.1004348.

43 Salil P. Vadhan. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012. doi:10.1561/0400000010.

44 Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019 – 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 733–764. Springer, 2019. doi:10.1007/978-3-030-26954-8_24.

45 Jiaheng Zhang, Tianyi Liu, Weijie Wang, Yinuo Zhang, Dawn Song, Xiang Xie, and Yupeng Zhang. Doubly efficient interactive proofs for general arithmetic circuits with linear prover time. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15–19, 2021*, pages 159–177. ACM, 2021. doi:10.1145/3460120.3484767.

A Interactive Proofs For Alternating Algorithms

Now we a sketch of our interactive protocols for alternating algorithms. This still uses the same Razborov-Smolensky degree reduction technique used for nondeterministic algorithms to reduce the degree of large fan in AND and ORs. The main conceptual challenge is rewriting the alternating algorithm in the correct format. So we do this first. For full proofs, see the full paper [11].

A.1 Alternation Reductions For Bounded Space

To prove our interactive protocol with alternating algorithms, we first must convert our algorithm into a simpler, layered algorithm. This is closely related to the reduction from an alternating algorithm to a low depth circuit by Ruzzo and Tompa [37], and a similar reduction was used by Fortnow and Lund [16] in their interactive proof for alternating algorithms.

▶ **Definition 22** (M with d Alternations). *For any $M : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ and integer d, define M with time T and d alternations inductively on d as a function $B^d : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ by*

$d = 1$: $B^1(u,v) = M(u,v)$.

d is even

$$\begin{aligned} B^d(u,v) =& \forall w \in \{0,1\}^S : M(u,w) \implies B^{d-1}(w,v) \\ =& \neg \bigvee_{w\in\{0,1\}^S} (M(u,w) \wedge \neg B^{d-1}(w,v)) \end{aligned}$$

d is odd

$$\begin{aligned} B^d(u,v) =& \exists w \in \{0,1\}^S : M(u,w) \wedge B^{d-1}(w,v) \\ =& \bigvee_{w\in\{0,1\}^S} M(u,w) \wedge B^{d-1}(w,v). \end{aligned}$$

Our interactive proof will focus on this intermediate representation of an alternating circuit as a matrix M with d quantifiers of S variables between them. Any alternating algorithm can be converted to a problem of a matrix M (which is the computation graph of a nondeterministic algorithm) with alternations. The idea is that the quantifiers guess the states at which the alternating algorithm switches quantifiers. A more detailed relationship is shown in the full version [11].

▶ **Lemma 23** (Layered Alternating Programs). *For any $L \in \mathbf{ATISP}_d[T,S]$, there is a nondeterministic algorithm A running in time $T' = O(T)$ and space $S' = O(S)$ such that on any input x, if M is the adjacency matrix of the computation graph of A on input x, then $x \in L$ if and only if the $M^{(T')}$ with d alternations, B^d as defined in Definition 22, has $B^d(a,b) = 1$ for some unique starting state a and unique accepting state b.*

A.2 Interactive Proof For Layered Alternations

Now the rest of the proof closely follows the proof for nondeterministic algorithms, defining M with alternations relative to D, showing how an alternation reduction for M with alternations relative to D, and a protocol to show that D is bad.

A subtle difference is that our interactive protocols actually reduce a statement about our alternating algorithm, to a statement about $M^{(T)}$, where M is the adjacency matrix of a nondeterministic algorithm. So we then have to apply our interactive proofs for nondeterministic algorithms. That is, we reduce our statement about alternating algorithms to one about nondeterministic ones, which we already developed the tools for.

▶ **Definition 24** (M with d Alternations, Relative to D). *For any $M : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$, and $D : \{0,1\}^\ell \times \{0,1\}^S \to \{0,1\}$, we define M with d alternations, relative to D, inductively on d as a function $B_D^d : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ by*

$d = 1$: $B_D^1(u,v) = M(u,v)$.

d is even $B_D^d(u,v) = \prod_{k\in\{0,1\}^\ell}(1 + \sum_{w\in\{0,1\}^S} D_r(k,w)(M(u,w) + M(u,w)B_D^{d-1}(w,v))) \mod 2$.

d is odd $B_D^d(u,v) = 1 + \prod_{k\in\{0,1\}^\ell}(1 + \sum_{w\in\{0,1\}^S} D_r(k,w)(M(u,w)B_D^{d-1}(w,v))) \mod 2$.

Now there is an interactive protocol for reducing the number of alternations by 1. Similar to our product reduction for nondeterministic algorithms, it uses our extended product reduction, Lemma 17, and a sum check.

▶ **Lemma 25** (IP for M with d Alternations, Relative To D, Single Step)**.** *For any $M : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ and integer $d > 1$, $D : \{0,1\}^\ell \times \{0,1\}^S \to \{0,1\}$ let $B^d_D : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ be M with d layered alternations relative to D, as defined in Definition 24. Let $\widehat{B^d_D}$ be the multilinear extension of B^d_D.*

Then there is an $O(\ell S)$ round interactive protocol with $O(\ell S \log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $\ell S \tilde{O}(\log(|\mathbb{F}|))$, space $O((\ell + S)\log(|\mathbb{F}|))$, and a prover P (given the truth table of M, B^d_D and D) that runs in time $2^{O(\ell+S)}\tilde{O}(\log(|\mathbb{F}|))$ which takes as input $u, v \in \mathbb{F}^S$, and $\alpha \in \mathbb{F}$ such that

Completeness: *If $\widehat{B^d_D}(u,v) = \alpha$, then when V interacts with P, V outputs a $u', v', w' \in \mathbb{F}^S$, $j' \in \mathbb{F}^\ell$, and $\alpha', \beta', \gamma' \in \mathbb{F}$ such that $\widehat{B^{d-1}_D}(w',v') = \alpha'$, $\widehat{D}(j',w') = \beta'$, and $\widehat{M}(u',w') = \gamma'$.*

Soundness: *If $\widehat{B^d_D}(u,v) \neq \alpha$, then for any prover P' with probability at most $\frac{(\ell+1)(6S+1)}{|\mathbb{F}|}$ will V output a $u', v', w' \in \mathbb{F}^S$, $j' \in \mathbb{F}^\ell$, and $\alpha', \beta', \gamma' \in \mathbb{F}$ such that $\widehat{B^{d-1}_D}(w',v') = \alpha'$, $\widehat{D}(j',w') = \beta'$, and $\widehat{M}(u',w') = \gamma'$.*

Applying this many times gives an interactive protocol reducing a statement about our alternating algorithm to one about a nondeterministic one, which we can solve using the ideas in Lemma 21.

▶ **Lemma 26** (IP for M with d Alternations, Relative To D)**.** *For any $M : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ and integer d, $D : \{0,1\}^\ell \times \{0,1\}^S \to \{0,1\}$ let $B^d_D : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$ be M with d layered alternations relative to D, as defined in Definition 24. Let $\widehat{B^d_D}$ be the multilinear extension of B^d_D.*

Then there is an $O(\ell S d)$ round interactive protocol with $O(\ell S d \log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $\ell S d \tilde{O}(\log(|\mathbb{F}|))$, space $O((\ell + S)\log(|\mathbb{F}|))$, and a prover P (given the truth table of M, B^d_D and D) that runs in time $d2^{O(\ell+S)}\tilde{O}(\log(|\mathbb{F}|))$ which takes as input $u, v \in \mathbb{F}^S$, and $\alpha \in \mathbb{F}$ such that

Completeness: *If $\widehat{B^d_D}(u,v) = \alpha$, then when V interacts with P, V outputs a $u', v', w' \in \mathbb{F}^S$, $j' \in \mathbb{F}^\ell$, and $\alpha', \beta' \in \mathbb{F}$ such that $\widehat{M}(u',v') = \alpha'$ and $\widehat{D}(j',w') = \beta'$.*

Soundness: *If $\widehat{B^d_D}(u,v) \neq \alpha$, then for any prover P' with probability at most $\frac{d(\ell+2)(6S+2)}{|\mathbb{F}|}$ will V output a $u', v', w' \in \mathbb{F}^S$, $j' \in \mathbb{F}^\ell$, and $\alpha', \beta' \in \mathbb{F}$ such that $\widehat{M}(u',v') = \alpha'$ and $\widehat{D}(j',w') = \beta'$.*

This would be enough if D was always good, but D may be bad, which must be handled to get perfect completeness. First, let us define what it means for D to be good.

▶ **Definition 27** (D is good for M up to d Alternations)**.** *For any $M : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$, d and $D : \{0,1\}^\ell \times \{0,1\}^S \to \{0,1\}$, we say that D is good for M with up to d alternations if for all $k \in [d]$ with $k > 1$ we have $B^k_D = B^k$.*

But when D is bad, we give a protocol showing where it is bad. A similar protocol exists for non deterministic algorithms. The idea is to find the first quantifier that D is not good for, and tell them both which clause it has the wrong value on, and which input should have given it a different value.

For instance, if D would claim $(\forall y : \phi(x, y)) = 1$, but it isn't, the prover says which y gives this wrong claim, and which x would have $\phi(x, y) = 0$. Our protocol can then show the verifier that indeed D claims that $\phi(x, y) = 1$, but $\phi(x, y) = 0$ since D is correct all the way up to the quantification on y.

More detailed proofs can be found in the full version [11].

▶ **Lemma 28** (Proving D is Bad for M with Alternations). *For some $M : \{0,1\}^S \times \{0,1\}^S \to \{0,1\}$, and integer d, let $\widehat{M}$ be the multilinear extension of M. Let ℓ be an integer and $D : \mathbb{F}^\ell \times \mathbb{F}^S \to \mathbb{F}$ a multilinear function.*

Then there is a round $O(\ell S d)$ interactive protocol with $O(\ell S d \log(|\mathbb{F}|))$ bits of communication, a verifier V that runs in time $\ell S d \tilde{O}(\log(|\mathbb{F}|))$ and space $O((\ell + S)\log(|\mathbb{F}|))$, and a prover P (with access tot he truth table of M) that runs in time $d2^{O(\ell+S)}\tilde{O}(\log(|\mathbb{F}|))$ such that

Completeness: *If D is not good for M up to d alternations, then when V interacts with P, V outputs $u', v', w' \in \mathbb{F}^S$, $j' \in \mathbb{F}^\ell$, and $\alpha', \beta' \in \mathbb{F}$ such that $\widehat{M}(u', v') = \alpha'$ and $\widehat{D}(j', w') = \beta'$.*

Soundness: *If D is good for M up to d alternations, then when V interacts with P, with probability at most $\frac{d(\ell+2)(6S+2)}{|\mathbb{F}|}$ will V output $u', v', w' \in \mathbb{F}^S$, $j' \in \mathbb{F}^\ell$, and $\alpha', \beta' \in \mathbb{F}$ such that $\widehat{M}(u', v') = \alpha'$ and $\widehat{D}(j', w') = \beta'$.*

Combining all of these gives our main theorem. Again, we see that the polylogarithmic factor overhead is $O(\log(S)^2 \text{polylog}(\log(S)))$. As noted in the nondeterministic section, this is worse than the $O(\log(S)\text{polylog}(\log(S)))$ factor overhead for deterministic algorithms in [9].

For a full proof, see [11].

On the Complexity of Triangle Counting Using Emptiness Queries

Arijit Bishnu
Indian Statistical Institute, Kolkata, India

Arijit Ghosh
Indian Statistical Institute, Kolkata, India

Gopinath Mishra
University of Warwick, Coventry, UK

Abstract

Beame et al. [ITCS'18 & TALG'20] introduced and used the Bipartite Independent Set (BIS) and Independent Set (IS) oracle access to an unknown, simple, unweighted and undirected graph and solved the edge estimation problem. The introduction of this oracle set forth a series of works in a short time that either solved open questions mentioned by Beame et al. or were generalizations of their work as in Dell and Lapinskas [STOC'18 and TOCT'21], Dell, Lapinskas, and Meeks [SODA'20 and SICOMP'22], Bhattacharya et al. [ISAAC'19 & TOCS'21], and Chen et al. [SODA'20]. Edge estimation using BIS can be done using polylogarithmic queries, while IS queries need sub-linear but more than polylogarithmic queries. Chen et al. improved Beame et al.'s upper bound result for edge estimation using IS and also showed an almost matching lower bound. Beame et al. in their introductory work asked a few open questions out of which one was on estimating structures of higher order than edges, like triangles and cliques, using BIS queries.

In this work, we almost resolve the query complexity of estimating triangles using BIS oracle. While doing so, we prove a lower bound for an even stronger query oracle called Edge Emptiness (EE) oracle, recently introduced by Assadi, Chakrabarty, and Khanna [ESA'21] to test graph connectivity.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Lower bounds and information complexity

Keywords and phrases Triangle Counting, Emptiness Queries, Bipartite Independent Set Query

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.48

Category RANDOM

Related Version *Full Version*: https://arxiv.org/abs/2110.03836

Funding *Gopinath Mishra*: Research supported in part by the Centre for Discrete Mathematics and its Applications (DIMAP), by EPSRC award EP/V01305X/13.

1 Introduction

Motivated by connections to group testing, to emptiness versus counting questions in computational geometry, and to the complexity of decision versus counting problems, Beame et al. introduced the Bipartite Independent Set (shortened as BIS) and Independent Set (shortened as IS) oracles as a counterpoint to the local queries [18, 16, 19]. The BIS query oracle can be seen in the lineage of the query oracles [24, 25, 23] that go beyond the local queries. The local queries for a graph $G = (V(G), E(G))$ are: (i) degree query: given $u \in V(G)$, the oracle reports the degree of u in $V(G)$; (ii) neighbour query: given $(u \in V(G))$, the oracle reports the i-th neighbor of u, if it exists; otherwise, the oracle reports

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 48; pp. 48:1–48:22
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

NULL; (iii) ADJACENCY query: given u, $v \in V(G)$, the oracle reports whether $\{u, v\} \in E(G)$. There is another related query oracle RANDOMEDGE query: the oracle reports an edge uniformly at random when we query.[1]

Let us start by looking into the formal definitions of BIS and IS.

▶ **Definition 1.1** (BIPARTITE INDEPENDENT SET). Given disjoint subsets $U, U' \subseteq V(G)$, a BIS query answers whether there exists an edge between U and U' in G.

▶ **Definition 1.2** (INDEPENDENT SET). Given a subset $U \subseteq V(G)$, a IS query answers whether there exists an edge between vertices of U in G.

The introduction of this new type of oracle access to a graph spawned a series of works that either solved open questions [12, 13] mentioned in Beame et al. or were generalizations [13, 8, 6]. Beame et al. used BIS and IS queries to estimate the number of edges in a graph [4, 5]. One of their striking observations was that BIS queries were more effective than IS queries for estimating edges. This observation also fits in with the fact that IS queries can be simulated in a randomized fashion using polylogarithmic BIS queries.[2] Edge estimation using BIS was also solved in [12] albeit in a higher query complexity than [4]. There were later generalizations of the BIS oracle to estimate higher order structures like triangles and hyperedges [13, 7, 6]. On the IS front, Beame et al.'s result for edge estimation using IS oracle was improved in [11] with an almost matching lower bound and thereby resolving the query complexity of estimating edges using IS oracle. One can observe the interest generated in these (bipartite) independent set based oracles in a short span of time. The results are summarized in Table 1: a cursory glance would tell us that commensurate higher order queries were needed for estimating higher order structures (TRIPARTITE INDEPENDENT SET (shortened as TIS) for counting triangles, COLORFUL INDEPENDENCE ORACLE (shortened as CID) for counting hyperedges) if polylogarithmic number of queries is the benchmark. We provide the definitions of TIS and CID below.

▶ **Definition 1.3** (TRIPARTITE INDEPENDENT SET (TIS) [7]). Given three disjoint subsets A, B, C of the vertex set V of a graph $G(V, E)$, the TIS oracle reports whether there exists a triangle having endpoints in A, B and C.

▶ **Definition 1.4** (COLORFUL INDEPENDENCE ORACLE (CID) [9, 13]). Given d pairwise disjoint subsets of vertices $A_1, \ldots, A_d \subseteq U(\mathcal{H})$ of a hypergraph $\mathcal{H}$ ($U(\mathcal{H})$ is the vertex set of the hypergraph $\mathcal{H}$) as input, CID query oracle answers whether $m(A_1, \ldots, A_d) \neq 0$, where $m(A_1, \ldots, A_d)$ denotes the number of hyperedges in $\mathcal{H}$ having exactly one vertex in each A_i, $\forall i \in \{1, 2, \ldots, d\}$.

Notice the use of number of disjoint subsets in the definition of TIS and CID. That is why, we call TIS and CID as higher order query oracles than BIS and IS.

1.1 The open questions suggested by Beame et al. [4, 5]

For a work that has spawned many interesting results in such a short span of time, let us focus on the open problems and future research directions mentioned in [4, 5].

[1] Note that that, in RANDOMEDGE query, the probability space is the set of all edges. So, it is not actually a local query.

[2] Let us consider an IS query with input U. Let us partition U into two parts X and Y by putting each vertex in U to X or Y independently and uniformly at random. Then we make a BIS query with inputs X and Y, and report U is an independent set if and only if BIS reports that there is no edge with one endpoint in each of X and Y. Observe that we will be correct with at least probability $1/2$. We can boost up the probability by repeating the above procedure suitable number of times.

Table 1 The whole gamut of results involving Local queries [17], BIS, IS and its generalizations. † Δ is the maximum number of triangles on an edge. ‡ Both these results estimate the number of hyperedges in a d-uniform hypergraph, where d is treated as a constant. Here n, m and T denote the number of vertices, edges and triangles in a graph G, respectively. $\widetilde{\mathcal{O}}(\cdot)$ and $\widetilde{\Omega}(\cdot)$ hide a multiplicative factor of *poly*$(\log n, 1/\varepsilon)$ and $1/$*poly*$(\log n, 1/\varepsilon)$, respectively.

Work	Oracle used	Structure estimated	Upper bound Lower bound	Any other problem solved?
[19]	Local	edge	$\widetilde{\mathcal{O}}\left(\frac{n}{\sqrt{m}}\right)$	Approximating average distance.
			$\Omega\left(\frac{n}{\sqrt{m}}\right)$	
[11]	IS	edge	$\widetilde{\mathcal{O}}\left(\min\left\{\sqrt{m}, n/\sqrt{m}\right\}\right)$	–
			$\widetilde{\Omega}\left(\min\left\{\sqrt{m}, n/\sqrt{m}\right\}\right)$	–
[4]	BIS	edge	$poly(\log n, 1/\varepsilon) = \widetilde{\mathcal{O}}(1)$	Edge estimation using IS queries.
			–	
[7]	TIS	triangle	$poly(\log n, \Delta, 1/\varepsilon)^{\dagger}$	–
			–	–
[13], [6] ‡	CID	hyperedge	$poly(\log n, 1/\varepsilon) = \widetilde{\mathcal{O}}(1)$	[13] resolved Q2 in positive.
			–	
[14]	Local	triangle	$\widetilde{\mathcal{O}}\left(\frac{n}{T^{1/3}} + \min\left\{m, \frac{m^{3/2}}{T}\right\}\right)$	–
			$\Omega\left(\frac{n}{T^{1/3}} + \min\left\{m, \frac{m^{3/2}}{T}\right\}\right)$	–
[2]	Local+ Random Edge	triangle	$\widetilde{\mathcal{O}}\left(\min\left\{m, \frac{m^{3/2}}{T}\right\}\right)$	Estimated number of arbitrary subgraphs.
			$\Omega\left(\min\left\{m, \frac{m^{3/2}}{T}\right\}\right)$	
This work	BIS	triangle	$\widetilde{\mathcal{O}}\left(\min\left\{\frac{m}{\sqrt{T}}, \frac{m^{3/2}}{T}\right\}\right)$	–
			$\widetilde{\Omega}\left(\min\left\{\sqrt{T}, \frac{m^{3/2}}{T}\right\}\right)$	–

Q1 Can we estimate the number of cliques using polylogarithmic number of BIS queries?

Q2 Can polylogarithmic number of BIS queries sample an edge uniformly at random?

Q3 Can BIS or IS queries possibly be used in combination with local queries for graph parameter estimation problems?

Q4 What other oracles, besides subset queries, allow estimating graph parameters with a polylogarithmic number of queries?

Answers to the above questions and a discussion

Only Q2 has been resolved till now in the positive [13] as can be observed from Table 1. At its core, Q1 asks if a query oracle can step up, that is, if it can estimate a structure that is of a higher order than what the oracle was designed for. The framing of Q1 seems that Beame et al. expected a polylogarithmic query complexity for estimation of the number of cliques using BIS. Pertinent to these questions, we also want to bring to focus a work [22] where the authors mention that it seems to them that estimation of higher order structures will require higher order queries (see the discussion after Proposition 23 of [22]). They showed that $\Omega(n^2/\log n)$ BIS queries are required to separate *triangle free* graph instances from graph instances having at least one triangle. This lower bound follows directly from the communication complexity of triangle freeness testing [3]. However, the full complexity of triangle estimation using emptiness queries like BIS remains elusive. It seems to us that the observations in [4, 5] and [22] about the power of BIS in estimating higher order structures stand in contrast. In this backdrop, we have almost resolved the lower bound for estimating triangles using BIS queries, and our upper bound result show that even if BIS can not estimate triangles using polylogarithmic queries but it is still more powerful than LOCAL + RANDOM EDGE queries on graph for estimating triangles (see Table 1). BIS has an inherent asymmetry in its structure in the following sense – when BIS says that there exists no edge between two disjoint sets, then BIS stands as a witness to the existence of two sets of vertices having no interdependence, while a yes answer implies that there can be any number of edges, varying from one to the product of the cardinality of the two sets, going across the two sets. We feel that this property of BIS gives it its power, but on the other hand, also makes it difficult to prove lower bounds. That is probably the reason why works related to upper bound for BIS and its generalizations exist, whereas works on lower bound were not forthcoming. Though not on BIS, the work of Chen et al. using IS queries gave an interesting lower bound construction for IS oracle. Our work goes one step further in being able to prove a lower bound for the BIS oracle which is much stronger than IS oracle. We have almost resolved the open question Q1 by proving almost matching lower and upper bounds involving BIS for estimating triangles.

1.2 A stronger oracle than BIS, our main result and its consequences

Now we define EDGE EMPTINESS (shortened as EE) query oracle which is stronger than both BIS and IS. The EDGE EMPTINESS query is a form of a *subset query* [24, 25, 23] where a subset query with a subset $P \subseteq U$ asks whether $P \cap T$ is empty or not, where T is also a subset of the universe U. The EDGE EMPTINESS query operates with U being the set of all vertex pairs in G, T being the set of edges E in G, and P being a subset of pairs of vertices of V.

▶ **Definition 1.5** (EDGE EMPTINESS). Given a subset $P \subseteq \binom{V(G)}{2}$, a EE query answers whether there exists an $\{u, v\} \in P$ such that $\{u, v\}$ is an edge in G.

EE query is recently introduce by Assadi et al. [1].[3] Note that BIS and IS queries can be simulated by an EE query. [4] We prove our lower bound in terms of the *stronger*[5] EE queries that will directly imply the lower bound in terms of BIS. But we prove matching upper bound in terms of BIS. Our main results are stated below in an informal setting. The formal statements are given in Theorems 3.1 and A.1.

▶ **Theorem 1.6** (Main lower bound (informal statement)). *Let m, n, $t \in \mathbb{N}$. Any (randomized) algorithm that has* EE *query access to a graph $G(V, E)$ with n vertices and $\Theta(m)$ edges, requires $\widetilde{\Omega}\left(\min\left\{\sqrt{t}, \frac{m^{3/2}}{t}\right\}\right)$* EE *queries to decide whether the number of triangles in G is at most t or at least $2t$.*

▶ **Theorem 1.7** (An upper bound (informal statement)). *There exists an algorithm, that has* BIS *query access to a graph $G(V, E)$, finds a $(1 \pm \varepsilon)$-approximation to the number of triangles in G with high probabilility, and makes $\widetilde{\mathcal{O}}\left(\min\left\{\frac{m}{\sqrt{T}}, \frac{m^{3/2}}{T}\right\}\right)$* BIS *queries in expectaton. Here m, n, T denote the number of vertices, edges and triangles in G.*

[3] Assadi et al. [1] named EE query as OR query in their paper.

[4] Let us consider a BIS query with inpus A and B. Let P be the set of vertex pairs with one vertex from each of A and B. We call EE oracle with input P, and report there is an edge having one vertex in each of A and B if and only if the EE oracle reports that there exists an $\{u, v\} \in P$ that forms an edge in G. Similarly, we can simulate an IS query with input U by using an EE query with input $P = \binom{U}{2}$.

[5] For an example to show how EE query can be powerful than that of BIS or IS, $\mathcal{O}(\log \log n)$ EE queries [24] are enough to estimate the number of edges in a graph, as opposed to high query complexity (compared to $\mathcal{O}(\log \log n)$) when we have BIS or IS queries (See Table 1).

Note that Edge Emptiness query is the *strongest* subset query on edges of the graph. Informally speaking, our lower bound states that no subset query on edges can estimate the number of triangles in a graph by using polylogarithmic queries. However, the results of Bhattacharya et al. [6] and Dell et al. [13] imply that polylogarithmic TIS queries are enough to estimate the number of triangles in the graph. Note that TIS query is also a subset query on triangles in the graph. To complement our lower bound result, we also give an algorithm (see Theorem 1.7) for estimating the number of triangles in a graph with BIS queries that matches our lower bound. Here we would also like to mention that the number of BIS queries our algorithm uses is less than that of the number of local queries [17] needed to estimate the number of triangles in a graph. This implies that we are resolving Q3 in positive in the sense that BIS queries are efficient queries for triangle estimation vis-a-vis local queries [14] coupled with even random edge queries [2] (see Table 1).

1.3 Notations

Throughout the paper, the graphs are undirected and simple. For a graph $G(V,E)$, $V(G)$ and $E(G)$ denote the set of vertices and edges, respectively; $|V(G)| = n$, $|E(G)| = m$ and the number of triangles is T, unless otherwise specified. We use $\binom{V(G)}{2}$ to denote the set of vertex pairs in G. Note that $E(G) \subseteq \binom{V(G)}{2}$. For $P \subseteq \binom{V(G)}{2}$, $V(P)$ represents the set of vertices that belong to at least one pair in P. The neighborhood of a vertex $v \in V(G)$ is denoted by $N_G(v)$, and $|N_G(v)|$ is called the degree of vertex v in G. $\Gamma(\{x,y\})$ denotes the set $N_G(x) \cap N_G(y)$, that is, the set of common neighbors of x and y in G. If $e = \{x,y\} \in E(G)$, $\Gamma(e)$ denotes the set of vertices that forms triangles with e as one of their edges. The induced degree of a vertex v in $Z \subseteq V(G) \setminus \{v\}$ is the cardinality of $N_G(v) \cap Z$. For $X \subseteq V(G)$, the subgraph of G induced by X is denoted by $G[X]$. Note that $E(G[X]) = \{\{x,y\} : x,y \in X\}$. For two disjoint sets A, $B \subset V(G)$, the bipartite subgraph of G induced by A and B is denoted by $G[A,B]$. Note that $E(G[A,B])$ is the set of edges having one vertex in A and the other vertex in B.

Throughout the paper, $\varepsilon \in (0,1)$ is the approximation parameter. When we say a is a $(1\pm\varepsilon)$-approximation of b, then $(1-\varepsilon)b \leq a \leq (1+\varepsilon)b$. Polylogarithmic means *poly* $(\log n, 1/\varepsilon)$. $\widetilde{\mathcal{O}}(\cdot)$ and $\widetilde{\Omega}(\cdot)$ hide a multiplicative factor of *poly*$(\log n, 1/\varepsilon)$ and $1/$*poly*$(\log n, 1/\varepsilon)$, respectively. We have avoided floor and ceiling for simplicity of presentation. The constants in this paper are not taken optimally. We have taken them to let the calculation work with clarity. However, those can be changed to other suitable and appropriate constants.

1.4 Paper organization

We start with the technical overview of our lower and upper bounds in Section 2.1 and Section 2.2, respectively. The detailed lower and upper bound proofs are in Section 3 and Appendix A, respectively. The missing proofs of Section 3 are presented in the full version of the paper [10].

2 Technical overview

In this section, we discuss about the techniques and proof overview, The overview of the lower bound is discussed in Section 2.1 and that of the upper bound is discussed in Section 2.2.

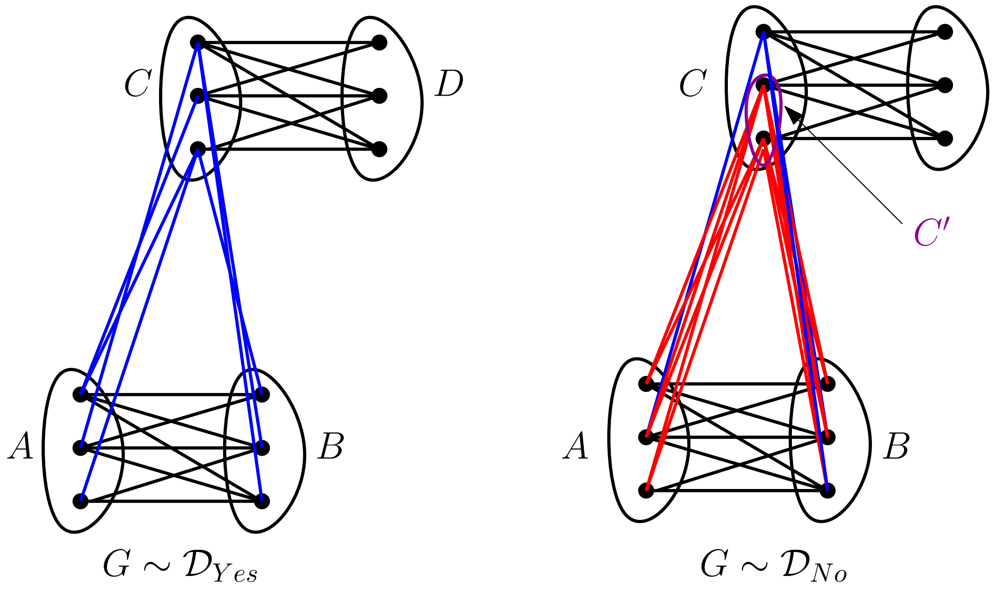


Figure 1 Illustration of $G \sim \mathcal{D}_{\mathbf{Yes}}$ and $G \sim \mathcal{D}_{\mathbf{No}}$ when $t = \Omega(m \log n)$.

2.1 Overview for the proof of our lower bound (Theorem 1.6)

Let us consider m, n, t as in Theorem 1.6. We prove the desired bound for BIS (stated in Theorem 1.6) by proving the lower bound is $\widetilde{\Omega}\left(\frac{m^{3/2}}{t}\right)$ when $t \geq km \log n$ and $\widetilde{\Omega}\left(\sqrt{t}\right)$ when $t < km \log n$ for EE query access, where k is a suitably chosen constant. In this Section, we discuss the overview of the proof when $t \geq km \log n$. The desired lower bound when $t < km \log n$ can be proved by using a reduction from the case when $t \geq km \log n$. The main intuition behind the lower bound is to "hide" a suitably generated vertex set such that a large number of queries is necessary to detect such a vertex.

The idea for the lower bound of $\widetilde{\Omega}\left(\frac{m^{3/2}}{t}\right)$ when $t \geq km \log n$

We prove by using Yao's method [21]. There are two distributions $\mathcal{D}_{\mathbf{Yes}}$ and $\mathcal{D}_{\mathbf{No}}$ (as described below) from which G is sampled satisfying $\mathbb{P}\left(G \sim \mathcal{D}_{\mathbf{Yes}}\right) = \mathbb{P}\left(G \sim \mathcal{D}_{\mathbf{No}}\right) = 1/2$. Note that, for each $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$, $|V(G)| = n = \Theta(\sqrt{m})$,[6] and $|E(G)| = \Theta(m)$ with a probability of at least $1 - o(1)$. But the number of triangles in each $G \sim \mathcal{D}_{\mathbf{No}}$ is at least two factor more than that of the number of triangles in any $G \sim \mathcal{D}_{\mathbf{Yes}}$, with a probability of at least $1 - o(1)$.

$\mathcal{D}_{\mathbf{Yes}}$: The vertex set $V(G)$ (with $|V(G)| = \Theta(\sqrt{m})$) is partitioned into four parts A, B, C, D uniformly at random. Vertex set A forms a biclique with vertex set B and vertex set C forms a biclique with vertex set D. Then every vertex pair $\{x, y\}$, with $x \in A \cup B$ and $y \in C$, is added as an edge to graph G with probability $\Theta\left(\sqrt{\frac{t}{m^{3/2}}}\right)$;

$\mathcal{D}_{\mathbf{No}}$: The vertex set $V(G)$ (with $|V(G)| = \Theta(\sqrt{m})$) is partitioned into four parts A, B, C, D uniformly at random. Vertex set A forms a biclique with vertex set B and vertex set C forms a biclique with vertex set D. Then every vertex pair $\{x, y\}$, with $x \in A \cup B$ and

[6] Without loss of generality, we assume that $\sqrt{m}$ is an integer. The proof can be extended to any graph having $n \geq \sqrt{m}$ vertices and m edges by adding $n - \sqrt{m}$ isolated vertices.

$y \in C$, is added as an edge to graph G with probability $\Theta\left(\sqrt{\frac{t}{m^{3/2}}}\right)$. Then each vertex of C is sampled with probability $\Theta\left(\frac{t}{m^{3/2}}\right)$. Let C' be the sampled set. Each vertex of C' is connected to every vertex of $A \cup B$ with an edge;

See Figure 1 for an illustration of the above construction. The constants, including k, in the order notations above are suitably set to have the following:

When $G \sim \mathcal{D}_{\text{Yes}}$: The number of triangles in each graph is at most t, with a probability of at least $1 - o(1)$;

When $G \sim \mathcal{D}_{\text{No}}$: $|C'| = \Theta\left(\frac{t}{m}\right)$ with a probability of at least $1 - o(1)$. Hence, the number of triangles in each $G \sim \mathcal{D}_{\mathbf{No}}$ is at least $2t$, with a probability of at least $1 - o(1)$.

Now, consider a particular EE query with input $P \subseteq \binom{V(G)}{2}$. Here, we divide the discussion into two parts, based on $|P| \leq \tau$ and $|P| > \tau$, where $\tau = \Theta(\log^2 n)$ is a threshold. If we query with the number of vertex pairs more than the threshold, chances are more we will not be able to distinguish between $G \sim \mathcal{D}_{\mathbf{Yes}}$ and $G \sim \mathcal{D}_{\mathbf{No}}$. When $|P| > \tau$, we can show that there exists a vertex pair $\{x, y\} \in P$ such that $\{x, y\}$ is an edge in G, with a probability of at least $1 - o(1)$, irrespective of whether $G \sim \mathcal{D}_{\mathbf{Yes}}$ or $G \sim \mathcal{D}_{\mathbf{No}}$. Intuitively, this is because the number of vertices and edges in G are $\Theta(\sqrt{m})$ and $\Theta(m)$, respectively. So, EE queries with input $P \subseteq \binom{V(G)}{2}$ such that $|P| > \tau$ will not be useful to distinguish whether $G \sim \mathcal{D}_{\mathbf{Yes}}$ or $G \sim \mathcal{D}_{\mathbf{No}}$.

We prove the desired lower bound by proving $\widetilde{\Omega}\left(\frac{m^{3/2}}{t}\right)$ EE queries are necessary to decide whether $G \sim \mathcal{D}_{\mathbf{Yes}}$ or $G \sim \mathcal{D}_{\mathbf{No}}$ with a probability of at least $2/3$. Note that $C' = \emptyset$ when $G \sim \mathcal{D}_{\mathbf{Yes}}$. So, the number of EE queries needed to decide whether $G \sim \mathcal{D}_{\mathbf{Yes}}$ or $G \sim \mathcal{D}_{\mathbf{No}}$, is at least the number of EE queries needed to *touch* at least one vertex of C' when $G \sim \mathcal{D}_{\mathbf{No}}$. Here, by touching at least a vertex of C', we mean $V(P) \cap C' \neq \emptyset$. As we have argued that only EE query with input $P \subseteq \binom{V(G)}{2}$ with $|P| \leq \tau$ can be useful, the probability that we touch a vertex in C' with such a query is at most $p = \mathcal{O}\left(\frac{C'}{\sqrt{m}} \cdot \tau\right) = \mathcal{O}\left(\frac{t \log^2 n}{m^{3/2}}\right)$. Hence the number of EE queries to touch at least a vertex of C', is at least $1/p$, that is, $\widetilde{\Omega}\left(\frac{m^{3/2}}{t}\right)$.

To let the the above discussion work, when $G \sim \mathcal{D}_{\mathbf{No}}$, $|C'| = \Theta\left(\frac{t}{m}\right)$ must be at least $\Omega(\log n)$. But $|C'| = \Theta\left(\frac{t}{m}\right)$ with a probability of at least $1 - o(1)$. Because of this, we take $t \geq km \log n$ in the above discussion. The formal statement of the lower bound, when $t \geq km \log n$, is given in Lemma 3.2 in Section 3. What we have discussed here is just an overview, the formal proof of Lemma 3.2 is much more invloved and delicate, which is presented in Section 3.1.

As we have already mentioned, the desired lower bound of $\widetilde{\Omega}\left(\sqrt{t}\right)$ when $t < km \log n$ can be proved by a reduction from the case when $t \geq km \log n$. The formal statement of the lower bound, when $t < km \log n$, is given in Lemma 3.3 in Section 3, and the proof is presented in Section 3.2.

2.2 Overview for our upper bound (Theorem 1.7)

We establish the upper bound claimed in Theorem 1.7 by giving two algorithms that report a $(1 \pm \varepsilon)$-approximation to the number of triangles in the graph:

(i) Triangle-Est-High(G, ε) that makes $\widetilde{\mathcal{O}}\left(\frac{m^{3/2}}{T}\right)$ BIS queries;

(ii) Triangle-Est-Low(G, ε) that makes $\widetilde{\mathcal{O}}\left(\frac{m+T}{\sqrt{T}}\right)$ BIS queries.

Informally speaking, our final algorithm Triangle-Est calls Triangle-Est-High(G, ε) and Triangle-Est-Low(G, ε) when $T = \Omega(m)$ and $T = \mathcal{O}(m)$, respectively. Observe that, if Triangle-Est knows T within a constant factor, then it can decide which one to use among Triangle-Est-High(G,ε) and Triangle-Est-Low(G,ε). If Triangle-Est does not know T within a constant factor, then it starts from a guess $L = \binom{n}{3}/2$ and makes a geometric search on L until the output of Triangle-Est is consistent with L. Depending on whether $L = \Omega(m)$ or $L = \mathcal{O}(m)$, Triangle-Est decides which one among Triangle-Est-High(G,ε) and Triangle-Est-Low(G,ε) to call. This guessing technique is very standard by now in property testing literature [19, 14, 15, 2]. Another point to note is that we do not know m. However, we can estimate m by using $poly(\log n)$ BIS queries (see Table 1). An estimate of m will perfectly work for us in this case.

Algorithm Triangle-Est-High(G,ε)

Algorithm Triangle-Est-High is inspired by the triangle estimation algorithm of Assadi et al. [2], where we have Adjacency, Degree, Random neighbor and Random Edge queries. Please see Appendix A.2 for formal definitions of these queries. Note that the algorithm by Assadi et al. can be *suitably* modified even if we have approximate versions of Degree, Random neighbor and Random Edge queries. Also refer Appendix A.2 for formal definitions of approximate version of the above queries. By Corollary A.8, $\widetilde{\mathcal{O}}(1)$ BIS queries are enough to simulate the approximate versions of Degree and Random neighbor, with a probability of at least $1 - o(1)$. By Proposition A.7, approximate version of Random Edge queries can also be simulated by $\widetilde{\mathcal{O}}(1)$ BIS queries, with a probability of at least $1 - o(1)$. Putting everything together, we get Triangle-Est-High(G,ε) for triangle estimation that makes $\widetilde{\mathcal{O}}(1)$ BIS queries. The formal statement of the corresponding triangle estimation result is given in Lemma A.2, and algorithm Triangle-Est-High(G,ε) is described in Appendix A.2.

Algorithm Triangle-Est-Low(G,ε)

This algorithm is inspired by the two pass streaming algorithm for triangle estimation by McGregor et al. [20]. Basically, we show that the steps of McGregor et al.'s algorithm can be executed by using $\widetilde{\mathcal{O}}\left(\frac{m+T}{\sqrt{T}}\right)$ BIS queries. To do so, we have used the fact that, given any $X \subseteq V(G)$, all the edges of the subgraph induced by X can be enumerated by using $\widetilde{\mathcal{O}}\left(|E(G[X])|\right)$ BIS queries (see Proposition A.4 for the formal statement). The formal statement of the corresponding triangle estimation result is given in Lemma A.3, and algorithm Triangle-Est-Low(G,ε) is described in Appendix A.3 along with its correctness proof and query complexity analysis.

3 Lower bound for estimating triangles using Edge Emptiness queries

In this Section, we prove the main lower bound result as sketched in Theorem 1.6; the formal theorem statement is stated below. As mentioned earlier, the lower bound proofs will be for the stronger query oracle EE. This will imply the lower bound for BIS.

▶ **Theorem 3.1** (Main lower bound result). *Let $m, n, t \in \mathbb{N}$ be such that $1 \leq t \leq \frac{m^{3/2}}{2}$. Any (randomized) algorithm that has* EE *oracle access to a graph $G(V,E)$ must make $\widetilde{\Omega}\left(\min\left\{\sqrt{t}, \frac{m^{3/2}}{t}\right\}\right)$* EE *queries to decide whether the number of triangles in G is at most t or at least $2t$ with a probability of at least $2/3$, where G has $n \geq 4\sqrt{m}$ vertices and $\Theta\left(m\right)$ edges.*

We prove the above theorem by proving Lemmas 3.2 and 3.3, as stated below. Note that Lemmas 3.2 and 3.3 talk about the desired lower bound when the number of triangles in the graph is *large* ($\Omega(m \log n)$) and *small* ($\mathcal{O}(m \log n)$), respectively.

▶ **Lemma 3.2** (Lower bound when there are *large* number of triangles)**.** *Let m, n, $t \in \mathbb{N}$ be such that $t \geq \frac{m \log n}{8}$. Any (randomized) algorithm that has* EE *oracle access to a graph $G(V, E)$ must make $\widetilde{\Omega}\left(\frac{m^{3/2}}{t}\right)$* EE *queries to decide whether the number of triangles in G is at most t or at least $2t$ with a probability of at least $2/3$, where G has $n \geq 4\sqrt{m}$ vertices, $\Theta(m)$ edges.*

▶ **Lemma 3.3** (Lower bound when there are *small* number of triangles)**.** *Let m, n, $t \in \mathbb{N}$ be such that $t < \frac{m \log n}{8}$. Any (randomized) algorithm that has* EE *oracle access to a graph $G(V, E)$ must make $\widetilde{\Omega}\left(\sqrt{t}\right)$* EE *queries to decide whether the number of triangles in G is at most t or at least $2t$ with a probability of at least $2/3$, where G has $n \geq 4\sqrt{m}$ vertices and $\Theta(m)$ edges.*

We first show Lemma 3.2 in Section 3.1, and then Lemma 3.3 in Section 3.2. Note that the proof of Lemma 3.3 will use Lemma 3.2.

3.1 Proof of Lemma 3.2

Without loss of generality, assume that $\sqrt{m}$ is an integer. We prove for the case when $n = 4\sqrt{m}$. But, we can make the proof work for any $n \geq 4\sqrt{m}$ by adding $n - 4\sqrt{m}$ isolated vertices. Note that $t \geq \frac{m \log n}{8}$ here. We further assume that $t \leq \frac{m^{3/2}}{128}$, and $m = \Omega(\log^2 n)$. Otherwise, the stated lower bound of $\widetilde{\Omega}\left(\frac{m^{3/2}}{t}\right)$ trivially follows as $\widetilde{\Omega}(\cdot)$ hides a multiplicative factor of $\frac{1}{poly(\log n)}$.

We use Yao's min-max principle to prove the lower bound. To do so, we consider two distributions $\mathcal{D}_{\mathbf{Yes}}$ and $\mathcal{D}_{\mathbf{No}}$ on graphs where

- Any graph $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$ has $4\sqrt{m}$ vertices;
- Any graph $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$ has $\Theta(m)$ edges with a probability of at least $1 - o(1)$;
- The number of triangles in any graph $G \sim \mathcal{D}_{\mathbf{Yes}}$ is at most t with a probability of at least $1 - o(1)$, and any graph $G \sim \mathcal{D}_{\mathbf{No}}$ has at least $2t$ triangles with a probability of at least $1 - o(1)$.

Note that, if we can show that any deterministic algorithm that distinguishes graphs from $\mathcal{D}_{\mathbf{Yes}}$ and $\mathcal{D}_{\mathbf{No}}$, with a probability of at least $2/3$, must make $\widetilde{\Omega}\left(\frac{m^{3/2}}{t}\right)$ EE queries, then we are done with the proof of Lemma 3.2.

3.1.1 The (hard) distribution for the input, its properties, and the proof set up

$\mathcal{D}_{\text{Yes}}$: A graph $G \sim \mathcal{D}_{\mathbf{Yes}}$ is sampled as follows:

- Partition the vertex set $V(G)$ into 4 parts A, B, C, D, by initializing A, B, C, D as empty sets, and then putting each vertex in $V(G)$ into one of the parts uniformly at random and independent of other vertices;
- Connect each vertex of A with every vertex of B with an edge to form a biclique. Also, connect each vertex of C with every vertex of D with an edge to form another biclique;
- For every $\{x, y\}$ where $x \in A \cup B$ and $y \in C$, add edge $\{x, y\}$ to G with probability $\sqrt{\frac{t}{16m^{3/2}}}$.

$\mathcal{D}_{\text{No}}$: A graph $G \sim \mathcal{D}_{\mathbf{No}}$ is sampled as follows:

- Partition the vertex set $V(G)$ into 4 parts A, B, C, D, by initializing A, B, C, D as empty sets, and then putting each vertex in $V(G)$ into one of the partitions uniformly at random and independent of other vertices;
- Connect each vertex of A with every vertex of B with an edge to form a biclique. Also, connect each vertex of C with every vertex of D with an edge to form another biclique;

- For every $\{x, y\}$ where $x \in A \cup B$ and $y \in C$, add edge $\{x, y\}$ to G with probability $\sqrt{\frac{t}{16m^{3/2}}}$.
- Select $C' \subseteq C$ by putting each $x \in C$ into C' with a probability of at least $\frac{32t}{m^{3/2}}$, independently, and then, add each edge in $\{x, y : x \in A \cup B, y \in C'\}$ to G.

The following observation establishes the number of vertices, edges, and the number of triangles in the graphs that can be sampled from $\mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$. The proof uses *large deviation inequalities* and is presented in the full version of the paper.

▶ **Observation 3.4** (Properties of the graph $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$).

(i) For $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$, the number of vertices in G is $4\sqrt{m}$. Also, $\frac{\sqrt{m}}{2} \le |A|, |B|, |C|, |D| \le 2\sqrt{m}$ holds with a probability of at least $1 - o(1)$, and the number of edges in G is $\Theta(m)$ with a probability of at least $1 - o(1)$;

(ii) If $G \sim \mathcal{D}_{\mathbf{Yes}}$, then there are at most t triangles in G with a probability of at least $1 - o(1)$,

(iii) If $G \sim \mathcal{D}_{\mathbf{No}}$, $\frac{8t}{m} \le |C'| \le \frac{64t}{m}$ with a probability of at least $1 - o(1)$, and there are at least $2t$ triangles in G with a probability of at least $1 - o(1)$.

The following remark is regarding the connection between graphs in $\mathcal{D}_{\mathbf{Yes}}$ and that in $\mathcal{D}_{\mathbf{No}}$. This will be used later in our proof, particularly in the proof of Claim 3.12.

▶ **Remark 1** (A graph $G' \sim \mathcal{D}_{\mathbf{No}}$ can be generated from a graph $G \sim \mathcal{D}_{\mathbf{Yes}}$). Let us first generate a graph $G \sim \mathcal{D}_{\mathbf{Yes}}$. Select $C' \subseteq C$ by putting each $x \in C$ into C' with a probability of at least $\frac{32t}{m^{3/2}}$, and then, add each edge in $\{x, y : x \in A \cup B, y \in C'\}$ to G to generate G', then (the resulting graph) $G' \sim \mathcal{D}_{\mathbf{No}}$.

The following observation says that a $\{x, y\} \in \binom{V(G)}{2}$ (with some condition) forms an edge with a probability of at least a constant. It will be used while we prove Claim 3.11.

▶ **Observation 3.5** (Any vertex pair $\{x, y\}$ is an edge in G with constant probability). Let $\{x, y\} \in \binom{V(G)}{2}$, and we are in the process of generating $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$. Let at most one of x and y has been put into one of the parts out of A, B, C and D. Then $\{x, y\}$ is an edge in G with probability at least $\frac{1}{4}$.

The above observation follows from the description of $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$ – each vertex in $V(G)$ is put into one of the parts out of A, B, C, D uniformly at random, each vertex of A is connected with every vertex in B, and each vertex of C is connected with every vertex in D.

In order to prove Lemma 3.2, by contradiction, assume that there is a randomized algorithm that makes $q = o\left(\frac{m^{3/2}}{t}\frac{1}{\log^2 n}\right)$ EE queries and decides whether the number of triangles in the input graph is at most t or at least $2t$, with a probability of at least 2/3. Then there exists a deterministic algorithm ALG that makes q EE queries and decides the following (when the input graph $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$ be such that both $G \sim \mathcal{D}_{\mathbf{Yes}}$ and $G \sim \mathcal{D}_{\mathbf{No}}$ holds with probability 1/2) –

$$\mathbb{P}_{G \sim \mathcal{D}_{\mathbf{No}}}(\text{ALG(G) reports NO}) - \mathbb{P}_{G \sim \mathcal{D}_{\mathbf{Yes}}}(\text{ALG(G) reports NO}) \ge \frac{1}{3} - o(1).$$

(Here $\mathbb{P}_{G \sim \mathcal{D}_{\mathbf{No}}}(\mathcal{E})$ and $\mathbb{P}_{G \sim \mathcal{D}_{\mathbf{Yes}}}(\mathcal{E})$ denote the probability of the event $\mathcal{E}$ under the conditional space $G \sim \mathcal{D}_{\mathbf{No}}$ and $G \sim \mathcal{D}_{\mathbf{Yes}}$, respectively.) Hence, we will be done with the proof of Lemma 3.2 by showing the following lemma.

▶ **Lemma 3.6** (Lower bound on the number of EE queries when $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$)**.** *Let the unknown graph G be such that $G \sim \mathcal{D}_{\mathbf{Yes}}$ and $G \sim \mathcal{D}_{\mathbf{No}}$ hold with equal probabilities. Consider any deterministic algorithm* ALG *that has* EE *access to G, and makes $q = o\left(\frac{m^{3/2}}{t}\frac{1}{\log^2 n}\right)$* EE *queries to G. Then*

$$\mathbb{P}_{G\sim\mathcal{D}_{\mathbf{No}}}(\text{ALG}(G) \textit{ reports } \text{NO}) - \mathbb{P}_{G\sim\mathcal{D}_{\mathbf{Yes}}}(\text{ALG}(G) \textit{ reports } \text{NO}) \le o(1).$$

Next, we define an augmented EE oracle (EE* oracle). EE* is tailor-made for the graphs coming from $\mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$. Moreover, it is *stronger* than EE, that is, any EE query can be simulated by a EE* query. We will prove the claimed lower bound in Lemma 3.6 when we have access to EE* oracle. Note that this will imply Lemma 3.6.

Before getting into the formal description of EE* oracle, note that the algorithm (with EE* access) maintains a four tuple data structure initialized with $\emptyset$. With each query to EE* oracle, the oracle updates the data structure and returns the updated data structure to the algorithm. Note that the updated data structure is a function of all previously made EE* queries, and it is enough to answer corresponding EE queries.

3.1.2 Augmented Edge Emptiness oracle (EE*)

Before describing the EE* query oracle and its interplay with the algorithm, first we present the data structure $(E_Q, V(E_Q), e, \ell_v)$ that the algorithm maintains with the help of EE* oracle. The data structure keeps track of the following information.

Information maintained by $(E_Q, V(E_Q), e, \ell_v)$

- E_Q is a subset of $\binom{V(G)}{2}$ that have been seen by the algorithm till now, $V(E_Q)$ is the set of vertices present in any vertex pair in E_Q.
- $e : \binom{V(E_Q)}{2} \to \{0,1\}$ such that $e(\{x,y\}) = 1$ means the algorithm knows that $\{x,y\}$ is an edge in G, $e(x,y) = 0$ means that $\{x,y\}$ is not an edge in G.
- $\ell_v : V(E_Q) \to \{A, B, C, C', D\}$, where

$$\ell_v(x) = \begin{cases} A, & x \in A \\ B, & x \in B \\ C, & x \in C \setminus C' \\ C', & x \in C' \\ D, & x \in D \end{cases}$$

Intuitively speaking, unless the algorithm knows about the presence of some vertex in C', it cannot distinguish whether the unknown graph $G \sim \mathcal{D}_{\mathbf{Yes}}$ or $G \sim \mathcal{D}_{\mathbf{No}}$. So, we define the notion of good and bad vertices, along with good and bad data structures. This notion will be used later in our proof.

▶ **Definition 3.7** (Bad vertex)**.** A vertex $x \in V(E_Q)$ is said to be a *bad* vertex if $\ell_v(x) = C'$. $(E_Q, V(E_Q), e, \ell_v)$ is said to be *good* if there does not exist any bad vertex in $V(E_Q)$.

EE* oracle and its interplay with the algorithm

The algorithm initializes the data structure $(E_Q, V(E_Q), e, \ell_v)$ with $E_Q = \emptyset$, $V(E_Q) = \emptyset$. So, e and ℓ_v are initialized with trivial functions with domain $\emptyset$. At the beginning of each round, the algorithm queries the EE* oracle with a subset $P \subseteq \binom{V(G)}{2}$ deterministically. Note that the choice of P depends on the current status of the data structure. Now, we explain how EE* oracle responds to the query and how the data structure is updated.

(1) If $|P| \leq \tau = 25\log^2 n$, the oracle sets $E_Q \leftarrow E_Q \cup P$, and changes $V(E_Q)$ accordingly. The oracle also sets the function e and ℓ_v as per their definitions, and then it sends the updated data structure to the algorithm.

(2) Otherwise (if $|P| > \tau$), the oracle finds a random subset $P' \subseteq P$ such that $|P'| = \tau$. The oracle checks if there is a pair $\{u, v\} \in P'$ such that $\{u, v\}$ is an edge. If yes, then the oracle responds as in (1) with P being replaced by P'. If no, the oracle sends the data structure corresponding to the entire graph along with a FAILURE signal.[7]

Owing to the way EE* oracle updates the data structure after each EE* query, we can make some assumptions on the inputs to the EE* oracle, as described in Remark 2. It will actually be useful when we prove Claim 3.11.

▶ **Remark 2** (Some assumptions on the EE* query). Let $(E_Q, V(E_Q), e, \ell_v)$ be the data structure just before the algorithm makes EE query with input P, and let $\left(E'_Q, V\left(E'_Q\right), e', \ell'_v\right)$ be the data structure updated by EE* oracle after the algorithm makes EE* query with input P. Without loss of generality, we assume that

(i) P is disjoint from $\binom{V(E_Q)}{2}$. It is because EE* maintains whether $\{x, y\}$ is an edge or not for each $\{x, y\} \in \binom{V(E_Q)}{2}$;

(ii) When $x, z \in V(E_Q)$, there does not exist $\{x, y\}$ and $\{y, z\}$ in P. It is because the oracle updates the data structure in the same way in each of the following three cases when $x, z \in V(E_Q)$ – (i) $\{x, y\}$ and $\{y, z\}$ are in P, (ii) $\{x, y\} \in P$ and $\{y, z\} \notin P$, and (iii) $\{x, y\} \notin P$ and $\{y, z\} \in P$. By the description of EE* oracle and its interplay with the algorithm, in all the three cases, the updated data structure $\left(E'_Q, V\left(E'_Q\right), e', \ell'_v\right)$ contains labels of all the three vertices x, y, z along with the information whether $\{x, y\}$ and $\{y, z\}$ form edges in G or not. So, instead of having both $\{x, y\}$ and $\{y, z\}$ in P with $x, z \in V(E_Q)$, it is *equivalent* to have exactly one among $\{x, y\}$ and $\{y, z\}$ in P.

In the following observation, we formally show that EE* oracle is stronger than that of EE. Then we prove Lemma 3.9 that says that $\Omega\left(\frac{m^{3/2}}{t}\frac{1}{\log^2 n}\right)$ EE* queries are necessary to distinguish between $G \sim \mathcal{D}_{\mathbf{Yes}}$ and $G \sim \mathcal{D}_{\mathbf{No}}$. Note that Lemma 3.9 will imply Lemma 3.6.

▶ **Observation 3.8** (EE* is stronger than EE). Let $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$. Each EE query to G can be simulated by using an EE* query to G.

Proof. Let us consider an EE query with input $P \subseteq \binom{V(G)}{2}$. We make a EE* query with the same input P, and answer the EE query as follows depending on whether $|P| \leq \tau$ or $|P| > \tau$.

$|P| \leq \tau$: The EE* oracle updates the data structure and let $\left(E'_Q, V\left(E'_Q\right), e', \ell'_v\right)$ be the updated data structure. It contains the the information about each $\{x, y\} \in P$ whether it forms an edge in G or not. So, from $\left(E'_Q, V\left(E'_Q\right), e', \ell'_v\right)$, the EE query with input P can be answered as follows: there exists an edge $\{x, y\} \in P$ with $\{x, y\} \in E(G)$ if and only if $e'(\{x, y\}) = 1$.

$|P| > \tau$: In this case, the EE* oracle finds a random subset $P' \subseteq P$ such that $|P'| = \tau$. It checks if there is an $\{x, y\} \in P'$ such that $\{x, y\}$ is an edge. If yes, the updated data structure contains the the information about each $\{x, y\} \in P'$ whether it forms an edge. In this case, we can report that there exists an $\{x, y\} \in P$ such that $\{x, y\}$ is an edge in G. If there is no $\{x, y\} \in P'$ such that $\{x, y\}$ is an edge, then (by the description of

[7] We later argue that FAILURE signal is sent with a very low probability.

EE oracle and its interplay with the algorithm) the EE* oracle sends the data structure corresponding to the entire graph. Obviously, we can report whether there exists an $\{x, y\} \in P$ such that $\{x, y\} \in E(G)$ or not.

Hence, in any case, we can report the answer to EE query with input P. ◀

We are left with proving the following technical lemma. As noted earlier, this will imply Lemma 3.6.

▶ **Lemma 3.9** (Lower bound on the number of EE* queries when $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$)**.** *Let the unknown graph G be such that $G \sim \mathcal{D}_{\mathbf{Yes}}$ and $G \sim \mathcal{D}_{\mathbf{No}}$ hold with equal probabilities. Consider any deterministic algorithm* ALG* *that has* EE* *access to G, and makes $q = o\left(\frac{m^{3/2}}{t}\frac{1}{\log^2 n}\right)$* EE* *queries to G. Then*

$$\mathbb{P}_{G \sim \mathcal{D}_{\mathbf{No}}}(\text{ALG}^* \textit{ (G) reports } \text{NO}) - \mathbb{P}_{G \sim \mathcal{D}_{\mathbf{Yes}}}(\text{ALG}^* \textit{ (G) reports } \text{NO}) \leq o(1).$$

3.1.3 Proof of Lemma 3.9

For clarity of explanation, we first describe ALG* as a decision tree. Then we will prove Lemma 3.9.

Decision tree view of ALG*

- Each internal node of $\mathcal{T}$ is labeled with a nonempty subset $\binom{V(G)}{2}$ and each leaf node is labeled with YES or NO;
- Each edge in the tree is labeled with a data structure $(E_Q, V(E_Q), e, \ell_v)$;
- The algorithm starts the execution from the root node r by setting r as the current node. Note that for the root node r, $E_Q = V(E_Q) = \emptyset$ and e and ℓ_v are the trivial functions. As the algorithm ALG* is deterministic, the first EE* query is same irrespective of the graph $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$ that we are querying. By making that query, we get an updated data structure from the oracle and let $\{r, u\}$ be the edge that is labeled with the updated data structure. Then ALG* sets u as the current node.
- If the current node u is not a leaf node in $\mathcal{T}$, ALG* makes a EE* query with a subset $P \subseteq \binom{V(G)}{2}$, where P is determined by the label of the node u. Note that P satisfies the condition described in Remark 2. The oracle updates the knowledge structure and ALG* moves to a child of u depending on the updated data structure;
- If the current node u is a leaf node in $\mathcal{T}$, report YES or NO according to the label of u.

Now, we define the notion of *good* and *bad* nodes in $\mathcal{T}$. The following definition is inspired from Definition 3.7.

▶ **Definition 3.10** (Bad node in the decision tree)**.** Let u be a node of $\mathcal{T}$ and $(E_Q, V(E_Q), e, \ell_v)$ be the current data structure. u is said to be *good* if there does not exist x in $V(E_Q)$ such that $\ell_v(x) = C'$. Otherwise, u is a *bad* node.

If $G \sim \mathcal{D}_{\mathbf{Yes}}$, then ALG* will never encounter a bad node. In other words, when ALG* reaches a bad node of the tree $\mathcal{T}$, then it can (easily) decide $G \sim \mathcal{D}_{\mathbf{No}}$. However, the inverse in not true. From this fact, consider two claims (Claims 3.11 and 3.12) about the traversal of the decision tree $\mathcal{T}$ when the graph $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$. These claims will be useful to show Lemma 3.9. Intuitively, Claim 3.11 says that the probability of reaching a bad node is very low when $G \sim \mathcal{D}_{\mathbf{No}}$. Claim 3.12 says that the probability to reach any particular good node is more when $G \sim \mathcal{D}_{\mathbf{Yes}}$ as compared to that of when $G \sim \mathcal{D}_{\mathbf{No}}$.

▶ **Claim 3.11** (Probability of reaching a bad node is very low when $G \sim \mathcal{D}_{\mathbf{No}}$). *Let $G \sim \mathcal{D}_{\mathbf{No}}$. Then the probability that* ALG* *reaches a bad node of the decision tree is $o(1)$. That is,*

$$\mathbb{P}_{G \sim \mathcal{D}_{\mathbf{No}}}(\text{ALG}^* \textit{ reaches a bad node}) = o(1).$$

▶ **Claim 3.12** (A technical claim to prove Lemma 3.9). *For any good node in the decision tree $\mathcal{T}$, the following holds:*

$$\mathbb{P}_{G \sim \mathcal{D}_{\mathbf{No}}}(\text{ALG}^* \textit{ reaches } v) \le \mathbb{P}_{G \sim \mathcal{D}_{\mathbf{Yes}}}(\text{ALG}^* \textit{ reaches } v).$$

The proofs of Claims 3.11 and 3.12 are non trivial and are in the full version due to paucity of space.

Now, we will prove Lemma 3.9.

Proof of Lemma 3.9. Let $\mathcal{L}_{\mathbf{No}}$ denote the set of leaf nodes of the decision tree $\mathcal{T}$ that are labeled NO. Also, let $\mathcal{L}_g \subseteq \mathcal{L}_{\mathbf{No}}$ be the set of leaf nodes that are good and labeled as NO.

$$\begin{aligned}
&\mathbb{P}_{G \sim \mathcal{D}_{\mathbf{No}}}(\text{ALG}^* \text{ (G) reports NO})\\
&\le \sum_{v \in \mathcal{L}_{\mathbf{No}}} \mathbb{P}_{G \sim \mathcal{D}_{\mathbf{No}}}(\text{ALG}^* \ (G) \text{ reaches } u)\\
&= \sum_{u \in \mathcal{L}_g} \mathbb{P}_{G \sim \mathcal{D}_{\mathbf{No}}}(\text{ALG}^* \ (G) \text{ reaches } u) + \sum_{u \in \mathcal{L}_{\mathbf{No}} \setminus \mathcal{L}_g} \mathbb{P}_{G \sim \mathcal{D}_{\mathbf{No}}}(\text{ALG}^* \ (G) \text{ reaches } u)\\
&\le \sum_{u \in \mathcal{L}_g} \mathbb{P}_{G \sim \mathcal{D}_{\mathbf{No}}}(\text{ALG}^* \ (G) \text{ reaches } u) + \mathbb{P}_{G \sim \mathcal{D}_{\mathbf{No}}}(\text{ALG}^* \ (G) \text{ reaches a bad node})\\
&\le \sum_{u \in \mathcal{L}_g} \mathbb{P}_{G \sim \mathcal{D}_{\mathbf{Yes}}}(\text{ALG}^* \ (G) \text{ reaches } u) + o(1) \qquad \text{(By Claims 3.11 and Claims 3.12)}\\
&\le \mathbb{P}_{G \sim \mathcal{D}_{\mathbf{Yes}}}(\text{ALG}^* \ (G) \text{ reports NO}) + o(1)
\end{aligned}$$

◀

3.2 Proof of Lemma 3.3

We assume that $t = \omega(\log^7 n)$. Otherwise, as $\widetilde{\Omega}(\cdot)$ hides a multiplicative term of $\frac{1}{poly(\log n)}$, the stated lower bound is trivial. Assume, for a contradiction, that there is an algorithm $\mathcal{A}$ for $t < \frac{m \log n}{8}$ such that

- it has EE oracle access to a graph $G_1(V_1, E_1)$ with $\Theta(\sqrt{m})$ vertices and $\Theta(m)$ edges;
- makes $o\left(\sqrt{t}\frac{1}{\log^{3.5} n}\right)$ EE queries;
- decides whether the number of triangles in G_1 is at most t or at least $2t$ with a probability of at least $2/3$.

Now we give an algorithm $\mathcal{A}'$ for

- it has EE oracle access to a graph $G_2(V_2, E_2)$ with $4\sqrt{t/\log n}$ vertices and $8t/\log n$ edges, that is, $t = \frac{|E_2| \log n}{8}$;
- makes $o\left(\frac{\sqrt{t}}{\log^{3.5} n}\right) = o\left(\frac{|E_2|^{3/2}}{t}\frac{1}{\log^2 n}\right)$ EE queries;
- decides whether the number of triangles in G_2 is at most t or at least $2t$ with a probability of at least $2/3$.

Description of $\mathcal{A}'$ using $\mathcal{A}$

- Let the unknown graph be $G_1 = G_2 \cup G'$ such that $V(G_1) = V(G_2) \sqcup V(G')$ and $E(G_1) = E(G_2) \sqcup E(G')$, where G' is a graph having $\Theta(\sqrt{m - t/\log n})$ vertices (disjoint from $V(G_2)$), $\Theta\left(\sqrt{(m - 8t/\log n)}\right)$ edges, and no triangles.[8] The number of vertices and edges in G_1 are $\Theta(\sqrt{m})$ and $\Theta(m)$, respectively. Also, the number of triangles in G_1 is same as in G_2;
- As an EE query to G_1 can be answered using one EE query to G_2, we can consider having EE query access to graph G_1;
- We run algorithm $\mathcal{A}$ assuming G_1 as the unknown graph;
- We report the output of $\mathcal{A}$ as the output of $\mathcal{A}$.

The correctness of $\mathcal{A}'$ follows from the correctness of $\mathcal{A}$. The number of queries made by the algorithm $\mathcal{A}'$ is $o\left(\sqrt{t}\frac{1}{\log^{3.5} n}\right)$. Recalling that $\mathcal{A}$ works on graph $G_2(V_2, E_2)$ satisfying $t = \frac{|E_2| \log n}{8}$ and by Lemma 3.2, algorithm $\mathcal{A}'$ does not exist as such an algorithm requires at least $\Omega\left(\frac{|E_2|^{3/2}}{t}\frac{1}{\log^2 n}\right)$ EE queries, which is $\Omega\left(\frac{\sqrt{t}}{\log^{3.5} n}\right)$.

Hence, we are done with the proof of Lemma 3.3.

4 Conclusion

We touched upon two open questions of Beame et al. [5] in this paper. We resolved the query complexity of triangle estimation when we have a Bipartite Independent Set oracle access to the unknown graph when $T = \Omega(m)$. But the query complexity of triangle counting remain illusive when $T = o(m)$ though we believe that our upper bound of $\widetilde{\mathcal{O}}(m/\sqrt{T})$ BIS queries is tight in this regard. It is also interesting if our upper bound can be improved when $T = o(m)$.

References

1 Sepehr Assadi, Deeparnab Chakrabarty, and Sanjeev Khanna. Graph Connectivity and Single Element Recovery via Linear and OR Queries. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume bisbisbisbis204 of *LIPIcs*, pages 7:1–7:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ESA.2021.7.

2 Sepehr Assadi, Michael Kapralov, and Sanjeev Khanna. A Simple Sublinear-Time Algorithm for Counting Arbitrary Subgraphs via Edge Sampling. In *Proceedings of the 10th Innovations in Theoretical Computer Science Conference, ITCS*, volume 124, pages 6:1–6:20, 2019.

3 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, SODA*, pages 623–632, 2002.

4 Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge Estimation with Independent Set Oracles. In *Proceedings of the 9th Innovations in Theoretical Computer Science Conference, ITCS*, volume 94, pages 38:1–38:21, 2018.

5 Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge Estimation with Independent Set Oracles. *ACM Trans. Algorithms*, 16(4):52:1–52:27, 2020.

[8] The constants in $\Theta(\sqrt{m - 8t/\log n})$ and $\Theta(m - 8t/\log n)$ are chosen suitably such that the graph G_1 satisfies the requirement of $\mathcal{A}$.

6 Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Hyperedge Estimation using Polylogarithmic Subset Queries. *CoRR*, abs/1908.04196, 2019.

7 Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Triangle Estimation Using Tripartite Independent Set Queries. In *Proceedings of the 30th International Symposium on Algorithms and Computation, ISAAC*, volume 149, pages 19:1–19:17, 2019.

8 Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. On Triangle Estimation Using Tripartite Independent Set Queries. *Theory Comput. Syst.*, 65(8):1165–1192, 2021.

9 Arijit Bishnu, Arijit Ghosh, Sudeshna Kolay, Gopinath Mishra, and Saket Saurabh. Parameterized Query Complexity of Hitting Set Using Stability of Sunflowers. In *Proceedings of the 29th International Symposium on Algorithms and Computation, ISAAC*, volume 123, pages 25:1–25:12, 2018.

10 Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. On the Complexity of Triangle Counting using Emptiness Queries. *CoRR*, abs/2110.03836, 2021. `arXiv:2110.03836`.

11 Xi Chen, Amit Levi, and Erik Waingarten. Nearly Optimal Edge Estimation with Independent Set Queries. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2916–2935, 2020.

12 Holger Dell and John Lapinskas. Fine-Grained Reductions from Approximate Counting to Decision. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 281–288, 2018.

13 Holger Dell, John Lapinskas, and Kitty Meeks. Approximately Counting and Sampling Small Witnesses Using a Colourful Decision Oracle. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2201–2211, 2020.

14 Talya Eden, Amit Levi, Dana Ron, and C. Seshadhri. Approximately Counting Triangles in Sublinear Time. *SIAM J. Comput.*, 46(5):1603–1646, 2017.

15 Talya Eden, Dana Ron, and C. Seshadhri. On Approximating the Number of k-Cliques in Sublinear Time. *SIAM J. Comput.*, 49(4):747–771, 2020.

16 Uriel 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.

17 Oded Goldreich. *Introduction to Property Testing.* Cambridge University Press, 2017.

18 Oded Goldreich and Dana Ron. Property Testing in Bounded Degree Graphs. *Algorithmica*, 32(2):302–343, 2002.

19 Oded Goldreich and Dana Ron. Approximating Average Parameters of Graphs. *Random Struct. Algorithms*, 32(4):473–493, 2008.

20 Andrew McGregor, Sofya Vorotnikova, and Hoa T. Vu. Better Algorithms for Counting Triangles in Data Streams. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS*, pages 401–411, 2016.

21 Rajeev Motwani and Prabhakar Raghavan. Randomized Algorithms. In *Algorithms and Theory of Computation Handbook*, Chapman & Hall/CRC Applied Algorithms and Data Structures series. CRC Press, 1999.

22 Cyrus Rashtchian, David P. Woodruff, and Hanlin Zhu. Vector-Matrix-Vector Queries for Solving Linear Algebra, Statistics, and Graph Problems. In *Proceedings of the 24th International Conference on Randomization and Computation, RANDOM*, volume 176, pages 26:1–26:20, 2020.

23 Dana Ron and Gilad Tsur. The Power of an Example: Hidden Set Size Approximation Using Group Queries and Conditional Sampling. *ACM Trans. Comput. Theory*, 8(4):15:1–15:19, 2016.

24 Larry J. Stockmeyer. The Complexity of Approximate Counting (Preliminary Version). In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, STOC*, pages 118–126, 1983.

25 Larry J. Stockmeyer. On Approximation Algorithms for #P. *SIAM J. Comput.*, 14(4):849–861, 1985.

A Upper bound for estimating triangles using BIS

In this Section, we prove Theorem 1.7, which is formally stated as follows:

▶ **Theorem A.1** (Upper bound matching the lower bound in Theorem 3.1). *There exists an algorithm* Triangle-Est(G, ε) *that has* BIS *query access to a graph* $G(V, E)$ *having* n *vertices, takes a parameter* $\varepsilon \in (0, 1)$ *as input, and reports a* $(1 \pm \varepsilon)$*-approximation to the number of triangles in* G *with a probability of at least* $1 - o(1)$*. Moreover, the expected number of* BIS *queries made by the algorithm is* $\widetilde{\mathcal{O}}\left(\min\{\frac{m}{\sqrt{T}}, \frac{m^{3/2}}{T}\}\right)$*, where* m *and* T *denote the number of edges and triangles in* G*, respectively.*

In order to prove the above theorem, we first prove Lemmas A.2 and A.3.

▶ **Lemma A.2** (Upper bound when the number of triangles is *large*). *There exists an algorithm* Triangle-Est-High(G, ε) *that has* BIS *query access to a graph* $G(V, E)$ *having* n *vertices,* $\varepsilon \in (0, 1)$ *as input, and reports a* $(1 \pm \varepsilon)$*-approximation to the number of triangles in* G *with a probability of at least* $1 - o(1)$*. Moreover, the expected number of* BIS *queries made by the algorithm is* $\widetilde{\mathcal{O}}\left(\frac{m^{3/2}}{T}\right)$*, where* m *and* T *denote the number of edges and triangles in* G*, respectively.*

▶ **Lemma A.3** (Upper bound when the number of triangles is *small*). *There exists an algorithm* Triangle-Est-Low(G, ε) *that has* BIS *query access to a graph* $G(V, E)$ *having* n *vertices,* $\varepsilon \in (0, 1)$ *as input, and reports a* $(1 \pm \varepsilon)$*-approximation to the number of triangles in* G *with a probability of at least* $1 - o(1)$*. Moreover, the expected number of* BIS *queries made by the algorithm is* $\widetilde{\mathcal{O}}\left(\frac{m+T}{\sqrt{T}}\right)$*, where* m *and* T *denote the number of edges and triangles in* G*, respectively.*

Our final algorithm Triangle-Est(G, ε) (as stated in Theorem A.1) is a combination of Triangle-Est-High(G, ε) and Triangle-Est-Low(G, ε). Informally, Triangle-Est (G, ε) calls Triangle-Est-High(G, ε) and Triangle-Est-Low (G, ε) when $T = \Omega(m)$ and $T = \mathcal{O}(m)$, respectively. Observe that, if Triangle-Est knows T within a constant factor, then it can decide which one to use among Triangle-Est-High(G, ε) and Triangle-Est-Low(G, ε). If Triangle-Est does not know T within a constant factor, then it starts from a guess $L = \binom{n}{3}/2$ and updates L by making a *geometric* search until the output of Triangle-Est is consistent with L. Depending on whether $L = \Omega(m)$ or $L = \mathcal{O}(m)$, Triangle-Est decides which one among Triangle-Est-High(G, ε) and Triangle-Est-Low(G, ε) to call. This guessing technique is standard in the property testing literature. It has been used several times in the literature (for example in [19, 14], generalized in [15], and used directly in [2]). So, we explain Triangle-Est-High(G, ε) and Triangle-Est-Low(G, ε) assuming a promised lower bound on L, and the respective query complexities will be in terms of L instead of T.

Another important thing to observe is that to execute the above discussed steps of algorithm Triangle-Est(G, ε) must know m. But we note that an estimate of m will be good enough for our purpose, and that can be estimated by using $\widetilde{\mathcal{O}}(1)$ BIS queries (see Table 1).

In Appendix A.1, we discuss some properties of BIS and some tasks it can perform. These properties will be useful while describing our Triangle-Est-High(G, ε) and Triangle-Est-Low(G, ε), and proving Lemma A.2 and Lemma A.3, in Section A.2 and Section A.3, respectively.

A.1 Some preliminaries about BIS

Let $G(V, E)$ be the unknown graph to which we have BIS query access. One can compute the exact number of edges using $\widetilde{\mathcal{O}}(|E(G)|)$ queries [5] deterministically. Also, we can estimate the number of edges in graph G [5, 6, 13] and sample an edge from G *almost uniformly* [13], with a probability of at least $1 - o(1)$, and making $\widetilde{\mathcal{O}}(1)$ BIS queries. Here, we would like to note that, all three results we mentioned above hold for induced subgraphs as well as induced bipartite subgraph, as formally described below. Those will be used when we design our upper bounds in Appendix A.2 and A.3.

▶ **Proposition A.4** (Exact edge estimation using BIS [5]). *There exists a deterministic algorithm that has* BIS *query access to an unknown graph $G(V, E)$ with n vertices, takes $X \subseteq V(G)$ (alternatively, two disjoint subsets A, B) as input, makes $\widetilde{\mathcal{O}}(|E(G[X])|)$ (alternatively, $\widetilde{\mathcal{O}}(|E(G[A, B])|)$)* BIS *queries, and reports all the edges in $E(G)$ (alternatively, $E(G[A, B])$).*

▶ **Proposition A.5** (Approximate edge estimation using BIS [6, 13]). *There exists an algorithm that has* BIS *query access to an unknown graph $G(V, E)$ with n vertices, takes $X \subseteq V(G)$ (alternatively, two disjoint subsets A, B) and a parameter $\varepsilon \in (0, 1)$ as inputs, makes $\widetilde{\mathcal{O}}(1)$* BIS *queries, and reports a $(1 \pm \varepsilon)$-approximation to $|E(G[X])|$ (alternatively, $|E(G[A, B])|$), with a probability of at least $1 - o(1)$.*

To state the next proposition, we need the following definition.

▶ **Definition A.6** (Approximate uniform sample from a set). For a nonempty set X and $\varepsilon \in (0, 1)$, getting a $(1 \pm \varepsilon)$-approximate uniform sample from X means getting a sample from a distribution on X such that the probability of getting $x \in X$ lies in $[(1-\varepsilon)/|X|, (1+\varepsilon)/|X|]$.

▶ **Proposition A.7** (Approximate edge sampling using BIS [13]). *There exists an algorithm that has* BIS *query access to an unknown graph $G(V, E)$ with n vertices, takes $X \subseteq V(G)$ (alternatively, two disjoint subsets A, B) and a parameter $\varepsilon \in (0, 1)$ as inputs, makes $\widetilde{\mathcal{O}}(1)$* BIS *queries, and reports a $(1 \pm \varepsilon)$-approximate uniform sample from $E(G[X])$ (alternatively, $E(G[A, B]$), with a probability of at least $1 - o(1)$.*

Observe that the following corollary follows from Propositions A.4, A.5 and A.7, by taking $A = \{v\}$ and $B = Z$, where $v \in V(G)$ and $Z \subseteq V(G) \setminus \{v\}$.

▶ **Corollary A.8** (BIS query can extract useful information about the neighborhood of a given vertex).

(i) Entire neighbourhood of a vertex using BIS: *There exists a deterministic algorithm that has* BIS *query access to an unknown graph $G(V, E)$ with n vertices, takes $v \in V(G)$ and $Z \subseteq V(G) \setminus \{v\}$ as input, makes $\widetilde{\mathcal{O}}(|N_G(v) \cap Z|)$* BIS *queries, and reports all the neighbors of v in Z.*

(ii) Approximate degree using BIS: *There exists an algorithm that has* BIS *query access to an unknown graph $G(V, E)$ with n vertices, takes $v \in V(G)$, $Z \subseteq V(G) \setminus \{v\}$ and $\varepsilon \in (0, 1)$ as inputs, makes $\widetilde{\mathcal{O}}(1)$* BIS *queries, and reports a $(1 \pm \varepsilon)$-approximation to $|N_G(v) \cap Z|$, with a probability of at least $1 - o(1)$.*

(iii) Finding an approximate random neighbor of a vertex using BIS: *There exists an algorithm that has* BIS *query access to an unknown graph $G(V, E)$ with n vertices, takes $v \in V(G)$, $Z \subseteq V(G) \setminus \{v\}$ and $\varepsilon \in (0, 1)$ as inputs, makes $\widetilde{\mathcal{O}}(1)$* BIS *queries, and reports a $(1 \pm \varepsilon)$-approximate uniform sample from the set $N_G(v) \cap Z$, with a probability of at least $1 - o(1)$.*

A.2 Algorithm Triangle-Est-High and proof of Lemma A.2

Algorithm Triangle-Est-High is inspired by the triangle estimation algorithm of Assadi et al. [2] [9] when we have the following query access to the unknown graph.

Adjacency Query: Given vertices $u, v \in V(G)$ as input, the oracle reports whether (u, v) is an edge or not;

Degree Query: Given a vertex $u \in V(G)$ as input, the oracle reports the degree of vertex u in G;

Random Neighbor Query: Given a vertex $u \in V(G)$, the oracle reports a neighbor of u uniformly at random if the degree of u is nonzero. Otherwise, the oracle reports a special symbol $\perp$;

Random Edge Query: With this query, the oracle reports an edge from the graph G uniformly at random.

The number of queries to the oracle made by Assadi et al.'s algorithm [2] is $\widetilde{\mathcal{O}}\left(\frac{m^{3/2}}{L}\right)$, where m denotes the number of edges and L is a promised lower bound on the number of triangles in G. Also, note that, the triangle estimation algorithm by Assadi et al. [2] can be *suitably* modified even if we have approximate versions of Degree, Random Neighbor and Random Edge queries, as described below.

Apx Degree Query: Given a vertex $u \in V(G)$ and $\varepsilon \in (0, 1)$ as input, the oracle reports a $(1 \pm \varepsilon)$-approximation to the degree of vertex u in G;

Apx Random Neighbor Query: Given a vertex $u \in V(G)$ and $\varepsilon \in (0, 1)$ as input, the oracle reports a $(1 \pm \varepsilon)$-approximate uniform sample from $N_G(u)$ if the degree of u is nonzero. Otherwise, the oracle reports a special symbol $\perp$;

Apx Random Edge Query: Given $\varepsilon \in (0, 1)$, the oracle reports a $(1 \pm \varepsilon)$-approximate uniform sample from $E(G)$.

From Corollary A.8, $\widetilde{\mathcal{O}}(1)$ BIS queries are enough to simulate Apx Degree Query and Apx Random Neighbor Query, with a probability of at least $1 - o(1)$. Also, by Proposition A.7, Apx Random Edge Query can be simulated by $\widetilde{\mathcal{O}}(1)$ BIS queries, with a probability of at least $1 - o(1)$. Moreover, a BIS query can trivially simulate an Adjacency Query. Combining these facts with the fact that the triangle estimation algorithm by Assadi et al. [2] can be suitably modified even if we have approximate versions of Degree, Random Neighbor and Random Edge queries, we are done with the proof of Lemma A.2.

A.3 Algorithm Triangle-Est-Low and the proof of Lemma A.3

Algorithm Triangle-Est-Low is inspired by the streaming algorithm for triangle counting by McGregor et al. [20]. Algorithm Triangle-Est-Low extracts a subset of edges by making BIS queries in a specific way as explained below. Later, we discuss that those sets of edges will be enough to estimate the number of triangles in G.

[9] Actually, they have given an algorithm for estimating the number of copies of any given subgraph of fixed size.

Generating a random sample $S \subseteq V(G)$ and exploring its neighborhood

Algorithm Triangle-Est-Low adds each vertex in $V(G)$ to S with probability $\widetilde{\mathcal{O}}\left(\frac{1}{\sqrt{L}}\right)$. Recall Corollary A.8 (i). For each $v \in S$, we find all the neighbors of v (the set $N_G(v)$) by making $\widetilde{\mathcal{O}}(|N_G(v)|)$ BIS queries. Let E_S be the set of all the edges having at least one end point in S, that is, $E_S = \{(v,w) : v \in S \text{ and } w \in N_G(v)\}$. After finding E_S, we do the following. For each $v \in S$, we find all the edges in the subgraph induced by $N_G(v)$ by using $\widetilde{\mathcal{O}}\left(|E(G[N_G(v)])|\right)$ BIS queries. This is again possible by Corollary A.8 (i). Note that $|E(G[N_G(v)])|$ is the number of edges in the subgraph of G induced by $N_G(v)$. Let E'_S be the set of all edges present in the subgraph induced by $N_G(v)$ for some $v \in S$, that is, $E'_S = \bigcup\limits_{v \in S} E(G[N_G(v)])$. Later, we argue that the number of BIS queries that we make to generate E_S and E'_S is bounded in expectation.

Apart from S, E_S and E'_S, Triangle-Est-Low extracts some more required edges by making BIS queries, as explained below.

Generating F, a set of $(1 \pm \mathcal{O}(\varepsilon))$-approximate uniform sample from $E(G)$, and exploring the subgraphs induced by sets $N_G(v) \cap V(F)$ for each $v \in V(F)$

Algorithm Triangle-Est-Low calls the algorithm corresponding to Proposition A.7, for $\widetilde{\mathcal{O}}\left(\frac{m}{\sqrt{L}}\right)$ times. By this process, we get a set F of $(1 \pm \mathcal{O}(\varepsilon))$-approximate uniform sample from $E(G)$, with a probability of at least $1 - o(1)$. Note that $|F| = \widetilde{\mathcal{O}}\left(\frac{m}{\sqrt{L}}\right)$, and the number of BIS queries we make to generate F is $\widetilde{\mathcal{O}}\left(\frac{m}{\sqrt{L}}\right)$. Let $V(F)$ be the set of vertices present in at least one edge in F. For each vertex $v \in V(F)$, we find all the edges in the subgraph of G induced by $N_G(v) \cap V(F)$, by using $\widetilde{\mathcal{O}}\left(|E(G[N_G(v) \cap V(F)])|\right)$ BIS queries (see Corollary A.8 (i)). Note that $|E\left(G[N_G(v) \cap V(F)]\right)|$ is the number of edges in the subgraph of G induced by $N_G(v) \cap F$. Let E_F be the set of all the edges that are present in subgraphs induced by $N_G(v) \cap V(F)$ for some $v \in V(F)$, that is, $E_F = \bigcup\limits_{v \in V(F)} E(G[N_G(v) \cap V(F)])$. Later we show that the expected number of BIS queries needed to find F and E_F is bounded.

In algorithm Triangle-Est-Low, BIS queries are made only to generate S, E_S, E'_S, F and E_F. After these sets are generated, no more BIS queries are made by the algorithm. We formally prove the query complexity of Triangle-Est-Low. But, first, we show that S, E_S, E'_S, F and E_F can be carefully used to estimate T, the number of triangles in G.

Connection with streaming algorithm for triangle counting by McGregor et al. [20]

(Estimating the number of triangles from S, E_S, E'_S, F and E_F)

McGregor et al. [20] gave a two-pass algorithm that estimates the number of triangles in a graph G when the edges of G arrive in an arbitrary order. Moreover, the space complexity of their algorithm is $\widetilde{\mathcal{O}}\left(\frac{m}{\sqrt{L}}\right)$. Note that their algorithm assumes a lower bound L on the number of triangles in the graph. The high level sketch of their algorithm is as follows:

- Generate a subset X of $V(G)$ by sampling each vertex in $V(G)$ with probability $\widetilde{\mathcal{O}}\left(\frac{1}{\sqrt{L}}\right)$;
- In the first pass, the edges having at least one vertex in X is found, and let it be E_X. Also, in the first pass, a subset of edges Y is generated by sampling each edge with probability $\widetilde{\mathcal{O}}\left(\frac{1}{\sqrt{L}}\right)$;
- In the second pass, for each edge $e = \{x, y\}$ in the stream, their algorithm finds the vertices in X with which e forms a triangle. Also, for each edge, $e = \{x, y\}$ in the stream, their algorithm finds the pairs of edges in Y that forms a triangle with e. Let Z be the set of *useful* edges in the second pass, that is, the set of edges that either forms a triangle with a vertex in X or forms a triangle with two edges in Y.

Note that their algorithm does not talk about the set Z. We are introducing it for our analysis. Executing the two passes described above is straightforward. They have proved that performing two passes as described is good enough to estimate the number of triangles in the graph.

Now, we compare the information maintained by our algorithm with the information maintained by McGregor et al.'s algorithm. S and E_S in our algorithm Triangle-Est-Low follows the same probability distribution as that of X and E_X, respectively, in McGregor et al.'s algorithm. Recall that F is a set of $\widetilde{\mathcal{O}}\left(\frac{m}{\sqrt{L}}\right)$ $(1 \pm \varepsilon)$-approximate sample from $E(G)$ with a probability $1 - o(1)$. But Y in McGregor et al.'s algorithm is generated by sampling each edge with probability $\widetilde{\mathcal{O}}\left(\frac{1}{\sqrt{L}}\right)$. But observe that the *total variation* distance between the probability distributions of F and Y is $o(1)$.

Before discussing about E'_S and E_F in our algorithm Triangle-Est-Low, consider the following observation about the set Z. Note that we have defined Z while describing the second pass of McGregor et al.'s algorithm.

▶ **Observation A.9.** Consider X, E_X, and Y generated by the first pass of McGregor et al.'s algorithm. Let E'_X be the edges in the subgraph induced by $N_G(v)$ for some $v \in X$, and let E_Y be the set of edges in the subgraph induced by $N_G(v) \cap V(Y)$. Here $V(Y)$ denotes the set of vertices present in at least one vertex of Y. Then for each edge $e \notin E'_X \cup E_Y$, there is neither a vertex in X with which e forms a triangle in G nor there are two edges in Y with which e forms a triangle in G. Then the set of useful edges Z is $E'_X \cup E_Y$.

By the above observation, E'_S and E_F in our algorithm Triangle-Est-Low are essentially enough for maintaining the information and executing the same steps as that of the second pass of McGregor et al.'s algorithm.

Putting everything together, algorithm Triangle-Est-Low outputs a $(1 \pm \varepsilon)$-approximation to the number of triangles in the graph.

Query complexity analysis

The set S can be generated without making any BIS queries. The number of BIS queries we make to find the set E_S is at most $\sum\limits_{v \in S} \widetilde{\mathcal{O}}(|N_G(v)|)$, which in expectation is

$$\mathbb{E}\left[\sum_{v \in S} \widetilde{\mathcal{O}}(|N_G(v)|)\right] = \sum_{v \in V(G)} \Pr(v \in S) \cdot \widetilde{\mathcal{O}}(|N_G(v)|) = \widetilde{\mathcal{O}}\left(\frac{m}{\sqrt{L}}\right).$$

The number of BIS queries we make to find the set E'_S is at most $\sum\limits_{v \in S} \widetilde{\mathcal{O}}\left(|E(G[N_G(v)])|\right)$. Note that $|E(G[N_G(v)]|$ is T_v, that is, the number of triangles having v as one of the vertex. So, the expected number of BIS queries we make to find E'_S is at most

$$\mathbb{E}\left[\sum_{v \in S} \widetilde{\mathcal{O}}(T_v)\right] = \sum_{v \in V(G)} \Pr(v \in S) \cdot \widetilde{\mathcal{O}}(T_v) = \widetilde{\mathcal{O}}\left(\frac{T}{\sqrt{L}}\right).$$

The number of BIS queries we make to find the set F is $\widetilde{\mathcal{O}}(|F|) = \widetilde{\mathcal{O}}\left(\frac{m}{\sqrt{L}}\right)$

The number of BIS queries to generate E_F is at most $\sum\limits_{v \in V(F)} \widetilde{\mathcal{O}}\left(|E(G[N_G(v) \cap V(F)])|\right)$. Observe that an edge $\{x, y\}$ is present in E_F if there exists a $z \in V(G)$ such that $\{x, y, z\}$

forms a triangle in G and $\{x, z\}$ and $\{y, z\}$ are in F. So, the probability that an edge $\{x, y\}$ is in $E(G[N_G(v) \cap V(F)]$ is at most $|\Gamma(\{x, y\})| \cdot \widetilde{\mathcal{O}}\left(\frac{1}{L}\right)$, where $\Gamma(\{x, y\})$ denotes the set of common neighbors of x and y in G. So, the expected number of BIS queries to enumerate all the edges in E_F is at most

$$\sum_{\{x,y\} \in E(G)} \widetilde{\mathcal{O}}\left(\frac{|\Gamma(\{x, y\}|)}{\sqrt{L}}\right) = \widetilde{\mathcal{O}}\left(\frac{T}{\sqrt{L}}\right).$$

Hence, the expected number of BIS queries made by the algorithm is $\widetilde{\mathcal{O}}\left(\frac{m+T}{\sqrt{L}}\right)$.

Fine Grained Analysis of High Dimensional Random Walks

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

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

Abstract

One of the most important properties of high dimensional expanders is that high dimensional random walks converge rapidly. This property has proven to be extremely useful in a variety of fields in the theory of computer science from agreement testing to sampling, coding theory and more. In this paper we present a state of the art result in a line of works analyzing the convergence of high dimensional random walks [13, 10, 15, 1], by presenting a *structured* version of the result of [1]. While previous works examined the expansion in the viewpoint of the worst possible eigenvalue, in this work we relate the expansion of a function to the entire spectrum of the random walk operator using the structure of the function; We call such a theorem a Fine Grained High Order Random Walk Theorem. In sufficiently structured cases the fine grained result that we present here can be much better than the worst case while in the worst case our result is equivalent to [1].

In order to prove the Fine Grained High Order Random Walk Theorem we introduce a way to bootstrap the expansion of random walks on the vertices of a complex into a fine grained understanding of higher order random walks, provided that the expansion is good enough.

In addition, our *single* bootstrapping theorem can simultaneously yield our Fine Grained High Order Random Walk Theorem as well as the well known Trickling down Theorem. Prior to this work, High order Random walks theorems and Tricking down Theorem have been obtained from different proof methods.

2012 ACM Subject Classification Theory of computation → Expander graphs and randomness extractors; Theory of computation → Random walks and Markov chains

Keywords and phrases High Dimensional Expanders, High Dimensional Random Walks, Local Spectral Expansion, Local to Global, Trickling Down

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.49

Category RANDOM

Related Version *Full Version*: https://arxiv.org/abs/2208.03241 [11]

Funding Research supported by ERC and ISF.

1 Introduction

In recent years much attention has been given to the field of high dimensional expanders which are high dimensioanl analogues of expander graphs. One extremely useful property of high dimensional expanders is that higher dimensional random walks (which are higher dimensional analogues of random walks on graphs) converge rapidly to their stationary distribution (For example, this property was used in [10, 14, 4, 9] and more). Consequently there has been some work studying the convergence of higher dimensional random walks [10, 13, 15, 1]. In this paper we improve upon these convergence results by relating the structure of the function to its expansion. Specifically we present the following improvements:

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 49; pp. 49:1–49:22
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Fine grained analysis of random walk

Prior to this paper, the state of the art analysis of high dimensional random walks was done by Alev and Lau in [1] following [10, 13, 15]. Their work analyzed the eigenvalues of an important random walk called the *down-up random walk*. Their result, however, was only useful for the worst case analysis as it did not relate the structure of the function to its expansion and thus was forced to consider the worst possible function. In this paper we present an improvement upon Alev and Lau's result by finding a connection between the structure of a function and how well it expands and can therefore yield better results on cochains that posses a "nice" structure.

In two-sided spectral expanders a fine grained analysis of high dimensional random walks was already proven, based on Fourier analysis in high dimensional expanders [8, 15]. In the two sided case even stronger results following from hypercontractivity are known [5, 6, 12] and more. Our result is, importantly, about *one-sided* spectral expanders as there are cases when the use of one-sided high dimensional expansion is crucial - for example in Anari et al's breakthrough proof of the fast convergence of the basis exchange walk [4] and thus showed an algorithm that samples a basis of a matroid. This result started a wave of sampling results that use high dimensional expanders [7, 3, 2] to name a few examples. This result relies heavily on fast convergence of high dimensional random walks on one-sided expanders (As they show that that the basis exchange corresponds to a down-up walk on the top dimension of a one-sided high dimensional expander). In this work we present, to our knowledge, the first result to show a fine grained analysis of the random walk operator in one-sided local spectral expanders.

Replacing eigendecomposition

Previous fine grained analysis of high dimensional random walks relied on finding approximate eigendecomposition of the high dimensional random walk. We present a new approach to finding a fine grained understanding of the high dimensioanl random walks: bootstrapping an understanding of the expansion of random walks on the vertices of the complex. We show that, if the expansion of these random walks beat the expansion of high dimensional random walks on the vertices of local structures[1], we can bootstrap it into a *fine grained* understanding of higher dimensional random walks.

Unification of the High order random walk theorem and the Tricking down theorem

In order to perform our fine grained analysis of the higher dimensional random walk operators we develop a new bootstrapping framework. This new framework is fairly generic and seems to be of independent interest as it can be used to prove another central theorem in the theory of high dimensional expansion, namely the trickling down theorem [16]. We comment that prior to this work these two important theorems were obtained by different proof techniques.

Before we can state our results more formally, we have to define the high dimensional analogs of expander graphs as graphs do not posses high dimensions. This high dimensional object is called a "simplicial complex" and is defined as:

▶ **Definition 1** (Simplicial complex). *A set X is a* simplicial complex *if it is closed downwards meaning that if $\sigma \in X$ and $\tau \subseteq \sigma$ then $\tau \in X$. We call members of X the* faces *of X.*

[1] Specifically, high dimensional random walks on the vertices of the links of vertices.

Simplicial complexes can be thought of as hyper-graphs with closure property (i.e. every subset of a hyper-edge is a hyper-edge). We are interested in higher dimensions and therefore it would be useful to define the dimension of these higher dimensional objects:

▶ **Definition 2** (Dimension). *Let X be a simplicial complex and let $\sigma \in X$ be a face of X. Define the dimension of σ to be $\dim(\sigma) = |\sigma| - 1$. We also denote the set of all faces of dimension i in X as $X(i)$. Also define the dimension of the complex X as $\dim(X) = \max_{\sigma \in X}\{\dim(\sigma)\}$. Note that there is a single (-1)-dimensional face - the empty face.*

Of particular interest are simplicial complexes whose maximal faces are of the same dimension, defined below:

▶ **Definition 3** (Pure simplicial complex). *A simplicial complex X is a* pure simplicial complex *if every face $\sigma \in X$ is contained in some $(\dim(X))$-dimensional face.*

Throughout this paper we will assume that every simplicial complex is pure. In most cases we will be interested in weighted pure simplicial complexes. In weighted pure simplicial complexes the top dimensional faces are weighted and the weight of the rest of the faces follows from there as described here:

▶ **Definition 4** (Weight). *Let X be a pure d-dimensional simplicial complex. Define its weight function $\mathrm{w} : X \to [0,1]$ to be a function such that:*

- $\sum_{\sigma \in X(d)} \mathrm{w}(X) = 1$
- *For every face τ of dimension $i < d$ it holds that $\mathrm{w}(\tau) = \frac{1}{\binom{d+1}{i+1}} \sum_{\substack{\sigma \in X(d) \\ \tau \subseteq \sigma}} \mathrm{w}(\sigma)$.*

It is important to note that we think of unweighted complex as complexes that satisfy $\forall \sigma \in X(d) : \mathrm{w}(\sigma) = \frac{1}{|X(d)|}$. While the top dimensional faces of unweighted complexes all have the same weight, the same cannot be said for lower dimensional faces.

It is also important to note that the sum of weights in every dimension is exactly 1 and therefore for every k the weight function can, and at times will, be thought of as a distribution on $X(k)$.

One key property of high dimensional expanders is that they exhibit *local to global* phenomena. These phenomena are at the main interest of this paper. It is therefore useful to consider local views of the simplicial complex which we define as follows:

▶ **Definition 5** (Link). *Let X be a simplicial complex and $\sigma \in X$ be a face of X. Define the* link *of σ in X as $X_\sigma = \{\tau \setminus \sigma | \sigma \subseteq \tau\}$. It is easy to see that the link of any face is a simplicial complex.*

The weight of faces in the links is induced by the weights of the faces in the original complex. Specifically, we denote by w_σ the weight function in the link of $\sigma \in X(i)$ and it holds that $\forall \tau \in X_\sigma(j) : \mathrm{w}_\sigma(\tau) = \frac{\mathrm{w}(\tau \cup \sigma)}{\binom{i+j+2}{i+1}\mathrm{w}(\sigma)}$. Generally speaking, the local to global phenomena are ways to derive properties of the entire complex by only looking at local views (i.e. links).

Another important substructure of a simplicial complex is its skeletons

▶ **Definition 6** (Skeleton). *Let (X, w) be a weighted pure d-dimensional simplicial complex and let $i \leq d$. Define the i-skeleton of X as the following weighted simplicial complex:*

$$X^{(i)} = \{\sigma \in X | \dim \sigma \leq i\}$$

With the original weight function.

In many cases we will think of the 1-skeleton of a simplicial complex as a graph. In addition, it is important to note that even if the original complex is unweighted, the skeletons of said complex might be.

We are now ready to define a high dimensional expander[2].

▶ **Definition 7** (Local spectral expander). *A pure d-dimensional simplicial complex X is a λ-local spectral expander[3] if for every face σ of dimension at most $d-2$ it holds that $X_\sigma^{(1)}$ is a λ-spectral expander[4]. Note that this includes $\sigma = \emptyset$, i.e. the entire complex.*

Much like graphs, simplicial complexes also support random walks. In graphs, the random walks are of the form vertex-edge-vertex - the walk might move between two vertices if they are connected by an edge. The high dimensional analogue of these random walks travel between two k-dimensional faces if they are part of a common $(k+1)$-dimensional face. Our particular random walk of interest is the higher dimensional analogue of the non-lazy random walk, defined as follows:

▶ **Definition 8** (Non-lazy up-down operator informal, for formal see 24). *Define the k-dimensional non-lazy up-down random walk, $(M')_k^+$ as the k dimensional analogue of the non-lazy random walk on the vertices of a graph: A walk that moves between two k-dimensional faces if they are contained in a $(k+1)$-dimensional face and never stays in place.*

We are going to improve our understanding of how these higher dimensional random walks apply to structured states. These states correspond to another natural structure on high dimensional expanders called cochains that is defined as follows:

▶ **Definition 9** (Cochains). *Let X be a pure d-dimensional simplicial complex. For $-1 \leq k \leq d$ define a k-dimensional cochain F to be any function from $X(k)$ to $\mathbb{R}$. We also denote by $C^k(X;\mathbb{R})$ the set of all k-dimensional cochains.*

We are going to be interested in ways of viewing the cochains in the links of the complex. For now we will only introduce one such way. Namely localization:

▶ **Definition 10** (Localization). *Let X be a pure d-dimensional simplicial complex, k, i be dimensions such that $i < k$ and $F \in C^k(X;\mathbb{R})$. Also let $\sigma \in X(i)$. Define the localization of F to σ to be:*

$$F_\sigma(\tau) = F(\sigma \cup \tau).$$

We note that there is a very natural inner product defined on the cochains of a simplicial complex, defined as follows:

▶ **Definition 11** (Inner product). *Let X be a pure d-dimensional simplicial complex and let $F, G \in C^k(X;\mathbb{R})$. Define the inner product of F and G to be:*

$$\langle F, G \rangle = \sum_{\sigma \in X(k)} \mathrm{w}(\sigma) F(\sigma) G(\sigma).$$

[2] There is no singular definition of high dimensional expander but in this paper we only use the algebraic definition - local spectral expansion.

[3] Much like one dimensional expanders, in high dimensions there is also notion of one-sided vs. two-sided local spectral expansion. The definition we use throughout the paper is that of *one*-sided local spectral expander. The difference being that in two-sided local spectral expander the underlying graph of every link is a two-sided expander rather than a one-sided expander.

[4] The complexes are weighted and therefore their expansion property is defined as the second largest eigenvalue of the non-lazy random walk. A random walk that walks from a face to one of its neighbours with probability equal to the proportion between the weight of the edge that connects them and the sum of the weights of the edges that include said vertex.

1.1 Main Results

In this paper we present a state of the art analysis the high dimensional analogue of the non-lazy random walk. We are specifically interested in going beyond Alev and Lau's worst case result [1, Theorem 1.5] and relate the structure of the cochain to its expansion. Specifically, we define:

▶ **Definition 12** (i-level cochain, informal. For formal see Definition 29)**.** *A cochain F is an i-level cochain with respect to localization if for every $\sigma \in X(i-1)$ it holds that $\langle F_\sigma, \mathbb{1} \rangle = 0$.*[5]

Previously the best analysis of high dimensional random walk was due to Alev and Lau [1, Theorem 1.5] who showed that:

▶ **Theorem 13** (Alev and Lau [1], restated)**.** *Let X be a pure d-dimensioanl high dimensional expander. Define $\gamma_i = \max_{\sigma \in X(i)} \{\lambda_2(X_\sigma)\}$. For any dimension k and any k-dimensional 0-level cochain F it holds that:*

$$\left\langle (M')_k^+ F, F \right\rangle \leq \left(1 - \frac{1}{k+1} \prod_{j=-1}^{k-2} (1-\gamma_i)\right) \|F\|^2.$$

We show that this result can be vastly improved for structured cochains. Specifically, we show the following decomposition of the high dimensional random walks:

▶ **Theorem 14** (Fine grained analysis of high dimensional random walks, Informal. For formal see Theorem 54)**.** *Let $F \in C^k(X; \mathbb{R})$ and let $F_0, \cdots, F_k$ be an orthogonal decomposition of F such that F_i is an i-level cochain and $F = \sum_{i=0}^{k} F_i$. In addition, let $\gamma_i = \max_{\sigma \in X(i)} \{\lambda_2(X_\sigma)\}$ (Where $\lambda_2(X_\sigma)$ is the second largest eigenvalue of the underlying graph of X_σ) then:*

$$\left\langle (M')_k^+ F, F \right\rangle \leq \sum_{i=0}^{k} \left(1 - \frac{1}{k-i+1} \prod_{j=i-1}^{k-1} (1-\gamma_j)\right) \|F_i\|^2.$$

Cases where we improve upon previous results

In the worst case (i.e. no assumption is made on the structure of F) our result matches that of Alev and Lau. In cases where the cochain is structured, however, our Theorem yields strictly better results than what was previously known. We also give some examples of families of structured cochains on which our result is *strictly better* than the result of Alev and Lau - Specifically, we show two families of cochains that are highly structured: The first is a set of cochains associated with a different form of high dimensional expansion, the minimal cochains and the second is the indicator function of a balanced set of faces (for more information see the full version).

Comparison with other known decompositions

Similar decompositions of the high dimensional random walks were already known for two-sided high dimensional expanders [8, 15]. These decomposition relied on finding approximate eigenspaces of the walk operator. Unlike the case in two-sided high dimensional expanders,

[5] We note that every j-dimensional i-level cochain corresponds to a cochain in the i^{th} dimension that has been "lifted" up to the j^{th} dimension. A more formal version of this statement can be found in Lemma 53.

in one-sided high dimensional expanders the eigenspaces of the high dimensional random walks are not currently understood, even approximately. We do, however, understand the expansion of key random walks on the 0-dimensional cochains. We follow this by showing that if this expansion is strong enough (i.e. these random walks converge fast enough) we can boost it to all levels and get a decomposition theorem without eigendecomposition. In order to apply our bootstrapping method we have to show that the expansion of the aforementioned 0-dimensional cochains "beats" the expansion of cochains of higher levels. Note that the decomposition achieved by our bootstrapping theorem differs from the decomposition known in two-sided expanders in one *crucial* way: In our decomposition the level functions are *not* approximate eigenfunctions. Moreover, applying the non-lazy up-down random walk operator to any one of them yields a cochain that is *not* orthogonal to many of the other F_js. This allows us to sidestep a major technical barrier in previous works as we do not rely on the existence of an eigendecomposition of the walk operator (or even an approximate decomposition of that operator). Note that many results that use local spectral expanders, such as their usage in proving the existence of cosystolic expanders, agreement testing, locally testable codes and more only require one application of the walk operator and thus we believe that our result will prove to be influential with regards to these fields.

Our main tool for proving the decomposition of high dimensional random walks is a bootstrapping theorem that reduces decomposition of higher dimensional random walk to an understanding of highly structured cochains (for example cochains that correlate with 0-dimensional cochains). The bootstrapping theorem is fairly general and thus we bring a special case of it here:

▶ **Theorem 15** (Bootstrapping theorem, informal. For formal see Theorem 34). *Let X be a simplicial complex and k be a dimension. Every $F \in C^k(X;\mathbb{R})$ that is orthogonal to the constant functions can be decomposed into $F = \sum_{i=0}^{k} F_i$ such that the cochains F_i are both:*

1. i-level cochains with respect to localization.
2. Orthogonal: *For every $i \neq j$ it holds that F_i is orthogonal to F_j.*

We can use a solution of some recursive formula on 0-level cochains and values $\{\lambda_i\}_{i=0}^{k}$ in order to bootstrap a decomposition of the following form:

$$\left\langle (M')_k^+ F, F \right\rangle = \sum_{i=0}^{k} \lambda_i \left\| F_i \right\|^2.$$

Solving the recursive formula in the bootstrapping theorem requires us to gain some "advantage" for highly structured cochains in the complex. We can therefore view this Theorem as a tool that allows us to bootstrap an advantage we have to a decomposition of the non-lazy random walks.

As we said, our bootstrapping theorem is fairly generic and can also yield the celebrated Oppenheim's trickling down theorem [16, Theorem 4.1]:

▶ **Theorem 16** (Trickling Down, [16]). *Let X be a pure d-dimensional simplicial complex. If it holds that:*

- *For every vertex v: $X_v^{(1)}$ is a λ spectral expander.*
- *X is connected.*

Then it holds that $X^{(1)}$ is a $\frac{\lambda}{1-\lambda}$ spectral expander.

In order to prove the trickling down we use Theorem 15 while defining the i-level cochains differently. For more detail, see Section A.

1.2 Proof Layout

As we mentioned before, we analyze the *non-lazy up-down random walk* - a high dimensional analogue of the non-lazy random walk in graphs. We would like to get a decomposition of the non-lazy random walk operator. In order to do that we decompose the space of cochains to spaces which we term level cochains. Our proof is then comprised of two steps: A bootstrapping Lemma that reduces the problem of decomposing the non-lazy up-down random walk operator to a simple recursive condition about the 0-level cochains and an advantage that allows us to solve said recursive condition. We note that solving the recursive condition is considerably simpler than proving the decomposition directly as the 0-level cochains are very structured objects.

Link viewers

Local-to-global arguments are typically structured by taking a global cochain, decomposing it to local cochains (i.e. cochains on the links of the vertices of the complex) and then using a property of the cochains in the links in order to argue about the original cochain. Two extremely useful ways of decomposing a global cochain to cochains in the links are *restriction* and *localization*. These two methods share many properties, for example:

- They preserve constant functions - Any global constant function is constant locally.
- They are linear - The local view of sum of any two global cochains is the sum of the local views.
- They interact well with the inner product - The expected value of the local inner products of two cochains is their inner product.

We start by identifying these properties and defining a general object that satisfy them called a *link viewer*. A link viewer Λ defines the local view of a cochain F in the link of σ which we denote by $\Lambda_\sigma F$. For the rest of this section, unless stated otherwise, we will think of the *localization link viewer* defined as $\Lambda^\ell_\sigma F = F_\sigma$.

Level cochains

Any link viewer decomposes the space of cochains to subspaces called *i-level cochains*. These spaces are the set of cochains whose expected value is 0 when viewed from any i-dimensional face. We think of these cochains as not correlated with any i-dimensional face. For example, under the localization link viewer the i-level cochains are cochains G that satisfy: $\forall \sigma \in X(i) : \mathbb{E}_{\tau \in X_\sigma(k-i-1)}\left[\Lambda^\ell_\sigma G(\tau)\right] = \mathbb{E}_{\tau \in X_\sigma(k-i-1)}\left[G_\sigma(\tau)\right] = 0$. Note that any i-level cochain is also an $(i-1)$-level cochain and that of F is an i-level cochain then $\Lambda_v F$ is an $(i-1)$-level cochain in the link of every vertex v. It is therefore useful to define *proper level cochains* as well - a proper i-level cochain is an i-level cochain that is orthogonal to every $(i-1)$-level cohain. Every cochain F can be decomposed to $F = \sum_{i=-1}^{p} F_i$ such that every F_i is a proper i-level cochain. This decomposition plays a key role in our proof.

The bootstrapping argument

We can now present our proof for the bootstrapping argument. We assume that the non-lazy up-down random walk expands locally and show that this yields that the non-lazy up-down random walk expands globally. We restrict our attention only to 0-level cochains (which are not necessarily proper) as (-1)-level cochains correspond to the trivial eigenvalue in which we are not interested. Let $F = \sum_{i=0}^{p} F_i$ be a 0-level cochain. Consider its localization to the links of vertices $\Lambda_v F = \sum_{i=0}^{p} \Lambda_v F_i$. Note that $\sum_{i=1}^{p} \Lambda_v F_i$ is a 0-level cochain in X_v(as the localization to links of vertices of i-level cochains is an $(i-1)$-level cochain). Note that $\Lambda_v F_0$ is *not* a 0-level cochain. We can, however, apply our decomposition theorem to some part of $\Lambda_v F_0$ by decomposing it to two parts:

- A 0-level cochain - $\Lambda_v F_0 - \mathbb{E}\left[\Lambda_v F_0\right]$
- A remainder - $\mathbb{E}\left[\Lambda_v F_0\right]$.

We can therefore apply our decomposition theorem to $\sum_{i=0}^{p} \Lambda_v F_i - \mathbb{E}\left[\Lambda_v F_0\right]$. Note that a key step in this decomposition is that despite the fact that these new localized cochains are *not* orthogonal to each other, they are *all* orthogonal to the constants. This allows us to separate them from the constant part of every localization. All we have left to do is to compensate for the remainder. This is done using the advantage step which we will describe next.

The Advantage

In the advantage step we try to bound the remainder we have left in the bootstrapping stage. The advantage required in order to achieve the random walk decomposition Theorem is done by considering the localization link viewer that showing the following observations:

1. Any 0-level function F_0 is, in a sense, a high dimensional description of some other 0-dimensional cochain $G \in C^0(X;\mathbb{R})$. This is true in the sense that $\|F_0\| = \|G\|$ and $\mathbb{E}\left[\Lambda_v^{\ell} F_0\right] = \left\|M_0^{+k} G\right\|$ where M_0^{+k} is the random walk the applied the up step k times followed by applying the down step k times.
2. There is a connection between the random walk on the underlying graph of the complex and M^{+k}. The key claim we use is that once the random walk has performed its first up step *it can already "see" all the vertices it will ever see* (this is due to the structure of simplicial complexes - if two vertices share a k-dimensional face then by the closure property they also share an edge). Going further up only decreases the probability of staying in place. Therefore M^{+k} is a weighted sum of the underlying graph's non-lazy random walk transition matrix and a lazy component (which corresponds to staying in place). We note that this observation can be generalized to any random walk on k-dimensional faces that "walks through" faces of dimension higher than $2k$.

Using these observations as well as our understanding of the 0-dimensional random walk we obtain the advantage we seek.

2 The Signless Differential and Its Adjoint Operator

One of the key operators induced by any simplicial complex is its *signless differential operator*. The signless differential operator is an averaging operator that accepts a k-dimensional cochain and returns a $(k+1)$-dimensional cochain. We adopt the terminology of [15] and consider repeated application of the signless differential and its adjoint operator.

▶ **Definition 17** (Signless differential). *The signless differential operator* $d_k : C^k(X;\mathbb{R}) \to C^{k+1}(X;\mathbb{R})$ *in the following way:*

$$d_k F(\sigma) = \mathbb{E}_{\tau \in X(k)}\left[F(\tau) | \tau \subseteq \sigma\right] = \sum_{\tau \in \binom{\sigma}{k+1}} \frac{1}{k+2} F(\tau).$$

When the dimension is clear from context it will be omitted from the notation. Also define $d_k^* : C^{k+1}(X;\mathbb{R}) \to C^k(X;\mathbb{R})$ *to be the adjoint operator of* d_k.

▶ Note. The signless differential does not meet the definition of a differential as $d_k d_{k-1} \neq 0$.

▶ **Lemma 18** ([15, Lemma 3.6], [8]). *It holds that:*

$$d_k^* F(\tau) = \mathbb{E}_{\sigma \in X(k+1)} \left[F(\sigma) | \tau \subseteq \sigma\right] = \mathbb{E}_{\sigma \in X_\tau(0)} \left[F_\tau(\sigma)\right].$$

For proof see the full version.

We will be interested in repeated application of the signless differential and its adjoint operator. To that effect it will be useful to present them explicitly. Lemmas 19 and 20 will present repeated applications of these operators explicitly (and the proofs of both Lemmas can be found in the full version of this paper).

▶ **Lemma 19.** *Let F be an i-dimensional cochain. Then:*

$$d_{j-1} \cdots d_i F(\sigma) = \mathbb{E}_{\tau \in X(i)} \left[F(\tau) | \tau \subseteq \sigma\right].$$

▶ **Lemma 20.** *Let F be an i-dimensional cochain. Then:*

$$d_i^* \cdots d_{j-1}^* F(\sigma) = \mathbb{E}_{\tau \in X_\sigma(j-i-1)} \left[F_\sigma(\tau)\right].$$

3 Up-Down and Down-Up Operators

In this section we will present two objects of interest - the up-down and down-up walks. These are natural operators that result from considering a standard walk on the k-dimensional faces of a simplicial complex using the following two steps: A *down step* where, given a k-dimensional face, the walk moves to a $(k-1)$-dimensional face that is contained in it with equal probability. And an *up step* in which, given a k-dimensional face, the walk moves to a $(k+1)$-dimensional face with probability proportional to its weight. Applying the up step followed by the down step yields the up-down walk while applying the down step followed by the up step yields the down-up walk.

▶ **Definition 21** (Up-down and down-up operators). *Let X be a simplicial complex. Then define the up-down random walk to be:*

$$\left[M_k^+\right]_{\sigma,\tau} = \begin{cases} \frac{1}{k+2} & \sigma = \tau \\ \frac{\mathrm{w}_\sigma(\tau \setminus \sigma)}{k+2} & \sigma \cup \tau \in X(k+1) \\ 0 & \textit{Otherwise.} \end{cases}$$

And the down-up random walk to be:

$$\left[M_k^-\right]_{\sigma,\tau} = \begin{cases} \frac{1}{k+1} \sum_{\tau' \in \binom{\sigma}{k}} \mathrm{w}_{\tau'}\left(\sigma \setminus \tau'\right) & \sigma = \tau \\ \frac{1}{k+1} \mathrm{w}_{\sigma \cap \tau}\left(\tau \setminus \sigma\right) & \sigma \cap \tau \in X(k-1) \\ 0 & \textit{Otherwise.} \end{cases}$$

We would now like to present characterization the up-down random walk and the down-up random walk using the signless differential (this characterization had appeared in [15, Corollary 3.7] and is given here for completeness). In order to do that, consider the following Lemma:

▶ **Lemma 22.** *Let X be a simplicial complex, it holds that:*

$$M_k^+ = d_k^* d_k \qquad M_k^- = d_{k-1} d_{k-1}^*.$$

For proof, see the full version of this paper.

We are also going to be interested in applying the signless differential and its adjoint operator multiple times in a row. We will therefore define the k-dimensional i-up-down operator and the k-dimensional i-down-up operator in the following way:

▶ **Definition 23.** *Let M_k^{+i} be the k-dimensional i-up-down operator and M_k^{-i} be the k-dimensional i-down-up operator defined as follows:*

$$M_k^{+i} = d_k^* \cdots d_{k+i-1}^* d_{k+i-1} \cdots d_k \qquad\qquad M_k^{-i} = d_{k-1} \cdots d_{k-i-1} d_{k-i-1}^* \cdots d_{k-1}^*.$$

Recall that the i-up-down operator and the i-down-up operators can be presented explicitly using Lemma 19 and Lemma 20. In addition, note that the i-up-down operator corresponds to applying the up step i times and then applying the down step i times. Likewise the i-down-up operator corresponds to applying the down step i times and then applying the up step i times.

4 The Non-Lazy Walk Operator

Our main object of study is going to be the non-lazy k-dimensional random walk operator. This operator is a generalization of the non-lazy random walk operator in graphs to higher dimensions. In graphs the non-lazy random walk operator moves between two vertices if they have an edge connecting them. The higher dimensional version of this operator is going to be something very similar: It is going to move between two k-faces if there is a $(k+1)$-face that contains both faces.

▶ **Definition 24** (The Non-Lazy k-dimensional Random Walk Operator)**.** *Let X be a pure d-dimensional simplicial complex define the k-dimensional random walk operator to be the following operator:*

$$\left[(M')_k^+\right]_{\sigma,\tau} = \begin{cases} \frac{\mathrm{w}_\sigma(\tau\setminus\sigma)}{k+1} & \sigma\cup\tau\in X(k+1) \\ 0 & \textit{Otherwise.} \end{cases}$$

One can also think of the non-lazy random walk operator as the regular up-down operator with the lazy part removed. Formally:

▶ **Observation 25.** *For every dimension k it holds that $(M')_k^+ = \frac{k+2}{k+1} M_k^+ - \frac{1}{k+1} I$.*

Of specific interest is the 0-dimensional non-lazy up-down operator as it can be used to describe *every* one of the 0-dimensional i-up-down operators. This is because every one of these operators ultimately move between a vertex and its neighbours. The more steps one takes up the complex the more mixed the result is. By that we mean that the non-lazy walk operator has a larger effect on the result. Quantitatively, this can be formulated via the following Lemma.

▶ **Lemma 26.** *It holds that $(M')_0^+ = \frac{i+1}{i} M_0^{+i} - \frac{1}{i} I$.*

For proof, see the full version.

5 Analyzing the Non-Lazy Random Walk Operator

We are now ready to start analyzing the random walk operators. In order to do so we are going to use a *local to global* argument. In order to apply a local to global argument we must understand how to view a cochain through the links. Specifically we will be interested in viewing methods that satisfy the following:

▶ **Definition 27** (Link viewer)**.** *A* link viewer *is any transformation Λ that accepts a face σ and a cochain in X. It then returns a cochain in X_σ. In addition, a link viewer satisfies the following:*

- *For every face σ it holds that Λ_σ is linear.*
- *For every face σ it holds that $\Lambda_\sigma \mathbb{1} = \mathbb{1}$.*
- *For every two cochains F, G and any two faces of the same dimension σ,τ it holds that $\dim(F) - \dim(\Lambda_\sigma F) = \dim(G) - \dim(\Lambda_\tau G)$. In addition, denote the dimensional difference for vertices by $\Delta(\Lambda) = \dim(F) - \dim(\Lambda_v F)$.*
- *Viewing of a cochain in a link is only determined by the cochain and the face for which it is the link (and not the path taken to achieve said view). Formally: For every $\tau \subseteq \sigma \in X$ it holds that $\Lambda_\sigma = \Lambda_{\sigma\setminus\tau}\Lambda_\tau$.*
- *For every dimension i it holds that $\langle F, G\rangle = \mathbb{E}_{\sigma\in X(i)}\left[\langle \Lambda_\sigma F, \Lambda_\sigma G\rangle\right]$.*

Of particular interest are link viewers that view the non-lazy random walk operator in "the right way":

▶ **Definition 28.** *A link viewer respects the non-lazy up-down random walk if for every k:*

$$\left\langle (M')^+_k F, F\right\rangle = \mathbb{E}_{v\in X(0)}\left[\left\langle (M')^+_{k-\Delta(\Lambda)} \Lambda_v F, \Lambda_v F\right\rangle\right].$$

▶ **Definition 29** (i-level cochain)**.** *A cochain F is an i-level cochain with respect to Λ if it holds that $\forall\sigma\in X(i-1) : \langle\Lambda_\sigma F, \mathbb{1}\rangle = 0$.*

When the link viewer is clear from context we will simply refer to them as i-level cochains. Denote the set of i-level j-dimensional cochains in X by $C^j_{\Lambda,i}(X;\mathbb{R})$.

▶ **Lemma 30.** *For every dimension $j \le i$: $C^k_{\Lambda,i}(X;\mathbb{R}) \subseteq C^k_{\Lambda,j}(X;\mathbb{R})$.*

Proof. Let $F \in C^k_{\Lambda,i}(X;\mathbb{R})$ and let σ be a $(j-1)$-dimensional face and note the following:

$$\langle\Lambda_\sigma F, \mathbb{1}\rangle = \mathbb{E}_{\tau\in X_\sigma(i-j)}\left[\langle\Lambda_\tau\Lambda_\sigma F, \Lambda_\tau\mathbb{1}\rangle\right] = \mathbb{E}_{\tau\in X_\sigma(i-j)}\left[\langle\Lambda_{\tau\sigma} F, \mathbb{1}\rangle\right]. = 0$$

Where the last equality is due to the fact that $\sigma\cup\tau$ is an $(i-1)$-dimensional face. ◀

Of particular interest are i-level cochains that are orthogonal to the $(i+1)$-level cochains. We will therefore define the following:

▶ **Definition 31** (Proper i-level cochain)**.** *A cochain F is a proper i-level cochain with respect to Λ if it holds that $F \in C^j_{\Lambda,i}(X;\mathbb{R}) \cap \left(C^j_{\Lambda,i+1}(X;\mathbb{R})\right)^\perp$.*

When the link viewer is clear from context we will simply refer to them as proper i-level cochains. Denote the set of proper i-level j-dimensional cochains in X by $C^j_{\Lambda,\hat{i}}(X;\mathbb{R})$.

Consider the following key property of proper level cochains:

▶ **Lemma 32.** *Let $i < j$ and let F be a proper i-level cochain and G be a proper j-level cochain then $\langle F, G\rangle = 0$.*

Proof. Due to Lemma 30 it holds that G is also a $i+1$ level cochain and thus is orthogonal to G by definition. ◀

Note that considering proper level cochains is one of the key ingredients of this paper. In previous results (For example [15]) instead of considering proper i-level cochains the authors essentially considered cochains in $\left(C^k_{\Lambda,i}(X;\mathbb{R})\right)^\perp$. Using proper level cochains allows us to separate the different levels completely and achieve a proper decomposition. Since analyzing the decomposition with only pure level cochains in mind is hard (as even if F is a level cochain the same cannot be said about $\Lambda_\sigma F$) we resort to first show the following technical theorem.

▶ **Theorem 33** (Bootstrapping Theorem, Technical Version). *Let X be a pure d-dimensional simplicial complex, Λ be a link viewer that respects the non-lazy up-down random walk and for every i let $F_i \in C^k_{\Lambda,i}(X;\mathbb{R})$ and $F = \sum_{i=0}^p F_i$. Denote $r = k - \Delta(\Lambda)$ and suppose that there are values of $\{\lambda_{\sigma,i,j}\}_{\sigma\in X, i\in[d], j\in[d]}$ ($\lambda_{\sigma,i,j}$ is the contraction of an i-level j-dimensional cochain in the link of σ) such that for every $\sigma \in X$ and G_0 a 0-level cochain:*

$$\begin{cases}\lambda_{\sigma,1,k}\left\|G_0\right\|^2+\mathbb{E}_{v\in X_\sigma(0)}\left[\left(1-\lambda_{\sigma\cup\{v\},1,r}\right)\left\|M_r^{-r}\Lambda_v G_0\right\|^2\right]\le\lambda_{\sigma,0,k}\left\|G_0\right\|^2\\ \max_{v\in X_\sigma(0)}\left\{\lambda_{\sigma\cup\{v\},i-1,r}\right\}\le\lambda_{\sigma,i,k}.\end{cases}$$

Then:

$$\left\langle (M')_k^+ F, F\right\rangle \le \sum_{i=0}^{p} \lambda_{\emptyset,i,k}\left\|F_i\right\|^2+\sum_{i=0}^{p}\sum_{\substack{j=1\\ i<j}}^{p} c_{i,j}\left\langle F_i,F_j\right\rangle.$$

For some constants $\{c_{i,j}\}$ and where p is the number of level functions that span the space orthogonal to the constants.

Proof. Let r be the dimension of $\Lambda_\sigma F$. Consider the following:

$$\begin{aligned}\left\langle (M')_k^+ F, F\right\rangle &= \left\langle (M')_k^+ \sum_{i=0}^{p} F_i, \sum_{i=0}^{p} F_i\right\rangle\\ &= \mathbb{E}_{v\in X(0)}\left[\left\langle \Lambda_v\left((M')_r^+\sum_{i=0}^{p}F_i\right),\Lambda_v\left(\sum_{i=0}^{p}F_i\right)\right\rangle\right]\\ &= \mathbb{E}_{v\in X(0)}\left[\left\langle (M')_r^+\sum_{i=0}^{p}\Lambda_vF_i,\sum_{i=0}^{p}\Lambda_vF_i\right\rangle\right]\\ &= \mathbb{E}_{v\in X(0)}\left[\left\langle (M')_r^+\left(I-M_r^{-r}\right)\sum_{i=0}^{p}\Lambda_vF_i,\left(I-M_r^{-r}\right)\sum_{i=0}^{p}\Lambda_vF_i\right\rangle\right] \qquad (1)\\ &\quad+\mathbb{E}_{v\in X(0)}\left[\left\langle (M')_r^+M_r^{-r}\sum_{i=0}^{p}\Lambda_vF_i,M_r^{-r}\sum_{i=0}^{p}\Lambda_vF_i\right\rangle\right]\\ &= \mathbb{E}_{v\in X(0)}\left[\left\langle (M')_r^+\left(I-M_r^{-r}\right)\sum_{i=0}^{p}\Lambda_vF_i,\left(I-M_r^{-r}\right)\sum_{i=0}^{p}\Lambda_vF_i\right\rangle\right]\\ &\quad+\mathbb{E}_{v\in X(0)}\left[\left\|M_r^{-r}\Lambda_vF_0\right\|^2\right].\end{aligned}$$

We will now like to apply our Theorem to the localized cochains in the links. For that, consider the following level cochains:

level	cochain	square of norm
$k-1$	$F^v_{k-1} := \Lambda_v F_k$	$\left\|\Lambda_v F_k\right\|^2$
$\vdots$	$\vdots$	$\vdots$
1	$F^v_1 := \Lambda_v F_2$	$\left\|\Lambda_v F_2\right\|^2$
0	$F^v_0 := \Lambda_v F_1 + \left(I - M_r^{-r}\right)\Lambda_v F_0$	$\left\|\Lambda_v F_1\right\|^2 + \left\|\left(I-M_r^{-r}\right)\Lambda_v F_0\right\|^2 + 2\left\langle \Lambda_v F_1, \Lambda_v F_0\right\rangle$

Note that, by definition, for every $i \geq 2$ it holds that $\Lambda_v F_i \in C^r_{\Lambda,i-1}(X_v;\mathbb{R})$. In addition $\Lambda_v F_1 + (I - M_r^{-r})\Lambda_v F_0 \in C^r_{\Lambda,0}(X_v;\mathbb{R})$. We can therefore apply the Theorem to every link which (after some manipulation of the mixed terms that can be found in the full version) yields that, for every link v, it holds that:

$$\left\langle (M')^{+}_{v,k}\left(I - M_r^{-r}\right)\Lambda_v F, \left(I - M_r^{-r}\right)\Lambda_v F\right\rangle \le \sum_{i=0}^{k-1} \lambda_{v,i,r}\left\|F_i^v\right\|^2 + \sum_{i=0}^{p}\sum_{\substack{j=1\\ i<j}}^{p} c_{i,j}\left\langle F_i^v, F_j^v\right\rangle$$

$$= \sum_{i=0}^{k-1} \lambda_{v,i,r}\left\|\Lambda_v F_i\right\|^2 + \lambda_{v,0,r}\left\|\left(I - M_r^{-r}\right)\Lambda_v F_0\right\|^2 + \sum_{i=0}^{p}\sum_{\substack{j=1\\ i<j}}^{p} c'_{i,j}\left\langle \Lambda_v F_i, \Lambda_v F_j\right\rangle .$$

Combining this with 1 and noting that $\lambda_{\emptyset,i,k} \ge \max_{v\in X(0)}\left\{\lambda_{v,i-1,r}\right\}$ yields that:

$$\begin{aligned}\left\langle (M')^{+}_{k} F, F\right\rangle \le &\sum_{i=1}^{k} \lambda_{\emptyset,i,k}\left\|F_i\right\|^2 \\ &+ \mathbb{E}_{v\in X(0)}\left[\left\|M_r^{-r}\Lambda_v F_0\right\|^2 + \lambda_{v,0,r}\left\|\left(I - M_r^{-r}\right)\Lambda_v F_0\right\|^2\right] \\ &+ \sum_{i=0}^{p}\sum_{\substack{j=1\\ i<j}}^{p} c'_{i,j}\left\langle F_i, F_j\right\rangle .\end{aligned}$$

Therefore all we have to do is prove that:

$$\mathbb{E}_{v\in X(0)}\left[\left\|M_r^{-r}\Lambda_v F_0\right\|^2 + \lambda_{v,0,r}\left\|\left(I - M_r^{-r}\right)\Lambda_v F_0\right\|^2\right] \le \lambda_{\emptyset,0,k}\left\|F_0\right\|^2$$

This follows directly from our choice of λs:

$$\begin{aligned}&\mathbb{E}_{v\in X(0)}\left[\left\|M_r^{-r}\Lambda_v F_0\right\|^2 + \lambda_{v,0,r}\left\|\left(I - M_r^{-r}\right)\Lambda_v F_0\right\|^2\right] \\ &\quad= \mathbb{E}_{v\in X(0)}\left[\left(1 - \lambda_{v,0,r}\right)\left\|M_r^{-r}\Lambda_v F_0\right\|^2 + \lambda_{v,0,r}\left(\left\|M_r^{-r}\Lambda_v F_0\right\|^2 + \left\|\left(I - M_r^{-r}\right)\Lambda_v F_0\right\|^2\right)\right] \\ &\quad= \mathbb{E}_{v\in X(0)}\left[\left(1 - \lambda_{v,0,r}\right)\left\|M_r^{-r}\Lambda_v F_0\right\|^2 + \lambda_{v,0,r}\left\|\Lambda_v F_0\right\|^2\right] \\ &\quad\le \mathbb{E}_{v\in X(0)}\left[\left(1 - \lambda_{v,0,r}\right)\left\|M_r^{-r}\Lambda_v F_0\right\|^2\right] + \lambda_{\emptyset,1,k}\left\|F_0\right\|^2 \le \lambda_{\emptyset,0,k}\left\|F_0\right\|^2 .\end{aligned}$$ ◀

▶ **Theorem 34** (Bootstrapping Theorem). *Let X be a pure d-dimensional simplicial complex, Λ be a link viewer that respects the non-lazy up-down random walk and for every i let $F_i \in C^k_{\Lambda,\hat{i}}\left(X;\mathbb{R}\right)$ be a proper i-level cochain and $F = \sum_{i=0}^{p} F_i$. Denote $r = k - \Delta\left(\Lambda\right)$ and suppose that there are values of $\left\{\lambda_{\sigma,i,j}\right\}_{\sigma\in X, i\in[d], j\in[d]}$ such that for every $\sigma \in X$ and G_0 a 0-level cochain:*

$$\begin{cases}\lambda_{\sigma,1,k}\left\|G_0\right\|^2 + \mathbb{E}_{v\in X_\sigma(0)}\left[\left(1 - \lambda_{\sigma\cup\{v\},1,r}\right)\left\|M_r^{-r}\Lambda_v G_0\right\|^2\right] \le \lambda_{\sigma,0,k}\left\|G_0\right\|^2 \\ \max_{v\in X_\sigma(0)}\left\{\lambda_{\sigma\cup\{v\},i-1,r}\right\} \le \lambda_{\sigma,i,k}\end{cases}$$

Then:

$$\left\langle (M')^{+}_{k} F, F\right\rangle \le \sum_{i=0}^{p} \lambda_{\emptyset,i,k}\left\|F_i\right\|^2 .$$

For some constants $\left\{c_{i,j}\right\}$ and where p is the number of level functions that span the space orthogonal to the constants.

Proof. Note that the difference between this Theorem and Theorem 33 is the choice of F_is. Specifically, in this Theorem the the cochains F_i are chosen to be *proper* i-level cochains. Therefore, due to Lemma 32, they are orthogonal to each other. This allows us to apply Theorem 33 and note that:

$$\left\langle (M')^{+}_{k} F, F\right\rangle \le \sum_{i=0}^{p} \lambda_{\emptyset,i,k}\left\|F_i\right\|^2 + \sum_{i=0}^{p}\sum_{\substack{j=1\\ i<j}}^{p} c''_{i,j}\left\langle F_i, F_j\right\rangle = \sum_{i=0}^{p} \lambda_{\emptyset,i,k}\left\|F_i\right\|^2 .$$ ◀

It is important to note that, unlike the decomposition known in the two-sided case, this decomposition is not a decomposition to approximate eigenfunctions. When applying the walk operator to a level function the result might be spread over multiple levels. Theorem 34 also yields a decomposition to the up-down operator:

▶ **Corollary 35.** *With the same assumptions as Theorem 34 it holds that*

$$\left\langle M_k^+ F, F \right\rangle \leq \sum_{i=0}^{p} \left(\frac{k+1}{k+2} \lambda_{\emptyset,i,k} - \frac{1}{k+1} \right) \|F_i\|^2.$$

Proof. The following holds:

$$\begin{aligned}\left\langle M_k^+ F, F \right\rangle &= \frac{k+2}{k+1} \left\langle \left(M'\right)_k^+ F, F \right\rangle - \frac{1}{k+1} \langle F, F \rangle \\ &\leq \frac{k+2}{k+1} \langle \lambda_{\emptyset,i,k} F, F \rangle - \frac{1}{k+1} \|F\|^2 \ = \ \frac{k+2}{k+1} \lambda_{\emptyset,i,k} \|F\|^2 - \frac{1}{k+1} \|F\|^2 \\ &= \sum_{i=0}^{p} \left(\frac{k+1}{k+2} \lambda_{\emptyset,i,k} - \frac{1}{k+1} \right) \|F_i\|^2.\end{aligned}$$ ◀

We use the bootstrapping theorem in order to show both a fine grained analysis of higher order random walks (for full proof, see Appendix B) and the trickling down theorem (for full proof, see Appendix A).

6 Fine Grained Analysis of High Dimensional Random Walks From the Bootstrapping Theorem

In this short section we will present a shortened version of our fine grained analysis of higher order random walks. For the a more detailed version of this analysis, see Appendix B and the full version of this paper.

We define the localization link viewer in the following way:

▶ **Definition 36** (Localization)**.** *Given a simplicial complex X and a cochain $F \in C^k(X; \mathbb{R})$ define the localization link viewer in the following way $\forall \sigma \in X : \Lambda_\sigma^\ell F = F_\sigma$.*

First, we show that it is indeed a link viewer that respects the non-lazy random walk. We then prove the following:

▶ **Lemma 37.** *Let X be a pure d-dimensional simplicial complex and let $F \in C^k(F; \mathbb{R})$ be a cochain. Then $\langle F, \mathbb{1} \rangle = 0 \Leftrightarrow F \in \ker\left(d_{-1}^* \cdots d_{k-1}^*\right)$.*

We use this observation to gain the following advantage:

▶ **Lemma 38** (The advantage)**.** *Let X be a d dimensional simplicial complex whose 1-skeleton is a γ spectral expander. Also let $F \in C_{\Lambda^\ell,0}^k(X; \mathbb{R})$ then:*

$$\left\| d_0^* \cdots d_{k-1}^* F \right\|^2 \leq \left(1 - \frac{k}{k+1}(1-\gamma) \right) \|F\|^2.$$

For more details on the proof of this Lemma, see Lemma 53.

We then use this advantage together with our bootstrapping theorem in order to show the following:

▶ **Theorem 39** (Random walk decomposition). *Let X be a d-dimensional pure simplicial complex. Also, assume that for every face σ of dimension smaller than $d-2$ it holds that $\lambda_2\left((M')^+_{\sigma,0}\right) \leq \lambda_\sigma$. Denote by $\gamma_{\tau,i} = \max_{\sigma\in X_\tau(i)}(\lambda_\sigma)$. For every set of proper level cochains $F_i \in C^k_{\Lambda^\ell,\hat{i}}(X;\mathbb{R})$ it holds that:*

$$\left\langle (M')^+_k \sum_{i=0}^k F_i, \sum_{i=0}^k F_i \right\rangle \leq \sum_{i=0}^k \left(1 - \frac{1}{k-i+1}\prod_{j=i-1}^{k-1}(1-\gamma_j)\right)\|F_i\|^2.$$

Proof sketch, for full proof see Theorem 54. We will prove this theorem by applying Theorem 34 to the k-dimensional non-lazy random walk operator. We start by noting that the space of k-dimensional cochains that are orthogonal to the constants is comprised of exactly k level functions as the space orthogonal to the constants is exactly $\ker(d_0^* \cdots d_k^*)$.

We prove the rest of this theorem using a recursive argument. First note that for $k=0$ the claim holds trivially as $\left\langle (M')^+_0 F, F\right\rangle \leq \lambda_\sigma \|F\|^2 = \gamma_{-1}\|F\|^2$.

Assume that for every non-empty face σ it holds that $\lambda_{\sigma,i,k} \leq 1 - \frac{1}{k-i+1}\prod_{j=i-1}^{k-1}(1-\gamma_{\sigma,j})$. Note that for every $i \geq 1$:

$$\begin{aligned}\lambda_{\emptyset,i,k} &= \max_{v\in X(0)}\{\lambda_{v,i-1,k-1}\}\\ &\leq \max_{v\in X(0)}\left\{1 - \frac{1}{k-i+1}\prod_{j=i-2}^{k-2}(1-\gamma_{\sigma,j})\right\} \leq 1 - \frac{1}{k-i+1}\prod_{j=i-1}^{k-1}(1-\gamma_{\emptyset,j}).\end{aligned}$$

Consider the left hand side of the recursive formula:

$$\lambda_{\emptyset,1,k}\|F_0\|^2 + \mathbb{E}_{v\in X(0)}\left[(1-\lambda_{\emptyset,1,k})\left\|M^{-(k-1)}_{(k-1)}\Lambda^\ell_v F_0\right\|^2\right] \leq \lambda_{\emptyset,0,k}\|F_0\|. \tag{2}$$

We show that:

$$\begin{aligned}&\lambda_{\emptyset,1,k}\|F_0\|^2 + \mathbb{E}_{v\in X(0)}\left[(1-\lambda_{\emptyset,1,k})\left\|M^{-(k-1)}_{(k-1)}\Lambda^\ell_v F_0\right\|^2\right]\\ &\qquad = \lambda_{\emptyset,1,k}\|F_0\|^2 + (1-\lambda_{\emptyset,1,k})\left\|d_0^*\cdots d_{k-1}^* F_0\right\|^2.\end{aligned}$$

Consider, again, the left hand side of inequality 2 and note that due to Lemma 53 it suffices to solve the following:

$$\lambda_{\emptyset,1,k}\left\|F^{=0}\right\|^2 + (1-\lambda_{\emptyset,1,k})\left(1 - \frac{k}{k+1}(1-\gamma_{-1})\right)\left\|F^{=0}\right\|^2 \leq \lambda_{\emptyset,0,k}\left\|F^{=0}\right\|^2.$$

We show that if we set $\lambda_{\emptyset,0,k} = 1 - \frac{1}{k+1}\prod_{j=-1}^{k-1}(1-\gamma_{\emptyset,j})$ we get that for every face σ and dimensions i,k:

$$\lambda_{\sigma,i,k} \leq 1 - \frac{1}{k-i+1}\prod_{j=i-1}^{k-1}(1-\gamma_{\sigma,j}).$$

Applying Theorem 34 proves the decomposition. ◀

References

1 Vedat Levi Alev and Lap Chi Lau. Improved analysis of higher order random walks and applications. *CoRR*, abs/2001.02827, 2020. arXiv:2001.02827.

2 Nima Anari, Vishesh Jain, Frederic Koehler, Huy Tuan Pham, and Thuy-Duong Vuong. Entropic independence: optimal mixing of down-up random walks. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1418–1430, 2022.

3 Nima Anari, Kuikui Liu, and Shayan Oveis Gharan. Spectral independence in high-dimensional expanders and applications to the hardcore model. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1319–1330. IEEE, 2020. doi:10.1109/FOCS46700.2020.00125.

4 Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials ii: High-dimensional walks and an fpras for counting bases of a matroid, 2018. doi:10.48550/arXiv.1811.01816.

5 Mitali Bafna, Max Hopkins, Tali Kaufman, and Shachar Lovett. High dimensional expanders: Eigenstripping, pseudorandomness, and unique games, 2020. doi:10.48550/arXiv.2011.04658.

6 Mitali Bafna, Max Hopkins, Tali Kaufman, and Shachar Lovett. Hypercontractivity on high dimensional expanders: a local-to-global approach for higher moments, 2021. doi:10.48550/arXiv.2111.09444.

7 Zongchen Chen, Kuikui Liu, and Eric Vigoda. Optimal mixing of glauber dynamics: Entropy factorization via high-dimensional expansion. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1537–1550, 2021.

8 Yotam Dikstein, Irit Dinur, Yuval Filmus, and Prahladh Harsha. Boolean function analysis on high-dimensional expanders. In Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20-22, 2018 – Princeton, NJ, USA*, volume 116 of *LIPIcs*, pages 38:1–38:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.38.

9 Irit Dinur, Prahladh Harsha, Tali Kaufman, Inbal Livni Navon, and Amnon Ta-Shma. List decoding with double samplers. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2134–2153. SIAM, 2019. doi:10.1137/1.9781611975482.129.

10 Irit Dinur and Tali Kaufman. High dimensional expanders imply agreement expanders. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 974–985. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.94.

11 Roy Gotlib and Tali Kaufman. Fine grained analysis of high dimensional random walks, 2023. arXiv:2208.03241.

12 Tom Gur, Noam Lifshitz, and Siqi Liu. Hypercontractivity on high dimensional expanders. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20–24, 2022*, pages 176–184. ACM, 2022. doi:10.1145/3519935.3520004.

13 Tali Kaufman and David Mass. High dimensional random walks and colorful expansion. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, volume 67 of *LIPIcs*, pages 4:1–4:27. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ITCS.2017.4.

14 Tali Kaufman and David Mass. Local-to-global agreement expansion via the variance method. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 74:1–74:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ITCS.2020.74.

15 Tali Kaufman and Izhar Oppenheim. High order random walks: Beyond spectral gap. In Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20-22, 2018 – Princeton, NJ, USA*, volume 116 of *LIPIcs*, pages 47:1–47:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.47.

16 Izhar Oppenheim. Local spectral expansion approach to high dimensional expanders part i: Descent of spectral gaps, 2017. doi:10.48550/arXiv.1709.04431.

A Trickling Down

Before we present our random walk decomposition Theorem, let us start with a “warm up”: An alternative proof for the trickling down theorem [16] that is based on Theorem 34. We believe that the fact that the main tool presented here can be used to prove the trickling down theorem is of independent interest: it shows that there is a single, local to global argument at the heart of both claims. In the trickling down theorem we are interested in the connection between the 0-dimensional non-lazy random walk on the vertices of a complex and the 0-dimensional non-lazy up-down random walk on the links of the vertices of the complex. It will, therefore, be natural to consider a link viewer that does not incur a decrease in dimension. One such link viewer is the restriction link viewer defined as:

▶ **Definition 40** (Restriction). *Given a simplicial complex X and a cochain $F \in C^k(X;\mathbb{R})$ define the restriction link viewer in the following way: $\forall \sigma \in X : \Lambda^r_\sigma F(\tau) = F_0(\tau)$.*

This link viewer maps the 0-dimensional non-lazy up-down random walk to the 0-dimensional non-lazy up-down on the links of the vertices. We will show that applying Theorem 34 to the restriction link viewer yields the trickling down theorem. However, before applying Theorem 34, we must first show that restriction is indeed a link viewer that respects the non-lazy up-down random walk. We state here that this is indeed the case and leave the proofs of these claims to the full version of this paper.

▶ **Lemma 41.** *The restriction link viewer is a link viewer.*

▶ **Lemma 42.** *The restriction link viewer respects the non-lazy random walk operator.*

▶ **Lemma 43.** *It holds for every vertex v and every cochain $F \in C^0(X;\mathbb{R})$ that:*

$$M^-_{v,0}\Lambda^r_v F_0 = \langle \Lambda^r_v F, \mathbb{1}\rangle_v = \mathbb{E}_{u\in X_v(0)}\left[\Lambda^r_v F(u)\right] = (M')^+_0 F(v)$$

Proof.

$$(M')^+_0 F(v) = \left[(M')^+_0 F\right]_v = \sum_{u\in X(0)} \left[(M')^+_0\right]_{v,u} F(u) = \sum_{\substack{u\in X(0)\\ u\cup v\in X(1)}} \mathrm{w}_v\left(u\setminus v\right) F(u)$$

$$= \sum_{u\in X_v(0)} \mathrm{w}_v\left(u\right)\Lambda^r_v F(u) = \mathbb{E}_{u\in X_v(0)}\left[\Lambda^r_v F(u)\right]. \qquad ◀$$

▶ **Corollary 44** (The advantage). *If X’s 1-skeleton is a λ-spectral expander it holds for every vertex v and every cochain $F \in C^0(X;\mathbb{R})$ that:*

$$\left\|d^*_0\Lambda^r_v F_0\right\| = \left\|M^-_0\Lambda^r_v F_0\right\| \le \lambda^2\left\|F\right\|^2.$$

Proof. Note that the 1-skeleton of X is a λ-spectral expander and $F \in C^0(X;\mathbb{R})$ therefore $\left\|(M')^+_0 F\right\|^2 \le \lambda^2\left\|F\right\|^2$. Combining this with Lemma 43 yields:

$$\left\|M^-_0\Lambda^r_v F_0\right\|^2 = \left\|(M')^+_0 F\right\|^2 = \left\|(M')^+_0 F_0\right\|^2 \le \lambda^2\left\|F_0\right\|^2. \qquad ◀$$

We can now show how applying Theorem 34 to the restriction link viewer yields Oppenheim's trickling down theorem [16, Theorem 4.1].

▶ **Theorem 45** (Trickling Down, restated Theorem 16). *If it holds that:*

- *For every vertex v: X_v is a $\lambda_{v,0,k}$ spectral expander.*
- *X is connected.*

Then it holds that $\lambda_{\emptyset,0,k} = \frac{\lambda_{\emptyset,1,k}}{1-\lambda_{\emptyset,1,k}}$.

Proof. Consider the following:

$$\lambda_{\emptyset,1,k} \left\|F_0\right\|^2 + (1 - \lambda_{\emptyset,1,k})\, \mathbb{E}_{v \in X(0)} \left[\left\|M^{-}_{v,0} \Lambda^{r}_{v} F_0\right\|^2\right] \leq \lambda_{\emptyset,0,k} \left\|F_0\right\|^2 .$$

Using Corollary 44 it suffices to find values of $\lambda_{\sigma,i,j}$ such that:

$$\lambda_{\emptyset,1,k} \left\|F_0\right\|^2 + (1 - \lambda_{\emptyset,1,k}) \lambda^2_{\emptyset,0,k} \left\|F_0\right\|^2 \leq \lambda_{\emptyset,0,k} \left\|F_0\right\|^2 .$$

Therefore solving the following inequality would bound $\lambda_{\emptyset,0,k}$:

$$\lambda_{\emptyset,1,k} + (1 - \lambda_{\emptyset,1,k}) \lambda^2_{\emptyset,0,k} \leq \lambda_{\emptyset,0,k}.$$

Note that picking $\lambda_{\emptyset,0,k} = \frac{\lambda_{\emptyset,1,k}}{1-\lambda_{\emptyset,1,k}}$ satisfies the inequality and thus proves the Theorem. ◀

B Decomposition of the Random Walk Operators

We are now ready to present the random walk decomposition theorem based on Theorem 34. Unlike the tricking down theorem, here the assumption we have is only on the expansion of the no-lazy up-down random walk on the vertices. We will, therefore, be interested in a link viewer that decreases the dimension of the cochain. Namely, the localization link viewer, defined as follows:

▶ **Definition 46** (Localization). *Given a simplicial complex X and a cochain $F \in C^k(X; \mathbb{R})$ define the localization link viewer in the following way: $\forall \sigma \in X : \Lambda^{\ell}_{\sigma} F = F_\sigma$.*

As with the trickling down theorem, we will be interested in applying Theorem 34 to the localization link viewer. We start by proving that the localization link viewer that respects the non-lazy up-down random walk. Full proofs of these claims can be found in the full version of this paper.

▶ **Lemma 47.** *The localization link viewer is a link viewer.*

▶ **Lemma 48.** *The localization link viewer respects the non-lazy random walk operator.*

B.1 Gaining the Advantage

Before we present the exact Lemma we use as the advantage step we should expand our understanding of Λ^{ℓ}'s level functions. We will begin by characterising the space of cochains that are orthogonal to the eigenspace of 1 (i.e. the constant functions).

▶ **Lemma 49.** *Let X be a pure d-dimensional simplicial complex and let $F \in C^k(F; \mathbb{R})$ be a cochain. Then: $\langle F, \mathbb{1} \rangle = 0 \Leftrightarrow F \in \ker\left(d^*_{-1} \cdots d^*_{k-1}\right)$.*

Proof. Note that, due to Lemma 20 it holds that: $\langle F, \mathbb{1} \rangle = \mathbb{E}_{\sigma \in X(k)} [F(\sigma)] = d^*_{-1} \cdots d^*_{k-1}(\emptyset)$. This proves the lemma. ◀

Now that we understand the constant part of a cochain we are ready to move on to understanding cochains of a higher level.

▶ **Lemma 50.** *Let X be a pure d-dimensional simplicial complex, i be a dimension and $F \in C^k(X;\mathbb{R})$ be a cochain. Then: $\forall \sigma \in X(i) : \left\langle \Lambda_\sigma^\ell F, \mathbb{1} \right\rangle_\sigma = 0 \Leftrightarrow F \in \ker\left(d_i^* \cdots d_{k-1}^*\right)$.*

Proof. Using Lemma 20 we prove that:

$$\begin{aligned}\forall \sigma \in X(i) : \left\langle \Lambda_\sigma^\ell F, \mathbb{1} \right\rangle_\sigma &= \sum_{\tau \in X_\sigma(0)} \mathrm{w}_\sigma(\tau) \Lambda_\sigma^\ell F(\tau) \\ &= d_{\sigma,i-1}^* \cdots d_{\sigma,k-2}^* F_\sigma(\emptyset) = \left(d_i^* \cdots d_{k-1}^* F\right)_\sigma(\emptyset) = d_i^* \cdots d_{k-1}^* F(\sigma).\end{aligned}$$

Which proves the Lemma. ◀

▶ **Corollary 51.** *It holds that F is a proper k-dimensional i-level cochain iff $F \in \operatorname{Im}(d_{k-1} \cdots d_i) \cap \ker\left(d_{i-1}^* \cdots d_{k-1}^*\right)$.*

Proof. The Corollary holds due the definition of i-level cochains and Lemma 50. ◀

Note that cochains that pure i-level cochains can be thought of as originating in the i-dimensional faces. For example, for every pure 0-level cochain there is a 0-dimensional cochain G such that $F = d \cdots dG$. We also note that any cochain that is not originated in the vertices can be distributed along the links in the sense that they remain orthogonal to the constants when applying the localization link viewer. We will therefore be interested in the cochains that originated in the vertices (as these are exactly the cochains which the local perspective seems to miss). Specifically, we show that the following hold:

▶ **Lemma 52.** *Let F be a k-dimensional proper 0-level cochain then there exists $F^{=0} \in C^0(X;\mathbb{R})$ such that:*

1. $d_{-1}^* F^{=0} = 0$
2. $\|F\|^2 = \left\|F^{=0}\right\|^2$
3. $\left\|d_0^* \cdots d_{k-1}^* F\right\|^2 = \left\|d_{k-1} \cdots d_0 F^{=0}\right\|^2$

The proof of this Lemma can be found in the full version of this paper.

We are now ready to present the advantage we use:

▶ **Lemma 53** (The advantage). *Let X be a d dimensional simplicial complex whose 1-skeleton is a γ spectral expander. Also let $F \in C_{\Lambda^\ell,0}^k(X;\mathbb{R})$ then:*

$$\left\|d_0^* \cdots d_{k-1}^* F\right\|^2 \leq \left(1 - \frac{k}{k+1}(1-\gamma)\right) \|F\|^2$$

Proof. Let F_0 be the projection of F into $C_{\Lambda^\ell,0}^k(X;\mathbb{R}) \cap \left(C_{\Lambda^\ell,1}^k(X;\mathbb{R})\right)^\perp$. Due to Lemma 50 it holds that for every dimension i that $C_{\Lambda^\ell,i}^k(X;\mathbb{R}) = \ker\left(d_{i-1}^* \cdots d_{k-1}^*\right)$ and therefore $F_0 \in \operatorname{Im}(d_{k-1} \cdots d_0) \cap \ker\left(d_{-1}^* \cdots d_{k-1}^*\right)$. We can therefore use Lemma 52 to find $F^{=0}$ such that:

1. $d_{-1}^* F^{=0} = 0$
2. $\|F_0\|^2 = \left\|F^{=0}\right\|^2$
3. $\left\|d_0^* \cdots d_{k-1}^* F_0\right\|^2 = \left\|d_{k-1} \cdots d_0 F^{=0}\right\|^2$

Due to Lemma 26 it holds that:

$$M_0^{+k} = \frac{k}{k+1}\left(M'\right)_0^+ + \frac{1}{k+1} I$$

And therefore:

$$\begin{aligned}
&\left\|d_{k-1}\cdots d_0 F^{=0}\right\|^2 = \left\langle d_{k-1}\cdots d_0 F^{=0}, d_{k-1}\cdots d_0 F^{=0}\right\rangle = \left\langle M_0^{+k} F^{=0}, F^{=0}\right\rangle = \\
&= \left\langle \left(\frac{k}{k+1}\left(M'\right)_0^+ - \frac{1}{k+1} I\right) F^{=0}, F^{=0}\right\rangle = \\
&= \frac{k}{k+1}\left\langle \left(M'\right)_0^+ F^{=0}, F^{=0}\right\rangle + \frac{1}{k+1}\left\langle F^{=0}, F^{=0}\right\rangle \le \\
&\le \left(\frac{k}{k+1}\gamma + \frac{1}{k+1}\right)\left\|F^{=0}\right\|^2 = \\
&= \left(\frac{k}{k+1}\gamma - \frac{k}{k+1} + 1\right)\left\|F^{=0}\right\|^2 = \\
&\qquad = \left(1 - \frac{k}{k+1}\left(1-\gamma\right)\right)\left\|F^{=0}\right\|^2 = \left(1 - \frac{k}{k+1}\left(1-\gamma\right)\right)\left\|F_0\right\|^2
\end{aligned}$$

And thus:

$$\begin{aligned}
\left\|d_0^* \cdots d_{k-1}^* F\right\|^2 &= \left\|d_{k-1}\cdots d_0 F_0\right\|^2 \\
&\le \left(1 - \frac{k}{k+1}\left(1-\gamma\right)\right)\left\|F_0\right\|^2 \le \left(1 - \frac{k}{k+1}\left(1-\gamma\right)\right)\left\|F\right\|^2 .
\end{aligned}$$

◀

B.2 Decomposing the Random Walk Operators

Now that we have developed the tools we need, we can move on to strengthening the result of Alev and Lau [1] by showing a decomposition of the random walk operators. We will do so by applying Theorem 34 to the localization link viewer:

▶ **Theorem 54** (Random walk decomposition). *Let X be a d-dimensional pure simplicial complex. Also, assume that for every face σ of dimension smaller than $d-2$ it holds that $\lambda_2\left(\left(M'\right)_{\sigma,0}^+\right) \le \lambda_\sigma$. Denote by $\gamma_{\tau,i} = \max_{\sigma\in X_\tau(i)}\left(\lambda_\sigma\right)$. For every set of proper level cochains $F_i \in C^k_{\Lambda^\ell,\hat{i}}\left(X;\mathbb{R}\right)$ it holds that:*

$$\left\langle \left(M'\right)_k^+ \sum_{i=0}^{k} F_i, \sum_{i=0}^{k} F_i\right\rangle \le \sum_{i=0}^{k}\left(1 - \frac{1}{k-i+1}\prod_{j=i-1}^{k-1}\left(1-\gamma_j\right)\right)\left\|F_i\right\|^2.$$

Proof. We will prove this theorem by applying Theorem 34 to the k-dimensional non-lazy random walk operator. We start by noting that the space of k-dimensional cochains that are orthogonal to the constants is comprised of exactly k level functions as the space orthogonal to the constants is exactly $\ker\left(d_0^*\cdots d_k^*\right)$.

We prove the rest of this theorem using a recursive argument. First note that for $k=0$ the claim holds trivially as:

$$\left\langle \left(M'\right)_0^+ F, F\right\rangle \le \lambda_\sigma \left\|F\right\|^2 = \gamma_{-1}\left\|F\right\|^2 .$$

Assume that for every non-empty face σ it holds that $\lambda_{\sigma,i,k} \le 1 - \frac{1}{k-i+1}\prod_{j=i-1}^{k-1}\left(1-\gamma_{\sigma,j}\right)$ and note that for every $i \ge 1$:

$$\begin{aligned}\lambda_{\emptyset,i,k} &= \max_{v\in X(0)}\left\{\lambda_{v,i-1,k-1}\right\}\\ &\leq \max_{v\in X(0)}\left\{1-\frac{1}{k-i+1}\prod_{j=i-2}^{k-2}\left(1-\gamma_{\sigma,j}\right)\right\} \;\leq\; 1-\frac{1}{k-i+1}\prod_{j=i-1}^{k-1}\left(1-\gamma_{\emptyset,j}\right).\end{aligned}$$

Consider the left hand side of the recursive formula:

$$\lambda_{\emptyset,1,k}\left\|F_0\right\|^2+\mathbb{E}_{v\in X(0)}\left[\left(1-\lambda_{\emptyset,1,k}\right)\left\|M_{(k-1)}^{-(k-1)}\Lambda_v^{\ell}F_0\right\|^2\right]\leq\lambda_{\emptyset,0,k}\left\|F_0\right\|. \tag{3}$$

And note that:

$$\begin{aligned}&\lambda_{\emptyset,1,k}\left\|F_0\right\|^2+\mathbb{E}_{v\in X(0)}\left[\left(1-\lambda_{\emptyset,1,k}\right)\left\|M_{(k-1)}^{-(k-1)}\Lambda_v^{\ell}F_0\right\|^2\right]\\ &\qquad=\lambda_{\emptyset,1,k}\left\|F_0\right\|^2+\mathbb{E}_{v\in X(0)}\left[\left(1-\lambda_{\emptyset,1,k}\right)\left\|d_{-1}^*\cdots d_{k-2}^*\Lambda_v^{\ell}F_0\right\|^2\right]\\ &\qquad=\lambda_{\emptyset,1,k}\left\|F_0\right\|^2+\mathbb{E}_{v\in X(0)}\left[\left(1-\lambda_{\emptyset,1,k}\right)\left\|\Lambda_v^{\ell}d_0^*\cdots d_{k-1}^*F_0\right\|^2\right]\\ &\qquad=\lambda_{\emptyset,1,k}\left\|F_0\right\|^2+\left(1-\lambda_{\emptyset,1,k}\right)\left\|d_0^*\cdots d_{k-1}^*F_0\right\|^2.\end{aligned}$$

Consider, again, the left hand side of inequality 3 and note that due to Lemma 53 it suffices to solve the following:

$$\begin{aligned}&\lambda_{\emptyset,1,k}\left\|F^{=0}\right\|^2+\left(1-\lambda_{\emptyset,1,k}\right)\left(1-\frac{k}{k+1}\left(1-\gamma_{-1}\right)\right)\left\|F^{=0}\right\|^2\leq\lambda_{\emptyset,0,k}\left\|F^{=0}\right\|^2\\ &\lambda_{\emptyset,1,k}+\left(1-\lambda_{\emptyset,1,k}\right)\left(1-\frac{k}{k+1}\left(1-\gamma_{-1}\right)\right)\leq\lambda_{\emptyset,0,k}\\ &\qquad\qquad\frac{k}{k+1}\left(1-\gamma_{-1}\right)\lambda_{\emptyset,1,k}+\left(1-\frac{k}{k+1}\left(1-\gamma_{-1}\right)\right)\leq\lambda_{\emptyset,0,k}.\end{aligned}$$

Consider the following:

$$\begin{aligned}&\frac{k}{k+1}\left(1-\gamma_{-1}\right)\lambda_{\emptyset,1,k}+\left(1-\frac{k}{k+1}\left(1-\gamma_{-1}\right)\right)\\ &\qquad\leq\frac{k}{k+1}\left(1-\gamma_{-1}\right)\left(1-\frac{1}{k}\prod_{j=0}^{k-1}\left(1-\gamma_{\emptyset,j}\right)\right)+1-\frac{k}{k+1}\left(1-\gamma_{-1}\right)\\ &\qquad=\frac{k}{k+1}\left(1-\gamma_{-1}\right)-\frac{1}{k+1}\prod_{j=-1}^{k-1}\left(1-\gamma_{\emptyset,j}\right)+1-\frac{k}{k+1}\left(1-\gamma_{-1}\right)\\ &\qquad=1-\frac{1}{k+1}\prod_{j=-1}^{k-1}\left(1-\gamma_{\emptyset,j}\right).\end{aligned}$$

Thus if we set:

$$\lambda_{\emptyset,0,k}=1-\frac{1}{k+1}\prod_{j=-1}^{k-1}\left(1-\gamma_{\emptyset,j}\right).$$

We get that for every face σ and dimensions i, k:

$$\lambda_{\sigma,i,k}\leq 1-\frac{1}{k-i+1}\prod_{j=i-1}^{k-1}\left(1-\gamma_{\sigma,j}\right).$$

Applying Theorem 34 proves the decomposition. ◀

Note that the decomposition presented in Theorem 54 is the first decomposition theorem to offer a proper (i.e. not approximate) decomposition of the k-dimensional random walk whose components are orthogonal to each other.

A Deterministic Construction of a Large Distance Code from the Wozencraft Ensemble

Venkatesan Guruswami
Department of EECS, University of California, Berkeley, CA, USA

Shilun Li
Department of Mathematics, University of California, Berkeley, CA, USA

Abstract

We present an explicit construction of a sequence of rate 1/2 Wozencraft ensemble codes (over any fixed finite field $\mathbb{F}_q$) that achieve minimum distance $\Omega(\sqrt{k})$ where k is the message length. The coefficients of the Wozencraft ensemble codes are constructed using Sidon Sets and the cyclic structure of $\mathbb{F}_{q^k}$ where $k+1$ is prime with q a primitive root modulo $k+1$. Assuming Artin's conjecture, there are infinitely many such k for any prime power q.

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization; Theory of computation → Error-correcting codes

Keywords and phrases Algebraic codes, Pseudorandomness, Explicit Construction, Wozencraft Ensemble, Sidon Sets

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.50

Category RANDOM

Related Version *Full Version*: `https://arxiv.org/abs/2305.02484`

Funding *Venkatesan Guruswami*: Research supported by a Simons Investigator Award and NSF grants CCF-2210823 and CCF-2228287.
Shilun Li: Research supported by University of California, Berkeley under Berkeley Fellowship.

1 Introduction

The explicit construction of binary error-correcting codes with a rate vs. distance trade-off approaching that of random constructions, i.e., the so-called Gilbert-Varshamov (GV) bound, remains an outstanding challenge in coding theory and combinatorics.

For large n, a random binary linear code of rate $R \in (0,1)$, defined for example as the column span of a random matrix $G \in \mathbb{F}_2^{n \times Rn}$, has relative distance $h^{-1}(1-R)$ with high probability, where $h^{-1}(\cdot)$ is the inverse of the binary entropy function. There is a similar GV bound $h_q^{-1}(1-R)$, involving the q-ary entropy function, for codes over other finite fields $\mathbb{F}_q$.

While explicit constructions meeting the GV bound remain elusive[1], there are known derandomizations showing that codes drawn randomly from much smaller, structured ensembles can also achieve the GV bound. One of the most classical and famous such ensemble is the Wozencraft ensemble, which consists of codes $C_{\mathrm{WE}}^{\alpha} = \{(x, \alpha x) : x \in \mathbb{F}_{q^k}\}$ as α varies over nonzero elements of the field $\mathbb{F}_{q^k}$, and one uses some fixed basis to express elements of $\mathbb{F}_{q^k}$ as length k vectors over $\mathbb{F}_q$. Note that each code C_{WE}^{α} has rate 1/2. The construction of Wozencraft ensemble codes C_{WE}^{α} first appeared in a paper by Massey [13], who attributed the discovery of these codes to John M. Wozencraft.

[1] Ta-Shma [19] recently constructed explicit binary codes near the GV bound for low rates. The codes have distance $\frac{1-\epsilon}{2}$ and rate $\Omega(\epsilon^{2+o(1)})$ which is asymptotically close to the rate $\Omega(\epsilon^2)$ guaranteed by the GV bound.

It is a standard exercise to show that for most choices of α, the code C^{α}_{WE} has distance close to $h_q^{-1}(1/2)$ and thus achieves the GV bound. Puncturing the Wozencraft ensemble gives codes of higher rates that also meet the GV bound. The property of the Wozencraft ensemble and its punctured variants behind this phenomenon is that every nonzero word appears as a codeword in an equal number of codes in the ensemble (if it appears in any code of the ensemble at all).

Using this property, Justesen [10] in 1972 gave the first strongly explicit asymptotically good binary linear codes, by concatenating an outer Reed–Solomon code with different inner codes drawn from the Wozencraft ensemble for different positions of the Reed-Solomon code. The Justesen construction achieves a trade-off between rate vs. distance called the Zyablov bound for rates at least 0.3. There are variants of Wozencraft codes which give Zyablov-bound-achieving codes for lower rates as well (see Section 5). In recent years, Wozencraft ensemble codes have also found other varied uses, for example in constructing covering codes of small block length [14] and write-once-memory codes [16, 17].

Given that for most α, the code C^{α}_{WE} meets the GV bound, it is a natural question whether one can find an explicit α for which the code has good distance (even if it doesn't quite meet the GV bound). Gaborit and Zemor [8] showed that it suffices to consider random α in a subset of size $\approx q^k/k$ and used it to show the existence of linear codes which are a factor k larger in terms of size than the Gilbert-Varshamov bound (such a result was shown earlier for general codes in [9]).

However, it remains an outstanding challenge to find some α in deterministic poly(k) time for which C^{α}_{WE} has distance $\Omega(k)$. This question is relatively well-known, eg. it received mention in a blog post by Dick Lipton [12], but has resisted progress. To the best of our knowledge, even an explicit α for which C^{α}_{WE} has distance $k^{\Omega(1)}$ was not known.

For certain structured fields $\mathbb{F}_{q^k}$ (of which there are an infinite family under Artin's conjecture), we give an explicit construction of $\alpha \in \mathbb{F}_{q^k}$ for which C^{α}_{WE} has distance $\Omega(\sqrt{k})$. We also give an explicit puncturing of these codes to achieve any desired rate $r < 1$, and $\Omega_r(\sqrt{k})$ distance (the constant in the $\Omega()$ depends on r). Our theorems are informally stated below.

▶ **Theorem 1** (Informal). *Fix a field $\mathbb{F}_q$ and consider an integer k such that $k+1$ is prime and q is a primitive root modulo $k+1$. There exist $\alpha^* \in \mathbb{F}_{q^k}$ which can be constructed in deterministic poly(k) time such that:*

- *$C^{\alpha^*}_{WE}$ has distance $\Omega(\sqrt{k})$.*
- *For any $r > \frac{1}{2}$, there is an explicit puncturing of $C^{\alpha^*}_{WE}$ with rate at least r and distance $\Omega_r(\sqrt{k})$.*

Please refer to Theorem 7 and Theorem 17 for construction of α^* and choice of puncturing.

2 Preliminaries

Throughout this paper, we will assume the alphabet has size q, where q is a prime power. Furthermore, we will assume k' is a prime such that q is a primitive root modulo k'. Assuming Artin's conjecture, such k' exists infinitely often at sufficiently high density and can be efficiently found in deterministic poly(k') time.

Denote $k = k'-1$ for ease of notation. Let $p(x) = 1+x+x^2+\ldots+x^{k'-1}$ be the cyclotomic polynomial which is irreducible over $\mathbb{F}_q$ (see Proposition 3). Note that $\mathbb{F}_q[x]/(p(x)) \cong \mathbb{F}_{q^k}$ for the extension field $\mathbb{F}_q[x]/(p(x))$ and we will fix the representation of $\mathbb{F}_{q^k}$ as polynomials in x of degree less than k, with operations performed modulo $p(x)$. We will fix the $\mathbb{F}_q$-linear isomorphism $\varphi : \mathbb{F}_{q^k} \to \mathbb{F}_q^k$ that maps a polynomial of degree less than k to its coefficient vector. That is, $\varphi(\sum_{i=0}^{k-1} a_i x^i) = (a_0, a_1, \ldots, a_{k-1})$. For $\alpha \in \mathbb{F}_{q^k}$, define $\mathrm{wt}(\alpha)$ to be the Hamming weight of $\varphi(\alpha)$.

The Wozencraft ensemble is a classic family of codes defined as follows.

▶ **Definition 2.** *For $\alpha \in \mathbb{F}_{q^k}$, the* ***Wozencraft ensemble code*** *C_{WE}^{α} parameterized by α is given by*

$$C_{WE}^{\alpha} = \{(\varphi(x), \varphi(\alpha x)) : x \in \mathbb{F}_{q^k}\} \ .$$

Note that C_{WE}^{α} is an $\mathbb{F}_q$-linear code of rate $1/2$.

▶ **Proposition 3.** *If k' is a prime such that q is a primitive root modulo k', then $p(x) = \sum_{i=0}^{k'-1} x^i$ is an irreducible polynomial of degree $k' - 1$ in $\mathbb{F}_q[x]$.*

Proof. Since q is a primitive root modulo k', $d = k' - 1$ is the smallest integer satisfying $q^d \equiv 1 \bmod k'$. Note that a field extension $\mathbb{F}_{q^d}$ of $\mathbb{F}_q$ contains a primitive k'-th root of unity ζ if and only if $k' | q^d - 1$, i.e., $q^d \equiv 1 \bmod k'$. So $\mathbb{F}_{q^{k'-1}}$ is the smallest field extension of $\mathbb{F}_q$ containing ζ and the mimimal polynomial of ζ has degree $k' - 1$. Since $p = \sum_{i=0}^{k'-1} x^i$ is a degree $k' - 1$ polynomial such that $p(\zeta) = 0$, p is the minimal polynomial of ζ and thus irreducible. ◀

We will use Sidon sets [18, 1] to construct the parameter α for C_{WE}^{α}.

▶ **Definition 4.** *A* ***Sidon set*** *is a set of integers $A = \{a_1, \ldots, a_d\}$ where $a_1 < a_2 < \ldots < a_d$ such that for all $i, j, k, l \in [d]$ with $i \neq j$ and $k \neq l$,*

$$a_i - a_j = a_k - a_l \iff i = k \text{ and } j = l.$$

A ***Sidon set modulo n*** *is a* ***Sidon set*** *such that for all $i, j, k, l \in [d]$ with $i \neq j$ and $k \neq l$,*

$$a_i - a_j \equiv a_k - a_l \pmod{n} \iff i = k \text{ and } j = l.$$

Size d of the Sidon set A is referred to as its ***order*** *and $a_d - a_1$ as its* ***length.***

▶ Remark 5. For any Sidon set with order d and length m, the $\binom{d}{2}$ distances between each pair of points need to be distinct. So $m \geq \binom{d}{2}$ and this gives a trivial upper bound $d \leq \sqrt{2m}$. This upper bound on d can be improved to $d \leq \sqrt{m} + O(m^{1/4})$ [7] and further to $d \leq \sqrt{m} + m^{1/4} + 1$ [11]. On the other hand, the maximal d given m satisfies $d \geq \sqrt{m} - O(m^{5/16})$ [7] but it is believed that we can have $d > \sqrt{m}$ [6].

We will introduce the Bose-Chowla construction of Sidon sets [4, 3].

▶ **Theorem 6** (Bose-Chowla, [4]). *Let p be a power of a prime, g be a primitive root in $\mathbb{F}_{p^2}$. Then the sequence of p integers*

$$A = \{i \in [p^2 - 2] : g^i + g^{pi} = 1\}$$

forms a Sidon set modulo $p^2 - 1$.

This construction of Sidon set has order $d = p$ and length at most $m(d) = p^2 - 2$. They are asymptotically optimal in the sense that $\lim_{d\to\infty} \sqrt{m(d)}/d = 1$. Given p, such construction can be done in $O(p^2)$ time by fixing $s = 1$ and finding a primitive root g via naive search. We will later define our parameter α to be $\sum_{a \in A} x^a$ where A is a Sidon set and show that C_{WE}^{α} has good distance.

3 Rate 1/2 Construction

In this section, we will give explicit construction of rate 1/2 Wozencraft ensemble codes with minimum distance $\Omega(\sqrt{k})$ using Sidon sets. To begin, we provide an intuitive explanation for the natural occurrence of Sidon sets in this specific context. Subsequently, we proceed with the analysis of the minimum distance of our construction.

3.1 Motivation

Fix a set of indices $A \subseteq [k]$ and an element $\alpha = \sum_{a\in A} x^a \in \mathbb{F}_{q^k}$ with coefficients either 0 or 1. Take any $y = \sum_{s\in S} b_s x^s$ where S is the set of non-zero indices of the coefficients of y, so $b_s \neq 0$ for all $s \in S$. To establish a lower bound on $\Delta(C^{\alpha}_{\mathrm{WE}})$, we would like show a lower bound on the weight of the product $\alpha y \in \mathbb{F}_{q^k}$ for any y. To simplify the analysis, we consider a ring extension of $\mathbb{F}_{q^k}$ (described in Section 3.2), in which the coefficient c_j in front of x^j of the product αy can be expressed as $c_j = \sum_{a\in A, s\in S} \mathbf{1}\{a + s \equiv j \pmod{k'}\} b_s$. We would like to establish a lower bound on the number of non-zero coefficients $c_j \neq 0$, which would transform into a lower bound on the weight of αy in $\mathbb{F}_{q^k}$. It is sufficient to demonstrate the existence of numerous choices of j satisfying $j \equiv a+s \pmod{k'}$ for a unique combination of a and s. In this case, c_j corresponds to the sum of one non-zero element and is therefore non-zero. To ensure there are an abundance of such choices for j with this uniqueness property, it is desirable to minimize collisions of the form $a + s \equiv a' + s' \pmod{k'}$ where $a, a' \in A$ and $s, s' \in S$. Since S can be selected adversarially with respect to A, it is advantageous to have $(a - a') \mod k'$ be unique, which is exactly the property of Sidon sets modulo k'. The result of this construction is stated formally as follows, where the proof is propounded to Section 3.2:

▶ **Theorem 7.** *Let d be the largest prime smaller than $\sqrt{k}$ and let $A = \{a_1, \ldots, a_d\}$ be a Bose-Chowla Sidon set with order d. Define $\alpha^* = \sum_{a\in A} x^a \in \mathbb{F}_{q^k}$. Then $\Delta(C^{\alpha^*}_{WE}) \geq d = \Omega(\sqrt{k})$.*

▶ Remark 8. Since there exists a prime between $[\frac{1}{2}\sqrt{k}, \sqrt{k}]$ by Bertrand–Chebyshev theorem [5], d can be found efficiently via naive search and $d \geq \frac{1}{2}\sqrt{k}$. Moreover, Baker, Harman and Pintz [2] showed that there exists a prime in the interval $[\sqrt{k} - k^{0.27}, \sqrt{k},]$ for k sufficiently large. So $d = (1 - o(1))\sqrt{k}$ and the constructed code $C^{\alpha^*}_{\mathrm{WE}}$ has distance asymptotically $\Delta(C^{\alpha^*}_{\mathrm{WE}}) \geq (1 - o(1))\sqrt{k}$ as $k \to \infty$.

It is also worthwhile to note that when $a + s \equiv j \pmod{k'}$ holds for more than one pair of (a, s), c_j may still be non-zero as it is a sum of multiple non-zero elements. In fact, for α with large weight, it is common that αy has substantial weight yet few choices of j satisfy the uniqueness property. One possible approach to improving the construction of A involves analyzing scenarios where $j \equiv a + s \pmod{k'}$ for multiple pairs of (a, s).

3.2 Proof of Theorem 7

To analyze the minimum distance of C^{α}_{WE}, it is helpful to define the ring $R = \mathbb{F}_q[x]/(x^{k'} - 1)$, which consists of polynomials of degree less than $k' = k + 1$. We can identify $\mathbb{F}_{q^k} \cong R/(p)$ by the map sending $f \in R$ to $(f \mod p) \in \mathbb{F}_{q^k}$. In addition, we can consider $\mathbb{F}_{q^k} \subseteq R$ and extend φ to the $\mathbb{F}_q$-linear map $\widetilde{\varphi} : R \to \mathbb{F}_q^{k'}$ mapping polynomials of degree less than k' to its coefficient vector. Define $\widetilde{\mathrm{wt}}(f)$ to be the Hamming weight of $\widetilde{\varphi}(f)$ for any $f \in R$. The following lemma gives the relationship between $\widetilde{\mathrm{wt}}(f)$ and $\mathrm{wt}(f \mod p)$.

▶ **Lemma 9.** *For any $f \in R$, $\mathrm{wt}(f \mod p) \geq \min\{\widetilde{\mathrm{wt}}(f), k - \widetilde{\mathrm{wt}}(f)\}$.*

Proof. For any $f \in R$, let us write $f = \sum_{i=0}^{k} b_i x^i$. Then

$$f \bmod p = f - b_k p = \sum_{i=0}^{k-1} (b_i - b_k) x^i.$$

If $b_k = 0$, then $\mathrm{wt}(f \bmod p) = \widetilde{\mathrm{wt}}(f)$; if $b_k \neq 0$, then

$$\mathrm{wt}(f \bmod p) = |\{i : b_i \neq b_k\}| \geq k - \widetilde{\mathrm{wt}}(f).$$

So $\mathrm{wt}(f \bmod p) \geq \min\{\widetilde{\mathrm{wt}}(f), k - \widetilde{\mathrm{wt}}(f)\}$. ◀

▶ **Lemma 10.** *Given $\alpha \in \mathbb{F}_{q^k} \subseteq R$, suppose that for every $y \in R$ with $\widetilde{\mathrm{wt}}(y) \leq c(k)$ the condition*

$$c(k) - \widetilde{\mathrm{wt}}(y) \leq \widetilde{\mathrm{wt}}(\alpha y) \leq k - \Big(c(k) - \widetilde{\mathrm{wt}}(y)\Big)$$

holds, where the product αy is taken in R. Then $\Delta(C_{WE}^{\alpha}) \geq c(k)$, where $\Delta(C_{WE}^{\alpha})$ denotes the distance of the code C_{WE}^{α}.

Proof. Take any non-zero $y \in \mathbb{F}_{q^k} \subseteq R$. Its corresponding codeword $C_{\mathrm{WE}}^{\alpha}(y) = (\varphi(y), \varphi(\alpha y \bmod p))$ has Hamming weight $\mathrm{wt}(y) + \mathrm{wt}(\alpha y \bmod p)$. By Lemma 9, the above condition implies

$$\mathrm{wt}(y) + \mathrm{wt}(\alpha y \bmod p) \geq \widetilde{\mathrm{wt}}(y) + \min\{\widetilde{\mathrm{wt}}(\alpha y), k - \widetilde{\mathrm{wt}}(\alpha y)\} \geq c(k).$$

Since C_{WE}^{α} is a linear code, $\Delta(C_{\mathrm{WE}}^{\alpha}) \geq c(k)$. ◀

We can now prove Theorem 7 by showing that α^* satisfies the condition of Lemma 10 with $c(k) = d$.

Proof of Theorem 7. Note that α is an element of $\mathbb{F}_{q^k}$ since it has degree at most $k-2$ by construction. Let us check that the condition of Lemma 10 indeed holds for $\alpha^* = \sum_{a \in A} x^a$ and $c(k) = d$. For any $y \in R$ with $\widetilde{\mathrm{wt}}(y) = w$, we can write $y = \sum_{i=1}^{w} b_{s_i} x^{s_i}$ where $b_{s_i} \neq 0$ for all $1 \leq i \leq w$. We will denote $S = \{s_1, \ldots, s_w\} \subseteq [k']$ the set non-zero coefficient indices of y, where we define $[k'] = \{0, \ldots, k'-1 = k\}$. The coefficients of the product $\alpha^* y = \sum_{j=0}^{k} c_j x^j$ are given by

$$c_j = \sum_{\substack{a \in A \\ s \in S}} \mathbf{1}\{a + s \equiv j \pmod{k'}\} b_s.$$

This motivates us to define the shifted set

$$(j - A)_k = \{(j - a_1) \bmod k', \ldots, (j - a_d) \bmod k'\}$$

which gives us

$$c_j = \sum_{s \in (j-A)_k \cap S} b_s.$$

We will denote by

$$J_m = \{j \in [k'] : |(j - A)_k \cap S| = m\}$$

the indices j such that $|(j-A)_k \cap S|$ has size m. It is evident that if $(j-A)_k \cap S = \emptyset$ then $c_j = 0$, and if $|(j-A)_k \cap S| = 1$ then $c_j \neq 0$. So

$$|J_1| \leq \widetilde{\mathrm{wt}}(\alpha^* y) \leq k - |J_0|.$$

Looking at the conditions of Lemma 10, it would be sufficient to take $c(k)$ such that

$$c(k) - w \leq \min\{|J_0|, |J_1|\}$$

for all w. We make the following claims on lower bounds of $|J_0|$ and $|J_1|$ which will be proven in Section 3.3.

▷ Claim 11. J_1 has size at least $wd - w(w-1)$.

▷ Claim 12. J_0 has size at least $k - wd$.

Assuming the claims, it suffices to take $c(k)$ such that

$$\begin{aligned} c(k) &\leq \min_{1 \leq w \leq c(k)} wd - w(w-1) + w = \min\{d+1, (d-c(k)+2)c(k)\}, \\ c(k) &\leq \min_{1 \leq w \leq c(k)} k - wd + w = k - (d-1)c(k). \end{aligned}$$

Solving the two inequalities, it suffices to take $c(k) \leq d$. So $\alpha^* = \sum_{a \in A} x^a$, $c(k) = d$ satisfy the condition of Lemma 10. ◀

3.3 Proofs of Claim 11 and Claim 12

To show a lower bound on $|J_1|$, it is desired that the sets $J_2, .., J_w$ are small. The following lemma gives an upper bound on the sizes of $J_2, .., J_w$:

▶ **Lemma 13.** *The sets $J_2, \ldots, J_w$ defined in the proof of Theorem 7 satisfy:*

$$\sum_{m=2}^{w} (m-1)|J_m| \leq \sum_{m=2}^{w} \binom{m}{2} |J_m| \leq 2\binom{w}{2} = w(w-1).$$

Proof. It is evident that the first inequality holds. Let us now show that for any two distinct $s, s' \in S$, we have $\{s, s'\} \subseteq (j-A)_k$ for at most two choices of j. Take any distinct pair $s, s' \in S$. Without loss of generality, assume $s < s'$. Suppose $\{s, s'\} \subseteq (j-A)_k$, then we can write $s \equiv j - a_l \pmod{k'}$ and $s' \equiv j - a_{l'} \pmod{k'}$ for some $l, l' \in [d]$. Then $a_l - a_{l'} \equiv s' - s \pmod{k'}$. So

$$a_l - a_{l'} = \begin{cases} s' - s & \text{if } l > l', \\ s' - s - k' & \text{if } l < l'. \end{cases}$$

In both cases, by definition of Sidon sets, l, l' are uniquely determined. Thus $j \equiv a_l + s$ is also uniquely determined.

For any $j_m \in J_m$, $|(j_m - A) \cap S| = m$ so there are $\binom{m}{2}$ distinct pairs $\{s, s'\} \subseteq (j_m - A)_k$. So the total count of such distinct pairs for all $j \in \cup_{m=2}^{w} J_m$ is $\sum_{m=2}^{w} \binom{m}{2} |J_m|$, which must not exceed $2\binom{w}{2}$ since each pair can only occur in $(j-A)_k$ for two choices of j. This gives

$$\sum_{m=2}^{w} \binom{m}{2} |J_m| \leq 2\binom{w}{2} = w(w-1)$$

which completes the proof. ◀

We can now obtain a lower bound on the size of J_1.

▷ Claim. J_1 has size at least $wd - w(w-1)$.

Proof. For each $s \in S$, there are $|A| = d$ number of j such that $s \in (j - A)_k$. When $w = 1$, the lemma holds as $|J_1| = d$. When $w \geq 2$, $|J_1|$ equals $|S|d = wd$ minus the times we overcount:

$$|J_1| = wd - \sum_{m=2}^{w} (m-1)|J_m|.$$

The proof is complete via the bound of Lemma 13:

$$|J_1| = wd - \sum_{m=2}^{w} (m-1)|J_m| \geq wd - w(w-1). \qquad \lhd$$

▷ Claim. J_0 has size at least $k - wd$.

Proof. For any $s \in S$, there are $|A| = d$ values of $j \in [k']$ such that $s \in (j - A)_k$. So there are at most $|S|d = wd$ indices j such that $|(j - A)_k \cap S| > 0$. ◁

4 Higher Rate Construction

In the last section, we gave an explicit construction of rate 1/2 Wozencraft ensemble code $C_{\mathrm{WE}}^{\alpha^*}$ achieving minimum distance $\Omega(\sqrt{k})$. In this section, we will show that appropriate puncturing of $C_{\mathrm{WE}}^{\alpha^*}$ will give us codes of rates $r \in (1/2, 1)$ and minimum distance at least $\Omega\left(\left(1 - \sqrt{2 - \frac{1}{r}}\right)\sqrt{k}\right)$.

▶ **Definition 14.** *Let $C_{WE}^{\alpha,r}$ be the rate r punctured code given by removing the last $(2 - \frac{1}{r})k$ message bits of each codeword in C_{WE}^{α}.*

Let φ_r denote the map sending any polynomial f to the first $(\frac{1}{r} - 1)k$ least significant coefficients. Let $\mathrm{wt}_r(f)$ denote the Hamming weight of $\varphi_r(f)$.

▶ **Lemma 15.** *For any $f \in R$, $\mathrm{wt}_r(f \bmod p) \geq \min\{\mathrm{wt}_r(f), (\frac{1}{r} - 1)k - \mathrm{wt}_r(f)\}$.*

Proof. Writing $f = \sum_{i=0}^{k} b_i x^i$, we have

$$f \bmod p = f - b_k p = \sum_{i=0}^{k-1} (b_i - b_k) x^i.$$

If $b_k = 0$, then $\mathrm{wt}_r(f \bmod p) = \mathrm{wt}_r(f)$; if $b_k \neq 0$, then

$$\mathrm{wt}_r(f \bmod p) = \left|\left\{i \in [(\frac{1}{r} - 1)k] \mid b_i \neq b_k\right\}\right| \geq (\frac{1}{r} - 1)k - \mathrm{wt}_r(f).$$

So $\mathrm{wt}_r(f \bmod p) \geq \min\{\mathrm{wt}_r(f), (\frac{1}{r} - 1)k - \mathrm{wt}_r(f)\}$. ◀

▶ **Lemma 16.** *Let $\alpha \in \mathbb{F}_{q^k} \subseteq R$. Suppose for every $y \in R$ with $\widetilde{\mathrm{wt}}(y) \leq c(k)$ the condition*

$$c(k) - \widetilde{\mathrm{wt}}(y) \leq \mathrm{wt}_r(\alpha y) \leq (\frac{1}{r} - 1)k - \left(c(k) - \widetilde{\mathrm{wt}}(y)\right)$$

holds, where the product αy is taken in R. Then $\Delta(C_{WE}^{\alpha,r}) \geq c(k)$.

Proof. Take any non-zero $y \in \mathbb{F}_{q^k} \subseteq R$. Its corresponding codeword $C_{\mathrm{WE}}^{\alpha,r}(y) = (\varphi(y), \varphi_r(\alpha y \bmod p))$ has hamming weight $\mathrm{wt}(y) + \mathrm{wt}_r(\alpha y \bmod p)$. By Lemma 15, the above condition implies

$$\mathrm{wt}(y) + \mathrm{wt}(\alpha y \bmod p) \geq \widetilde{\mathrm{wt}}(y) + \min\{\mathrm{wt}_r(\alpha y), (\frac{1}{r} - 1)k - \mathrm{wt}_r(\alpha y)\} \geq c(k).$$

Since $C_{\mathrm{WE}}^{\alpha,r}$ is a linear code, $\Delta(C_{\mathrm{WE}}^{\alpha,r}) \geq c(k)$. ◀

The following theorem establishes a lower bound on the minimum distance of the punctured code $C_{\mathrm{WE}}^{\alpha^*,r}$ with α^* constructed using Sidon sets as outlined in Theorem 7.

▶ **Theorem 17.** *For any rate $r > 1/2$, using the construction of α^* by Theorem 7, the condition of Lemma 16 is satisfied with $c(k) = \Omega\left(\left(1 - \sqrt{2 - \frac{1}{r}}\right)\sqrt{k}\right)$. Thus the punctured code $C_{WE}^{\alpha^*,r}$ has minimum distance at least $\Omega\left(\left(1 - \sqrt{2 - \frac{1}{r}}\right)\sqrt{k}\right)$.*

The proof is omitted[2] since it follows a similar outline as the proof of Theorem 7. We present two claims similar to Claim 11 and Claim 12 that substantiate this argument. For any $y = \sum_{s \in S} b_s x^s$ with weight $|S| = w$, denote $J_m^r = \{j \in [(\frac{1}{r} - 1)k] : |(j - A)_k \cap S| = m\}$ the non-punctured indices j such that $|(j - A)_k \cap S|$ has size m, where A is the Sidon set in the construction of α^*. Then analogous to Claim 11 and Claim 12, we can lower bound the size of J_1^r and J_2^r:

▷ Claim 18. J_1^r has size at least $w\left(\left(1 - \sqrt{(2 - \frac{1}{r})} - o(1)\right)\sqrt{k} - (2 - \frac{1}{r})^{\frac{1}{4}} k^{\frac{1}{4}} - 1\right) - w^2$.

▷ Claim 19. J_0^r has size at least $(\frac{1}{r} - 1)k - w\left(\sqrt{(\frac{1}{r} - 1)k} + (\frac{1}{r} - 1)^{\frac{1}{4}} k^{\frac{1}{4}} + 1\right)$.

To justify the claims, we will require the following theorem by Lindström [11] which bounds the order of any Sidon set via its length.

▶ **Theorem 20** (Lindström [11]). *For any Sidon set with length m, the order is at most $\sqrt{m} + m^{1/4} + 1$.*

The key observation is that for any $s \in S$, the shifted set $(s + A)_k$ is a Sidon set. Moreover, it is partitioned into two parts after puncturing: the remaining part $(s + A)_k \cap [(\frac{1}{r} - 1)k]$ and the removed part $(s + A)_k \cap ([k'] \setminus [(\frac{1}{r} - 1)k])$, where each part is itself a Sidon set. Applying Theorem 20 to the two parts and recalling that $|A| = (1 - o(1))\sqrt{k}$ by Remark 8, we can then obtain the results described in Claim 18 and Claim 19.

5 Open Questions

It is well known that Wozencraft ensemble codes C_{WE}^{α} satisfy the Gilbert-Varshamov bound for most α. Concretely,

$$\lim_{k \to \infty} \Pr_{\alpha \in S} [\Delta(C_{\mathrm{WE}}^{\alpha}) \geq d(k)] = 1,$$

where $S = \mathbb{F}_{q^k}^*$, $d(k) = \left(h_q^{-1}(\frac{1}{2}) - \epsilon\right) \cdot 2k$ with h_q the q-ary entropy function and $\epsilon > 0$ chosen arbitrarily. In this paper, we have proposed a construction of $S = \{\alpha^*\}$ such that the equation above holds with $d(k) = \Omega(\sqrt{k})$. It is of natural interest to reduce the size of the

[2] The proof appears in detail in the full version of the paper posted at https://arxiv.org/abs/2305.02484.

ensemble and find $S, d(k)$ satisfying the equation above with $|S|$ small and $d(k)$ large. For example, can one construct S such that the equation above is satisfied via $|S| = O(2^{o(k)})$ and $d(k) = \Omega(k)$, or $|S| = \text{poly}(k)$ and $d(k) = \Omega(k^c)$ with $c > \frac{1}{2}$? In addition, are there any barriers to such constructions, as in, would such constructions imply progress on some other explicit construction challenge?

In 1973, Weldon [20] proposed an ensemble of codes, a code that generalizes Wozencraft ensemble codes. The Weldon ensemble was used by him, and later Shen [15], to construct explicit concatenated codes achieving the Zyablov bound for rates less than 0.3, thus improving upon Justesen codes [10] for low rates. Weldon codes[3] of rate $1/(t+1)$ are indexed by $\alpha_1, \alpha_2, \ldots, \alpha_t \in \mathbb{F}_{q^k}$ and defined as

$$C_{\text{WE}}^{\alpha_1,\ldots,\alpha_t} = (\varphi(x), \varphi(\alpha_1 x), \ldots, \varphi(\alpha_t x)).$$

For some fixed $t > 2$, can one find explicit α_i's where the distance of $C_{\text{WE}}^{\alpha_1,\ldots,\alpha_t}$ is asymptotically larger than $\sqrt{k}$?

References

1 Wallace C Babcock. Intermodulation interference in radio systems frequency of occurrence and control by channel selection. *The Bell System Technical Journal*, 32(1):63–73, 1953.

2 Roger C Baker, Glyn Harman, and János Pintz. The difference between consecutive primes, ii. *Proceedings of the London Mathematical Society*, 83(3):532–562, 2001.

3 R. C. Bose. An affine analogue of singer's theorem. *The Journal of the Indian Mathematical Society*, 6(0), 1942. URL: https://www.i-scholar.in/index.php/JIMSIMS/article/view/151305.

4 Raj Chandra Bose and Sarvadaman Chowla. Theorems in the additive theory of numbers. Technical report, North Carolina State University. Dept. of Statistics, 1960.

5 Pafnutij Lvovič Čebyšev. Mémoire sur les nombres premiers, 1850.

6 Apostolos Dimitromanolakis. Analysis of the golomb ruler and the sidon set problems and determination of large near-optimal golomb rulers. *Diploma thesis, Department of Electronic and Computer Engineering, Technical University of Crete*, 2002.

7 Paul Erdos and Pál Turán. On a problem of sidon in additive number theory, and on some related problems. *J. London Math. Soc*, 16(4):212–215, 1941.

8 Philippe Gaborit and Gilles Zemor. Asymptotic improvement of the gilbert–varshamov bound for linear codes. *IEEE Transactions on Information Theory*, 54(9):3865–3872, 2008.

9 Tao Jiang and Alexander Vardy. Asymptotic improvement of the Gilbert-Varshamov bound on the size of binary codes. *IEEE Trans. Inf. Theory*, 50(8):1655–1664, 2004.

10 Jørn Justesen. Class of constructive asymptotically good algebraic codes. *IEEE Transactions on Information Theory*, 18(5):652–656, 1972.

11 Bernt Lindström. On b2-sequences of vectors. *Journal of number Theory*, 4(3):261–265, 1972.

12 Dick Lipton. An error correcting code from the book. https://rjlipton.wpcomstaging.com/2011/08/08/an-error-correcting-code-from-the-book/, August 2011. URL: https://rjlipton.wpcomstaging.com/2011/08/08/an-error-correcting-code-from-the-book/.

13 James L Massey. Threshold decoding, 1963.

14 Aditya Potukuchi and Yihan Zhang. Improved efficiency for covering codes matching the sphere-covering bound. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 102–107. IEEE, 2020.

[3] The ensemble codes which Weldon [20] proposed can have any rate of $n_1/(n_1+n_2)$ with n_1, n_2 positive integers. However, he only used codes of rate $1/(t+1)$ to construct the concatenated code.

15 B-Z Shen. A justesen construction of binary concatenated codes that asymptotically meet the zyablov bound for low rate. *IEEE Transactions on Information Theory*, 39(1):239–242, 1993.

16 Amir Shpilka. Capacity achieving two-write wom codes. In *Latin American Symposium on Theoretical Informatics*, pages 631–642. Springer, 2012.

17 Amir Shpilka. New constructions of wom codes using the wozencraft ensemble. *IEEE Transactions on Information Theory*, 59(7):4520–4529, 2013.

18 Simon Sidon. Ein satz über trigonometrische polynome und seine anwendung in der theorie der fourier-reihen. *Mathematische Annalen*, 106(1):536–539, 1932.

19 Amnon Ta-Shma. Explicit, almost optimal, epsilon-balanced codes. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 238–251, 2017.

20 E.J. Weldon. Justesen's construction–the low-rate case (corresp.). *IEEE Transactions on Information Theory*, 19(5):711–713, 1973. doi:10.1109/TIT.1973.1055068.

NP-Hardness of Almost Coloring Almost 3-Colorable Graphs

Yahli Hecht
School of Computer Science, Tel Aviv University, Israel

Dor Minzer
Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA, USA

Muli Safra
School of Computer Science, Tel Aviv University, Israel

Abstract

A graph $G = (V, E)$ is said to be (k, δ) almost colorable if there is a subset of vertices $V' \subseteq V$ of size at least $(1 - \delta)|V|$ such that the induced subgraph of G on V' is k-colorable. We prove that for all k, there exists $\delta > 0$ such for all $\varepsilon > 0$, given a graph G it is NP-hard (under randomized reductions) to distinguish between:

1. Yes case: G is $(3, \varepsilon)$ almost colorable.
2. No case: G is not (k, δ) almost colorable.

This improves upon an earlier result of Khot et al. [16], who showed a weaker result wherein in the "yes case" the graph is $(4, \varepsilon)$ almost colorable.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases PCP, Hardness of approximation

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.51

Category RANDOM

Funding *Yahli Hecht*: Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 835152).
Dor Minzer: Supported by an Sloan Research Fellowship, NSF CCF award 2227876 and NSF CAREER award 2239160.
Muli Safra: Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 835152), and by an ISF grant and a BSF grant.

1 Introduction

The graph coloring problem is one of the most basic combinatorial optimization problems studied in theoretical computer science. A graph $G = (V, E)$ is *k-colorable* if there exists a vertex coloring $\mathsf{col}\colon V \to \{1, \ldots, k\}$ such that for each edge $e = (u, v) \in E$, it holds that $\mathsf{col}(u) \neq \mathsf{col}(v)$. The chromatic number of G, denoted by $\chi(G)$, is defined to be the smallest integer k so that G is k-colorable. What is the computational complexity of finding the chromatic number of a graph?

It has long been known that computing the chromatic number of a graph is NP-hard [10]. Using the PCP theorem, one can improve this result and show that even approximating the chromatic number of a graph (within any constant factor) is NP-hard [12]. Thus, the next natural question to ask is how hard is it to find somewhat efficient coloring of a graph, provided that a very efficient one exists. Specifically, given a $k \in \mathbb{N}$, what is the smallest k' such that given a k-colorable graph, one may efficiently color it using k'?

As the case of $k = 2$ is the 2-coloring problem which is known to be in P, the first interesting case to consider is that of $k = 3$. Despite significant effort, the best unconditional result along these lines, due to [2], asserts that such graphs are NP-hard to color using 5 colors. In terms of conditional results, based on Khot's 2-to-2 Games Conjecture (with perfect completeness) [13], Dinur et al. [7] proved that it is NP-hard to C-color 4-colorable graphs for all $C > 0$. This result was recently improved by Guruswami and Sandeep [8], who showed that it is NP-hard to C-color 3-colorable graphs for any $C > 0$.[1] The gap between the ratios known to be hard and those known to be efficiently achievable is huge: on the algorithmic front, the state of the art result (due to [11]) asserts that one may efficiently find an $n^{0.2-\varepsilon}$-coloring of a given 3-colorable graph, where n is the number of vertices of the graph and $\varepsilon > 0$ is an absolute constant.

Due to the lack of progress towards unconditional hardness results for the graph coloring problem, one is motivated to consider relaxations of the problem that are easier to work with that still capture its essence. Most relevant to us is the almost coloring relaxation. Here, we say a graph $G = (V, E)$ is (k, δ) almost colorable if there exists a subset of vertices $V' \subseteq V$ with $|V'| \geqslant (1-\delta)|V|$ such that the induced subgraph of G on V' is k-colorable. That is, that the graph $G' = (V', E')$, where $E' = \{e = (u, v) \mid u, v \in V'\}$, is k-colorable. With regards to this notion, based on the hardness of 2-to-2 Games with imperfect completeness [15, 6, 5, 16], one can show that almost coloring almost 4-colorable graphs is NP-hard with any constant number of colors. More specifically, using ideas from [7], one can show that for all $\varepsilon, \delta > 0$, given a graph $G = (V, E)$, it is NP-hard to distinguish between:

1. Yes case: G is $(4, \varepsilon)$ almost colorable.
2. No case: G does not contain an independent set of fractional size δ.

In particular, it is NP-hard to $(k, 1-\varepsilon)$ almost color a given $(4, \varepsilon)$ almost colorable graph. We remark that the "independent set" conclusion in the no case is in fact stronger, and one often manages to achieve it via the typical style of PCP reductions for coloring problems.

The distinction between 3-coloring and 4-coloring may seem minor at first glance. This distinction hides within it a technical barrier related to the difference between Unique-Games and 2-to-2 Games. Indeed, the only type of problems that seem to facilitate results for 3-coloring are Unique-Games [7] (which inherently lack perfect completeness) and a certain variant of 2-to-2 games called Rich 2-to-2 Games [4, 3]. Both problems are conjectured to be NP-hard, but the proof of this assertion seems out of reach of current techniques. Using standard 2-to-2 Games, which are now known to be NP-hard (with imperfect completeness), there seem to have been fundamental difficulties beyond 4-colorable graphs.

Recently, Guruswami and Sandeep [8] observed that there there are transformations from the algebraic CSPs world [17] (which date back to [9]) that reduce the chromatic number of a graph from 4 to 3. Using these ideas, and combining them with the reduction of [7], Guruswami and Sandeep managed to prove the aforementioned conditional result, asserting that assuming the d-to-d conjecture of Khot with perfect completeness, it is NP-hard to C-color a given 3-colorable graph, for all constants $C > 0$. We remark that interestingly their reduction fails to establish the stronger "independent set" type conclusion in the no case.

The main result of this paper is that while this reduction does not preserve the independent set conclusion, it does preserve the intermediate, strictly weaker conclusion of almost coloring. More precisely, we use these ideas to get an (unconditional) NP-hardness result regarding almost coloring almost 3-colorable graphs with a constant number of colors. Specifically, we prove:

► **Theorem 1.** *For all $k \in \mathbb{N}$, there exists $\delta > 0$ such that for all $\varepsilon > 0$, given a graph $G = (V, E)$ it is NP-hard (under randomized reductions) to distinguish between:*

1. *Yes case: G is $(3, \varepsilon)$ almost colorable.*
2. *No case: G is not (k, δ) almost colorable.*

[1] In fact, for the result of Guruswami et al./ it suffices to assume that the d-to-d Games Conjecture of Khot [13] holds for some constant $d \in \mathbb{N}$.

Proof Idea

While the reduction of Guruswami and Sandeep [8] does not preserve the notion of almost coloring for all graphs, it turns out that it does so for bounded degree graphs. Indeed, this is the main insight behind the proof of Theorem 1. Thus, our proof of Theorem 1 starts with a hard instance of 2-to-2 Games with imperfect completeness and uses the reduction of [7] to establish hardness of almost coloring almost 4-colorable graphs for instances with certain regularity properties. We then use random sparsification to further reduce such instances to instances with bounded maximal degree. Finally, we use line-graph based reductions as in [8] to reduce this problem to the problem of almost coloring almost 3-colorable graphs.

We remark that just like in the result of Guruswami and Sandeep, our result also fails to establish the stronger "independent set" type conclusion in the no case. Indeed, strengthening Theorem 1 in that manner would automatically improve upon the best known hardness of approximation result for Vertex Cover, which is a long standing challenge. Currently, it is known that it is NP-hard to approximate the size of the smallest vertex cover of a graph within factor $\sqrt{2}-\varepsilon$ for all $\varepsilon > 0$, and an "independent-set"-type strengthening of Theorem 1 would improve this factor to $3/2-\varepsilon$, for all $\varepsilon > 0$. While we do not know how to prove such strengthening, we believe it may be doable and discuss such a possibility in Section 4.

2 Preliminaries

In this section, we present a few notions that will be necessary in the proof of Theorem 1.

2.1 Induced Coloring and Constraint Satisfaction Problems

For an integer $k \in \mathbb{N}$, the k-induced coloring of a graph G is the fractional size of the largest induced subgraph of G which is k-colorable. More precisely:

▶ **Definition 2.** *For a graph $G = (V, E)$, the induced-coloring $\mathsf{iCol}_k(G)$ is the fractional size of the largest induced subgraph that is k-colorable. That is, it is the maximum of $|V'| / |V|$ among all induced sub-graphs (V', E') of G that are k-colorable.*

According to this notation, the fact that a graph G is (k, δ) almost colorable is equivalent to $\mathsf{iCol}_k(G) \geqslant 1-\delta$. Next, we define the label cover problem, followed by the definition of a specific type of label cover instances called 2-to-2 Games.

▶ **Definition 3.** *An instance of Label Cover $\Psi = (G, \Sigma, \Phi)$ consists of a graph $G = (V, E)$, a finite alphabet Σ and a collection of constraints $\Phi = \{\Phi_e\}_{e\in E}$ one for each edge of G. The constraint on an edge $e \in E$ specifies tuples $\Phi_e \subseteq \Sigma \times \Sigma$ that are deemed satisfactory.*

Given an instance of label cover Ψ, the goal is to find an assignment $A\colon V \to \Sigma$ satisfying as many of the constraints as possible. Here, we say A satisfies the constraint on an edge $e = (u, v) \in E$ if $(A(u), A(v)) \in \Phi_e$. We denote by $\mathsf{sat}(\Psi)$ the maximum fraction of constraints that are satisfied in Ψ by any assignment A. We will mostly be interested in a specific type of label cover instances, known as 2-to-2 instances, in which the constraints take a special form.

▶ **Definition 4.** *A label cover instance $(G = (V, E), \Phi, 2R)$ is called a 2-to-2 Games instance if for every edge $e = (u, v) \in E$, the constraint Φ_e is of the form $\cup_{i=1}^{R} A_i \times B_i$, where $\{A_i\}_{i=1}^{R}$ and $\{B_i\}_{i=1}^{R}$ are two partitions of Σ into sets of size 2. Equivalently, the constraint Φ_e takes the form*

$$\left\{(i,j) \in \{1,\ldots,2R\} \;\middle|\; (\pi^{-1}(i), \sigma^{-1}(j)) \in \{(2k,2k),(2k,2k+1),(2k+1,2k),(2k+1,2k+1)\}\right\}$$

for some permutations $\pi, \sigma : [2R] \to [2R]$.

Typically, given a 2-to-2 Games instances Ψ, the goal is to find an assignment to the vertices satisfying as many of the constraints as possible. For our purposes, it will be useful for define a variation of this parameter denoted by $\mathsf{iSat}(\Psi)$.

▶ **Definition 5.** *For an integer $m \in \mathbb{N}$ and a 2-to-2 Games instance $\Psi = (G = (V, E), \Sigma, \Phi)$, we define $\mathsf{iSat}(\Psi)$ to be the fractional size of the largest $V' \subseteq V$ for which there is an assignment $A\colon V' \to \Sigma$ satisfying all of the constraints of Ψ inside V'.*

2.2 Fourier Analysis

Our reduction requires a few basic notions from discrete Fourier analysis which we present next. We refer the reader to [19] for a more thorough presentation.

2.2.1 The Fourier Decomposition over Product Spaces

Let $q \in \mathbb{N}$ be an integer, and denote $[q] = \{0, 1, \ldots, q-1\}$. We will consider the product space $L_2([q]^n, \mu)$ where μ is the uniform measure over $[q]^n$ and the inner product between $f, g\colon [q]^n \to \mathbb{R}$ is defined as

$$\langle f, g \rangle = \mathop{\mathbb{E}}_{x\sim\mu}[f(x)g(x)].$$

We may pick an orthonormal basis $\alpha_0, \ldots, \alpha_{q-1}\colon [q] \to \mathbb{R}$ of $L_2([q], \mu)$ so that $\alpha_0 \equiv 1$, and we do so canonically. Given this basis, we may tensorize it to get an orthonormal basis of $L_2([q]^n, \mu)$ as $\{\alpha_{j_1,\ldots,j_n}\}_{j_1,\ldots,j_n\in[q]}$ where $\alpha_{j_1,\ldots,j_n}$ is defined as

$$\alpha_{j_1,\ldots,j_n}(x_1, \ldots, x_n) = \prod_{i=1}^{n} \alpha_{j_i}(x_i).$$

Given the orthonormal basis $\{\alpha_{\vec{j}}\}_{\vec{j}\in[q]^n}$, we may decompose any $f\colon [q]^n \to \mathbb{R}$ as a linear combination of the basis functions, and the coefficients in this linear combination are given by the Fourier coefficients:

▶ **Definition 6.** *For $f\colon [q]^n \to \mathbb{R}$ and $\vec{j} \in [q]^n$, we define $\hat{f}(\vec{j}) \overset{def}{=} \left\langle f, \alpha_{\vec{j}} \right\rangle$.*

Also, given $\vec{j} \in [q]^n$, we define the degree of $\vec{j}$, $\mathsf{deg}(\vec{j})$, to be the number of indices $i = 1, \ldots, n$ such that $j_i \neq 0$. We define the action of a permutation on a function:

▶ **Definition 7.** *For a permutation $\pi\colon [n] \to [n]$ and a function $f\colon [q]^n \to \mathbb{R}$, we define $f^\pi\colon [q]^n \to \mathbb{R}$ by $f^\pi(x_1, \ldots, x_n) \overset{def}{=} f(x_{\pi(1)}, \ldots, x_{\pi(n)}) = f(x^\pi)$.*

2.2.2 Low Degree Influences and a Corollary of the Invariance Principle

Our analysis requires the notion of low degree influences, which is defined in terms of the Fourier coefficients of a function f in the following way:

▶ **Definition 8.** *For a function $f\colon [q]^n \to \mathbb{R}$, a coordinate $i = 1, \ldots, n$ and an integer d, the degree d influence of i on f is defined as*

$$I_i^{\leqslant d}[f] \overset{def}{=} \sum_{\substack{\vec{j}\in[q]^n \\ \vec{j}_i\neq 0, \mathsf{deg}(\vec{j})\leqslant d}} \hat{f}^2(\vec{j}).$$

The following well known fact asserts that the number of coordinates with significant low degree influence is relatively small.

▶ **Fact 9.** *For a function $f\colon [q]^n \to \mathbb{R}$ with $\|f\|_2 \leqslant 1$ and $\tau > 0$, the number of coordinates $i = 1, \dots, n$ with $I_i^{\leqslant d}[f] \geqslant \tau$ is at most $\frac{d}{\tau}$.*

In the reduction from 2-to-2 Games to 4-coloring we will have functions operating on the space $[q^2]^n$, and we will want to sometimes view it as $[q]^{2n}$. To do so, we will use the natural identification between $[q]^2$ and $[q^2]$ and thus define an identification between the spaces $[q^2]^n$ and $[q]^{2n}$ in the following way:

▶ **Definition 10.** *For $x = (x_1, \dots, x_{2n}) \in [q]^{2n}$, we denote $\overline{x} \overset{def}{=} ((x_1, x_2), \dots, (x_{2n-1}, x_{2n})) \in [q^2]^n$.*

Thus, given a function $f\colon [q]^{2n} \to \mathbb{R}$, one may consider the function $\overline{f}\colon [q^2]^n \to \mathbb{R}$ defined by the natural identification above as:

$$\overline{f}((x_1, x_2), \dots, (x_{2n-1}, x_{2n})) = f(x_1, \dots, x_{2n}).$$

The following facts from [7] will be used in the analysis of our reduction. The first claim related low-degree influences of f and of $\overline{f}$:

▷ Claim 11. For a function $f\colon [q]^{2n} \to \mathbb{R}$, a coordinate $1 \leqslant i \leqslant n$ and a degree parameter d, we have that

$$I_i^{\leqslant d}[\overline{f}] \leqslant I_{2i-1}^{\leqslant 2d}[f] + I_{2i}^{\leqslant 2d}[f].$$

Next, we need a corollary of the invariance principle which is also used in [7], and towards this end, we define the parameter $\Gamma_\rho(\mu, \tau)$:

▶ **Definition 12.** *Let Φ be the cumulative distribution function of $\mathcal{N}(0,1)$. We denote*

$$\Gamma_\rho(\mu, \tau) \overset{def}{=} \Pr\left[X \leqslant \Phi^{-1}(\mu) \wedge Y \geqslant \Phi^{-1}(1-\tau)\right],$$

where (X, Y) are ρ-correlated Gaussian random variables. If $\mu = \tau$, we write $\Gamma_\rho(\mu)$, we also omit ρ when clear from context.

Asymptotics for the value $\Gamma_\rho(\mu, \tau)$ exist (and are not too hard to establish), however, we will only require the fact that $\Gamma_\rho(\varepsilon)$ is a positive constant for all $\rho < 1$ and $\varepsilon > 0$. With the parameter $\Gamma_\rho(\mu, \tau)$ in hand, we now state the corollary of the invariance principle necessary for our analysis which can also be found in [7].

▶ **Theorem 13.** *Let $q \in \mathbb{N}$ be an integer and let T be a connected symmetric Markov chain on $[q]$, with eigenvalues $\lambda_0 = 1 \geqslant \ldots \geqslant \lambda_{q-1}$. If $\rho := \max(|\lambda_1|, |\lambda_{q-1}|) < 1$, then for any $\mu, \tau > 0$ there exist $\delta > 0$ and $d \in \mathbb{N}$, for which the following holds. For all functions $f, g\colon [q]^n \to [0,1]$, if $\mathbb{E}[f] \geqslant \mu, \mathbb{E}[g] \geqslant \tau$, and $\langle f, T^{\otimes n} g\rangle \leqslant \frac{1}{2}\Gamma_\rho(\mu, \tau)$, then there exists a coordinate $i = 1, \dots, n$ such that $I_i^{\leqslant d}[f], I_i^{\leqslant d}[g] \geqslant \delta$.*

2.2.3 A Markov Chain on $[4]^2$

We end this section with the following claim due to [7] establishing the existence of a Markov chain on $[4]^2$ with certain properties:

▷ Claim 14. There exists a connected symmetric Markov chain T on $\{0,1,2,3\}^2$ with $\max(|\lambda_1|, |\lambda_{q-1}|) < 1$, such that if $T((x_1, x_2) \leftrightarrow (y_1, y_2)) > 0$ then $\{x_1, x_2\} \cap \{y_1, y_2\} = \emptyset$.

3 Hardness of Almost Coloring Almost 3-Colorable Graphs

3.1 The Starting Point: 2-to-2 Games

The starting point of our reduction is the hardness of 2-to-2 Games with imperfect completeness [15, 6, 5, 16] (see [18, 14] for an exposition). Specifically, we will use the following formulation:

▶ **Theorem 15.** *For all $s, \eta > 0$, given a 2-to-2 Games instance Ψ, it is NP-hard to distinguish between:*

- *YES case:* $\mathsf{iSat}(\Psi) \geqslant 1 - \eta$.
- *NO case:* $\mathsf{sat}(\Psi) \leqslant s$.

Additionally, in both the Yes and No cases, the graph G underlying Ψ satisfies the following regularity property: for all $\varepsilon \leqslant \beta \leqslant 1$, any set $S \subseteq V$ of fractional size β contains at least $\Omega(\beta^2)$ fraction of the edges of G.

3.2 A Reduction from 2-to-2 Games to Almost 4-Coloring

In this section, we apply the reduction of [7] on instances from Theorem 15 to get the following hardness result for almost 4-coloring:

▶ **Theorem 16.** *For all $\varepsilon, \eta > 0$, given an edge weighted graph $G = (V, E, w)$, it is NP-hard to distinguish between:*

- *Yes case:* $\mathsf{iCol}_4(G) \geqslant 1 - \eta$
- *No case: any set $S \subseteq V$ of fractional size at least ε contains edges of total weight of at least $\Omega\left(\varepsilon^2 \Gamma_\rho(\varepsilon/2)\right)$, where $\rho \in (0, 1)$ is an absolute constant.*

Proof. Let $\Psi = (G = (V, E), \Sigma, \Phi)$ be a 2-to-2 Games instances from Theorem 15 with completeness $1 - \eta$ and soundness $s > 0$, and let T be the Markov chain from Claim 14. Without loss of generality, we assume that the alphabet Σ is $[2R]$, where $2R = |\Sigma|$. Below, when we write Γ, we mean $\Gamma = \Gamma_\rho$ where ρ is the absolute constant from Claim 14.

We construct a graph $G = (V', E')$ by replacing each vertex $u \in V$ by a block of vertices $\{u\} \times \{0, 1, 2, 3\}^{2R}$, denoted by $B[u]$. Thus, $V' = \bigcup_{u \in V} B[u]$. As for the edges of H, for an edge $\{u, v\} \in E$ in G with a 2-to-2 constraint described by the permutations $\pi, \sigma \in S_{2R}$, we add an edge between (u, x) and (v, y) if $T^{\otimes R}(\overline{x^\pi} \leftrightarrow \overline{y^\sigma}) > 0$, in which case we assign the edge the weight $\frac{1}{|E|} T^{\otimes R}(\overline{x^\pi} \leftrightarrow \overline{y^\sigma})$.

Completeness. If $\mathsf{iSat}(\Psi) \geqslant 1 - \eta$, then there is $S \subseteq V$ of fractional size $1 - \eta$ and a assignment $c \colon S \to [2R]$, satisfying all constraints within S. For each $u \in S$ we assign the block $B[u]$ by $A(u, x) = x_{c(u)}$, and note that this assignment forms a 4-coloring of $\bigcup_{u \in S} B[u]$. Thus, $\mathsf{iCol}_4(G') \geqslant 1 - \eta$.

Soundness. Assume towards contradiction that there is a set of vertices $S' \subseteq V'$ of fractional size ε that contains less than $c\varepsilon^2\Gamma(\varepsilon/2)$ of the edges (weighted), where $c > 0$ is an absolute constant to be determined. Define

$$S \overset{def}{=} \left\{ v \in V \;\middle|\; |B[v] \cap S'| \geqslant \frac{\varepsilon}{2} |B[v]| \right\}.$$

By an averaging argument we have that $|S| \geqslant \frac{\varepsilon}{2} |V|$, and for each $v \in S$ we define $f_v \colon \{0, 1, 2, 3\}^{2R} \to \{0, 1\}$ by $f_v(x) = 1_{(v,x) \in S}$. Thus, $\mathbb{E}[f_v] = \frac{|B[v] \cap S'|}{|B[v]|} \geqslant \frac{\varepsilon}{2}$ for all $v \in S$.

By the regularity condition on Ψ, the set S contains at least $\alpha\varepsilon^2$ fraction of the edges of Ψ, where $\alpha > 0$ is an absolute constant. Fix $u, v \in S$ between which there is an edge and let π, σ be the permutations defining the constraint on it. Note that the weight of edges between the block of u and the block of v is proportional to $\langle \overline{f_u^{\pi^{-1}}}, T^{\otimes R}\overline{f_v^{\sigma^{-1}}}\rangle$. Thus, as the total weight edges covered by S' is at most $c\varepsilon^2\Gamma(\varepsilon/2)$, it follows that for at least half of the edges (u, v) inside S we have that

$$\langle \overline{f_u^{\pi^{-1}}}, T^{\otimes R}\overline{f_v^{\sigma^{-1}}}\rangle \leqslant \frac{2c\varepsilon^2\Gamma(\varepsilon/2)}{\alpha\varepsilon^2} < \frac{1}{2}\Gamma(\varepsilon/2)$$

where we used the fact that c is sufficiently small. We refer to such edges (u, v) as good.

Fix a good edge (u, v). Applying Theorem 13 we find $d \in \mathbb{N}$ and $\delta > 0$ depending only on ε, such that there is $i \in \{1, \dots, R\}$ for which $I_i^{\leqslant d}[\overline{f_u^{\pi^{-1}}}] \geqslant \delta, I_i^{\leqslant d}[\overline{f_v^{\sigma^{-1}}}] \geqslant \delta$. Using Claim 11 we conclude that

$$I_{\pi(2i-1)}^{\leqslant 2d}[f_u] + I_{\pi(2i)}^{\leqslant 2d}[f_u] = I_{2i-1}^{\leqslant 2d}[f_u^{\pi^{-1}}] + I_{2i}^{\leqslant 2d}[f_u^{\pi^{-1}}] \geqslant I_i^{\leqslant d}[\overline{f_u^{\pi^{-1}}}] \geqslant \delta,$$

$$I_{\sigma(2i-1)}^{\leqslant 2d}[f_v] + I_{\sigma(2i)}^{\leqslant 2d}[f_v] = I_{2i-1}^{\leqslant 2d}[f_v^{\sigma^{-1}}] + I_{2i}^{\leqslant 2d}[f_v^{\sigma^{-1}}] \geqslant I_i^{\leqslant d}[\overline{f_v^{\sigma^{-1}}}] \geqslant \delta.$$

Taking $\mathsf{List}[u] = \left\{ i \in \{1, \dots, R\} \,\middle|\, I_i^{\leqslant 2d}[f_u] \geqslant \frac{\delta}{2} \right\}$, we conclude that for each good edge (u, v) it holds that the lists $\mathsf{List}[u]$ and $\mathsf{List}[v]$ contain a pair of assignments satisfying the constraint on (u, v). Also, by Fact 9 the size of each one of these lists is at most $\frac{4d}{\delta}$. Thus, if we choose for each $u \in S$ a label from $\mathsf{List}[u]$ uniformly at random, then we get an assignment that satisfies at least $\frac{\delta}{4d}$ of the good edges in expectation. Therefore, in expectation, it satisfies at least $\Omega(\varepsilon^2\delta/d)$ of the constraints of Ψ. In particular, there exists an assignment to Ψ satisfying at least $\Omega(\varepsilon^2\delta/d)$ of the constraints in it, and this is a contradiction provided that the soundness parameter s is sufficiently small. ◀

3.3 Sparsification: Hardness of Almost 4-coloring on Bounded Degree Graphs

In this section, we start with instances produced from Theorem 16, and reduce them to unweighted graph instances in which the maximal degree is bounded. To do so, we apply the random sparsification technique from [1]. More precisely, we show the following result:

▶ **Theorem 17.** *For all $\varepsilon, \eta > 0$, given a graph $G = (V, E)$ it is NP-hard under randomized reductions to distinguish between:*

- *Yes case:* $\mathsf{iCol}_4(G) \geqslant 1 - \eta$
- *No case: G has no independent set of size $\varepsilon |V|$.*

Moreover, the maximal degree of a vertex in G is at most $O\left(\frac{1}{\varepsilon^2\Gamma(\varepsilon/2))}\right)$.

Proof. Let $(G = (V, E), w)$ be a weighted graph as in Theorem 16, and construct a graph $G' = (V', E')$ in the following way:

Step 1: reduce the average degree and remove weight. Let $d^{-1} = C^{-1} \cdot \varepsilon^2\Gamma(\varepsilon/2)$ for a large absolute constant $C > 0$ to be determined. Independently sample dn edges from G (allowing repetitions), with probabilities proportional to the weights and include them, unweighted, in the graph G'.

Completeness. It is clear that $\mathsf{iCol}_4(G') \geqslant \mathsf{iCol}_4(G)$ as G' is a subgraph of G.

Soundness. Let $S \subseteq V$ be a set of fractional size ε, and recall that it contained at least $c\varepsilon^2\Gamma(\varepsilon/2)$ of the total weight, for some $c > 0$. Thus, the probability that S is an independent set in G' is at most

$$\left(1 - c\varepsilon^2\Gamma(\varepsilon/2)\right)^{dn} \leqslant e^{-c\varepsilon^2\Gamma(\varepsilon/2)dn} \leqslant e^{-ncC} \leqslant 2^{-2n}$$

for sufficiently large $C > 0$. As there are at most 2^n distinct such sets S, it follows from the union bound that the probability at least one of them is an independent set is at most $2^n \cdot 2^{-2n} \leqslant 2^{-n}$.

Step 2: Average to maximal degree. After step 1 the average degree is at most $2d$. We remove from G' vertices with degree higher than $4d$. By Markov's inequality, at most half of the vertices are removed. Consequently, if an almost coloring for $1 - \eta$ of the vertices existed, the same coloring colors at least $1 - 2\eta$ of the vertices. Similarly, if the largest independent set had fractional size at most ε, after removing vertices, no independent set has a fractional size larger than 2ε. ◀

3.4 Decreasing the Chromatic number

The final step in our proof is to apply a transformation on instances from Theorem 17 that reduces the almost coloring number in the "yes case" to 3, while keeping the soundness in the form of almost coloring. Toward this end, we use the *directed line graph*. The line graph was shown by [9, 20] to reduce the chromatic number (to be roughly logarithmic in the chromatic number of the original graph). This fact was used in [17, 8] to get hardness results for 3-colorable graphs. We also use this property of the line graph. For our purposes though, we show that a stronger guarantee can be made so long as the original graph has a bounded degree.

▶ **Definition 18.** *For a digraph $G = (V, E)$, the directed line graph of G, denoted by $\delta(G) = (V', E')$, is defined in the following way:*

- *The vertices are $V' = E(G)$, the (directed) edges of the original graph G.*
- *The edges are any pair of edges of G that shares a vertex, namely:*

$$E' = \{((u, v), (v, w)) \mid (u, v), (v, w) \in E(G)\}.$$

Applying the line-graph twice is denoted by $\delta^2(G) \overset{def}{=} \delta(\delta(G))$.

We use the following properties of the line graph. The first two lemmas address the completeness of the reduction.

▶ **Lemma 19.** *If G is 4-colorable, then $\chi(\delta^2(G)) \leqslant 3$.*

Proof. Given a 4-coloring $c : V \to \{0, 1, 2, 3\}$ of G, we can color $\delta^2(G)$ in the following way: for every vertex $((i, j), (j, k))$, if $c(j) \in \{0, 1, 2\}$ we will assign the color $c(j)$. Otherwise, we will assign any color in $\{0, 1, 2\} \setminus \{c(i), c(k)\}$. ◀

Next, we show that if G has bounded degree and a large induced subgraph which is 4-colorable, then $\delta^2(G)$ is almost 3-colorable.

▶ **Lemma 20.** *For a directed graph $G = (V, E)$ with bounded degree d, if $\mathsf{iCol}_4(G) \geqslant 1 - \eta$, then $\mathsf{iCol}_3(\delta^2(G)) \geqslant 1 - 6d^2\eta$.*

Proof. Since $\mathsf{iCol}_4(G) \geqslant 1-\eta$, there exists an induced subgraph H of G, where $|V(H)| \geqslant (1-\eta)\,|V(G)|$ and $\chi(H) \leqslant 4$. By Lemma 19 we get that $\delta^2(H)$ is 3-colorable. Therefore, it suffices to bound the fractional number of vertices of $\delta^2(G)$ that are not vertices of $\delta^2(H)$.

The set $V(\delta^2(G)) \setminus V(\delta^2(H))$ contains vertices of the form $((u,v),(v,w))$ when $\{u,v,w\} \not\subseteq V(H)$. The fractional size of $V(G)\setminus V(H)$ in $V(G)$ is at most η, and for each vertex $v \in V(G)$, there exists at most $3d^2$ vertices in $V(\delta^2(G))$ that include v. Therefore, at most $|V(G)|\,3d^2\eta$ of the vertices are not inside $V(\delta^2(H))$, which implies that $\left|V(\delta^2 H)\right| \geqslant (1-6d^2)\left|V(\delta^2 G)\right|$ and so $\mathsf{iCol}_3(\delta^2(G)) \geqslant 1-6d^2\eta$. ◀

The next lemma addresses the soundness of the reduction:

▶ **Lemma 21.** *Let $G=(V,E)$ be a directed graph with a maximal degree of at most d. Then*

$$\mathsf{iCol}_{\lfloor \log(Q)\rfloor}(\delta(G)) \leqslant \frac{d-1}{d} + \frac{\mathsf{iCol}_Q(G)}{d}.$$

Proof. For $S \subseteq V(\delta G) = E(G)$ and a partial $t = \lfloor \log(Q) \rfloor$ coloring $c\colon S \to [t]$, we can construct the following partial coloring of G. Let $H \subseteq V(G)$ be the set of vertices u such that $(u,v) \in S$ for all neighbours v of u in G. Define $f\colon H \to P([t])$ by

$$f(u) = \{c((u,v)) \mid (u,v) \in E(G)\}.$$

The function above is indeed a valid partial coloring, since for each $(u,v) \in E(G)$, such that u,v are both colored, $c((u,v)) \in f(u)$ and $c((u,v)) \notin f(v)$. This partial coloring has at most Q colors, since $|\mathcal{P}([t])| = 2^t \leqslant 2^{\log(Q)} = Q$.

Finally, by definition, we must have that $|H| \leqslant \mathsf{iCol}_Q(G)\,|V|$. On the other hand, if the size of S is denoted by $1-\eta$, then we have that H contains at least $(1-d\eta)\,|V|$ vertices. It follows that $1-d\eta \leqslant \mathsf{iCol}_Q(G)$, and so $\eta \geqslant \frac{1-\mathsf{iCol}_Q(G)}{d}$. ◀

We are now ready to complete the proof of Theorem 1, restated below:

▶ **Theorem 22.** *For all $k \in \mathbb{N}$ there is $\delta = \delta(k) < 1$ such that the following holds for all $\eta > 0$. Given an undirected graph $G=(V,E)$, it is NP-hard (under randomized reductions) to distinguish between:*

- *Yes case:* $\mathsf{iCol}_3(G) \geqslant 1-\eta$
- *No case:* $\mathsf{iCol}_k(G) \leqslant 1-\delta$

Moreover, the problem is NP-hard on instances with bounded degrees (depending only on k).

Proof. We start with a graph G from Theorem 17 with sufficiently small parameters, construct $\delta^2(G)$, and replace the directed edges with undirected ones. Let d be the bound on the maximal degree of G.

Completeness. If $\mathsf{iCol}_4(G) \geqslant 1-\eta$, then by Lemma 20 $\mathsf{iCol}_3(\delta^2(G)) \geqslant 1-6d^2\eta$, which is at least $1-\sqrt{\eta}$ provided η is small enough (recall that d only depends on the soundness parameter ε in Theorem 17).

Soundness. Suppose that the largest independent set in G has fractional size at most $1/Q$. By Lemma 21 we get that $\mathsf{iCol}_{\lfloor \log(Q/2)\rfloor}(\delta G) \leqslant \frac{2d-1}{2d} + \frac{1}{4d}$. Note that the maximum degree in $\delta(G)$ is at most $4d$, so applying Lemma 21 on $\delta(G)$ we get that

$$\mathsf{iCol}_{\lfloor \log(\lfloor \log(Q/2)\rfloor)\rfloor}(\delta^2(G)) \leqslant \frac{4d-1}{4d} + \frac{4d-1}{16d^2} = 1 - \frac{1}{16d^2}.$$ ◀

4 Discussion

As discussed in the introduction, we believe that it may be possible to improve Theorem 1 so that in the "no case", the graph does not contain an independent set of fractional size ε. More precisely, we believe that the following conjecture should hold:

▶ **Conjecture 23.** *For all $\varepsilon, \eta > 0$, given an undirected graph $G = (V, E)$, it is NP-hard to distinguish between:*

- *Yes case:* $\mathsf{iCol}_3(G) \geqslant 1 - \eta$
- *No case: G does not contain an independent size of fractional size ε.*

If true, Conjecture 23 would be a significant result in our opinion. To start with, it immediately implies an improvement on the best known hardness of approximation result for Vertex Cover (to factor $3/2 - \varepsilon$ for all $\varepsilon > 0$, where the state of the art NP-hardness result stands at $\sqrt{2} - \varepsilon$). Below, we show that a strong enough form of Theorem 1 in which δ is a sufficiently large function of k (more specifically, $\delta \geqslant 2^{-o(k)}$) implies that Conjecture 23 holds. We remark that unfortunately, in our proof $\delta = 2^{2^{-k^{O(1)}}}$, which is quantitatively not strong enough to conclude Conjecture 23.

▶ **Definition 24.** *Given a graph $G = (V, E)$ and $t \in \mathbb{N}$, define $G[t] = (V[t], E[t])$ as:*

- *The vertices are $(a_1, \ldots, a_t) \in V^t$.*
- *There is an edge between $A = (a_1, \ldots, a_t)$ and $B = (b_1, \ldots, b_t)$, if we can move from A to B by replacing one vertex with one of his neighbors.*

$$(b_1, \ldots, b_t) = (a_1, \ldots, a_{i-1}, u, a_{i+1} \ldots, a_t).$$

The following claim shows that the transformation from G to $G[t]$ has an amplification-type effect:

▷ Claim 25. If $\mathsf{iCol}_Q(G) \leqslant 1 - (1 - c)^Q$, then for all $\varepsilon > 0$, there exists $t = t(Q, \varepsilon)$ such that $G[t]$ has no independent set of size $c + \varepsilon$.

Proof. Fix t. For an independent set $I \subseteq V[t]$ we define

$$\mathsf{est}_I(v) := \frac{|\{(a_1, \ldots, a_{t-1}, v) \in V[t]\} \cap I|}{|\{(a_1, \ldots, a_{t-1}, v) \in V[t]\}|}$$

For any independent set I and $v \in V$, it holds that $\mathsf{est}_I(v) \leqslant \mathrm{IS}(G[t-1])$, where $\mathrm{IS}(G)$ denotes the fractional size of the largest IS in G. This holds since $\{(a_1, \ldots, a_{t-1}) \mid (a_1, \ldots, a_{t-1}, v) \in I\}$ is an IS in $G[t-1]$. It also follows that $\mathrm{IS}(G[1]) \geqslant \mathrm{IS}(G[2]) \geqslant \mathrm{IS}(G[3]) \geqslant \ldots$ (as $\mathbb{E}_v\left[\mathsf{est}(v)\right] = |I|/|V[t]|$).

Let $\rho > 0$ be a parameter. Since $\mathrm{IS}(G[1]) \geqslant \mathrm{IS}(G[2]) \geqslant \mathrm{IS}(G[3]) \geqslant \ldots$, there exists $t \leqslant \lceil 1/\rho^2 \rceil + 1$, for which $\mathrm{IS}(G[t-1]) - \mathrm{IS}(G[t]) \leqslant \rho^2$. Fixing a maximal independent $I \subseteq V[t]$, it holds that $\mathsf{est}_I(v) \leqslant \mathrm{IS}(G[t]) + \rho^2$ for all $v \in V$. By Markov's inequality, for at most $\rho|V|$ vertices of G, $\mathsf{est}_I(v) < \mathrm{IS}(G[t]) - \rho$. We say a vertex $v \in V$ is good if $\mathsf{est}_I(v) \geqslant \mathrm{IS}(G[t]) - \rho$.

Sampling $a_1, \ldots, a_{t-1} \in V$ uniformly and taking $\{u \mid (a_1, \ldots, a_{t-1}, u) \in I\}$ Q times, we get Q independent sets, which is a partial Q-coloring. For a good vertex v, the probability that it is included in one of the Q samples is at least

$$1 - (1 - \mathsf{est}_I(v))^Q \geqslant 1 - (1 - \mathrm{IS}(G[t]) + \rho)^Q,$$

so the expected fractional size of such Q coloring is at least:

$$(1-\rho)\Big(1-(1-\mathrm{IS}(G[t])+\rho)^{Q}\Big)\,|V|\,.$$

As this must be smaller than $\mathsf{iCol}_Q(G) \leqslant 1-(1-c)^Q$, it follows that $\mathrm{IS}(G[t]) \leqslant c+O(\rho)$, and the proof is concluded for small enough ρ. ◁

▶ **Corollary 26.** *If $\delta(Q) > \frac{1}{2^{o(Q)}}$ in Theorem 22, then Conjecture 23 holds.*

Proof. Start with a graph $G=(V,E)$ from Theorem 22 and take $G[t]$ for sufficiently large t. If $\mathsf{iCol}_3(G) \geqslant 1-\eta$ and $S \subseteq V$ is a set of fractional size at least $1-\eta$ such that the induced subgraph on S is 3-colorable, then we may color S^t by $(a_1,\dots,a_t) \to \sum_{i=1}^{t} c(a_i) \pmod 3$, when $c : S \to \{0,1,2\}$ is a 3-coloring of S.

The soundness follows from Claim 25. ◀

References

1 Per Austrin, Subhash Khot, and Muli Safra. Inapproximability of vertex cover and independent set in bounded degree graphs. *Theory Comput.*, 7(1):27–43, 2011. doi:10.4086/toc.2011.v007a003.

2 Libor Barto, Jakub Bulín, Andrei A. Krokhin, and Jakub Oprsal. Algebraic approach to promise constraint satisfaction. *J. ACM*, 68(4):28:1–28:66, 2021. doi:10.1145/3457606.

3 Mark Braverman, Subhash Khot, Noam Lifshitz, and Dor Minzer. An invariance principle for the multi-slice, with applications. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 228–236. IEEE, 2021. doi:10.1109/FOCS52979.2021.00030.

4 Mark Braverman, Subhash Khot, and Dor Minzer. On rich 2-to-1 games. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 27:1–27:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ITCS.2021.27.

5 Irit Dinur, Subhash Khot, Guy Kindler, Dor Minzer, and Muli Safra. On non-optimally expanding sets in grassmann graphs. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 940–951. ACM, 2018. doi:10.1145/3188745.3188806.

6 Irit Dinur, Subhash Khot, Guy Kindler, Dor Minzer, and Muli Safra. Towards a proof of the 2-to-1 games conjecture? In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 376–389. ACM, 2018. doi:10.1145/3188745.3188804.

7 Irit Dinur, Elchanan Mossel, and Oded Regev. Conditional hardness for approximate coloring. *SIAM J. Comput.*, 39(3):843–873, 2009. doi:10.1137/07068062X.

8 Venkatesan Guruswami and Sai Sandeep. d-To-1 Hardness of Coloring 3-Colorable Graphs with O(1) Colors. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 62:1–62:12, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2020.62.

9 C.C Harner and R.C Entringer. Arc colorings of digraphs. *Journal of Combinatorial Theory, Series B*, 13(3):219–225, 1972. doi:10.1016/0095-8956(72)90057-3.

10 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.

11 Ken-ichi Kawarabayashi and Mikkel Thorup. Coloring 3-colorable graphs with less than n 1/5 colors. *Journal of the ACM (JACM)*, 64(1):1–23, 2017.

12 Sanjeev Khanna, Nathan Linial, and Shmuel Safra. On the hardness of approximating the chromatic number. In *Second Israel Symposium on Theory of Computing Systems, ISTCS 1993, Natanya, Israel, June 7-9, 1993, Proceedings*, pages 250–260. IEEE Computer Society, 1993. doi:10.1109/ISTCS.1993.253464.

13 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity, Montréal, Québec, Canada, May 21-24, 2002*, page 25. IEEE Computer Society, 2002. doi:10.1109/CCC.2002.1004334.

14 Subhash Khot. On the proof of the 2-to-2 games conjecture. *Current Developments in Mathematics*, 2019(1):43–94, 2019.

15 Subhash Khot, Dor Minzer, and Muli Safra. On independent sets, 2-to-2 games, and grassmann graphs. 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 576–589. ACM, 2017. doi:10.1145/3055399.3055432.

16 Subhash Khot, Dor Minzer, and Muli Safra. Pseudorandom sets in grassmann graph have near-perfect expansion. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 592–601. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00062.

17 Andrei Krokhin, Jakub Oprš al, Marcin Wrochna, and Stanislav Živný. Topology and adjunction in promise constraint satisfaction. *SIAM Journal on Computing*, 52(1):38–79, February 2023. doi:10.1137/20m1378223.

18 Dor Minzer. *On Monotonicity Testing and the 2-to-2 Games Conjecture*, volume 49. Association for Computing Machinery, New York, NY, USA, 1 edition, 2022.

19 Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.

20 S. Poljak and V. Rödl. On the arc-chromatic number of a digraph. *Journal of Combinatorial Theory, Series B*, 31(2):190–198, 1981. doi:10.1016/S0095-8956(81)80024-X.

Extracting Mergers and Projections of Partitions

Swastik Kopparty
Department of Computer Science and Department of Mathematics, University of Toronto, Canada

Vishvajeet N
School of Informatics, University of Edinburgh,UK

Abstract

We study the problem of extracting randomness from somewhere-random sources, and related combinatorial phenomena: partition analogues of Shearer's lemma on projections.

A somewhere-random source is a tuple $(X_1, \ldots, X_t)$ of (possibly correlated) $\{0,1\}^n$-valued random variables X_i where for some unknown $i \in [t]$, X_i is guaranteed to be uniformly distributed. An *extracting merger* is a seeded device that takes a somewhere-random source as input and outputs nearly uniform random bits. We study the seed-length needed for extracting mergers with constant t and constant error.

Since a somewhere-random source has min-entropy at least n, a standard extractor can also serve as an extracting merger. Our goal is to understand whether the further structure of being somewhere-random rather than just having high entropy enables smaller seed-length, and towards this we show:

- Just like in the case of standard extractors, seedless extracting mergers with even just one output bit do not exist.
- Unlike the case of standard extractors, it *is* possible to have extracting mergers that output a constant number of bits using only constant seed. Furthermore, a random choice of merger does not work for this purpose!
- Nevertheless, just like in the case of standard extractors, an extracting merger which gets most of the entropy out (namely, having $\Omega(n)$ output bits) must have $\Omega(\log n)$ seed. This is the main technical result of our work, and is proved by a second-moment strengthening of the graph-theoretic approach of Radhakrishnan and Ta-Shma to extractors.

All this is in contrast to the status for condensing mergers (where the output is only required to have high min-entropy), whose seed-length/output-length tradeoffs can all be fully explained by using standard condensers.

Inspired by such considerations, we also formulate a new and basic class of problems in combinatorics: partition analogues of Shearer's lemma. We show basic results in this direction; in particular, we prove that in any partition of the 3-dimensional cube $[0,1]^3$ into two parts, one of the parts has an axis parallel 2-dimensional projection of area at least 3/4.

2012 ACM Subject Classification Theory of computation → Expander graphs and randomness extractors; Mathematics of computing → Combinatorics; Theory of computation → Complexity theory and logic; Mathematics of computing → Discrete mathematics

Keywords and phrases randomness extractors, randomness mergers, extracting mergers, partitions, projections of partitions, covers, projections of covers

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.52

Category RANDOM

Related Version *Full Version*: https://arxiv.org/abs/2306.16915 [13]

Funding *Swastik Kopparty*: Research supported by an NSERC Discovery Grant.
Vishvajeet N: Research supported by the ERC (grant agreement No. 947778).

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 52; pp. 52:1–52:22
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

We study the problem of extracting randomness from somewhere-random sources, and related combinatorial phenomena: partition analogues of Shearer's lemma on projections. For the (completely self-contained) combinatorics, see Section 1.2, Section 6 and Section 7.

A t-part somewhere-random source is a tuple $(X_1, \dots, X_t)$ of (possibly correlated) $\{0,1\}^n$-valued random variables X_i, where some unknown X_i is guaranteed to be uniformly distributed. We will take t to be constant and n growing throughout this paper. A *merger* is a seeded device that takes a somewhere-random source and purifies its randomness. Mergers have been extensively studied in the theory of extractors, and have played an important role in their development. In fact, there were at least 3 distinct points in the history of extractors [19, 15, 7] when the best known explicit extractor constructions were based on new advances in explicit merger constructions.

An important observation is that t-part somewhere-random sources are special cases of sources with (min) entropy rate $1/t$. Thus any randomness purifying device (such as an extractor, condenser or disperser) that can give guarantees when fed a source with entropy rate at least $1/t$ is automatically some kind of merger for t-part somewhere-random sources.

In the literature, mergers have only been studied in the *condensing* regime: where their output is required to have high entropy rate (rather than requiring the output to be near-uniform). It turns out that information-theoretically, condensing mergers are completely overshadowed by classical condensers. A condenser is a seeded device that takes in a source with sufficient entropy rate and outputs a random variable with high entropy rate. Thus a condenser that can operate on sources with entropy rate $1/t$ is automatically a condensing merger for t-part somewhere-random sources. It turns out that whatever parameter ranges are achievable by condensing mergers can be completely explained by condensers.

In this paper, we study mergers in the *extracting* regime: where their output is required to be near-uniform. Our main result is a characterization of the seed-length needed for such extracting mergers. Unlike the tragic case of condensing mergers and their relationship with condensers, extracting mergers are able to step out of the shadow of extractors, and carve a niche, albeit small, for themselves.

We also study extracting multimergers, where more random variables out of the given tuple of random variables are required to be uniform and independent. This leads us to a number of interesting combinatorial / geometric questions, for which we give some new and basic combinatorial theorems (such as a partition analogue of Shearer's lemma on projections of a set in a product space).

1.1 Overview of results

Our results are best viewed in contrast to the situation with classical extractors and condensers. An extractor takes a source with some min-entropy and an independent uniform seed, and outputs a nearly-uniform distributed random variable. A condenser takes a source with some min-entropy and an independent uniform seed, and outputs a source with higher min-entropy-rate.

Both extractors and condensers are functions of the form:

$$F : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m,$$

where d is the "seed-length" and m is the "output-length".

Consider a random source $\mathbf{X}$ that is $\{0,1\}^n$-valued and has entropy rate $1/t$ (which means that its min-entropy is $\geq n/t$).

In the case of extractors, for $(1-\epsilon)$-fraction of $j \in \{0,1\}^d$, the output $F(\mathbf{X}, j)$ is required to be ϵ-close in statistical distance to the uniform distribution over $\{0,1\}^m$. In the case of condensers, for $(1-\epsilon)$-fraction of $j \in \{0,1\}^d$, the output $F(\mathbf{X}, j)$ is required to be ϵ-close in statistical distance to some $\{0,1\}^m$-valued random variable with min-entropy $\geq k'$.

Extractors and condensers are qualitatively very different from the point of view of seed-length. We summarize their salient features below:

- There are no seedless extractors or condensers.
- There are condensers with **constant** seed-length $d = O(\log \frac{1}{\epsilon})$ which are **lossless** (we can take k' as large as $\frac{n}{t} + d$), provided $m > k' + \Omega(\log \frac{1}{\epsilon})$.
- The seed-length required for an extractor to extract one bit of entropy from a random source $(\{0,1\}^n)^t$ is $\log n + 2\log \frac{1}{\epsilon} + O(1)$. Furthermore, this seed-length suffices to extract almost all the entropy out of the source.

A merger takes in a t-part somewhere-random source (which is a special case of a source with entropy rate $\frac{1}{t}$) and an independent uniform seed, and outputs a source with purer randomness. This naturally creates two kinds of mergers - condensing mergers and extracting mergers. To the best of our knowledge, only condensing mergers have been studied in the literature, and the (non-constructive) existence results for condensing mergers all follow from the existence results for condensers mentioned above.

Let $E : (\{0,1\}^n)^t \times \{0,1\}^d \to \{0,1\}^m$ be an extracting merger, namely its output is guaranteed to be ϵ-close to uniform on $\{0,1\}^m$ whenever given a t-part somewhere-random source as input.

▶ **Theorem A** (Informal). *We have the following:*

- *There are no seedless extracting mergers, even with output length* 1.
- *There are extracting mergers with* **constant** *seed length $O(\log \frac{1}{\epsilon})$, which can output a constant number of nearly-uniform bits.*
- *Nevertheless, if the seed length required for an extracting merger to extract almost all (or even a constant fraction) the entropy out of a somewhere-random source is $\Theta(\log n)$.*

The first item is trivial. The second item is also not difficult, but it already gives a taste of why things are different with extracting mergers. Indeed, randomly-chosen functions are not extracting mergers. The third item in the above theorem is our main technical result. It is proved by a second-moment strengthening of the graph-theoretic approach of Radhakrishnan and Ta-Shma to extractors.

1.2 Projections of partitions

Our study of these questions about randomness extraction leads us to formulate and make progress on a new and natural combinatorial question: the partition analogue of the Shearer/Loomis-Whitney inequalities on volumes of projections. These questions arise when we consider the problem of extracting randomness from t-part s-where random sources (where s out of the t parts of the source are uniform and independent). We call devices that do this *extracting multimergers*. For the rest of this subsection we only focus on the combinatorial aspect.

Let A be an "nice" subset of the solid cube $[0,1]^3$ with (Lebesgue) volume α. Consider the three axis-parallel 2-dimensional projections: $\Pi_{XY}(A)$, $\Pi_{YZ}(A)$, $\Pi_{XZ}(A)$. The Shearer/Loomis-Whitney inequality [5, 14] implies that at least one of these three projections has area at least $\alpha^{2/3}$. This is tight, as witnessed by the case where A is a cube of side-length $\alpha^{1/3}$ (and this is roughly the only such example).

Now consider the following partition variant: Let A, B be "nice" subsets of $[0,1]^3$ that partition $[0,1]^3$. Consider the six axis-parallel 2-dimensional projections of these two sets: $\Pi_{XY}(A)$, $\Pi_{YZ}(A)$, $\Pi_{XZ}(A)$ and $\Pi_{XY}(B)$, $\Pi_{YZ}(B)$, $\Pi_{XZ}(B)$. How large can we guarantee that one of them is?

Using the previous inequality and the fact that at least one of A, B has volume at least $1/2$, we get that one of these six 2-dimensional projections has area at least $(1/2)^{2/3} \geq 0.6299$. For this bound to be tight, we would need both A and B to have volume $1/2$, and both A and B to be tight examples for the Shearer/Loomis-Whitney inequality. This would require us to be able to cover $[0,1]^3$ by two cubes of volume $1/2$ – which is clearly impossible. This suggests that there should be a better bound!

We show, using a delicate study of the sections of the cube and some seemingly lucky inequalities, a tight bound for this problem.

▶ **Theorem B** (Informal). *Let A, B be "nice" subsets of $[0,1]^3$ that partition $[0,1]^3$. Then at least one of the six 2-dimensional projections*

$$\Pi_{XY}(A), \Pi_{YZ}(A), \Pi_{XZ}(A), \Pi_{XY}(B), \Pi_{YZ}(B), \Pi_{XZ}(B),$$

has area at least $3/4$.

Such "projections of partitions" questions can be formulated in great generality, and apart from Theorem B (whose proof we find very interesting), we also make some general observations and make some slightly non-trivial progress. We think these are very natural combinatorial questions worthy of further study. Beyond having connections to mergers, these questions turn out to be related to the KKL and BKKKL theorems/conjectures [12, 2, 10, 9] on influences of Boolean functions on the solid cube $[0,1]^n$. For example, Theorem B implies that any 3-variable Boolean function $f : [0,1]^3 \to \{0,1\}$ has some variable and some bit b such that the "influence towards b" of that variable is at least $1/4$, and this is tight.

Another application of such results is to partition analogues of the Kruskal-Katona theorem. For example, Theorem B implies that for any partition of $\binom{[n]}{3}$ into two parts, one of the two parts has shadow with size at least $\left(\frac{3}{4} - o(1)\right)\binom{n}{2}$.

1.3 Related work

Mergers were introduced by Ta-Shma [19] in his thesis, and were used to construct state-of-the-art extractors at the time (these were condensing mergers). Later, [15] proposed a new condensing merger construction based on taking random linear combinations of vectors over finite fields, and used it in their construction of the first extractors optimal upto constant factors. This analysis was greatly improved by Dvir [6] through his solution to the finite field Kakeya conjecture. Subsequently, [8, 7] defined a higher degree polynomial variant of the [15] merger, and by developing the ideas from [6], were able to construct improved (constant seed) mergers and state-of-the-art extractors. Subsequently [20] showed how to get analogous explicit constructions of condensers (subsuming the [7] condensing mergers) by improving the [11] condensers.

Another interesting constant seed condensing merger is by [18], which was constructed on the way to multi-source extractors.

Our lower bounds for the seed length of extracting mergers are proved by developing ideas from the paper of Radhakrishnan and Ta-Shma [17]. A recent beautiful proof of [1] also achieved a similar result to [17] in a much cleaner way, but we were not able to adapt this approach to our setting.

Other papers relevant to the study of multimergers are related to resilient functions [3, 4, 16].

Finally, our combinatorial results are related to the KKL and BKKKL theorems/conjectures [12, 2, 10, 9] on influences of Boolean functions on the solid cube $[0,1]^n$.

1.4 Organization

We give the basic definitions of extracting mergers and extracting multimergers in Section 2. In Section 3 we start with a simple proof that seedless mergers do not exist. This is followed by showing the existence of mergers and multimergers in the extracting regime with constant seed-length. We prove our lower bound on the seed length of extracting mergers in Section 4, which culminates in Theorem 9. In Section 5 we explore the connection between *seedless* extracting mergers and projections of partition questions. Section 6 is devoted to proving Theorem 17, our (optimal) lower bound on partitioning the unit cube into 2 parts, and Section 7 is devoted to partitions of the cube into 3 parts.

2 Sources and Mergers

▶ **Definition 1** (k-source). *For any k, we say that a random variable X is a k-source if for all x, $\Pr[X = x] \leq 2^{-k}$*

2.1 Somewhere and s-where Random Sources

▶ **Definition 2** (Somewhere-Random Source). *For a domain D, a tuple $\mathbf{X} = (X_1, \ldots, X_t)$ of jointly distributed D-valued random variables is called a t-part somewhere random source if for some $i \in [t]$, the distribution of X_i is uniform over D.*

▶ **Definition 3** (s-where Random Source). *For a domain D and an integer $s > 0$, a tuple $\mathbf{X} = (X_1, \ldots, X_t)$ of jointly distributed D-valued random variables is called a t-part s-where random source if for some distinct $i_1, \ldots, i_s \in [t]$, the joint distribution of $(X_{i_1}, X_{i_2}, \ldots, X_{i_s})$ is uniform over D^s.*

2.2 Extracting Mergers and Multimergers

▶ **Definition 4** (Extracting Mergers). *Let n, t, d, m be integers, and let $\epsilon > 0$.*

A function $E : (\{0,1\}^n)^t \times \{0,1\}^d \to \{0,1\}^m$ is called an $\boldsymbol{(n, t, d, m, \epsilon)}$***-extracting merger*** *if the following holds.*

Suppose $\mathbf{X} = (X_1, \ldots, X_t)$ is a somewhere-random source where each X_i is $\{0,1\}^n$-valued. Then for at least $(1-\epsilon)$-fraction of $j \in \{0,1\}^d$, the distribution of:

$$Z = E(\mathbf{X}, j),$$

is ϵ-close to the uniform distribution on $\{0,1\}^m$.

We will sometimes refer to these as ϵ-extracting mergers (since n, d, t, m are related to the shape of E).

▶ **Definition 5** (Extracting Multimergers). *Let n, t, s, d, m be integers, and let $\epsilon > 0$.*

A function $E : (\{0,1\}^n)^t \times \{0,1\}^d \to \{0,1\}^m$ is called an $\boldsymbol{(n, d, t, m, \epsilon, s)}$***-extracting multimerger*** *if the following holds.*

Suppose $\mathbf{X} = (X_1, \ldots, X_t)$ *is an* s*-where random source where each* X_i *is* $\{0,1\}^n$*-valued. Then for at least* $(1-\epsilon)$*-fraction of* $j \in \{0,1\}^d$*, the distribution of:*

$$Z = E(\mathbf{X}, j),$$

is ϵ*-close to the uniform distribution on* $\{0,1\}^m$*.*

We will sometimes refer to these as (ϵ, s)-extracting multimergers (since n, d, t, m are related to the shape of E).

Observe that the $s = 1$ case in the above definition corresponds to extracting mergers.

Note on the definitions

In all our definitions, we chose to define the "strong" versions (where the output bits are required to be independent of the seed) for simplicity. In fact, our existence result for mergers is for the strong version, and our impossibility result is for the weak version.

3 Simple results about extracting mergers

For the rest of this paper, we only talk about *extracting* (not condensing) mergers and multimergers.

3.1 Seedless Mergers do not exist

We begin with the simple observation that there are no seedless extracting mergers.

▶ **Theorem 6** (There are no seedless mergers). *Let* n *be an integer and* $\varepsilon < 1/2$*. There does not exist a function* $M : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ *that is an* ε*-merger.*

Proof. Fix an $\varepsilon < 1/2$. Assume for the sake of contradiction there exists an ε-merger $M : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$.

In particular, this means for *every* function $f : \{0,1\}^n \to \{0,1\}^n$, when X is distributed uniformly over $\{0,1\}^n$, the distribution of $M(X, f(X))$ is ε-close to uniform on $\{0,1\}$ – and in particular, it has full support on $\{0,1\}$. We will now demonstrate a function $g : \{0,1\}^n \to \{0,1\}^n$ such that $M(g(Y), Y)$ is constant for uniformly distributed Y, thus contradicting the merger assumption.

Fix any $y \in \{0,1\}^n$. Consider the constant function $f_y : \{0,1\}^n \to \{0,1\}^n$ given by $f_y(x) = y$ for all x. By our hypothesis above, the distribution of $M(X, f_y(X))$ has full support $\{0,1\}$. Thus there exists $x \in \{0,1\}^n$ such that $M(x, y) = 0$. Pick one such x and call it $g(x)$.

Thus we have $M(g(y), y) = 0$ for all $y \in \{0,1\}^n$. We conclude that for uniform $Y \in \{0,1\}^n$, $M(g(Y), Y) = 0$, which is the desired contradiction. ◀

3.2 Extracting mergers with constant seed exist

We now show that constant seed extracting mergers with constant output length exist. While the proof is quite simple, it is interesting because (1) constant seed extractors do not exist, (2) a random choice of $E : (\{0,1\}^n)^t \times \{0,1\}^d \to \{0,1\}^m$ does not give a constant seed extracting mergers, and most importantly (3) as we will later see, the seed length still needs to be superconstant to produce a superconstant number of output bits, as we will see in the next section.

▶ **Theorem 7.** *Let n, t be integers and $\epsilon > 0$.*
Then for any integer $m \leq n$, setting:

$$d = \log m + \log(t-1) + 2\log\frac{1}{\epsilon} + O(1),$$

there exists a function $E : (\{0,1\}^n)^t \times \{0,1\}^d \to \{0,1\}^m$ that is an ε-extracting merger.

Thus with $O(\log t + \log \frac{1}{\epsilon})$ bits of seed, we can extract $\mathrm{poly}(\frac{1}{\epsilon})$ bits out.

Proof. We want to get an extracting merger $E((x_1, \dots, x_t), j)$, where the $x_i \in \{0,1\}^n$ and $j \in \{0,1\}^d$.

The nature of a somewhere-random source is that applying a truncation to each element of the source yields a smaller somewhere-random source. The idea of our extracting merger is to truncate our somewhere-random source, and to then apply a standard seeded extractor to the entire truncated source. The truncation makes the instance size smaller, enabling us to use a reduced seed length in the extractor.

We truncate each x_i to the first m bits, thus obtaining $x'_1, \dots, x'_t \in \{0,1\}^m$.

We can verify that our truncation to the first m bits produces a source $(X'_1, \dots, X'_t)$ of length mt and min-entropy m. By the standard result on existence of extractors (See Theorem 6.14 in [21]), there exists a strong (m, ϵ)-extractor $Ext_0 : \{0,1\}^{mt} \times \{0,1\}^d \to \{0,1\}^m$ with seed length $d = \log m + \log(t-1) + 2\log\frac{1}{\epsilon} + O(1)$.

We can thus define the function $E : (\{0,1\}^n)^t \times \{0,1\}^d \to \{0,1\}^m$:

$$E((x_1, \dots, x_t), j) = Ext_0((x'_1, \dots, x'_t), j).$$

Observe that the function E is an ϵ-extracting merger that uses a seed j of length d and outputs m bits as required. ◀

In contrast, a random $E : (\{0,1\}^n)^t \times \{0,1\}^d \to \{0,1\}$ is not an extracting merger at all! To see this, it suffices to fix $t = 2$. If E is chosen at random, then for every $j \in \{0,1\}^d$ and $x \in \{0,1\}^n$, it is very likely that there exists a $y \in \{0,1\}^n$ such that $E((x,y),j) = 0$. Define $f_j : \{0,1\}^n \to \{0,1\}^n$ by $f_j(x) =$ any such y. Then for every $j \in \{0,1\}^d$, $E(X, f_j(X), j)$ is constant when X is picked uniformly at random, showing that E is not a merger.

3.3 Extracting Multimergers

Using the same idea, we also get interesting multimergers.

▶ **Theorem 8.** *Let n, t, s be integers with $s < t$, and $\epsilon > 0$. Then for any integer $a \leq n$, setting $m = s \cdot a$ and:*

$$d = \log a + \ 2\log\frac{1}{\epsilon} + \log(t-s) + \Omega(1),$$

there exists a function $E : (\{0,1\}^n)^t \times \{0,1\}^d \to \{0,1\}^m$ that is an (ε, s)-extracting multimerger.

Taking for example $s = t-1$ and $a = \mathrm{poly}\left(\frac{1}{\epsilon}\right) \ll n$, we get that by investing $O(\log\frac{1}{\epsilon})$ bits of seed, we can extract $\mathrm{poly}\left(\frac{1}{\epsilon}\right) \cdot t$ bits of randomness from any t-part $(t-1)$-where random source $\mathbf{X} \in (\{0,1\}^n)^t$.

In this setting of parameters, the seed length does not even depend on t, and we could take t to be growing superconstantly while preserving constant seed-length.

3.4 Seedless Multimergers

Our final observation of this section is that for multimergers with large t and where s is a large fraction of t, seedless multimergers with small error *do* exist. Indeed, if $s = t - 1$, and we define $E : (\{0,1\}^n)^t \to \{0,1\}$ by

$$E(x_1, \ldots, x_t) = Maj(x_{11}, x_{21}, \ldots, x_{t1}),$$

it is easy to see that E is a seedless $(\epsilon, t-1)$-multimerger for $\epsilon = O(\frac{1}{\sqrt{t}})$. Replacing E with any resilient function gives other examples of seedless multimergers (including with larger output size).

Investigation of this phenomenon leads us to the projections of partitions question, and we explicitly give the connection and some results about it in a later section. Nevertheless, this seems like the tip of an iceberg.

4 Mergers with large output need large seed

In this section we show a lower bound on the seed-length for 2-source extracting mergers, essentially showing that the dependence of the seed-length for extracting mergers in Theorem 7 on m and ϵ is tight.

▶ **Theorem 9.** *Let $\epsilon < 1/40$. Let $E : (\{0,1\}^n)^2 \times \{0,1\}^d \to \{0,1\}^m$ be a ε-extracting merger. Then for $\epsilon \geq 2^{-\Omega(m)}$, we have:*

$$d \geq \log m + \log \frac{1}{\epsilon} - O(1).$$

and for $\epsilon < 2^{-\Omega(m)}$, we have:

$$d \geq \Omega(m).$$

For the proof of this theorem, the representation of the inputs and output of E in terms of bits is a distraction. So, letting $N = 2^n$, $D = 2^d$, $M = 2^m$ and identifying $\{0,1\}^n, \{0,1\}^d, \{0,1\}^m$ with $[N], [D], [M]$ respectively, we will view E as a function $E : [N]^2 \times [D] \to [M]$.

Recalling the ϵ-extracting merger property, we have that E is such that whenever X, Y are jointly distributed $[N]$-valued random variables, with at least one of them uniformly distributed, and J is picked uniformly from $[D]$ and independently of (X, Y), then the distribution of $E((X,Y), J)$ is ϵ-close to the uniform distribution on $[M]$.

We will show that for $\epsilon \geq M^{-\Omega(1)}$, we have:

$$D \geq \Omega\left(\frac{1}{\epsilon} \log M\right),$$

and for $\epsilon < M^{-\Omega(1)}$, we have:

$$D \geq \Omega\left(M^{\Omega(1)}\right).$$

Our proof is based on the following idea. Consider a uniformly random subset $S \subseteq [M]$ of size λM. For each $y \in [N]$, we look for an x such that for all $j \in [D]$, $E(x,y,j) \notin S$. If there is such an x, then we define $g(y) = x$. If such an x exists for most y, then for uniformly chosen $Y \in [N], J \in [D]$, we have $\Pr_{Y,J}[E(g(Y), Y, J) \in S] \ll \lambda - \epsilon$, contradicting the merger property. Thus for most S, for many y there is no such x; namely, for most S, for many y, for all x, there is some j, such that $E(x, y, j) \in S$. For this to happen for even one y turns out to be very abnormal, and we derive our lower bound on D by digging into its structure. This part uses a second moment variation of the Radhakrishnan-TaShma [17] approach to extractor lower bounds.

4.1 Abnormal conductors

A map $C : [N] \times [D] \to [M]$ is called a conductor (this is a general term capturing the shape of seeded extractors and seeded condensers). We will also view this as a bipartite multigraph with $[N]$ on the left, $[M]$ on the right and D labelled edges coming out of every left vertex.

If C is chosen at random, then for most $S \subseteq [M]$ of size λM and most $x \in [N]$, we expect about λ fraction of the edges coming out of x to land in S. But we do not expect that this will happen for *all* x! When C is chosen at random, then for most S there will be some $x \in [N]$ for which a very small ($\ll \lambda$) fraction of edges coming out of x lie in S. We capture this with the following definition.

▶ **Definition 10.** *Let $C : [N] \times [D] \to [M]$ be a conductor. Let S be a subset of $[M]$.*

We say the vertex $x \in [N]$ totally misses *S (under C) if*

$$|\{j \in [D] \mid C(x, j) \in S\}| = 0.$$

We say the vertex $x \in [N]$ mostly misses *S (under C) if*

$$|\{j \in [D] \mid C(x, j) \in S\}| < \frac{1}{2}\frac{|S|}{M}D.$$

▶ **Definition 11** (Abnormal conductors)**.** *Let $C : [N] \times [D] \to [M]$ be a conductor.*

We say that C is (γ, λ)-abnormal if

$$\Pr_{S \in \binom{[M]}{\lambda M}}[\exists x \in [N] \text{ s.t. } x \text{ mostly misses } S] < 1 - \gamma.$$

▶ **Lemma 12** (Extracting mergers contain abnormal conductors)**.** *Suppose $0 < \gamma < \frac{\lambda}{2} - \epsilon$. Suppose $E : [N]^2 \times [D] \to [M]$ is an ϵ-extracting merger. For $y \in [N]$, let $E_y : [N] \times [D] \to [M]$ be the function $E(\cdot, y, \cdot)$. Then for some $y \in [N]$, E_y is (γ, λ)-abnormal.*

Proof. Suppose not; namely that for all $y \in [N]$, we have that E_y is not (γ, λ)-abnormal.

Pick $S \in \binom{[M]}{\lambda M}$ uniformly at random.

Let B_y be the event that there exists some $x \in [N]$ that mostly misses S under E_y.

By our assumption, $\Pr[B_y] \geq 1 - \gamma$. So the expected number of y for which B_y happens is at least $(1 - \gamma)N$.

Thus there exists some particular choice of S for which B_y happens for at least $(1 - \gamma)N$ many ys. Call this choice S_0. Define $f : [N] \to [N]$ by defining $f(y)$ as follows:

$$f(y) = \begin{cases} \text{any } x \text{ that mostly misses } S_0 \text{ under } E_y & B_y \text{ happened,} \\ \text{arbitrary} & B_y \text{ did not happen.} \end{cases}$$

Then

$$\Pr_{Y \in [N], J \in [D]}[E(f(Y), Y, J) \in S_0] < \frac{\lambda}{2}(1 - \gamma) + \gamma < \lambda - \epsilon.$$

But $|S_0| = \lambda M$, and thus we get a contradiction to the ϵ-extracting merger property of E. This completes the proof. ◀

4.2 The structure of abnormal conductors

The previous lemma gave us a y for which E_y is abnormal. We now use show that abnormal conductors are very structured, and thus get a lower bound on D.

▶ **Lemma 13.** *Let $C : [N] \times [D] \to [M]$ be a (γ, λ)-abnormal conductor. Suppose $10\epsilon < \lambda < \frac{1}{2}$. Suppose that for $X \in [N]$ and $J \in [D]$ picked uniformly and independently, $C(X, J)$ is ϵ-close to the uniform distribution on $[M]$. Then*

$$D \geq \min \left\{ \Omega\left(\frac{1}{\lambda}\log(\lambda\gamma M)\right), \Omega\left((\lambda\gamma M)^{1/4}\right) \right\}.$$

Proof. We begin with a pruning phase to remove the high degree vertices from the right side. At first reading, it will be helpful to consider the case where $B = \emptyset$.

Let $\beta = \frac{\lambda}{5} - \epsilon$. Note that the average right degree is ND/M. Define the set of high-degree right vertices by:

$$B = \{z \in [M] \mid \text{ there are at least } \frac{1}{\beta}\frac{ND}{M} \text{ edges to } z\}.$$

Thus $|B| \leq \beta M$. By the hypothesis on $C(X, J)$, we have

$$\Pr_{X \in [N], J \in [D]}[C(X, J) \in B] \leq \beta + \epsilon.$$

Let G be the set of all vertices on the left that do not have too many edges to B; namely:

$$G = \{x \in [N] \mid x \text{ has at most } 2(\beta + \epsilon)D \text{ edges to } B\ \}.$$

Then $|G| \geq N/2$.

Now pick $S \in \binom{[M]}{\lambda M}$ uniformly at random. If there is a vertex $x \in G$ that totally misses $S \setminus B$, then by choice of G:

$$|\{j \mid C(x, j) \in S\}| \leq 2(\beta + \epsilon)D < \frac{1}{2}\lambda D,$$

namely, x mostly misses S.

By our hypothesis on the abnormality of C, the existence of such an x cannot happen too often. Thus:

$$\Pr_S[\exists x \in G \mid x \text{ totally misses } S \setminus B] < 1 - \gamma. \tag{1}$$

For each $x \in G$, let A_x be the event that x totally misses $S \setminus B$ under C.

We are interested in the event that some $x \in G$ totally misses $S \setminus B$, namely, the event $\bigvee_{x \in G} A_x$.

Observe that[1]

$$\Pr[A_x] \geq \frac{\binom{M-D}{\lambda M}}{\binom{M}{\lambda M}} \geq e^{-4\lambda D} =: p.$$

Define $A = \sum_{x \in G} A_x$. Then $\mathbf{E}[A] \geq |G|p$.

[1] Here we use the observation that $(1 - \frac{D}{(1-\lambda)M}) < e^{-\frac{2D}{(1-\lambda)M}} < e^{-4D/M}$, which follows from the fact that $1 - x < e^{-2x}$ for $x < 1/2$.

By the second moment method, we have:

$$\Pr[A=0] \leq \frac{\mathbf{Var}[A]}{\mathbf{E}[A]^2}.$$

But Equation (1) tells us that $\Pr[A=0] > \gamma$.
Thus $\mathbf{Var}[A] \geq \gamma \mathbf{E}[A]^2 \geq \gamma \cdot p^2 |G|^2$.
We now extract some structure from this.
We have:

$$\mathbf{Var}[A] = \sum_{x,x' \in G} \left(\Pr[A_x \wedge A_{x'}] - \Pr[A_x]\Pr[A_{x'}] \right).$$

Two simple observations about this expression:

- Each term in the sum above is at most 1.
- Furthermore, if x, x' have no common neighbors in $[M] \setminus B$, then the corresponding term of the sum above is ≤ 0. Indeed, if $U_x, U_{x'} \subseteq [M] \setminus B$ are the neighborhoods of x and x' in $[M] \setminus B$, and if they are disjoint, then:

$$\Pr[A_x] = \frac{\binom{M-|U_x|}{\lambda M}}{\binom{M}{\lambda M}},$$

$$\Pr[A_{x'}] = \frac{\binom{M-|U_{x'}|}{\lambda M}}{\binom{M}{\lambda M}},$$

$$\Pr[A_x \wedge A_{x'}] = \frac{\binom{M-|U_x \cup U_{x'}|}{\lambda M}}{\binom{M}{\lambda M}} = \frac{\binom{M-|U_x|-|U_{x'}|}{\lambda M}}{\binom{M}{\lambda M}}.$$

So

$$\frac{\Pr[A_x \wedge A_{x'}]}{\Pr[A_x]\Pr[A_{x'}]} = \prod_{i=0}^{\lambda M-1} \frac{(M-i)\cdot(M-|U_x|-|U_{x'}|-i)}{(M-|U_x|-i)\cdot(M-|U_{x'}|-i)} \leq 1.$$

Combining the largeness of $\mathbf{Var}[A]$ with these two observations tells us that there are many $x, x' \in G$ which have a common neighbor in $[M] \setminus B$. Specifically:

$$\gamma p^2 |G|^2 \leq \mathbf{Var}[A] \leq \sum_{x,x' \in G} \mathbf{1}[x, x' \text{ have a common neighbor in } [M] \setminus B].$$

Thus there are at least $\gamma p^2 |G|^2 \geq \frac{1}{4}\gamma p^2 N^2$ pairs x, x' from G that have a common neighbor in $[M] \setminus B$.

Now the initial pruning we did will help us. Since all the vertices in $[M] \setminus B$ have degree at most $\frac{1}{\beta}\frac{ND}{M}$, we can bound the number of such pairs x, x'. For every vertex $x \in G$, there are at most $D \cdot \frac{1}{\beta}\frac{ND}{M}$ vertices x' such that x and x' share a common neighbor in $[M] \setminus B$. Thus the total number of pairs x, x' from G that have a common neighbor in $[M] \setminus B$ is at most

$$N \cdot D \cdot \frac{1}{\beta}\frac{ND}{M} = \frac{1}{\beta}\frac{D^2}{M}N^2.$$

Thus $\frac{1}{4}\gamma p^2 \leq \frac{1}{\beta}\frac{D^2}{M}$. Since $p = e^{-4\lambda D}$, we get:

$$M \leq \frac{4}{\gamma\beta} D^2 e^{8\lambda D}.$$

This means that either $D \geq \Omega\left((\gamma\beta M)^{1/4}\right) = \Omega\left((\gamma\lambda M)^{1/4}\right)$, or else:

$$D \geq \Omega\left(\frac{1}{\lambda}\log(\gamma\beta M)\right) \geq \Omega\left(\frac{1}{\lambda}\log(\gamma\lambda M)\right).$$ ◀

4.3 Putting everything together

We now prove Theorem 9.

Proof. Let $E : [N]^2 \times [D] \to [M]$ be an ϵ-extracting merger.

Set $\lambda = 20\epsilon$ and $\gamma = \epsilon$. Lemma 12 tells us that there is some $y := y_0$ for which E_y is (λ, γ)-abnormal.

Now, since E is ϵ-extracting, we have that $E_y(X, J) = E(X, y, J)$ is ϵ-close to the uniform distribution on $[M]$ for uniform and independent $X \in [N]$ and $J \in [D]$. Thus Lemma 13 tells us that

$$D \geq \min\left\{\Omega\left(\frac{1}{\epsilon}\log(\epsilon^2 M)\right), \Omega\left((\epsilon^2 M)^{1/4}\right)\right\}$$

.

If $\epsilon \geq \frac{1}{M^{1/10}}$, then the first expression is smaller and

$$D \geq \Omega(\frac{1}{\epsilon}\log M),$$

and if $\epsilon < \frac{1}{M^{1/10}}$, then E is also a $\frac{1}{M^{1/10}}$-extracting merger, and thus using the above lower bound for $\frac{1}{M^{1/10}}$ in place of ϵ, we get that:

$$D \geq M^{\Omega(1)}.$$ ◀

5 Seedless Extracting Multimergers and Projections of Partitions

In this section, we study seedless multimergers. Here our understanding is far from complete, and we suggest many directions for research.

We begin by observing a connection between seedless multimergers and a very natural and clean geometric question: how do we partition the unit cube $[0, 1]^t$ into c parts to ensure that all s-dimensional axis-parallel projections of all parts are small? We then prove some interesting positive and negative results about special cases of this general question. We conclude by collecting a number of observations and questions about this natural partitioning problem.

5.1 Seedless Multimergers with 1 bit output

In Section 3.1 we have already seen that there are no seedless mergers (i.e., with $s = 1$). We now look into seedless multimergers.

Let us consider the simplest nontrivial situation: $t = 3$ and $s = 2$, and $m = 1$ (we only try to extract 1 bit of randomness), with n big. Suppose a given function $E : (\{0, 1\}^n)^3 \to \{0, 1\}$ is known to be a (ϵ, s)-multimerger. For convenience, we identify $\{0, 1\}^n$ with $[N]$, for $N = 2^n$.

By the multimerger property, for every function $f : [N]^2 \to [N]$, the distribution of $E(X, Y, f(X, Y))$ should be ϵ-close to uniform. Let

$$P_{XY,0} = \{(x, y) \in [N]^2 : \exists z \mid E(x, y, z) = 0\}.$$

Notice that this is the projection of $E^{-1}(0)$ to two coordinates.

If $P_{XY,0}$ is bigger than $\frac{1+\epsilon}{2}N^2$, then we can violate the multimerger property: we define $f : [N]^2 \to [N]$ by $f(x, y) = z$, if any, such that $E(x, y, z) = 0$". Then $E(X, Y, f(X, Y))$ for uniform and independent $X, Y \in [N]$ is ϵ-far from uniform.

We have a similar observation for all the other two dimensional projections, and also for the set $E^{-1}(1)$. Thus if a seedless one-bit output multimerger for 3-part 2-where random sources exists, then there is a partition of $[N]^3$ into 2 parts such that each part has all its 2-dimensional axis parallel projections have size at most $\frac{1+\epsilon}{2}N^2$.

The connection also goes in reverse. Suppose we have a partition A, B of $[N]^3$ for which each part has all its 2-dimensional axis parallel projections with size at most $\frac{1+\epsilon}{2}N^2$. Let $E : [N]^3 \to \{0,1\}$ be the unique function with $E^{-1}(0) = A$ and $E^{-1}(1) = B$. Suppose (X, Y, Z) is an $[N]^3$-valued random variable that is 2-where random. Then we claim that $E(X, Y, Z)$ is ϵ-close to the uniform distribution. Indeed, if (X, Y) is uniformly distributed over $[N]^2$ (the cases of (Y, Z) and (X, Z) being uniformly distributed are similar), then:

$$\begin{aligned}
\Pr[E(X,Y,Z) = 0] &\le \Pr_{X,Y}[\exists z \in [N] \text{ s.t. } E(X,Y,z) = 0] \\
&\le \Pr_{X,Y}[\exists z \in [N] \text{ s.t. } (X,Y,z) \in A] \\
&\le \frac{|\Pi_{XY}(A)|}{N^2} \\
&\le \frac{1+\epsilon}{2}, \\
\Pr[E(X,Y,Z) = 1] &\le \Pr_{X,Y}[\exists z \in [N] \text{ s.t. } E(X,Y,z) = 1] \\
&\le \Pr_{X,Y}[\exists z \in [N] \text{ s.t. } (X,Y,z) \in B] \\
&\le \frac{|\Pi_{XY}(B)|}{N^2} \\
&\le \frac{1+\epsilon}{2},
\end{aligned}$$

which implies the desired ϵ-closeness to uniform of $E(X, Y, Z)$.

The exact same argument applies to general t, s. We record this below.

▶ **Theorem 14.** *Let $N = 2^n$. There exists a seedless $(n, d, t, m, \epsilon, s)$-multimerger if and only if there is a partition of $[N]^t$ into two sets A, B such that for every subset $U \subseteq [t]$ of size s, the projections $\Pi_U(A)$ and $\Pi_U(B)$ onto the coordinates U are of size at most $\frac{1+\epsilon}{2}N^s$.*

Motivated by this, we consider general projections of partitions questions, where the set $[N]^t$ is partitioned into c parts, and we seek to minimize the maximum s-dimensional axis parallel projection of all the parts[2].

As noted in the introduction, there is a basic bound for this problem that comes from Shearer's lemma. It says that there is a lower bound of $\left(\frac{1}{c}\right)^{s/t} N^s$ on the size of some projection. This bound is usually not tight – but it sometimes is! Whenever c is a perfect t'th power, then this bound is tight, and is realized by a partition into product sets. But for other c this kind of partition does not work, and very interesting questions ensue. In particular, we would like to highlight the case of N gigantic, and $c = \text{poly}(t)$ (so that c is clearly not a perfect t'th power).

Here is one observation that gives a flavor of what happens for large t and s. When c is a constant, and $s = t - o(\sqrt{t})$, there is a partition so that all s-dimensional projections of size $\frac{1}{c} + o(1)$. This comes by considering suitable threshold partitions. Extensions of this are related to the BKKKL [2, 10] conjectures on low influence functions.

[2] For $c > 2$, the problem of getting such partitions with c parts is somewhat related to the problem of multimergers with $\log_2(c)$ bit output, but the connection is not as tight as for the case of $c = 2$

This question is also equivalent to a problem in the continuous domain about open covers of $[0,1]^t$. Here we want to minimize the maximum s-dimensional projection size when we cover $[0,1]^t$ by c open sets.

In the following sections, we discuss two results on partitioning in three dimensions. For the first result, we get (to our surprise!) the tight bound for partitioning the cube into two parts. For the second result, we get nontrivial bounds (both upper and lower) for partitioning the cube into three parts.

Apology

These questions are more naturally phrased as questions about covers rather than partitions. However we stick to the partition language because of the particular sequence of events that led us to these problems.

6 Partitioning the 3-dimensional cube into two parts

In this section, we prove a tight bound on the largest 2 dimensional axis parallel projection of a part when partitioning $[0,1]^3$ into 2 parts.

Let $\pi_{XY}, \pi_{YZ}, \pi_{XZ} : [0,1]^3 \to [0,1]^2$ be the 2-dimensional projection maps.

The following example gives a nice partitioning with small projections.

▶ **Definition 15** (Majority Partitioning Scheme). *We define the function $MAJ_3 : [0,1]^3 \to \{0,1\}$ as*

$$MAJ_3(x,y,z) = Maj(x_1,y_1,z_1)$$

where Maj denotes the Majority function on 3 bits, and where x_1, y_1, z_1 denote the indicator variables for whether $x > 1/2$, $y > 1/2$, $z > 1/2$ respectively.

We refer to the partition naturally induced by the output of MAJ_3 on the input space $[0,1]^3$ i.e. $\{MAJ_3^{-1}(0), MAJ_3^{-1}(1)\}$, as the Majority Partitioning Scheme.

We next record the observation tha all 2-dimensional axis-parallel projections of all parts in the majority partitioning scheme on $[N]^3$ are of size at most $\frac{3}{4}N^2$, which is stated in the following lemma:

▶ **Lemma 16** (Majority Partitions Optimally). *Every 2-dimensional projection of every partition in the majority partitioning scheme MAJ_3 on $[0,1]^3$ is of size at most $\frac{3}{4}$.*

In the other direction, we first prove a lower bound on projection sizes for a discrete version of the problem.

Let N be a large positive integer. We reuse notation and let $\pi_{XY}, \pi_{YZ}, \pi_{XZ} : [N]^3 \to [N]^2$ be the 2-dimensional projection maps.

▶ **Theorem 17.** *Let $A, B \subseteq [N]^3$ be a partition.*

Then one of the six 2-dimensional projections of A and B

$$\pi_{XY}(A), \pi_{YZ}(A), \pi_{XZ}(A), \pi_{XY}(B), \pi_{YZ}(B), \pi_{XZ}(B)$$

has size at least $\frac{3}{4}N^2$.

Proof of Theorem 17 can be found in the appendix.

By a simple discretization argument, we get the following corollary:

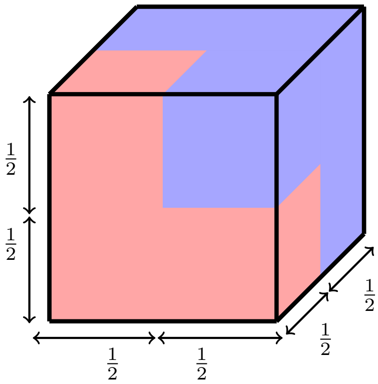


Figure 1 Majority Partitioning of the cube into 2 parts, where the partitioned sets are coloured in red and blue. Observe that *all* projections of the red set and the blue set are of equal size $\frac{3}{4}$, and Theorem 17 implies this partitioning is optimal.

▶ **Corollary 18.** *Any cover of* $[0,1]^3$ *by two open sets* A, B *has one of the following 6 sets:*

$$\Pi_{XY}(A), \Pi_{YZ}(A), \Pi_{XZ}(A), \Pi_{XY}(B), \Pi_{YZ}(B), \Pi_{XZ}(B)$$

having area at least $3/4$.

Thus we get that MAJ_3 is an *optimal* partition for partitioning $[N]^3$ into two parts.

7 Partitioning the 3-dimensional cube into three parts

In this section we study the case of partitioning the 3 dimensional cube $[0,1]^3$ into 3 parts.

We begin with a nice partition of $[0,1]^3$ into 3 parts so that each part has small 2-dimensional axis-parallel projections.

▶ **Definition 19** (Golden Ratio Partitioning Scheme). *Let u be the positive root of $x^2 + x = 1$. We define the function $GR_3 : [0,1]^3 \to \{0,1,2\}$ as*

$$GR_3(x,y,z) = \begin{cases} 0, & |x| > u, |y| > u \\ 1, & |x| \le u, |y| \le u, |z| \le \frac{1}{2} \\ 2, & \textit{otherwise.} \end{cases}$$

We refer to the partition into 3 parts naturally induced by the output of GR_3 on the input space $[0,1]^3$ i.e. $\{GR_3^{-1}(0), GR_3^{-1}(1), GR_3^{-1}(2)\}$, as the golden ratio partitioning scheme.

▶ **Lemma 20** (Golden Ratio Partitioning Bound). *Every 2-dimensional projection of every partition in the golden ratio partitioning scheme GR_3 on $[0,1]^3$ is of size $u \le 0.619$.*

We do not know if this is the optimal partition into 3 parts. For the rest of this section, we prove the best lower bound that we know. As in the previous section, we do this via an analogous discrete problem.

Let $\eta_0 \approx 0.5264$ be the real number $\in [0.5, 1.0]$ satisfying:

$$(2 - 3\eta_0) \cdot \left(2 - 2\sqrt{1 - \eta_0}\right) + (3\eta_0 - 1) = \frac{1}{6}(4 - \eta_0).$$

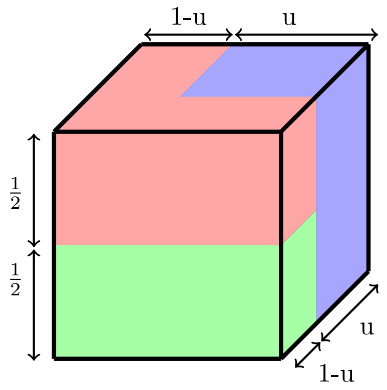


Figure 2 Golden Ratio Partitioning of the cube into 3 parts, where the partitioned sets are coloured in red, green and blue. The green and red parts are just translates of each other. Here u is the positive root of $x^2 + x = 1$.

▶ **Theorem 21.** *Let $A, B, C \subseteq [N]^3$ be a 3-partition of $[N]^3$. Then one of the nine 2-dimensional projections of A, B and C, i.e.,*
$\Pi_{XY}(A), \Pi_{XY}(B), \Pi_{XY}(C), \Pi_{YZ}(A), \Pi_{YZ}(B), \Pi_{YZ}(C), \Pi_{XZ}(A), \Pi_{XZ}(B), \Pi_{XZ}(C)$ *has size at least $\eta_0 N^2$.*

Proof of Theorem 21 can be found in the appendix.

This gives us a corresponding result about covers of $[0, 1]^3$ with 3 open sets.

▶ **Corollary 22.** *Any cover of $[0, 1]^3$ by three open sets A, B, C has one of the following 9 sets:*

$$\Pi_{XY}(A), \Pi_{YZ}(A), \Pi_{XZ}(A), \Pi_{XY}(B), \Pi_{YZ}(B), \Pi_{XZ}(B), \Pi_{XY}(C), \Pi_{YZ}(C), \Pi_{XZ}(C)$$

having area at least η_0.

References

1 Divesh Aggarwal, Siyao Guo, Maciej Obremski, João Ribeiro, and Noah Stephens-Davidowitz. Extractor Lower Bounds, Revisited. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, 2020.

2 Jean Bourgain, Jeff Kahn, Gil Kalai, Yitzhak Katznelson, and Nathan Linial. The Influence of Variables in Product Spaces. In *Israel Journal of Mathematics*, 1992.

3 Eshan Chattopadhyay, Jesse Goodman, Vipul Goyal, and Xin Li. Extractors for Adversarial Sources via Extremal Hypergraphs. In *Proceedings of the 52nd Annual ACM Symposium on Theory of Computing (STOC)*, 2020.

4 Eshan Chattopadhyay and David Zuckerman. Explicit Two-Source Extractors and Resilient Functions. In *Annals of Mathematics*, 2019.

5 F.R.K Chung, R.L Graham, P Frankl, and J.B Shearer. Some Intersection Theorems for Ordered Sets and Graphs. *Journal of Combinatorial Theory, Series A*, 1986.

6 Zeev Dvir. On The Size of Kakeya Sets in Finite Fields. In *Journal of the American Mathematical Society*, 2009.

7 Zeev Dvir, Swastik Kopparty, Shubhangi Saraf, and Madhu Sudan. Extensions to the Method of Multiplicities, with Applications to Kakeya Sets and Mergers. In *SIAM Journal on Computing*, 2013.

8 Zeev Dvir and Avi Wigderson. Kakeya Sets, New Mergers and Old Extractors. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2008.

9 Yuval Filmus, Lianna Hambardzumyan, Hamed Hatami, Pooya Hatami, and David Zuckerman. Biasing Boolean Functions and Collective Coin-Flipping Protocols over Arbitrary Product Distributions. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, 2019.

10 Ehud Friedgut. Influences in Product Spaces: KKL and BKKKL Revisited. *Comb. Probab. Comput*, 2004.

11 Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced Expanders and Randomness Extractors from Parvaresh–Vardy Codes*. In *Journal of the Association for Computing Machinery*, 2009.

12 J. Kahn, G. Kalai, and N. Linial. The influence of variables on Boolean functions. In *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science, (FOCS)*, 1988.

13 Swastik Kopparty and Vishvajeet N. Extracting mergers and projections of partitions, 2023. arXiv:2306.16915.

14 Lynn Harold Loomis and Hassler Whitney. An Inequality Related to the Isoperimetric Inequality. *Bull. AMS*, 1949.

15 Chi-Jen Lu, Omer Reingold, Salil Vadhan, and Avi Wigderson. Extractors: Optimal up to Constant Factors. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, 2003.

16 Raghu Meka. Explicit Resilient Functions Matching Ajtai-Linial. In *Proceedings of the 2017 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2017.

17 Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for Dispersers, Extractors, and Depth-Two Superconcentrators. In *SIAM Journal on Discrete Mathematics*, 2000.

18 Ran Raz. Extractors with Weak Random Seeds. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, 2005.

19 Amnon Ta-Shma. Refining Randomness. In *Thesis Submitted to the Hebrew University of Jerusalem*, 2000.

20 Amnon Ta-Shma and Christopher Umans. Better Condensers and New Extractors from Parvaresh-Vardy Codes. In *Proceedings of the 27th Conference on Computational Complexity (CCC)*, 2012.

21 Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 2012.

A Appendix

▶ **Theorem 23** (Restatement of Theorem 17)**.** *Let $A, B \subseteq [N]^3$ be a partition. Then one of the six 2-dimensional projections of A and B*

$$\pi_{XY}(A), \pi_{YZ}(A), \pi_{XZ}(A), \pi_{XY}(B), \pi_{YZ}(B), \pi_{XZ}(B)$$

has size at least $\frac{3}{4}N^2$.

Proof. Suppose $\pi_{XY}(A)$ and $\pi_{XY}(B)$ are both at most $\frac{3}{4}N^2$.

Fix $z \in [N]$. Let us consider the slice $S_z = [N]^2 \times \{z\}$, and focus on the X and Y projections of the sets $A \cap S_z$ and $B \cap S_z$ (so four projections in all, each being a subset of $[N]$).

Define[3]:

$$A_{Xz} = \{x \mid \forall y \in [N], (x, y, z) \in A\}.$$

[3] If A, B was merely a cover of $[N]^3$ rather than a partition, the correct definition would be

$$A_{Xz} = \{x \mid \nexists y \in [N] \text{ s.t. } (x, y, z) \in B\},$$

etc, and the rest of the proof would remain the same.

$B_{Xz} = \{x \mid \forall y \in [N], (x,y,z) \in B\}$.

$A_{Yz} = \{y \mid \forall x \in [N], (x,y,z) \in A\}$.

$B_{Yz} = \{y \mid \forall x \in [N], (x,y,z) \in B\}$.

Let $\alpha_{Xz}, \beta_{Xz}, \alpha_{Yz}, \beta_{Yz} \in [0,1]$ be their fractional sizes (= size divided by N).
Then we have the following:

- $A_{Xz} \cap B_{Xz} = \emptyset$ and $A_{Yz} \cap B_{Yz} = \emptyset$. Thus:

$$\alpha_{Xz} + \beta_{Xz} \leq 1, \tag{2}$$
$$\alpha_{Yz} + \beta_{Yz} \leq 1. \tag{3}$$

- $((A_{Xz} \times [N]) \cup ([N] \times A_{Yz})) \subseteq \pi_{XY}(A)$.
This is because any $(x,y) \in (A_{Xz} \times [N])$ has $(x,y,z) \in A$, and thus $(x,y) \in \pi_{XY}(A)$.
The fractional size of the left hand side is $1 - (1-\alpha_{Xz})(1-\alpha_{Yz})$, and the fractional size of the right hand side is $\leq 3/4$.
This gives us (after applying the AM-GM inequality[4]):

$$\alpha_{Xz} + \alpha_{Yz} \leq 1. \tag{4}$$

- Similarly,

$$((B_{Xz} \times [N]) \cup ([N] \times B_{Yz})) \subseteq \pi_{XY}(B),$$

$$\beta_{Xz} + \beta_{Yz} \leq 1. \tag{5}$$

- At most one of A_{Xz}, B_{Yz} can be nonempty, and at most one of $A_{Yz}, B_{X,z}$ can be nonempty. This is because $x \in A_{Xz}$ and $y \in B_{Yz}$ imply that $(x,y,z) \in A$ and $(x,y,z) \in B$ respectively. Thus at most one of α_{Xz}, β_{Yz}, and at most one of α_{Yz}, β_{Xz} can be nonzero.

Putting everything together, we get that only two of the four numbers $\alpha_{Xz}, \beta_{Xz}, \alpha_{Yz}, \beta_{Yz}$ can be nonzero, and furthermore, the sum of those two is bounded above by 1.

Therefore, for each $z \in [N]$,

$$\alpha_{Xz} + \beta_{Xz} + \alpha_{Yz} + \beta_{Yz} \leq 1.$$

Averaging in z, we get that

$$\mathbf{E}_z[\alpha_{Xz} + \beta_{Xz} + \alpha_{Yz} + \beta_{Yz}] \leq 1,$$

and thus one of the four numbers:

$$\mathbf{E}_z[\alpha_{Xz}], \mathbf{E}_z[\beta_{Xz}], \mathbf{E}_z[\alpha_{Yz}], \mathbf{E}_z[\beta_{Yz}]$$

is at most 1/4.

Finally, observe that $(1 - \mathbf{E}_z[\alpha_{Xz}])$ is the fractional size of $\pi_{XZ}(B)$ (and similarly for the other three numbers), and so one of the four projections

$$\pi_{XZ}(B), \pi_{XZ}(A), \pi_{YZ}(B), \pi_{YZ}(A)$$

has size at least $\frac{3}{4}N^2$. ◀

[4] For any non-negative real numbers x and y, $\sqrt{x \cdot y} \leq \frac{x+y}{2}$

▶ **Theorem 24** (Restatement of Theorem 21). *Let $A, B, C \subseteq [N]^3$ be a 3-partition of $[N]^3$. Then one of the nine 2-dimensional projections of A, B and C, i.e.,*

$\Pi_{XY}(A), \Pi_{XY}(B), \Pi_{XY}(C), \Pi_{YZ}(A), \Pi_{YZ}(B), \Pi_{YZ}(C), \Pi_{XZ}(A), \Pi_{XZ}(B), \Pi_{XZ}(C)$ *has size at least* $\eta_0 N^2$.

Before embarking on the proof of Theorem 21, we note down a simple set intersection lemma that will be useful.

▶ **Lemma 25** (Set intersection inequality). *Suppose U, V, W be arbitrary sets which have union equal to T.*

Then

$$|U| + |V| + |W| \geq 2|T| - (|U \setminus (V \cup W)| + |V \setminus (W \cup U)| + |W \setminus (U \cup V)|) + |U \cap V \cap W|.$$

This lemma gives a way to get a lower bound on the average size of three sets U, V, W that cover a set T by first proving an upper bound on the sizes of the "unique" parts $U \setminus (V \cup W), V \setminus (U \cup W), W \setminus (U \cup V)$. The proof is simple and omitted.

We now prove Theorem 21.

Proof. Consider any partition A, B and C of $[N]^3$ into 3 parts. Suppose $\Pi_{XY}(A)$, $\Pi_{XY}(B)$ and $\Pi_{XY}(C)$ are at most η_0. (If not we are done).

Fix $z \in [N]$.

Our first step is to consider the slice $S_z = [N]^2 \times \{z\}$, and focus on the X and Y projections of the 3 sets $A \cap S_z, B \cap S_z, C \cap S_z$ (so six projections in all, each being a subset of $[N]$).

Define:

$$A_{Xz} = \{x \in [N] \mid \exists y \in [N] \text{ s.t. } (x, y, z) \in A\}.$$

$$B_{Xz} = \{x \in [N] \mid \exists y \in [N] \text{ s.t. } (x, y, z) \in B\}.$$

$$C_{Xz} = \{x \in [N] \mid \exists y \in [N] \text{ s.t. } (x, y, z) \in C\}.$$

$$A_{Yz} = \{y \in [N] \mid \exists x \in [N] \text{ s.t. } (x, y, z) \in A\}.$$

$$B_{Yz} = \{y \in [N] \mid \exists x \in [N] \text{ s.t. } (x, y, z) \in B\}.$$

$$C_{Yz} = \{y \in [N] \mid \exists x \in [N] \text{ s.t. } (x, y, z) \in C\}.$$

Note that:

$$A_{Xz} \cup B_{Xz} \cup C_{Xz} = [N]$$

$$A_{Yz} \cup B_{Yz} \cup C_{Yz} = [N]$$

since A, B, C is a partition of $[N]^3$.

Next we identify the "pure" parts of these projections, defined below:

$$\widetilde{A}_{Xz} = A_{Xz} \setminus (B_{Xz} \cup C_{Xz})$$

$$\widetilde{B}_{Xz} = B_{Xz} \setminus (C_{Xz} \cup A_{Xz})$$

$$\widetilde{C}_{Xz} = C_{Xz} \setminus (A_{Xz} \cup B_{Xz})$$

$$\widetilde{A}_{Yz} = A_{Yz} \setminus (B_{Yz} \cup C_{Yz})$$

$$\widetilde{B}_{Yz} = B_{Yz} \setminus (C_{Yz} \cup A_{Yz})$$

$$\widetilde{C}_{Yz} = C_{Yz} \setminus (A_{Yz} \cup B_{Yz})$$

Furthermore, we have:

$$\{x \mid \Pi^{-1}_{XZ}(x, z) \subseteq A\} \subseteq \widetilde{A}_{Xz}$$

and five similar containments for $\widetilde{B}_{Xz}, \widetilde{C}_{Xz}, \widetilde{A}_{Yz}, \widetilde{B}_{Yz}, \widetilde{C}_{Yz}$.

Let $\widetilde{\alpha}_{Xz}, \widetilde{\beta}_{Xz}, \widetilde{\gamma}_{Xz}, \widetilde{\alpha}_{Yz}, \widetilde{\beta}_{Yz}, \widetilde{\gamma}_{Yz} \in [0, 1]$ be their fractional sizes.

Note that since the corresponding sets are disjoint, we have:

$$\widetilde{\alpha}_{Xz} + \widetilde{\beta}_{Xz} + \widetilde{\gamma}_{Xz} \le 1 \tag{6}$$

$$\widetilde{\alpha}_{Yz} + \widetilde{\beta}_{Yz} + \widetilde{\gamma}_{Zz} \le 1 \tag{7}$$

▶ **Lemma 26.** *For any $z \in [N]$, out of the 6 variables $\widetilde{\alpha}_{Xz}, \widetilde{\beta}_{Xz}, \widetilde{\gamma}_{Xz}, \widetilde{\alpha}_{Yz}, \widetilde{\beta}_{Yz}, \widetilde{\gamma}_{Yz}$, let H be the set of those variables that are nonzero. Then H is a subset of at least one of the following sets of variables:*

$$\{\widetilde{\alpha}_{Xz}, \widetilde{\alpha}_{Yz}\}, \{\widetilde{\beta}_{Xz}, \widetilde{\beta}_{Yz}\}, \{\widetilde{\gamma}_{Xz}, \widetilde{\gamma}_{Yz}\}, \{\widetilde{\alpha}_{Xz}, \widetilde{\beta}_{Xz}, \widetilde{\gamma}_{Xz}\}, \{\widetilde{\alpha}_{Yz}, \widetilde{\beta}_{Yz}, \widetilde{\gamma}_{Yz}\}$$

Proof. It is a consequence of the easy observation that $\widetilde{\alpha}_{Xz}$ and $\widetilde{\beta}_{Yz}$ cannot both be nonzero (and 5 similar easy observations). ◀

Let

$$\delta_{Xz} = \begin{cases} 1 & \widetilde{\alpha}_{Yz}, \widetilde{\beta}_{Yz}, \widetilde{\gamma}_{Yz} > 0 \\ 0 & \text{otherwise} \end{cases}.$$

$$\delta_{Yz} = \begin{cases} 1 & \widetilde{\alpha}_{Xz}, \widetilde{\beta}_{Xz}, \widetilde{\gamma}_{Xz} > 0 \\ 0 & \text{otherwise} \end{cases}.$$

Note that δ_{Xz} depends on the projections in the Y direction (and vice versa). The reason for this definition is the following observation: if $\delta_{Xz} = 1$, then we have

$$A_{Xz} \cap B_{Xz} \cap C_{Xz} = A_{Xz} = B_{Xz} = C_{Xz} = [N], \tag{8}$$

and similarly, if $\delta_{Yz} = 1$, then we have

$$A_{Yz} \cap B_{Yz} \cap C_{Yz} = A_{Yz} = B_{Yz} = C_{Yz} = [N], \tag{9}$$

which is something that our set intersection lemma can exploit.

Define

$$\lambda_{Xz} = \widetilde{\alpha}_{Xz} + \widetilde{\beta}_{Xz} + \widetilde{\gamma}_{Xz} - \delta_{Xz},$$

$$\lambda_{Yz} = \widetilde{\alpha}_{Yz} + \widetilde{\beta}_{Yz} + \widetilde{\gamma}_{Yz} - \delta_{Yz}.$$

$$\lambda_z = \lambda_{Xz} + \lambda_{Yz}.$$

Note that by Equations (6), (7), for all z,

$$\lambda_{Xz} \le 1. \tag{10}$$

$$\lambda_{Yz} \le 1. \tag{11}$$

By the set intersection lemma,

$$|A_{Xz}| + |B_{Xz}| + |C_{Xz}| \geq 2N - \left(|\widetilde{A}_{Xz}| + |\widetilde{B}_{Xz}| + |\widetilde{C}_{Xz}|\right) + |A_{Xz} \cap B_{Xz} \cap C_{Xz}|$$
$$\geq (2 - \lambda_{Xz})N$$

Similarly,

$$|A_{Yz}| + |B_{Yz}| + |C_{Yz}| \geq (2 - \lambda_{Yz})N$$

Summing over $z \in [N]$ and adding these two equations, we get:

$$\Pi_{XZ}(A) + \Pi_{XZ}(B) + \Pi_{XZ}(C) \geq (2 - \mathbf{E}_z[\lambda_{Xz}])\, N^2, \tag{12}$$
$$\Pi_{YZ}(A) + \Pi_{YZ}(B) + \Pi_{YZ}(C) \geq (2 - \mathbf{E}_z[\lambda_{Yz}])\, N^2, \tag{13}$$
$$\Pi_{XZ}(A) + \Pi_{XZ}(B) + \Pi_{XZ}(C) + \Pi_{YZ}(A) + \Pi_{YZ}(B) + \Pi_{YZ}(C) \geq (4 - \mathbf{E}_z[\lambda_z])\, N^2. \tag{14}$$

Our goal is now to get an upper bound on $\mathbf{E}_z[\lambda_z]$.

To get our main result, we will show that $\mathbf{E}_z[\lambda_z] \leq \lambda^* := 4 - 6\eta_0 \approx 0.856$ (or else we find a large projection in some other way). This will show that one of the 6 projections on the left hand side is at least $\eta_0 N^2$, as desired.

If we just want to get a projection of size $\geq \frac{1}{2}N^2$, then it suffices to show that $\lambda^* \leq 1$, and this turns out to be simpler.

Towards that end, we define α_X to be the fraction of x for which $\{x\} \times [N] \subseteq \Pi_{XY}(A)$. Similarly define $\alpha_Y, \beta_X, \beta_Y, \gamma_X, \gamma_Y$.

Note that since $\widetilde{A}_{Xz} \times [N] \times \{z\} \subseteq A$, we have:

$$\alpha_{Xz} \leq \alpha_X,$$

and 5 similar inequalities.

Note that $\alpha_X \leq \eta$, and 5 similar inequalities.

Define $u : [0,1] \to [0,2]$ by:

$$u(a) = 2 - 2\sqrt{1-a}.$$

Using the argument used to arrive at Equation (4) (by the AM-GM inequality), we have

$$\widetilde{\alpha}_{Xz} + \widetilde{\alpha}_{Yz} \leq u(\eta_0) \qquad (\text{ and thus } \alpha_X + \alpha_Y \leq u(\eta_0)\ ).$$

and 2 similar pairs of inequalities.

Let $g_X = \max\{\alpha_X + \beta_X, \beta_X + \gamma_X, \alpha_X + \gamma_X\}$., similarly q_Y.

By the inequalities above, we have: $g_X + g_Y \leq 2\eta_0 + u(\eta_0)$.

Now let $q_X = \Pr_{z \in [n]}[\text{exactly two of } \alpha_{Xz}, \beta_{Xz}, \gamma_{Xz} \text{ are nonzero}]$, similarly define q_Y.

$$q = \Pr_{z \in [n]}[\text{at most one of } \alpha_{Xz}, \beta_{Xz}, \gamma_{Xz} \text{ and at most one of } \alpha_{Yz}, \beta_{Yz}, \gamma_{Yz} \text{ is nonzero}]$$

We are now in a position to state a key lemma which will prove our lower bound:

▶ **Lemma 27.**

$$\mathbf{E}_z[\lambda_z] \leq q \cdot u(\eta_0) + q_X \cdot \min(g_X, 1) + q_Y \cdot \min(g_Y, 1)).$$

Proof. Let $z \in [N]$. We take cases on which of the 6 numbers $\widetilde{\alpha}_{Xz}, \widetilde{\beta}_{Xz}, \widetilde{\gamma}_{Xz}, \widetilde{\alpha}_{Yz}, \widetilde{\beta}_{Yz}, \widetilde{\gamma}_{Yz}$ are nonzero. By Lemma 26, there only a few cases to consider.

- If the first three numbers are nonzero or the second three numbers are nonzero, then λ_z is nonpositive because the sum of those three is at most 1 (by Equations (6), (7), and $\delta_{Xz} = 1$ or $\delta_{Yz} = 1$.
- If exactly two of the first three numbers are nonzero, then λ_z is at most $\min(g_X, 1)$. This happens for q_X fraction of the z's.
- If exactly two of the second three numbers are nonzero, then λ_z is at most $\min(g_Y, 1)$. This happens for q_Y fraction of the z's
- If at most one of the first three numbers and at most one of the second three numbers is nonzero, then λ_z is at most $2 - 2\sqrt{1-\eta_0}$. This happens for q fraction of the z's. ◀

Now $q+q_X+q_Y \leq 1$. At this point, we already see that $\mathbf{E}_z[\lambda_z] \leq 1$ (since $u(\eta_0) \approx 0.6237 \leq 1$), and this gives us the result that some projection has size at least $\frac{1}{2}N^2$.

To get our improved bound of $\eta_0 N^2$, we need one more idea.

▶ **Lemma 28.** $q_X + \mathbf{E}_z[\lambda_{Yz}] \leq 1$ *and* $q_Y + \mathbf{E}_z[\lambda_{Xz}] \leq 1$

Proof. We prove the first inequality, the second being similar. q_X is the fraction of z for which exactly two of $\{\widetilde{\alpha}_{Xz}, \widetilde{\beta}_{Xz}, \widetilde{\gamma}_{Xz}\}$ are nonzero. For such a z, we have $\widetilde{\alpha}_{Yz} = \widetilde{\beta}_{Yz} = \widetilde{\gamma}_{Yz} = \delta_{Yz} = 0$, and thus $\lambda_{Yz} = 0$. Along with Equations (10), (11), this completes the proof. ◀

By Equation (12), if $\mathbf{E}_z[\lambda_{Xz}]$ is at most $2 - 3\eta_0$, then we get a projection onto the XZ plane of size at least $\eta_0 N^2$, and we are done. Similarly, by Equation (13), if $\mathbf{E}_z[\lambda_{Yz}]$ is at most $2 - 3\eta_0$, then we get a projection onto the YZ plane of size at least $\eta_0 N^2$, and we are done. Thus we may assume that both $\mathbf{E}_z[\lambda_{Xz}]$ and $\mathbf{E}_z[\lambda_{Yz}]$ are at least $2 - 3\eta_0$.

By the previous lemma, we thus get that $q_X, q_Y \leq 3\eta_0 - 1$. Summarizing everything we know: $g_X + g_Y \leq 2\eta_0 + u(\eta_0)$, $q_X, q_Y \leq 3\eta_0 - 1$. and $q + q_X + q_Y \leq 1$. Under these constraints, we claim that: $q \cdot u(\eta_0) + q_X \cdot \min(g_X, 1) + q_Y \cdot \min(g_Y, 1) \leq \lambda^*$. By inspection, we see that the LHS is maximized when:

$$g_X = 1, q_X = 3\eta_0 - 1, q = 1 - q_X = 2 - 3\eta_0,$$

which makes it evaluate to:

$$(2 - 3\eta_0) \cdot u(\eta_0) + (3\eta_0 - 1) = \frac{1}{6}(4 - \eta_0) = \lambda^*,$$

where the first equality is the defining equation of η_0, and the second equality is the definition of λ^*. This completes the proof. ◀

Rapid Mixing of Global Markov Chains via Spectral Independence: The Unbounded Degree Case

Antonio Blanca
Pennsylvania State University, University Park, PA, USA

Xusheng Zhang
Pennsylvania State University, University Park, PA, USA

Abstract

We consider spin systems on general n-vertex graphs of unbounded degree and explore the effects of spectral independence on the rate of convergence to equilibrium of global Markov chains. Spectral independence is a novel way of quantifying the decay of correlations in spin system models, which has significantly advanced the study of Markov chains for spin systems. We prove that whenever spectral independence holds, the popular Swendsen–Wang dynamics for the q-state ferromagnetic Potts model on graphs of maximum degree Δ, where Δ is allowed to grow with n, converges in $O((\Delta \log n)^c)$ steps where $c > 0$ is a constant independent of Δ and n. We also show a similar mixing time bound for the block dynamics of general spin systems, again assuming that spectral independence holds. Finally, for *monotone* spin systems such as the Ising model and the hardcore model on bipartite graphs, we show that spectral independence implies that the mixing time of the systematic scan dynamics is $O(\Delta^c \log n)$ for a constant $c > 0$ independent of Δ and n. Systematic scan dynamics are widely popular but are notoriously difficult to analyze. This result implies optimal $O(\log n)$ mixing time bounds for any systematic scan dynamics of the ferromagnetic Ising model on general graphs up to the tree uniqueness threshold. Our main technical contribution is an improved factorization of the entropy functional: this is the common starting point for all our proofs. Specifically, we establish the so-called k-partite factorization of entropy with a constant that depends polynomially on the maximum degree of the graph.

2012 ACM Subject Classification Theory of computation → Random walks and Markov chains; Mathematics of computing → Gibbs sampling

Keywords and phrases Markov chains, Spectral Independence, Potts model, Mixing Time

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.53

Category RANDOM

Related Version *Full Version*: https://arxiv.org/abs/2307.00683

Funding The research is supported in part by NSF grant CCF-2143762.

Acknowledgements The authors thank the anonymous referees for their constructive comments during the manuscript review process.

1 Introduction

Spectral independence is a powerful new approach for quantifying the decay of correlations in spin system models. Initially introduced in [4], this condition has revolutionized the study of Markov chains for spin systems. In a series of important and recent contributions, spectral independence has been shown to be instrumental in determining the convergence rate of the Glauber dynamics, the simple single-site update Markov chain that updates the spin at a randomly chosen vertex in each step.

The first efforts in this series (see [4, 24, 25]) showed that spectral independence implies optimal $O(n \log n)$ mixing of the Glauber dynamics on n-vertex graphs of bounded degree for general spin systems. The unbounded degree case was studied in [20, 19, 3, 44], while [6]

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 53; pp. 53:1–53:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

explored the effects of this condition on the speed of convergence of global Markov chains (i.e., Markov chains that update the spins of a large number of vertices in each step) in the bounded degree setting. Research exploring the applications of spectral independence is ongoing. We contribute to this line of work by investigating how spectral independence affects the speed of convergence of *global Markov chains* for general spin systems on graphs of unbounded degree.

A *spin system* is defined on a graph $G = (V, E)$. There is a set $\mathcal{S} = \{1, \dots, q\}$ of spins or colors, and configurations are assignments of spin values from $\mathcal{S}$ to each vertex of G. The probability of a configuration $\sigma \in \mathcal{S}^V$ is given by the Gibbs distribution:

$$\mu(\sigma) = \frac{e^{-H(\sigma)}}{Z}, \tag{1}$$

where the normalizing factor Z is known as the partition function, and the Hamiltonian $H : \mathcal{S}^V \to \mathbb{R}$ contains terms that depend on the spin values at each vertex (a "vertex potential" or "external field") and at each pair of adjacent vertices (an "edge potential"); see Definition 24. A widely studied spin system, and one that we will pay close attention to in this paper, is the ferromagnetic Potts model, where for a real parameter $\beta > 0$, associated with inverse temperature in physical applications, the Hamiltonian is given by:

$$H(\sigma) = -\beta \sum_{\{u,v\}\in E} \mathbb{1}(\sigma_u = \sigma_v).$$

The classical ferromagnetic Ising model corresponds to the $q = 2$ case. (In this variant of the Potts model, the Hamiltonian only includes edge potentials, and there is no external field.) We shall use μ_{Ising} and μ_{Potts} for the Gibbs distributions corresponding to the Ising and Potts models. Other well-known, well-studied spin systems include uniform proper colorings and the hardcore model.

Spin systems provide a robust framework for studying interacting systems of simple elements and have a wide range of applications in computer science, statistical physics, and other fields. In such applications, generating samples from the Gibbs distribution (1) is a fundamental computational task and one in which Markov chain-based algorithms have been quite successful. A long line of work dating back to the 1980s relates the speed of convergence of Markov chains to various forms of decay of correlations in the model. Spectral independence, defined next, captures the decay of correlations in a novel way.

Roughly speaking, spectral independence holds when the spectral norm of a "pairwise" influence matrix is bounded. To formally define it, let us begin by introducing some notations. Let $\Omega \subseteq \mathcal{S}^V$ be the support of μ: the set of configurations σ such that $\mu(\sigma) > 0$. A *pinning* τ on a subset of vertices $\Lambda \subseteq V$ is a fixed partial configuration on Λ; i.e., a spin assignment from $\mathcal{S}^\Lambda$ to the vertices of Λ. For a pinning τ on $\Lambda \subseteq V$ and $U \subseteq V \setminus \Lambda$, we let $\Omega_U^\tau = \{\sigma_U \in \mathcal{S}^U : \mu(\sigma_U \mid \sigma_\Lambda = \tau) > 0\}$ be the set of partial configurations on U that are consistent with the pinning τ. We write $\Omega_u^\tau = \Omega_{\{u\}}^\tau$ if u is a single vertex. Let

$$\mathcal{P}^\tau := \{(u, s) : u \notin \Lambda, s \in \Omega_u^\tau\}$$

denote the set of consistent vertex-spin pairs in $\Omega_{V\setminus\Lambda}^\tau$ under μ. For each $\Lambda \subseteq V$ and pinning τ on Λ, we define the *signed pairwise influence matrix* $\Psi_\mu^\tau \in \mathbb{R}^{\mathcal{P}^\tau \times \mathcal{P}^\tau}$ to be the matrix with entries:

$$\Psi_\mu^\tau((u, a), (v, b)) = \mu(\sigma_v = b \mid \sigma_u = a, \sigma_\Lambda = \tau) - \mu(\sigma_v = b \mid \sigma_\Lambda = \tau)$$

for $u \neq v$, and $\Psi_\mu^\tau((u, a), (u, b)) = 0$ otherwise.

▶ **Definition 1** (Spectral Independence). *A distribution μ satisfies η-*spectral independence *if for every subset of vertices $\Lambda \subseteq V$ and every pinning $\tau \in \Omega_\Lambda$, the largest eigenvalue of the signed pairwise influence matrix Ψ_μ^τ, denoted $\lambda_1(\Psi_\mu^\tau)$, satisfies $\lambda_1(\Psi_\mu^\tau) \leq \eta$.*

There are several definitions of spectral independence in the literature; we use here the one from [22].

We show that spectral independence implies new upper bounds on the mixing time of several well-studied global Markov chains in the case where the maximum degree Δ of the underlying graph $G = (V, E)$ is unbounded; i.e., $\Delta \to \infty$ with n. The mixing time is defined as the number of steps required for a Markov chain to reach a distribution close in total variation distance to its stationary distribution, assuming a worst possible starting state; a formal definition is given in Section 2. The global Markov chains we consider include the Swendsen–Wang dynamics for the ferromagnetic q-state Potts, the systematic scan dynamics for monotone spin systems, and the block dynamics for general spin systems. These three dynamics are among the most popular and well-studied global Markov chains and present certain advantages (e.g., faster convergence and amenability to parallelization) to the Glauber dynamics.

1.1 The Swendsen–Wang dynamics

A canonical example of a global Markov chain is the Swendsen–Wang (SW) dynamics for the ferromagnetic q-state Potts model. The SW dynamics transitions from a configuration σ_t to σ_{t+1} by:

1. For each edge $e = \{u, v\} \in E$, if $\sigma_t(u) = \sigma_t(v)$, independently include e in the set A_t with probability $p = 1 - e^{-\beta}$;
2. Then, independently for each connected component $\mathcal{C}$ in (V, A_t), draw a spin $s \in \{1, \dots, q\}$ uniformly at random and set $\sigma_{t+1}(v) = s$ for all $v \in \mathcal{C}$.

The SW dynamics is ergodic and reversible with respect to μ_{Potts} and thus converges to it. This Markov chain originated in the late 1980s [53] as an alternative to the Glauber dynamics, which mixes exponentially slowly at low temperatures (large β). The SW dynamics bypasses key barriers that cause the slowdown of the Glauber dynamics at low temperatures. For the Ising model ($q = 2$), for instance, it was recently shown to converge in poly(n) steps on any n-vertex graph for any value of $\beta > 0$ [39]. (The conjectured mixing time is $\Theta(n^{1/4})$, but we seem to be far from proving such a conjecture.) For $q \geq 3$, on the other hand, the SW dynamics can converge exponentially slowly at certain "intermediate" temperatures regimes corresponding to first-order phase transitions; see [38, 15, 36, 37, 26].

Recently, η-spectral independence (with $\eta = O(1)$) was shown to imply that the mixing time of the SW dynamics is $O(\log n)$ on graphs of maximum degree $\Delta = O(1)$, i.e., bounded degree graphs [6]. This mixing time bound is optimal since the SW dynamics requires $\Omega(\log n)$ steps to mix in some cases where η and Δ are both $O(1)$ [7, 9]. However, it does not extend to the unbounded degree setting since the constant factor hidden by the big-O notation depends exponentially on the maximum degree Δ; this is the case even when $\eta = O(1)$ and $\beta\Delta = O(1)$. Our first result provides a mixing time bound that depends only polynomially on Δ.

▶ **Theorem 2.** *Let $q \geq 2$, $\beta > 0$, $\eta > 0$ and $\Delta \geq 3$. Suppose $G = (V, E)$ is an n-vertex graph of maximum degree Δ. Let μ_{Potts} be the Gibbs distribution of the q-state ferromagnetic Potts model on G with parameter β. If μ_{Potts} is η-spectrally independent with $\eta = O(1)$ and $\beta\Delta = O(1)$, then there exists a constant $c > 0$ such that the mixing time of the SW dynamics satisfies $T_{mix}(P_{SW}) = O\big((\Delta \log n)^c\big)$.*

The constant c has a near linear dependency on η and $\beta\Delta$; a more precise statement of Theorem 2 with a precise expression for c is given in Theorem 11.

Despite the expectation that the SW dynamics mixes in $O(\log n)$ steps in weakly correlated systems (i.e., when $\beta\Delta$ is small), proving sub-linear upper bounds on its mixing time has been difficult. Recently, various forms of decay of correlation (e.g., strong spatial mixing, entropy mixing, and spectral independence) have been used to obtain $O(\log n)$ bounds for the mixing time of the SW dynamics on cubes of the integer lattice graph $\mathbb{Z}^d$, regular trees, and general graphs of bounded degree (see [7, 9, 6]). However, for graphs of large degree, i.e., with $\Delta \to \infty$ with n, the only sub-linear mixing time bounds known either hold for the very distinctive mean-field model, where G is the complete graph [35, 11], or hold for very small values of β; i.e., $\beta \lesssim 1/(3\Delta)$ [43]. Our results provide new sub-linear mixing time bounds for graph families of sub-linear maximum degree, provided $\eta = O(1)$ and $\beta\Delta = O(1)$. These last two conditions go hand-in-hand: in all known cases where $\eta = O(1)$, we also have $\beta\Delta = O(1)$.

On graphs of degree at most Δ, η-spectral independence is supposed to hold with $\eta = O(1)$ whenever $\beta < \beta_u(q,\Delta)$, where $\beta_u(q,\Delta)$ is the threshold for the uniqueness/non-uniqueness phase transition on Δ-regular trees. This has been confirmed for the Ising model ($q = 2$) but not for the Potts model. Specifically, for the ferromagnetic Ising model, we have $\beta_u(2,\Delta) = \ln \frac{\Delta}{\Delta-2}$, and when $\beta \le (1-\delta)\beta_u(2,\Delta)$ for some $\delta \in (0,1)$, μ_{Ising} is η-spectrally independent with $\eta = O(1/\delta)$; see [24, 25]. In contrast, for the ferromagnetic Potts model with $q \ge 3$, there is no closed-form expression for $\beta_u(q,\Delta)$ (it is defined as the threshold value where an equation starts to have a double root), and for graphs of unbounded degree η-spectral independence is only known to hold when $\beta \le \frac{2(1-\delta)}{\Delta}$. As a result, we obtain the following corollary of Theorem 2.

► **Corollary 3.** *Let $\delta \in (0,1)$, $\Delta \ge 3$. Suppose that either $q = 2$ and $\beta < (1-\delta)\beta_u(2,\Delta)$, or $q \ge 3$ and $\beta \le \frac{2(1-\delta)}{\Delta}$. Then, there exists a constant $c = c(\delta) > 0$ such that the mixing time of the SW dynamics for the q-state ferromagnetic Potts model on any n-vertex graph of maximum degree Δ satisfies $T_{mix}(P_{SW}) = O\big((\Delta \log n)^c\big)$.*

We mention that other conditions known to imply spectral independence (e.g., those in [14]) are not well-suited for the unbounded degree setting since under those conditions, the best known bound for η depends polynomially on Δ. For another application of Theorem 2, see Section 3.3.1 where we provide a bound on the mixing of the SW dynamics on random graphs.

We comment briefly on our proof approach for Theorem 2. A mixing time bound for the SW dynamics can be deduced from the so-called *edge-spin* factorization of the entropy functional introduced in [7]. It was noted there that this factorization, in turn, follows from a different factorization of entropy known as *k-partite factorization*, or KPF. Spectral independence is known to imply KPF but with a loss of a multiplicative constant that depends exponentially on the maximum degree of the graph. Our proof of Theorem 2 follows this existing framework, but pays closer attention to establishing KPF with an optimized constant with a better dependence on the model parameters. This is done through a multi-scale analysis of the entropy functional; in each scale, we apply spectral independence to achieve a tighter KPF condition. Our new results for KPF not only hold for the Potts model, but also for a general class of spin systems, and we use it to establish new mixing time bounds for the systematic scan dynamics and block dynamics.

1.2 The systematic scan dynamics

Our next contribution pertains the *systematic scan dynamics*, which is a family of Markov chains closely related to the Glauber dynamics in the sense that updates occur at single vertices sequentially. The key difference is that the vertex updates happen according to a predetermined ordering ϕ of the vertices instead of at random vertices. These dynamics offer practical advantages since there is no need to randomly select vertices at each step, thereby reducing computation time.

There is a folklore belief that the mixing time of the systematic scan dynamics (properly scaled) is closely related to that of the Glauber dynamics. However, analyzing this type of dynamics has proven very challenging (see, e.g., [28, 41, 30, 29, 49, 40, 8]), and the best general condition under which the systematic scan dynamics is known to be optimally mixing is a Dobrushin-type condition due to Dyer, Goldberg, and Jerrum [30]. The new developments on Markov chain mixing stemming from spectral independence have not yet provided new results for this dynamics, even for the bounded degree case where much progress has already been made. We show that spectral independence implies optimal mixing of the systematic scan dynamics for *monotone* spin systems with *bounded marginals*; we define both of these notions next.

▶ **Definition 4** (Monotone spin system). *In a monotone system, there is a linear ordering of the spins at each vertex which induces a partial order $\preceq_q$ over the state space. A spin system is* monotone *with respect to the partial order $\preceq_q$ if for every $\Lambda \subseteq V$ and every pair of pinnings $\tau_1 \succeq_q \tau_2$ on $V \setminus \Lambda$, the conditional distribution $\mu(\cdot \mid \sigma_\Lambda = \tau_1)$ stochastically dominates $\mu(\cdot \mid \sigma_\Lambda = \tau_2)$.*

Canonical examples of monotone spin systems include the ferromagnetic Ising model and the hardcore model on bipartite graphs. As in earlier work (see [24, 25, 6]), our bounds on the mixing time will depend on a lower bound on the marginal probability of any vertex-spin pair. This is formalized as follows.

▶ **Definition 5** (Bounded marginals). *The distribution μ is said to be b-*marginally bounded *if for every $\Lambda \subseteq V$ and pinning $\tau \in \Omega_\Lambda$, and each $(v, s) \in \mathcal{P}^\tau$, we have $\mu(\sigma_v = s \mid \sigma_\Lambda = \tau) \geq b$.*

Before stating our result for the systematic scan dynamics of b-marginally bounded monotone spin systems, we note that this Markov chain updates in a single step each vertex once in the order prescribed by ϕ. Under a minimal assumption on the spin system (the same one required to ensure the ergodicity of the Glauber dynamics), the systematic scan dynamics is ergodic. Specifically, when the spin system is totally-connected (see Definition 25), the systematic scan dynamics is ergodic. Moreover, the systematic scan dynamics is not necessarily reversible with respect to μ, so, as in earlier works, we work with the symmetrized version of the dynamics in which, in each step, the vertices are updated according to ϕ first, and subsequently in the reverse order of ϕ. The resulting dynamics, which we denote by P_ϕ, is reversible with respect to μ. Our main result for the systematic scan dynamics is the following.

▶ **Theorem 6.** *Let $b > 0$, $\eta > 0$, and $\Delta \geq 3$. Suppose $G = (V, E)$ is an n-vertex graph of maximum degree Δ. Let μ be the distribution of a totally-connected monotone spin system on G. If μ is η-spectrally independent and b-marginally bounded, then for any ordering ϕ,*

$$T_{mix}(P_\phi) = \left(\frac{e^2\Delta}{b}\right)^{9+4\lceil \frac{2\eta}{b} \rceil} \cdot O(\log n).$$

The bound in this theorem is tight: for a particular ordering ϕ, we prove an $\Omega(\log n)$ mixing time lower bound that applies to settings where Δ, b and η are all $\Theta(1)$; see Lemma 26.

We present next several interesting consequences of Theorem 6. First, we obtain the following corollary using the known results about spectral independence for the ferromagnetic Ising model.

▶ **Corollary 7.** *Let $\delta \in (0,1), \Delta \geq 3$ and $0 < \beta < (1-\delta)\beta_u(2,\Delta)$. Suppose $G = (V,E)$ is an n-vertex graph of maximum degree Δ. For any ordering ϕ of the vertices of G, the mixing time of P_ϕ for the Ising model on G with parameter β satisfies $T_{mix}(P_\phi) = O(\log n)$.*

The constant hidden by the big-O notation is an absolute constant that depends only on the constant δ, even when Δ depends on n. This result, compared to the earlier conditions in [28, 41, 30], extends the parameter regime where the $O(\log n)$ mixing time bound applies; in fact, the parameter regime in Corollary 7 is tight, as the systematic scan dynamics undergoes an exponential slowdown when $\beta > \beta_u(2,\Delta)$ [49]. We also derive analogous results for the hardcore model on bipartite graphs; see Section 4.1.

Our next application concerns the specific but relevant case where the underlying graph is an n-vertex cube of the integer lattice graph $\mathbb{Z}^d$. In this context, it was proved in [8] that all systematic scan dynamics converge in $O(\log n(\log\log n)^2)$ steps whenever a well-known condition known as *strong spatial mixing (SSM)* holds. A pertinent open question is whether SSM implies spectral independence. In fact, spectral independence is often proved by adapting earlier arguments for establishing SSM (see, e.g., [4, 24]). Recently, it was proved in [23] that SSM on trees implies spectral independence on large-girth graphs. We show that for *general* spin systems on $\mathbb{Z}^d$, SSM implies η-spectral independence with $\eta = O(1)$.

▶ **Lemma 8.** *For a spin system on a d-dimensional cube $V \subseteq \mathbb{Z}^d$, SSM implies η-spectral independence, where $\eta = O(1)$.*

The formal definition of SSM is given later in Section 4. Lemma 8 does not assume monotonicity for the spin system and could be of independent interest. An interesting consequence of this lemma, when combined with Theorem 6 is the following.

▶ **Corollary 9.** *Let $d \geq 2$. For a b-marginally bounded monotone spin system on a d-dimensional cube $V \subseteq \mathbb{Z}^d$, SSM implies that the mixing time of any systematic scan P_ϕ is $O(\log n)$.*

For the ferromagnetic Ising model on $\mathbb{Z}^2$, SSM is known to hold for all $\beta < \beta_c(2) = \ln(1+\sqrt{2})$ (see [17, 45, 2, 5]), so by Corollary 9 we deduce that when $\beta < \beta_c(2)$, the mixing time of any systematic scan P_ϕ on an n-vertex square box of $\mathbb{Z}^2$ is $O(\log n)$; note that $\beta_c(2) > \beta_u(2,2d)$, the corresponding tree uniqueness threshold.

We comment briefly on the techniques used to establish our results for the systematic scan dynamics. Our starting point is again the k-partite factorization of entropy (KPF). Our improved bounds for KPF imply that a global Markov chain that updates a random independent set of vertices in each step is rapidly mixing. We then use the censoring technique from [34, 10] to relate the mixing time of this Markov chain to that of the systematic scan dynamics. To establish Lemma 8, we use SSM to construct a contractive coupling for a particular Markov chain. Our Markov chain is similar to the one from [31], but modified to update rectangles instead of balls, and thus match the variant of SSM that holds up to the critical threshold for the Ising model on $\mathbb{Z}^2$. This contractive coupling is then used to establish spectral independence using the machinery from [6].

1.3 The block dynamics

Our final result concerns a family of Markov chains known as the *block dynamics*. They are a natural generalization of the Glauber dynamics where a random subset of vertices (instead of a random vertex) is updated in each step. More precisely, let $\mathcal{B} := \{B_1, \dots, B_K\}$ be a collection of subsets of vertices (called blocks) such that $V = \cup_{i=1}^K B_i$. Let α be a distribution over $\mathcal{B}$. The *(heat-bath) block dynamics* with respect to $(\mathcal{B}, \alpha)$ is the Markov chain that, in each step, given a spin configuration σ_t, selects $B_i \in \mathcal{B}$ according to the distribution α and updates the configuration on B_i with a sample from the $\mu(\cdot \mid \sigma_t(V \setminus B_i))$; that is, from the conditional distribution on B_i given the spins of σ_t in $V \setminus B_i$. We denote this Markov chain (and its transition matrix) by $P_{\mathcal{B},\alpha}$. When the B_i's are each single vertices, and α is a uniform distribution over the blocks in $\mathcal{B}$, we obtain the Glauber dynamics. Our result for the mixing time of the block dynamics is the following.

▶ **Theorem 10.** *Let $b > 0$, $\eta > 0$ and $\Delta \geq 3$. Suppose $G = (V, E)$ is an n-vertex graph of maximum degree Δ. Let μ be a Gibbs distribution of a totally-connected spin system on G. Let $\mathcal{B} := \{B_1, \dots, B_K\}$ be any collection of blocks such that $V = \cup_{i=1}^K B_i$, and let α be a distribution over $\mathcal{B}$. If μ is η-spectrally independent and b-marginally bounded, then there exists a constant $C > 0$ such that the mixing time of block dynamics $P_{\mathcal{B},\alpha}$ satisfies:*

$$T_{mix}(P_{\mathcal{B},\alpha}) = O\Big(\alpha_{min}^{-1} \cdot \Big(\frac{C\Delta \log n \log\log n}{b^7}\Big)^{2+\lceil \frac{2\eta}{b} \rceil}\Big),$$

where $\alpha_{min} = \min_{v \in V} \sum_{B \in \mathcal{B}} \alpha_B$.

Previous results for the block dynamics only apply to the bounded degree case [9, 17, 6], so Theorem 10 provides the first bounds for its mixing time in the unbounded degree setting.

Organization. The rest of the paper is organized as follows. In Section 2, we provide a number of definitions and background results. In Sections 3 and 4, we provide proof sketches for our results for the SW dynamics and the systematic scan dynamics; that is, Theorems 2 and 6, respectively. Some of our proofs are deferred to the full version of the paper [12].

2 Mixing times and modified log-Sobolev inequalities

Let P be an irreducible and aperiodic (i.e., ergodic) Markov chain with state space Ω and stationary distribution μ. Let us assume that P is reversible with respect to μ, and let

$$d(t) := \max_{x \in \Omega} \|P^t(x, \cdot) - \mu\|_{TV} := \max_{x \in \Omega} \max_{A \subseteq \Omega} |P^t(x, A) - \mu(A)|,$$

where $P^t(x, \cdot)$ denotes the distribution of the chain at time t assuming $x \in \Omega$ as the starting state; $\|\cdot\|_{TV}$ denotes the total variation distance. Note that with a slight abuse of notation we use P for both the Markov chain and its transition matrix. For $\varepsilon > 0$, let

$$T_{mix}(P, \varepsilon) := \min\{t > 0 : d(t) \leq \varepsilon\},$$

and the *mixing time of* P is defined as $T_{mix}(P) = T_{mix}(P, 1/4)$.

For functions $f, g : \Omega \to \mathbb{R}$, the *Dirichlet form* of a reversible Markov chain P with stationary distribution μ is defined as

$$\mathcal{E}_P(f, g) = \langle f, (I - P)g \rangle_\mu = \frac{1}{2} \sum_{x, y \in \Omega} \mu(x)P(x, y)(f(x) - f(y))(g(x) - g(y)),$$

where $\langle f, g \rangle_\mu := \sum_{x \in \Omega} f(x)g(x)\mu(x)$.

The spectrum of the ergodic and reversible Markov chain P is real, and we let $1 = \lambda_1 > \lambda_2 \geq \cdots \geq \lambda_{|\Omega|} \geq -1$ denote its eigenvalues. The (absolute) spectral gap of P is defined by $\mathrm{GAP}(P) = 1 - \max\{|\lambda_2|, |\lambda_{|\Omega|}|\}$. When P is positive semidefinite, we have

$$\mathrm{GAP}(P) = 1 - \lambda_2 = \inf\left\{\frac{\mathcal{E}_P(f,f)}{\langle f,f\rangle_\mu} \mid f : \Omega \to \mathbb{R}, \langle f,f\rangle_\mu \neq 0\right\}.$$

For P reversible and ergodic, we have the following standard comparison between the spectral gap and the mixing time

$$T_{mix}(P,\varepsilon) = \frac{1}{\mathrm{GAP}(P)} \cdot \log\big(\frac{1}{\varepsilon\mu_{min}}\big), \tag{2}$$

where $\mu_{min} := \min_{x\in\Omega} \mu(x)$.

The expected value of a function $f : \Omega \to \mathbb{R}_{\geq 0}$ with respect to μ is defined as $\mathrm{E}_\mu[f] = \sum_{x\in\Omega} f(x)\mu(x)$. Similarly, the entropy of the function with respect to μ is given by

$$\mathrm{Ent}_\mu(f) := \mathrm{E}_\mu\Big[f \log \frac{f}{\mathrm{E}_\mu[f]}\Big] = \mathrm{E}_\mu[f \log f] - \mathrm{E}_\mu[f \log(\mathrm{E}_\mu[f])].$$

We say that the Markov chain P satisfies a *modified log-Sobolev inequality* (MLSI) with constant ρ if for every function $f : \Omega \to \mathbb{R}_{\geq 0}$,

$$\rho \cdot \mathrm{Ent}_\mu(f) \leq \mathcal{E}_P(f, \log f).$$

The smallest ρ satisfying the inequality above is called the *modified log-Sobolev constant* of P and is denoted by $\rho(P)$. A well-known general relationship (see [27, 13]) shows that

$$\frac{1 - 2\mu_{min}}{\log(1/\mu_{min} - 1)}\mathrm{GAP}(P) \leq \rho(P) \leq 2\mathrm{GAP}(P). \tag{3}$$

For distributions μ and ν over Ω, the relative entropy of ν with respect to μ, denoted as $\mathcal{H}(\nu \mid \mu)$, is defined as $\mathcal{H}(\nu \mid \mu) := \sum_{x\in\Omega} \nu(x) \log \frac{\nu(x)}{\mu(x)}$. A Markov chain P with stationary distribution μ is said to satisfy discrete *relative entropy decay* with rate $r > 0$ if for all distributions ν:

$$\mathcal{H}(\nu P \mid \mu) \leq (1 - r)\mathcal{H}(\nu \mid \mu). \tag{4}$$

It is a standard fact (see, e.g., Lemma 2.4 in [7]) that when (4) holds, then $\rho(P) \geq r$, and

$$T_{mix}(P,\varepsilon) \leq \frac{1}{r} \cdot \Big(\log\log\big(\frac{1}{\mu_{min}}\big) + \log\big(\frac{1}{2\varepsilon}\big)\Big). \tag{5}$$

3 Swendsen-Wang dynamics on general graphs

In this section, we consider the SW dynamics for the q-state ferromagnetic Potts models on general graphs. In particular, we establish Theorem 2 from the introduction, which is a direct corollary of the following more general result.

▶ **Theorem 11.** *Let $q \geq 2$, $\beta > 0$, $\eta > 0$, $b > 0$, $\Delta \geq 3$, and $\chi \geq 2$. Suppose $G = (V,E)$ is an n-vertex graph of maximum degree Δ and chromatic number χ. Let μ_{Potts} be the Gibbs distribution of the q-state ferromagnetic Potts model on G with parameter β. If μ_{Potts} is η-spectrally independent and b-marginally bounded, then there exists a universal constant $C > 1$ such that the modified log-Sobolev constant of the SW dynamics satisfies:*

$$\rho(P_{SW}) = \Omega\left(\frac{b^{7+6\kappa}}{\chi\cdot(C\Delta\log n)^{\kappa}\cdot(\log\log n)^{\kappa+1}}\right),$$

where $\kappa = 1 + \lceil\frac{2\eta}{b}\rceil$, *and*

$$T_{mix}(P_{SW}) = O\big(b^{-(7+6\kappa)}\cdot\chi\cdot(C\Delta\log n)^{\kappa}(\log\log n)^{\kappa+1}\cdot\log n\big).$$

Theorem 2 follows from this theorem by noting that $\chi\leq\Delta$ and that under the assumptions $\eta = O(1)$ and $\beta\Delta = O(1)$, we have $b = O(1)$ and $\kappa = O(1)$.

▶ Remark 12. When Δ is small, i.e., $\Delta = o(\log n)$, we can obtain slightly better bounds on $\rho(P_{SW})$ and $T_{mix}(P_{SW})$ and replace the $(C\Delta\log n\cdot\log\log n)^{\kappa}$ factor by a factor of $(C\Delta)^{6+4\lceil\frac{2\eta}{b}\rceil}$. This result is included in the full version of this paper [12].

Before proving Theorem 11, we provide a number of definitions and required background results in Section 3.1. We then sketch the proof of Theorem 11 in Sections 3.2 and include some applications of this result in Section 3.3.

3.1 Factorization of entropy

We present next several factorizations of the entropy functional $\mathrm{Ent}_{\mu}(f)$, which are instrumental in establishing the decay of the relative entropy for the SW dynamics. We introduce some useful notations first. For a pinning τ in $V\setminus\Lambda$ (i.e., $\tau\in\Omega_{V\setminus\Lambda}$), we let $\mu^{\tau}_{\Lambda}(\cdot) := \mu(\cdot\mid\sigma_{V\setminus\Lambda}=\tau)$. Given a function $f:\Omega\to\mathbb{R}_{\geq 0}$, subsets of vertices $B\subseteq\Lambda\subset V$, and $\tau\in\Omega_{V\setminus\Lambda}$, the function $f^{\tau}_{B}:\Omega^{\tau}_{B}\to\mathbb{R}_{\geq 0}$ is defined by:

$$f^{\tau}_{B}(\sigma) = \mathrm{E}_{\xi\sim\mu^{\tau}_{\Lambda\setminus B}}[f(\tau\cup\xi\cup\sigma)].$$

If $B=\Lambda$, we often write f^{τ} for f^{τ}_{B}, and if $\tau=\emptyset$, then we use f_B for f^{τ}_{B}. We use $\mathrm{Ent}^{\tau}_{B}(f^{\tau})$ to denote $\mathrm{Ent}_{\mu^{\tau}_{B}}(f^{\tau})$, and if the pinning τ on $V\setminus B$ is from a distribution π over $\Omega_{V\setminus B}$, we use $\mathrm{E}_{\tau\sim\pi}[\mathrm{Ent}^{\tau}_{B}(f^{\tau})]$ to denote the expected value of the function f on S over the random pinning τ.

Various forms of entropy factorization arise from bounding $\mathrm{Ent}_{\mu}(f)$ by different (weighted) sums of restricted entropies of the function f. The first one we introduced, is the so-called *ℓ-uniform block factorization of entropy* of *ℓ-UBF*. For an integer $\ell\leq n$, ℓ-UBF holds for μ with constant C_{UBF} if for all functions $f:\Omega\to\mathbb{R}_{\geq 0}$,

$$\frac{\ell}{n}\cdot\mathrm{Ent}_{\mu}(f)\leq C_{\mathrm{UBF}}\cdot\frac{1}{\binom{n}{\ell}}\sum_{S\in\binom{V}{\ell}}\mathrm{E}_{\tau\sim\mu_{V\setminus S}}\left[\mathrm{Ent}^{\tau}_{S}(f^{\tau})\right], \tag{6}$$

where $\binom{V}{\ell}$ denotes the collection of all subsets of V of size ℓ. An important special case is when $\ell=1$, in which case (6) is called *approximate tensorization of entropy (AT)*; this special case has been quite useful for establishing optimal mixing time bounds for the Glauber dynamics in various settings (see, e.g., [47, 16, 18, 46]). The following result will be useful for us.

▶ **Theorem 13** ([25, 6]). *Let b and η be fixed. For $\theta\in(0,1)$ and $n\geq\frac{2}{\theta}(\frac{4\eta}{b^2}+1)$, the following holds. If the Gibbs distribution μ of a spin system on an n-vertex graph is η-spectrally independent and b-marginally bounded, then $\lceil\theta n\rceil$-UBF holds with $C_{\mathrm{UBF}} = (e/\theta)^{\lceil\frac{2\eta}{b}\rceil}$. In addition, if $\theta < b^2/(12\Delta)$, then:*

$$\mathrm{Ent}_{\mu}(f)\leq C_{\mathrm{UBF}}\cdot\frac{18}{b^5\theta}\sum_{i=1}^{n}\mathrm{E}_{\tau\sim\mu_{V\setminus\{i\}}}\left[\mathrm{Ent}^{\tau}_{i}(f^{\tau})\right].$$

Note that the inequality in the theorem corresponds to AT with constant $C_{\mathrm{AT}} = C_{\mathrm{UBF}} \cdot \frac{18}{b^5\theta}$.

Another useful notion is *k-partite factorization of entropy* or KPF. Let $U_1, \dots, U_k$ be k disjoint independent sets of V such that $\bigcup_{i=1}^k U_i = V$. We say μ satisfies KPF with constant C_{KPF} if for all functions $f : \Omega \to \mathbb{R}_{\geq 0}$,

$$\mathrm{Ent}_\mu(f) \leq C_{\mathrm{KPF}} \sum_{i=1}^{k} \mathrm{E}_{\tau \sim \mu_{V \setminus U_i}} \left[\mathrm{Ent}_{U_i}^{\tau}(f^\tau)\right].$$

KPF was introduced in [6], where it was used to analyze global Markov chains. The interplay between KPF and UBF is intriguing and is further explored in this paper.

3.2 Proof of main result for the SW dynamics: Theorem 11

The main technical contribution in the proof of Theorem 11 is establishing KPF with a better (i.e., smaller) constant C_{KPF}. As in [6], KPF is then used to derive an improved "edge-spin" factorization of entropy which is known to imply the desired bounds on the modified log-Sobolev constant and on the mixing time of the SW dynamics.

▶ **Theorem 14.** *For a b-marginally bounded Gibbs distribution μ that satisfies η-spectral independence on an n-vertex graph $G = (V, E)$ of maximum degree Δ, if b and η are constants independent of Δ and n, and $\Delta \in [3, \frac{b^4 n}{10e(4\eta+b^2)}]$, then there exists an absolute constant $c > 0$ such that k-partite factorization of entropy holds for μ with constant $C_{\mathrm{KPF}} = (\Delta \log n)^c$. Specifically, for a set of k disjoint independent sets $V_1, \dots, V_k$ such that $\bigcup_{j=1}^k V_j = V$,*

$$\mathrm{Ent}_\mu(f) \leq 54 \cdot \frac{e^{13\kappa}}{b^{5+6\kappa}} \cdot (\Delta \log n)^{\kappa} \cdot (\log\log n)^{1+\kappa} \sum_{j=1}^{k} \mathrm{E}_{\tau \sim \mu_{V \setminus V_j}} [\mathrm{Ent}_{V_j}^{\tau}(f^\tau)], \tag{7}$$

where $\kappa = 1 + \lceil \frac{2\eta}{b} \rceil$. Moreover, if $\Delta^2 \leq \frac{b^4 n}{10e(4\eta+b^2)}$, then the following also holds

$$\mathrm{Ent}_\mu(f) \leq 72 \cdot \frac{e^{8\kappa}}{b^{5+4\kappa}} \cdot \Delta^{2+4\kappa} \sum_{j=1}^{k} \mathrm{E}_{\tau \sim \mu_{V \setminus V_j}} [\mathrm{Ent}_{V_j}^{\tau}(f^\tau)]. \tag{8}$$

This Theorem is proved in the full version [12].

▶ Remark 15. Let $\mathcal{B} = \{B_1, \dots, B_k\}$ be a collection of disjoint independent sets such that $V = \bigcup_{i=1}^k B_i$. The independent set dynamics $P_{\mathcal{B}}$ is a heat-bath block dynamics w.r.t. $\mathcal{B}$ and a uniform distribution over $\mathcal{B}$. If μ satisfies k-partite factorization of entropy with C_{KPF}, then $P_{\mathcal{B}}$ satisfies a relative entropy decay with rate $r \geq 1/(k \cdot C_{\mathrm{KPF}})$.

As mentioned, KPF was first studied in [6]; the constant proved there was

$$C_{\mathrm{KPF}} = b^{O(\Delta)} \cdot (b\Delta)^{O(\eta/b)},$$

so our new bound improves the dependence on Δ from exponential to polynomial.

With KPF on hand, the next step in the proof of Theorem 11 relies on the so-called edge-spin factorization of entropy. Let $\Omega_J := \Omega \times \{0,1\}^E$ be the set of joint configurations (σ, A) corresponding to pairs of a spin configuration $\sigma \in \Omega$ and an *edge configuration* (a subset of edges in a graph) $A \subseteq E$. For a q-state Potts model μ_{Potts} with parameter $p = 1 - e^{-\beta}$, we use ν to denote the *Edwards-Sokal* measure on Ω_J given by

$$\nu(\sigma, A) := \frac{1}{Z_J}(1-p)^{|E|-|A|} p^{|A|} \mathbf{1}(\sigma \sim A),$$

where $\sigma \sim A$ is the event that every edge in A has its two endpoints with the same spin in σ, and $Z_J := \sum_{(A,\sigma)\in\Omega_J}(1-p)^{|E|-|A|}p^{|A|}\mathbf{1}(\sigma \sim A)$ is a normalizing constant. Let $\nu(\cdot \mid \sigma)$ and $\nu(\cdot \mid A)$ denote the conditional measures obtained from ν by fixing the spin configuration to be σ or fixing the edge configuration to be A respectively. For a function $f : \Omega_J \to \mathbb{R}_{\geq 0}$, let $f^\sigma : \{0,1\}^{|E|} \to \mathbb{R}_{\geq 0}$ be the function given by $f^\sigma(A) = f(\sigma \cup A)$, and let $f^A : \Omega \to \mathbb{R}_{\geq 0}$ be the function given by $f^A(\sigma) = f(\sigma \cup A)$. We say that *edge-spin factorization of entropy* holds with constant C_{ES} if for all functions $f : \Omega_J \to \mathbb{R}_{\geq 0}$,

$$\mathrm{Ent}_\nu(f) \leq C_{\mathrm{ES}}\left(\mathrm{E}_{(\sigma,A)\sim\nu}\left[\mathrm{Ent}_{A\sim\nu(\cdot|\sigma)}(f^\sigma)\right] + \mathrm{E}_{(\sigma,A)\sim\nu}\left[\mathrm{Ent}_{\sigma\sim\nu(\cdot|A)}(f^A)\right]\right). \tag{9}$$

The following result from [6] will be useful for us.

▶ **Lemma 16** (Theorem 6.1 [6]). *Suppose the q-state ferromagnetic Potts model with parameter β on a graph G of maximum degree is $\Delta \geq 3$ satisfies KPF with constant C_{KPF}. Then, the edge-spin factorization of entropy holds with constant $C_{\mathrm{ES}} = O(\beta\Delta k e^{\beta\Delta}) \cdot C_{\mathrm{KPF}}$.*

The final ingredient in the proof of Theorem 11 is the following.

▶ **Lemma 17** (Lemma 1.8 [7]). *Suppose edge-spin factorization of entropy holds with constant C_{ES}. Then, the SW dynamics P_{SW} satisfies the relative entropy decay with rate $\Omega\left(\frac{1}{C_{\mathrm{ES}}}\right)$.*

We are now ready to prove Theorem 11. Since Theorem 14 requires an upper bound on the maximum degree Δ, when $\Delta = \Omega(n)$ we use a crude comparison argument to obtain a polynomial bound for the modified log-Sobolev constant and mixing time of the SW dynamics.

Proof of Theorem 11. First, we assume $\Delta \in [3, \frac{b^4 n}{10e(4\eta+b^2)}]$. By Theorem 14, μ_{Potts} satisfies χ-partite factorization of entropy with constant

$$C_{\mathrm{KPF}} = (\Delta \log n)^\kappa (\log\log n)^{1+\kappa} \cdot O\left(\frac{e^{13\kappa}}{b^{5+6\kappa}}\right).$$

It follows from Lemma 16 and Lemma 17 that the SW dynamics satisfies (4) with

$$r = \Omega\left(\frac{b^{5+6\kappa}}{\chi\beta\Delta e^{\beta\Delta}\cdot(\Delta\log n)^\kappa(\log\log n)^{1+\kappa}\cdot e^{13\kappa}}\right).$$

Note that $b \leq q^{-1}e^{-\beta\Delta}$, and so $\beta\Delta e^{\beta\Delta} \leq e^{2\beta\Delta} \leq b^{-2}$. Therefore, the mixing time bound follows from (5).

Next, let us consider the case when $\Delta = \Omega(n)$. In this case, it suffices to provide a $1/\mathrm{poly}(n)$ lower bound on the modified log-Sobolev constant of the SW dynamics, which can be obtained in a straightforward manner using the known bounds for the Potts Glauber dynamics and the comparison technology from [8].

Recall that $P_{\mathcal{B}}$ is the independent set dynamics; that is, the block dynamics with respect to a collection of disjoint independent sets $\{B_1, \ldots, B_k\}$; see Remark 15. From Theorem 3.2 in [32], we know that $\mathrm{GAP}(P_{GD}) \geq n^{-(2\eta+1)}$, where P_{GD} denotes the Potts Glauber dynamics. Since $\mathcal{E}_{P_{GD}}(f,f) \leq \mathcal{E}_{P_{\mathcal{B}}}(f,f)$ for any function f, it follows that $\mathrm{GAP}(P_{\mathcal{B}}) \geq n^{-(2\eta+1)}$. In addition, the comparison inequalities from [8] imply that

$$\mathrm{GAP}(P_{SW}) \geq \mathrm{GAP}(P_{\mathcal{B}}) \cdot \min_{i=1,\ldots,k}\ \min_{\tau\in\Omega_{V\setminus B_i}}\ \min_{v\in B_i} \mathrm{GAP}(P_v^\tau),$$

where P_v^τ is the transition matrix for the update at vertex v, with τ as the fixed boundary condition, that adds each monochromatic edge between v and its neighbors independently with probability $p := 1 - e^{-\beta}$, and assigns a new random spin to v only if no edge is added. From a simple coupling argument it follows that for any $v \in B_i$, $\mathrm{GAP}(P_v^\tau) \geq (1-p)^\Delta = e^{-\beta\Delta} \geq qb$. Thus, $\mathrm{GAP}(P_{SW}) \geq n^{-(2\eta+1)}qb$, and $\rho(P_{SW}) = \Omega(n^{-(2\eta+2)}b)$ by (3). The mixing time bound follows from (2). ◀

3.3 Applications of Theorem 11

In this section, we prove Corollary 3 from the introduction and present another application of Theorem 11 concerning the SW dynamics on a random graph generated from the classical Erdős-Rényi $G(n,p)$ model. For this, we first define Dobrushin's influence matrix.

▶ **Definition 18.** *The* Dobrushin influence matrix $A \in \mathbb{R}^{n\times n}$ *is defined by* $A(u,u) = 0$ *and for* $u \neq v$*,*

$$A(u,v) = \max_{(\sigma,\tau)\in S_{u,v}} d_{TV}(\mu_v(\cdot \mid \sigma), \mu_v(\cdot \mid \tau)),$$

where $S_{u,v}$ *contains the set of all pairs of partial configurations* (σ,τ) *in* $\Omega_{V\setminus\{v\}}$ *that can only disagree at* u*, namely,* $\sigma_w = \tau_w$ *if* $w \neq u$*.*

It is known that an upper bound on the spectral norm of A implies spectral independence. In particular, we have the following result from [6].

▶ **Proposition 19** (Theorem 1.13, [6]). *If the Dobrushin influence matrix* A *of a distribution* μ *satisfies* $\|A\| \leq 1-\varepsilon$ *for some* $\varepsilon > 0$*, then* μ *is spectral independent with constant* $\eta = 2/\varepsilon$*.*

For the ferromagnetic Ising model, $\beta_u(\Delta) := \ln \frac{\Delta}{\Delta-2}$ corresponds to the threshold value of the parameter β for the uniqueness/non-uniqueness phase transition on the Δ-regular tree. For the anti-ferromagnetic Ising model, the phase transition occurs at $\bar{\beta}_u(\Delta) := -\ln\frac{\Delta}{\Delta-2}$. If $\bar{\beta}_u(\Delta)(1-\delta) < \beta < \beta_u(\Delta)(1-\delta)$, we say the Ising model satisfies the *δ-uniqueness condition*. On a bounded degree graph, $\|A\| \leq 1-\delta$ for the Ising model is a strictly stronger condition than δ-uniqueness condition. However, due to the observation made in [3], if $\Delta \to \infty$, the two conditions are roughly equivalent.

▶ **Proposition 20.** *The Ising model with parameter* $\bar{\beta}_u(\Delta)(1-\delta) < \beta < \beta_u(\Delta)(1-\delta)$ *and* $\Delta \to \infty$ *satisfies* $\|A\| \leq 1-\delta/2$*.*

This proposition is proved in the full version [12]; we show next that Corollary 3 follows from Theorem 11. For this, we first restate the corollary in a more precise manner.

▶ **Corollary 21.** *Let* $\delta \in (0,1)$ *and* $\Delta \geq 3$*. For the ferromagnetic Ising model with* $\beta \leq (1-\delta)\beta_u(\Delta)$ *on any graph* G *of maximum degree* Δ *and chromatic number* χ*, or for the ferromagnetic* q*-state Potts model with* $q \geq 3$ *and* $\beta \leq \frac{2(1-\delta)}{\Delta}$ *on the same graph, the mixing time of the SW dynamics satisfies*

$$T_{mix}(P_{SW}) = O\big(\chi \cdot \Delta^{\kappa} \cdot (\log n \log\log n)^{1+\kappa}\big),$$

where $\kappa = 1 + \lceil \frac{4qe^2}{\delta} \rceil$*.*

Proof. If $\Delta = O(1)$, then the corollary was proved in a stronger form in [6]. Thus, we assume $\Delta \to \infty$.

We first show spectral independence. Let $q = 2$. Under the δ-uniqueness condition $0 < \beta < (1-\delta)\beta_u(\Delta)$, by Proposition 20 and Proposition 19, the Ising model μ_{Ising} satisfies $(4/\delta)$-spectral independence. For the q-state Potts model with $q \geq 3$, the Dobrushin influence matrix corresponding to μ_{Potts} satisfies $\|A\| \leq \frac{1}{2}\beta\Delta$; see proof of Theorem 2.13 in [54]. Thus, if $\beta \leq \frac{2(1-\delta)}{\Delta}$, then $\|A\| \leq 1-\delta$, and by Proposition 19, μ_{Potts} satisfies $(2/\delta)$-spectral independence.

Letting $N(v)$ denote the neighborhood of v, and noting that for any configuration η on $N(v)$ we have $\mu(\sigma_v = c \mid \sigma_{N(v)} = \eta) \geq 1/(qe^2)$, we deduce that μ_{Potts} and μ_{Ising} are both $(1/(qe^2))$-marginally bounded. Therefore, by noting that $\kappa = 1 + \lceil \frac{4qe^2}{\delta} \rceil$ is a constant that only depends on δ, the mixing time bound follows from Theorem 11

$$T_{mix}(P_{SW}) = O\big(b^{-(7+6\kappa)} \cdot \chi \cdot (C\Delta \log n)^{\kappa} (\log\log n)^{\kappa+1} \cdot \log n\big) = O\big(\chi \cdot \Delta^{\kappa} \cdot (\log n \log\log n)^{1+\kappa}\big),$$

as desired. ◀

3.3.1 The SW dynamics on random graphs

As another application of Theorem 11, we consider the SW dynamics on a random graph generated from the classical $G(n, \frac{d}{n})$ model in which each edge is included independently with probability $p = d/n$; we consider the case where d is a constant independent of n. In this setting, while a typical graph has $\tilde{O}(n)$ edges, its maximum degree is of order $\Theta(\frac{\log n}{\log\log n})$ with high probability. Our results imply that the SW dynamics has polylogarithmic mixing on this type of graph provided β is small enough.

▶ **Corollary 22.** *Let $\delta \in (0,1)$ and $d \in \mathbb{R}_{\geq 0}$ be constants independent of n. Suppose that $G \sim G(n, d/n)$ and G has maximum degree Δ. For the ferromagnetic Ising model with parameter $\beta < (1-\delta)\beta_u(\Delta)$ on G or the ferromagnetic q-Potts model with $q \geq 2$ and $\beta \leq \frac{2(1-\delta)}{\Delta}$ on the same graph, the SW dynamics has $(\log n)^{3+2\lceil \frac{4qe^2}{\delta} \rceil} \cdot O(\log\log n)$ mixing time, with high probability over the choice of the random graph G.*

Corollary 22 is established using Corollary 21 and the following fact about random graphs. The full proof is provided in the full version [12].

▶ **Proposition 23** ([1]). *Let $G \sim G(n, \frac{d}{n})$ for a fixed $d \in \mathbb{R}_{\geq 0}$, and let χ be the chromatic number of G. With high probability over the choice of G, $\chi = k_d$ or $\chi = k_d + 1$, where k_d is the smallest integer k such that $d < 2k \log k$.*

4 Systematic scan dynamics

In this section, we study the systematic scan dynamics for general spin systems, which we define next.

▶ **Definition 24** (Spin system). *Let $G = (V, E)$ be a graph and $\mathcal{S} = \{1, \dots, q\}$ a set of spins. Let $\Omega \subseteq \mathcal{S}^V$ be the set of possible spin configurations on G. We write σ_v for the spin assigned to v by σ. Given a configuration $\sigma \in \Omega$ and a subset Λ of V, we write $\sigma_\Lambda \in \mathcal{S}^\Lambda$ for the configuration of σ restricted to Λ. For a subset of vertices $\Lambda \subseteq V$, a* boundary condition *τ is an assignment of spins to (some) vertices in outer vertex boundary $\partial\Lambda \subseteq V \setminus \Lambda$ of Λ; namely, $\tau : (\partial\Lambda)_\tau \to \mathcal{S}$, with $(\partial\Lambda)_\tau \subseteq \partial\Lambda$. Note that a boundary condition is simply a pinning of a subset of vertices identified as being in the boundary of G. Given a boundary condition $\tau : (\partial V)_\tau \to \mathcal{S}$, the* Hamiltonian *$H : \Omega \to \mathbb{R}$ of a spin system is defined as*

$$H(\sigma) = -\sum_{\{v,u\} \in E} K(\sigma_v, \sigma_u) - \sum_{\{v,u\} \in E : u \in V, v \in (\partial V)_\tau} K(\sigma_v, \tau_v) - \sum_{v \in V} U(\sigma_v), \tag{10}$$

where $K : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ and $U : \mathcal{S} \to \mathbb{R}$ are respectively the symmetric edge interaction potential function *and the* spin potential function *of the system. The* Gibbs distribution *of a spin system with Hamiltonian H is defined as*

$$\mu(\sigma) = \frac{1}{Z_H} e^{-H(\sigma)},$$

where $Z_H := \sum_{\sigma \in \Omega} e^{-H(\sigma)}$. We use Ω for the set of configurations σ satisfying $\mu(\sigma) > 0$.

The Potts model, as defined in the introduction, corresponds to the spin system with $q \geq 2$, $K(x,y) = \beta \cdot \mathbb{1}(x = y)$, and $U(\sigma_v) = 0$ for all $v \in V$. In this section, we focus on the ferromagnetic Ising model where $\beta > 0$ and $\mathcal{S} = \{-1, +1\}$. Another important spin system is the hardcore model that can be defined by setting $\mathcal{S} = \{1, 0\}$, $K(x,y) = \infty$ if $x = y = 1$ and $K(x,y) = 0$ otherwise, and $U(x) = \mathbb{1}(x = 1) \cdot \ln \lambda$, where $\lambda > 0$ is referred to as the *fugacity* parameter of the model.

We restrict attention to *totally-connected* spin systems, as this ensures that the Glauber dynamics, the systematic scan dynamics, and the block dynamics are all irreducible Markov chains (and thus ergodic).

▶ **Definition 25.** *For a subset $\mathcal{C}_U$ of partial configurations on $U \subseteq V$, let $H[\mathcal{C}_U] = (\mathcal{C}_U, E[\mathcal{C}_U])$ be the induced subgraph where $E[\mathcal{C}_U]$ consists of all pairs of configurations on $\mathcal{C}_U$ that differ at exactly one vertex. We say that $\mathcal{C}_U$ is connected when $H[\mathcal{C}_U]$ is connected. For a pinning τ on $\Lambda \subseteq V$, we say $\Omega^\tau_{V\setminus\Lambda}$ is connected if $H[\Omega^\tau_{V\setminus\Lambda}]$ is connected. A distribution μ over $\mathcal{S}^V$ is* totally-connected *if for every $\Lambda \subseteq V$ and every pinning τ on Λ, $\Omega^\tau_{V\setminus\Lambda}$ is connected.*

Given an ordering $\phi = [v_1, \dots, v_n]$ of the vertices, a systematic scan dynamics performs heat-bath updates on $v_1, \dots, v_n$ sequentially in this order. Recall that a heat-bath update on v_i simply means the replacement of the spin on v_i by a new spin assignment generated according to the conditional distribution in v_i given the configuration in $V \setminus \{v_i\}$. Let $P_i \in \mathbb{R}^{|\Omega|\times|\Omega|}$ be the transition matrix corresponding to a heat-bath update on the vertex v_i. The transition matrix of the systematic scan dynamics for the ordering ϕ can be written as $\mathcal{S}_\phi := P_n \dots P_1$. In general, $\mathcal{S}_\phi$ is not reversible, so as in earlier works we work with the symmetrized version of the scan dynamics that updates the spins in the order ϕ and in addition updates the spins in the reverse order of ϕ [33, 48]. The transition matrix of the symmetrized systematic scan dynamics can then be written as

$$P_\phi := \prod_{i=1}^{n} P_i \prod_{i=0}^{n-1} P_{n-i}.$$

Henceforth, we only consider the symmetrized version of the dynamics. Since P_ϕ is a symmetrized product of reversible transition matrices, one can straightforwardly verify its reversibility with respect to μ; its ergodicity follows from the assumption that the spin system is totally-connected (see Definition 25).

We show tight mixing time bounds for P_ϕ for monotone spin systems (see Definition 4). Our main result for the systematic scan dynamics is Theorem 6 from the introduction; its proof is included in the full version of this paper [12]. We complement Theorem 6 with a lower bound for the mixing time of systematic scan dynamics for a particular ordering ϕ. Specifically, on a bipartite graph $G = (V_E \cup V_O, E)$, an *even-odd scan dynamics P_{EOE}* is a systematic scan dynamics with respect to an ordering ϕ such that v_e appears before v_o in ϕ for all $v_e \in V_E$ and $v_o \in V_O$. In other words,

$$P_\phi = \prod_{i:v_i\in V_E} P_i \prod_{i:v_i\in V_O} P_i \prod_{i:v_i\in V_O} P_i \prod_{i:v_i\in V_E} P_i.$$

The above expression is well-defined without specifying the ordering in which the vertices in V_E and V_O are updated since the updates commute.

▶ **Lemma 26.** *Let Δ be a constant and let G be an n-vertex connected bipartite graph with maximum degree Δ. The even-odd scan dynamics P_{EOE} for the ferromagnetic Ising model on G has mixing time $T_{mix}(P_{EOE}) = \Omega(\log n)$.*

The lower bound in Lemma 26 is proved in the full version of this paper [12] using the machinery from [42] and the fact that even-odd scan dynamics does not propagate disagreements quickly (under a standard coupling). Our proof can thus be extended to other scan orderings that propagate disagreements slowly; however, there are orderings that do propagate disagreements quickly (think of a box in $\mathbb{Z}^2$ with the vertices sorted in a "spiral" from the boundary of the box to its center). For this type of ordering, the technique does not provide the $\Omega(\log n)$ lower bound. In addition, while we focus on the ferromagnetic Ising model to ensure clarity in the proof, the established lower bound is expected to apply to a broader class of spin systems.

4.1 Applications of Theorem 6

We discuss next some applications of Theorem 6. As a first application, we can establish *optimal* mixing for the systematic scan dynamics on the ferromagnetic Ising model under the δ-uniqueness condition, improving the best known results that hold under the Dobrushin-type conditions [51, 28, 41]. This result was stated in Corollary 7 in the introduction and is proved next. For this, we recall that under δ-uniqueness condition, the Ising distribution μ_{Ising} satisfies spectral independence and the bounded marginals condition.

▶ **Proposition 27** ([24, 25])**.** *The ferromagnetic Ising model with parameter β such that $\bar{\beta}_u(\Delta)(1-\delta) < \beta < \beta_u(\Delta)(1-\delta)$ is $O(1/\delta)$-spectrally independent and b-marginally bounded with $b = O(1)$.*

Proof of Corollary 7. We fix $\delta \in (0,1)$ and first assume that Δ is a constant. By Proposition 27, the ferromagnetic Ising model with parameter $\beta < (1-\delta)\beta_u(\Delta)$ satisfies η-spectral independence and b-bounded marginals, where $\eta = O(1/\delta)$ and b is a constant. Since the ferromagnetic Ising model is a monotone system, it follows from Theorem 6 that $T_{mix} = O(\log n)$ for any ordering ϕ.

Now, when $\Delta \to \infty$ as $n \to \infty$, by Proposition 20, the Dobrushin's influence matrix A of ferromagnetic Ising model satisfies that $\|A\| \leq 1 - \delta/2$. Under this assumption, it is known that $T_{mix} = O(\log n)$ for any ordering ϕ; see [41]. ◀

We can similarly show mixing time bound for the systematic scan dynamics of the hardcore model on bipartite graphs under δ-uniqueness condition.

▶ **Corollary 28.** *Let $\delta \in (0,1)$ be a constant. Suppose G is an n-vertex bipartite graph of maximum degree $\Delta \geq 3$. For the hardcore model on G with fugacity λ such that $0 < \lambda < (1-\delta)\lambda_u(\Delta)$, where $\lambda_u(\Delta) = \frac{(\Delta-1)^{\Delta-1}}{(\Delta-2)^{\Delta}}$ is the tree uniqueness threshold on the Δ-regular tree, the systematic scan with respect to any ordering ϕ satisfies*

$$T_{mix}(P_\phi) = \Delta^{O(1/\delta)} \cdot O(\log n).$$

Proof of Corollary 28. The hardcore model on a bipartite graph $(V_1 \cup V_2, E)$ with fugacity $0 < \lambda < (1-\delta)\lambda_u(\Delta)$ is monotone, and [25, 3, 21] show that it satisfies $O(1/\delta)$-spectral independence and the $O(\lambda)$-bounded marginals condition. Theorem 6 then implies $\Delta^{O(1/\delta)} \cdot O(\log n)$ mixing of systematic scan for any ordering. ◀

We consider next the application of Theorem 6 to the special case where the underlying graph is a cube of the d-dimensional lattice graph $\mathbb{Z}^d$. We show that strong spatial mixing implies optimal $O(\log n)$ mixing of any systematic scan dynamics. Previously, under the same type of condition, [8] gave an $O(\log n(\log\log n)^2)$ mixing time bound for arbitrary orderings, and an $O(\log n)$ mixing time bound for a special class of scans that (deterministically) propagate disagreements slowly under the standard identity coupling. We first provide the definition of our SSM condition.

▶ **Definition 29.** *We say a spin system μ on $\mathbb{Z}^d$ satisfies the* strong spatial mixing (SSM) *condition if there exist constants $\alpha, \gamma, L > 0$ such that for every d-dimensional rectangle $\Lambda \subset \mathbb{Z}^d$ of side length between L and $2L$ and every subset $B \subset \Lambda$, with any pair (τ, τ') of boundary configurations on $\partial\Lambda$ that only differ at a vertex u, we have*

$$\|\mu_B^\tau(\cdot) - \mu_B^{\tau'}(\cdot)\|_{TV} \leq \gamma \cdot \exp(-\alpha \cdot dist(u, B)),$$

where $dist(\cdot,\cdot)$ denotes graph distance.

The definition above differs from other variants of SSM in the literature (e.g., [31, 8, 45]) in that Λ has been restricted to "regular enough" rectangles. In particular, our variant of SSM is easier to satisfy than those in [31, 45] but more restricting than the one in [8] (that only considers squares). Nevertheless, it follows from [17, 45, 2, 5] that for the ferromagnetic Ising model, this form of SSM holds up to a critical threshold temperature $\beta < \beta_c(2) = \ln(1 + \sqrt{2})$ on $\mathbb{Z}^2$.

Corollary 9 from the introduction states that for b-marginally bounded monotone spin system on d-dimensional cubes $V \subseteq \mathbb{Z}^d$, SSM implies that the mixing time of any systematic scan P_ϕ is $O(\log n)$. As mentioned there, this result in turn implies that any systematic scan dynamics for the ferromagnetic Ising model is mixing in $O(\log n)$ steps on boxes of $\mathbb{Z}^2$ when $\beta < \beta_c(2)$. Another interesting consequence of Corollary 9 is that we obtain $O(\log n)$ mixing time for any systematic scan dynamics P_ϕ for the hardcore model on $\mathbb{Z}^2$ when $\lambda < 2.538$, which is the best known condition for ensuring SSM [52, 50].

Our proof of Corollary 9 relies on Lemma 8 that is restated below. The proof of Lemma 8 is provided in the full version of this paper [12]. Remarkably, Lemma 8 generalizes beyond monotone systems and may be of independent interests.

▶ **Lemma 8.** *For a spin system on a d-dimensional cube $V \subseteq \mathbb{Z}^d$, SSM implies η-spectral independence, where $\eta = O(1)$.*

Proof of Corollary 9. Assume a monotone spin system satisfies SSM condition. Then the spin system satisfies η-spectral independence, where $\eta = O(1)$ by Lemma 8. By noting that $\Delta = 2^d$ the corollary follows from Theorem 6. ◀

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 Kenneth S. Alexander. On weak mixing in lattice models. *Probab. Theory Relat. Fields*, 110(441-471), 1998.

3 Nima Anari, Vishesh Jain, Frederic Koehler, Huy Tuan Pham, and Thuy-Duong Vuong. Entropic independence: Optimal mixing of down-up random walks. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 1418–1430, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519935.3520048.

4 Nima Anari, Kuikui Liu, and Shayan Oveis Gharan. Spectral independence in high-dimensional expanders and applications to the hardcore model. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1319–1330, 2020. doi:10.1109/FOCS46700.2020.00125.

5 Vincent Beffara and Hugo Duminil-Copin. The self-dual point of the two-dimensional random-cluster model is critical for $q \geq 1$. *Probab. Theory Relat. Fields*, 153(511-542), 2012.

6 Antonio Blanca, Pietro Caputo, Zongchen Chen, Daniel Parisi, Daniel Stefankovic, and Eric Vigoda. On mixing of Markov chains: coupling, spectral independence, and entropy factorization. *Electronic Journal of Probability*, 27:1–42, 2022. doi:10.1214/22-EJP867.

7 Antonio Blanca, Pietro Caputo, Daniel Parisi, Alistair Sinclair, and Eric Vigoda. Entropy decay in the Swendsen–Wang dynamics on $\mathbb{Z}^d$. *The Annals of Applied Probability*, 32(2):1018–1057, 2022. doi:10.1214/21-AAP1702.

8 Antonio Blanca, Pietro Caputo, Alistair Sinclair, and Eric Vigoda. Spatial mixing and nonlocal markov chains. *Random Structures & Algorithms*, 55(3):584–614, 2019. doi:10.1002/rsa.20844.

9 Antonio Blanca, Zongchen Chen, Daniel Stefankovic, and Eric Vigoda. The Swendsen–Wang dynamics on trees. *Random Structures & Algorithms*, 2023.

10 Antonio Blanca, Zongchen Chen, and Eric Vigoda. Swendsen-wang dynamics for general graphs in the tree uniqueness region. *Random Structures & Algorithms*, 56(2):373–400, 2020.

11 Antonio Blanca and Alistair Sinclair. Dynamics for the mean-field random-cluster model. In *Proceedings of APPROX/RANDOM*, 2015.

12 Antonio Blanca and Xusheng Zhang. Rapid mixing of global markov chains via spectral independence: the unbounded degree case. *arXiv preprint arXiv:2307.00683*, 2023.

13 Sergey Bobkov and Prasad Tetali. Modified log-sobolev inequalities, mixing and hypercontractivity. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 287–296, New York, NY, USA, 2003. Association for Computing Machinery. doi:10.1145/780542.780586.

14 Magnus Bordewich, Catherine Greenhill, and Viresh Patel. Mixing of the glauber dynamics for the ferromagnetic potts model. *Random Structures & Algorithms.*, 48(1):21–52, January 2016. URL: http://dro.dur.ac.uk/15252/.

15 Christian Borgs, Jennifer T. Chayes, and Prasad Tetali. Tight bounds for mixing of the Swendsen-Wang algorithm at the Potts transition point. *Probab. Theory Relat. Fields*, 152(3-4):509–557, 2012. doi:10.1007/s00440-010-0329-0.

16 Pietro Caputo, Georg Menz, and Prasad Tetali. Approximate tensorization of entropy at high temperature, 2014. doi:10.48550/arXiv.1405.0608.

17 Pietro Caputo and Daniel Parisi. Block factorization of the relative entropy via spatial mixing. *Communications in Mathematical Physics*, 388(2):793–818, October 2021. doi:10.1007/s00220-021-04237-1.

18 Filippo Cesi. Quasi-factorization of the entropy and logarithmic Sobolev inequalities for Gibbs random fields. *Probability Theory and Related Fields*, 120:569–584, 2001.

19 Xiaoyu Chen, Weiming Feng, Yitong Yin, and Xinyuan Zhang. Optimal mixing for two-state anti-ferromagnetic spin systems. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 588–599. IEEE, 2022.

20 Xiaoyu Chen, Weiming Feng, Yitong Yin, and Xinyuan Zhang. Rapid mixing of glauber dynamics via spectral independence for all degrees. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 137–148, 2022. doi:10.1109/FOCS52979.2021.00022.

21 Xiaoyu Chen, Jingcheng Liu, and Yitong Yin. Uniqueness and rapid mixing in the bipartite hardcore model, 2023. arXiv:2305.00186.

22 Zongchen Chen, Andreas Galanis, Daniel Stefankovic, and Eric Vigoda. Rapid mixing for colorings via spectral independence. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1548–1557, 2021.

23 Zongchen Chen, Kuikui Liu, Nitya Mani, and Ankur Moitra. Strong spatial mixing for colorings on trees and its algorithmic applications, 2023. arXiv:2304.01954.

24 Zongchen Chen, Kuikui Liu, and Eric Vigoda. Rapid mixing of Glauber dynamics up to uniqueness via contraction. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1307–1318. IEEE Computer Society, 2020.

25 Zongchen Chen, Kuikui Liu, and Eric Vigoda. Optimal mixing of glauber dynamics: Entropy factorization via high-dimensional expansion. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 1537–1550, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3406325.3451035.

26 Amin Coja-Oghlan, Andreas Galanis, Leslie Ann Goldberg, Jean Bernoulli Ravelomanana, Daniel Štefankovič, and Eric Vigoda. Metastability of the Potts ferromagnet on random regular graphs. *Communications in Mathematical Physics*, 2023. doi:10.1007/s00220-023-04644-6.

27 P. Diaconis and L. Saloff-Coste. Logarithmic Sobolev inequalities for finite Markov chains. *The Annals of Applied Probability*, 6(3):695–750, 1996. doi:10.1214/aoap/1034968224.

28 Martin Dyer, Leslie Ann Goldberg, and Mark Jerrum. Dobrushin conditions and systematic scan. In Josep Díaz, Klaus Jansen, José D. P. Rolim, and Uri Zwick, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 327–338, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

29 Martin Dyer, Leslie Ann Goldberg, and Mark Jerrum. Systematic scan for sampling colorings. *The Annals of Applied Probability*, 16(1):185–230, 2006.

30 Martin Dyer, Leslie Ann Goldberg, and Mark Jerrum. Matrix norms and rapid mixing for spin systems. *The Annals of Applied Probability*, 19(1):71–107, 2009.

31 Martin Dyer, Alistair Sinclair, Eric Vigoda, and Dror Weitz. Mixing in time and space for lattice spin systems: A combinatorial view. *Random Structures & Algorithms*, 24(4):461–479, 2004. doi:10.1002/rsa.20004.

32 Weiming Feng, Heng Guo, Yitong Yin, and Chihao Zhang. Rapid mixing from spectral independence beyond the boolean domain. *ACM Trans. Algorithms*, 18(3), October 2022. doi:10.1145/3531008.

33 James Allen Fill. Eigenvalue Bounds on Convergence to Stationarity for Nonreversible Markov Chains, with an Application to the Exclusion Process. *The Annals of Applied Probability*, 1(1):62–87, 1991. doi:10.1214/aoap/1177005981.

34 James Allen Fill and Jonas Kahn. Comparison inequalities and fastest-mixing Markov chains. *The Annals of Applied Probability*, 23(5):1778–1816, 2013. doi:10.1214/12-AAP886.

35 Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Swendsen-Wang algorithm on the mean-field Potts model. In *Proceedings of APPROX/RANDOM*, 2015.

36 Reza Gheissari and Eyal Lubetzky. Mixing times of critical two-dimensional Potts models. *Comm. Pure Appl. Math*, 71(5):994–1046, 2018.

37 Reza Gheissari, Eyal Lubetzky, and Yuval Peres. Exponentially slow mixing in the mean-field Swendsen–Wang dynamics. *Annales de l'Institut Henri Poincare (B)*, 2019. Extended abstract appeared in Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2018), pp. 1981–1988.

38 Vivek K. Gore and Mark R. Jerrum. The Swendsen-Wang process does not always mix rapidly. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 674–681, New York, NY, USA, 1997. Association for Computing Machinery. doi:10.1145/258533.258662.

39 Heng Guo and Mark Jerrum. Random cluster dynamics for the Ising model is rapidly mixing. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 1818–1827, 2017.

40 Heng Guo, Kaan Kara, and Ce Zhang. Layerwise systematic scan: Deep Boltzmann machines and beyond. In *International Conference on Artificial Intelligence and Statistics*, pages 178–187. PMLR, 2018.

41 Thomas P. Hayes. A simple condition implying rapid mixing of single-site dynamics on spin systems. In *The 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 39–46, 2006. doi:10.1109/FOCS.2006.6.

42 Thomas P. Hayes and Alistair Sinclair. A general lower bound for mixing of single-site dynamics on graphs. *The Annals of Applied Probability*, 17(3), June 2007. doi:10.1214/105051607000000104.

43 Mark Huber. A bounding chain for Swendsen-Wang. *Random Structures & Algorithms*, 22(1):43–59, 2003.

44 Vishesh Jain, Huy Tuan Pham, and Thuy-Duong Vuong. Spectral independence, coupling, and the spectral gap of the glauber dynamics. *Information Processing Letters*, 177:106268, 2022.

45 F. Martinelli, E. Olivieri, and R. H. Schonmann. For 2-D lattice spin systems weak mixing implies strong mixing. *Communications in Mathematical Physics*, 165(1):33–47, 1994. doi:cmp/1104271032.

46 Fabio Martinelli. Lectures on Glauber dynamics for discrete spin models. *Lectures on probability theory and statistics (Saint-Flour, 1997)*, 1717:93–191, 1999.

47 Katalin Marton. Logarithmic sobolev inequalities in discrete product spaces. *Combinatorics, Probability and Computing*, 28(6):919–935, 2019. doi:10.1017/S0963548319000099.

48 Ravi Montenegro and Prasad Tetali. Mathematical aspects of mixing times in markov chains. *Foundations and Trends® in Theoretical Computer Science*, 1(3):237–354, 2006. doi:10.1561/0400000003.

49 Yuval Peres and Peter Winkler. Can extra updates delay mixing? *Communications in Mathematical Physics*, 323(3):1007–1016, 2013.

50 Ricardo Restrepo, Jinwoo Shin, Prasad Tetali, Eric Vigoda, and Linji Yang. Improved mixing condition on the grid for counting and sampling independent sets. *Probab. Theory Relat. Fields*, 2013. doi:10.1007/s00440-012-0421-8.

51 BARRY SIMON. *The Statistical Mechanics of Lattice Gases, Volume I.* Princeton University Press, 1993. URL: http://www.jstor.org/stable/j.ctt7ztwsn.

52 Alistair Sinclair, Piyush Srivastava, Daniel Stefankovic, and Yitong Yin. Spatial mixing and the connective constant: Optimal bounds. *Probab. Theory Relat. Fields*, 168:153–197, 2017. doi:10.1007/s00440-016-0708-2.

53 Robert H. Swendsen and Jian-Sheng Wang. Nonuniversal critical dynamics in Monte Carlo simulations. *Phys. Rev. Lett.*, 58:86–88, 1987. doi:10.1103/PhysRevLett.58.86.

54 Mario Ullrich. Rapid mixing of swendsen–wang dynamics in two dimensions. *Dissertationes Mathematicae*, 502:1–64, 2014. doi:10.4064/dm502-0-1.

The Full Rank Condition for Sparse Random Matrices

Amin Coja-Oghlan ✉ 🏠
Department of Computer Science, TU Dortmund, Germany

Jane Gao ✉ 🏠
Department of Combinatorics and Optimization, University of Waterloo, Canada

Max Hahn-Klimroth ✉ 🏠
Department of Computer Science, TU Dortmund, Germany

Joon Lee ✉ 🏠
Communication Theory Laboratory, École Polytechnique Fédérale de Lausanne, Switzerland

Noela Müller ✉ 🏠
Department of Mathematics and Computer Science, Eindhoven University of Technology,
The Netherlands

Maurice Rolvien ✉ 🏠
Department of Computer Science, TU Dortmund, Germany

Abstract

We derive a sufficient condition for a sparse random matrix with given numbers of non-zero entries in the rows and columns having full row rank. Inspired by low-density parity check codes, the family of random matrices that we investigate is very general and encompasses both matrices over finite fields and $\{0, 1\}$-matrices over the rationals. The proof combines statistical physics-inspired coupling techniques with local limit arguments.

2012 ACM Subject Classification Mathematics of computing; Theory of computation → Error-correcting codes

Keywords and phrases random matrices, rank, finite fields, rationals

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.54

Category RANDOM

Related Version *Full Version*: https://arxiv.org/abs/2112.14090

Funding *Amin Coja-Oghlan*: DFG CO 646/3 and DFG CO 646/5
Max Hahn-Klimroth: DFG FOR 2975
Noela Müller: NWO Gravitation grant NETWORKS-024.002.003

1 Introduction

1.1 Background and motivation

While "continuous" random matrices such as, for example, a random $n \times n$-matrix with independent Gaussian entries have full rank almost surely for trivial reasons, the rank problem for random combinatorial matrices with entries drawn from discrete distributions poses deep mathematical challenges. In the 1960s Komlós, among the first to study this type of problem, proved that a random $n \times n$-matrix with independent ±1-entries has full rational rank with high probability [24]. An obvious lower bound on the singularity probability is the probability $2^{-n+o(n)}$ that two rows or columns coincide. The conjecture that this lower bound is tight inspired an impressive line of work (e.g., [23, 33]), which culminated in Tikhomirov's proof of the conjecture [34].

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 54; pp. 54:1–54:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

By comparison to the case of dense random matrices, relatively little is known about the sparse case where the average number of non-zero entries per row or column is bounded. Moreover, techniques developed for dense random matrices such as large deviations inequalities or Littlewood-Offord arguments do not easily carry over to the sparse case. Yet sparse random matrices are of key interest in computer science. Prominent applications include low-density parity check codes [32], data compression [1, 36] and hashing [16].

One of the first full rank theorems for sparse random matrices came in the guise of a random constraint satisfaction problem. Specifically, in the random k-XORSAT problem we form a random Boolean formula over n variables with m independent XOR-clauses of length k. The question is for what clause densities m/n such a random formula admits an (XOR-)satisfying assignment. Because Boolean XOR is equivalent to addition over the field $\mathbb{F}_2$, this question boils down to determining the threshold m/n up to which a random $m \times n$-matrix over $\mathbb{F}_2$ with precisely k ones per row has full row rank. For the case $k = 3$ this problem was solved by Dubois and Mandler [18] in 2002. Remarkably, the case of general k was solved only more than ten years later by Pittel and Sorkin [31]. Both proofs depend on delicate and technically demanding moment computations.

The contribution of the present paper is a sufficient condition for a sparse random combinatorial matrix to have full row rank. We derive this sufficient condition within the framework of a very general model of random matrices that hails from coding theory [32]. As a very special case this general result encompasses the random k-XORSAT problem. But in addition, we obtain a range of other important special cases such as matrices in which the number of non-zero entries per row or column follows a power law. In fact, the sufficient condition that we obtain is essentially necessary, too. The proof of the main result is based on a novel combination of statistical physics-inspired coupling arguments and local limit theorem techniques. These methods are conceptually more powerful than the method of moments as there exist concrete instances of the present random matrix model where the matrix provably has full rank even though the method of moments fails spectacularly.

1.2 Results

The random matrix model that we investigate allows to control the number of non-zero entries in the rows and columns. The model is identical to the type of model used to construct low-density parity check codes [9, 32]. Specifically, let $\boldsymbol{d} \geq 0$, $\boldsymbol{k} \geq 3$ be independent integer-valued random variables such that $\mathbb{E}[\boldsymbol{d}^{2+\eta}] + \mathbb{E}\left[\boldsymbol{k}^{2+\eta}\right] < \infty$ for an arbitrarily small $\eta > 0$. Let $(\boldsymbol{d}_i, \boldsymbol{k}_i)_{i \geq 1}$ be independent copies of $(\boldsymbol{d}, \boldsymbol{k})$ and set $d = \mathbb{E}[\boldsymbol{d}], k = \mathbb{E}[\boldsymbol{k}]$. Furthermore, let $\mathfrak{d}$ and $\mathfrak{k}$ be the greatest common divisors of the support of $\boldsymbol{d}$ and $\boldsymbol{k}$, respectively. Finally, let $n > 0$ be a large integer divisible by $\mathfrak{k}$ and let $\boldsymbol{m} \sim \mathrm{Po}(dn/k)$ be independent of $(\boldsymbol{d}_i, \boldsymbol{k}_i)_{i \geq 1}$. These definitions ensure that the event

$$\sum_{i=1}^{n} \boldsymbol{d}_i = \sum_{j=1}^{\boldsymbol{m}} \boldsymbol{k}_j \tag{1.1}$$

occurs with probability $\Omega(n^{-1/2})$ [9, Proposition 1.7]. Hence, assuming (1.1) occurs, let $\mathbb{G} = \mathbb{G}_n(\boldsymbol{d}, \boldsymbol{k})$ be a simple random bipartite graph on a set $\{a_1 \ldots, a_{\boldsymbol{m}}\}$ of *check nodes* and a set $\{x_1, \ldots, x_n\}$ of *variable nodes* such that the degree of a_i equals $\boldsymbol{k}_i$ and the degree of x_j equals $\boldsymbol{d}_j$ for all i, j. Adopting coding terminology, we refer to $\mathbb{G}$ as the *Tanner graph.* We need to assume that the second moment is bounded so that the *Tanner graph* is locally finite. The random graph $\mathbb{G}$ naturally induces a $\{0, 1\}$-matrix, namely the $\boldsymbol{m} \times n$-biadjacency matrix $\mathbb{B} = \mathbb{B}(\mathbb{G})$ of the bipartite graph. Explicitly,

$$\mathbb{B}_{ij} = \mathbb{1}\{a_i x_j \in E(\mathbb{G})\} \qquad\qquad (1 \leq i \leq \boldsymbol{m},\ 1 \leq i \leq n).$$

Let

$$D(z) = \sum_{\ell=0}^{\infty} \mathbb{P}\left[\boldsymbol{d} = \ell\right] z^{\ell} \qquad \text{and} \qquad K(z) = \sum_{\ell=0}^{\infty} \mathbb{P}\left[\boldsymbol{k} = \ell\right] z^{\ell}$$

be the probability generating functions of $\boldsymbol{d}$ and $\boldsymbol{k}$. Since $\mathbb{E}[\boldsymbol{d}^2] + \mathbb{E}[\boldsymbol{k}^2] < \infty$, the function

$$\Phi : [0,1] \to \mathbb{R}, \qquad z \mapsto D\left(1 - K'(z)/k\right) - \frac{d}{k}\left(1 - K(z) - (1-z)K'(z)\right) \tag{1.2}$$

is well-defined. The following result renders a sufficient condition for $\mathbb{B}$ to have full row rank.

▶ **Theorem 1.** *If*

$$\Phi(z) < \Phi(0) \qquad \textit{for all } 0 < z \leq 1, \tag{1.3}$$

then $\mathbb{B}$ has full rational row rank w.h.p.

Theorem 1 is a direct consequence of a significantly stronger result on matrices over finite fields. Specifically, suppose that $q \geq 2$ is a prime power, let $\mathbb{F}_q$ signify the field with q elements and let $\boldsymbol{\chi}$ be a random variable that takes values in $\mathbb{F}_q^* = \mathbb{F}_q \setminus \{0\}$. Let $(\boldsymbol{\chi}_{i,j})_{i,j \geq 1}$ be independent copies of $\boldsymbol{\chi}$. Finally, let $\mathbb{A} = \mathbb{A}_n(\boldsymbol{d}, \boldsymbol{k}, \boldsymbol{\chi})$ be the $\boldsymbol{m} \times n$-matrix with entries

$$\mathbb{A}_{i,j} = \mathbb{1}\left\{a_i x_j \in E(\mathbb{G})\right\} \cdot \boldsymbol{\chi}_{i,j} \in \mathbb{F}_q.$$

Hence, the i-th row of $\mathbb{A}$ contains $\boldsymbol{k}_i$ non-zero entries and the j-th column contains $\boldsymbol{d}_j$ non-zero entries.

▶ **Theorem 2.** *If q and $\mathfrak{d}$ are coprime and* (1.3) *is satisfied, then $\mathbb{A}$ has full row rank over $\mathbb{F}_q$ w.h.p.*

Theorem 1 follows from Theorem 2 and a few lines of linear algebra.

The sufficient condition (1.3) is quite close to being necessary, too. Indeed, the *normalised* rank of $\mathbb{A}$ (and $\mathbb{B}$) can be expressed in terms of the function Φ as follows [9, Theorem 1.1]:

$$\frac{\mathrm{rk}(\mathbb{A})}{n} \xrightarrow{n \to \infty} 1 - \max_{z \in [0,1]} \Phi(z) \qquad \text{in probability.} \tag{1.4}$$

Since $\boldsymbol{k} \geq 3$, the definition (1.2) ensures that $\Phi(0) = 1 - d/k$ and thus $n\Phi(0) \sim n - \boldsymbol{m}$ w.h.p. Hence, (1.4) implies that $\mathrm{rk}(\mathbb{A}) \leq \boldsymbol{m} - \Omega(n)$ w.h.p. unless $\Phi(z)$ attains its maximum at $z = 0$. In other words, $\mathbb{A}$ has full row rank *only if* $\Phi(z) \leq \Phi(0)$ for all $0 < z \leq 1$. Indeed, in Section 1.3 we will discover examples that require a strict inequality as in (1.3). The condition that q and $\mathfrak{d}$ be coprime is generally necessary as well, as we will see in Example 8 below.

▶ Remark 3. We emphasise that (1.4) does not guarantee that $\mathbb{A}$ has full row rank w.h.p. even if (1.3) is satisfied. In fact, due to the normalisation on the l.h.s., (1.4) only implies that $\mathrm{rk}(\mathbb{A}) = \boldsymbol{m} - o(n)$ w.h.p., rather than the far stronger statement that $\mathrm{rk}(\mathbb{A}) = \boldsymbol{m}$ w.h.p. delivered by Theorem 2.

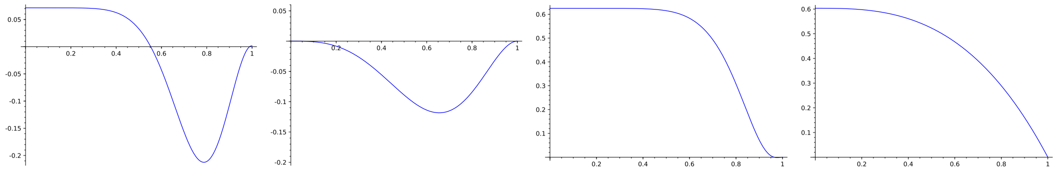


Figure 1 From left to right: the shape of Φ for Examples 4–7.

1.3 Examples

To illustrate the power of Theorems 1 and 2 we consider a few instructive special cases of distributions $\boldsymbol{d}, \boldsymbol{k}, \boldsymbol{\chi}$.

▶ **Example 4** (random k-XORSAT)**.** In random k-XORSAT we are handed a number of independent random constraints c_i of the type $c_i = y_{i1}$ XOR $\cdots$ XOR y_{ik} where each literal y_{ij} is either one of n available Boolean variables $x_1, \ldots, x_n$ or a negation $\neg x_1, \ldots, \neg x_n$. The goal to determine the maximum number of random constraints can be satisfied simultaneously w.h.p. Because Boolean XOR comes down to addition over $\mathbb{F}_2$ and since the clauses are drawn independently, XOR-satisfiability can be rephrased as the full rank problem for the random matrix $\mathbb{A}$ over $\mathbb{F}_q$ with $q = 2$, $\boldsymbol{k} = k$ fixed to a deterministic value and $\boldsymbol{d} \sim \mathrm{Po}(d)$ a Poisson variable. Hence, the generating functions of $\boldsymbol{d}, \boldsymbol{k}$ read $D(z) = \exp(d(z-1))$ and $K(z) = z^k$ and $\Phi_{d,k}(z) = \exp(-dz^{k-1}) - \frac{d}{k}\left(1 - kz^{k-1} + (k-1)z^k\right)$. Thus, Theorem 2 implies that for a given $k \geq 3$ the threshold of d up to which random k-XORSAT is satisfiable w.h.p. equals the largest d such that

$$\Phi_{d,k}(z) < \Phi_{d,k}(0) = 1 - d/k \qquad \text{for all } 0 < z \leq 1. \tag{1.5}$$

A few lines of calculus verify that (1.5) matches the formulas for the k-XORSAT threshold derived by combinatorial methods tailored to this specific case [18, 31]. Theorem 2 also encompasses the generalisations of XORSAT to other finite fields $\mathbb{F}_q$ from [5, 21].

▶ **Example 5** (identical distributions)**.** An interesting scenario arises when $\boldsymbol{d}, \boldsymbol{k}$ are identically distributed. For example, suppose that $\mathbb{P}[\boldsymbol{d} = 3] = \mathbb{P}[\boldsymbol{d} = 4] = \mathbb{P}[\boldsymbol{k} = 3] = \mathbb{P}[\boldsymbol{k} = 4] = 1/2$. Thus, $D(z) = K(z) = (z^3 + z^4)/2$. The resulting $\Phi(z)$ attains two identical maxima, namely $\Phi(0) = \Phi(1) = 0$. Since $\boldsymbol{k}_i, \boldsymbol{d}_i$ are chosen independently subject only to (1.1), the probability that $\mathbb{A}$ has more rows than columns works out to be $1/2 + o(1)$. As a consequence, $\mathbb{A}$ cannot have full row rank w.h.p. This shows that the condition that 0 be the *unique* maximiser of $\Phi(x)$ is generally necessary.

▶ **Example 6** (fixed $\boldsymbol{d}, \boldsymbol{k}$)**.** Suppose that both $\boldsymbol{d} = d, \boldsymbol{k} = k \geq 3$ are constants rather than genuinely random. Then $\Phi(z) = (1 - z^{k-1})^d - \frac{d}{k}(1 - kz^{k-1} + (k-1)z^k)$. Clearly, $\mathbb{A}$ cannot have full row rank unless $d \leq k$, while Theorem 2 implies that $\mathbb{A}$ has full row rank w.h.p. if $d < k$. This result was previously established via the second moment method [30]. But in the critical case $d = k$ the function $\Phi(z)$ attains its identical maxima at $z = 0$ and $z = 1$. Specifically, $0 = \Phi(0) = \Phi(1) > \Phi(z)$ for all $0 < z < 1$. Hence, Theorem 2 does not cover this special case. Nonetheless, Huang [22] and Mészáros [29] proved that the random $\{0, 1\}$-matrix $\mathbb{B}$ has full rational rank w.h.p. The proof is based on a delicate moment computation in combination with a precise local expansion via the Laplace method. However, numerical evidence indicates that the corresponding "d-regular" random matrix $\mathbb{A}$ over a finite field fails to have full rank w.h.p.

▶ **Example 7** (power laws). Let $\mathbb{P}(\boldsymbol{d} = \ell) \propto \ell^{-\alpha}$ for some $\alpha > 3$ and $\boldsymbol{k} = k \geq 3$. Thus,

$$D(z) = \frac{1}{\zeta(\alpha)} \sum_{\ell=1}^{\infty} \frac{z^\ell}{\ell^\alpha}, \quad K(z) = z^k,$$
$$\Phi(z) = D\left(1 - z^{k-1}\right) - \frac{\zeta^{-1}(\alpha)\zeta(\alpha-1)}{k}\left(1 - kz^{k-1} + (k-1)z^k\right)$$

and it can be verified that $\Phi'(z) < 0$ for all $z \in (0,1)$. Hence, (1.3) is always satisfied and Theorems 1 and 2 show that $\mathbb{A}, \mathbb{B}$ have full row rank.

▶ **Example 8** (zero row sums). Theorem 2 requires the assumption that q and the g.c.d. $\mathfrak{d}$ of the support of $\boldsymbol{d}$ be coprime. This assumption is indeed necessary. To see this, consider the case that $q = 2$, $\boldsymbol{\chi} = 1$, $\boldsymbol{d} = 4$ and $\boldsymbol{k} = 8$ deterministically. Then the rows of $\mathbb{A}$ always sum to zero. Hence, $\mathbb{A}$ cannot have full row rank.

2 Proof Strategy

The proof of the main theorem (Theorem 2) substantially extends the techniques behind the asymptotic rank formula (1.4) from [9]. As one key additional ingredient we require a new method to count "equitable" vectors in the kernel of $\mathbb{A}$, i.e., vectors in which each element of $\mathbb{F}_q$ occurs with frequency $1/q + o(1)$. This part of the proof, which involves the asymptotic enumeration of lattice points that satisfy certain arithmetic conditions, hinges on local limit techniques and algebraic arguments, specifically the identification of suitable bases of $\mathbb{Z}$-modules generate by lattice points. This argument generalises techniques that were also used in the study of adjacency matrices of random d-regular graphs [22, 29].

To motivate the proof strategy we first go down the "classical" path of the method of moments. We will discover where this proof strategy gets stuck and then work our way around the obstacle by means of physics-inspired coupling arguments. In statistical physics jargon, the moment calculation amounts to an "annealed" analysis. In a nutshell, the issue with such analyses is that unlikely events can render outsized contributions to moments of exponentially large random variables such as the number of vectors in the kernel of $\mathbb{A}$. Once we see where such large deviations hazards lurk, we will be able to replace the "annealed" strategy by a "quenched" approach that sidesteps these large deviations effects.

2.1 The method of moments

By extension of random k-XORSAT from Example 4, the full rank problem for the random matrix $\mathbb{A}$ over $\mathbb{F}_q$ can be viewed as a random constraint satisfaction problem. Specifically, choose a vector $\boldsymbol{y} \in \mathbb{F}_q^{\boldsymbol{m}}$ uniformly independently of $\mathbb{A}$. Then a solution to our random CSP is just a solution $x \in \mathbb{F}_q^n$ to the linear system $\mathbb{A}x = \boldsymbol{y}$. Thus, together with the corresponding entry of $\boldsymbol{y}$ each of the $\boldsymbol{m}$ rows of $\mathbb{A}$ induces a constraint.

Since the early 2000s the default method for approaching random CSPs has been the second moment method [2, 3]. Indeed, one of the first contributions to this line of work was the aforementioned work of Dubois and Mandler on random 3-XORSAT [18], which corresponds to the special case $q = 2$, $\boldsymbol{k} = 3$, $\boldsymbol{d} = \mathrm{Po}(d)$. A natural first stab at the full rank problem therefore appears to be to run the second moment routine on the number $\boldsymbol{Z} = \boldsymbol{Z}(\mathbb{A}, \boldsymbol{y})$ of solutions to $\mathbb{A}x = \boldsymbol{y}$. But clearly, to have any chance of success we need to condition on the degrees of the variable and check nodes, and a few more innocent pieces of information. Formally, let $\mathfrak{A}$ be the σ-algebra generated by $\boldsymbol{m}, (\boldsymbol{k}_i)_{i\geq1}, (\boldsymbol{d}_i)_{i\geq1}$ and by the numbers $\boldsymbol{m}(\chi_1, \ldots, \chi_\ell)$ of rows with non-zero entries $\chi_1, \ldots, \chi_\ell \in \mathbb{F}_q^*$. Let us write $\mathbb{P}_{\mathfrak{A}} = \mathbb{P}[\,\cdot \mid \mathfrak{A}]$ and $\mathbb{E}_{\mathfrak{A}} = \mathbb{E}[\,\cdot \mid \mathfrak{A}]$ for the conditional probability and expectation given $\mathfrak{A}$.

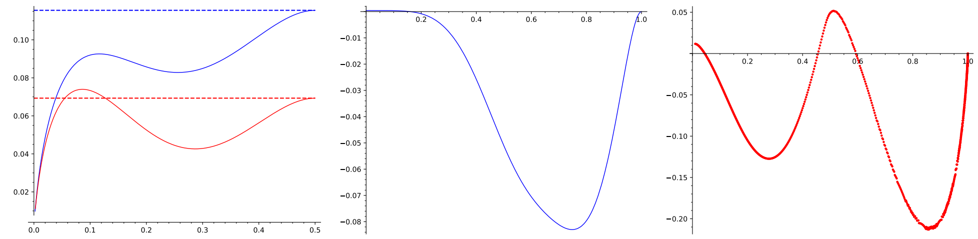


Figure 2 *Left:* the r.h.s. of (2.6) for $d = 2.5$ (blue) and $d = 2.7$ (red) in the interval $[0, \frac{1}{2}]$. *Middle:* the function $\Phi(z)$ from Example 9. *Right:* numerical lower bound on the moment formula from Example 9.

Since $\boldsymbol{y}$ is independent of $\mathbb{A}$, for any fixed $x \in \mathbb{F}_q^n$ the event $\mathbb{A}x = \boldsymbol{y}$ has probability $q^{-\boldsymbol{m}}$. As there are q^n choices of x, linearity of expectation yields

$$\mathbb{E}_{\mathfrak{A}}[\boldsymbol{Z}] = q^{n-\boldsymbol{m}}. \tag{2.1}$$

For the second moment method to succeed we need to verify that $\mathbb{E}_{\mathfrak{A}}[\boldsymbol{Z}^2] \sim \mathbb{E}_{\mathfrak{A}}[\boldsymbol{Z}]^2$. Then Chebyshev's inequality yields $\boldsymbol{Z} \sim \mathbb{E}_{\mathfrak{A}}[\boldsymbol{Z}]$ w.h.p., and thus $\mathbb{A}x = \boldsymbol{y}$ has a solution w.h.p., provided that $n \le \boldsymbol{m}$. This fact, in turn, would imply that $\mathbb{A}$ has full row rank w.h.p.; for if it were the case that $\operatorname{rk}\mathbb{A} < \boldsymbol{m}$, then the linear system $\mathbb{A}x = \boldsymbol{y}$ would fail to have a solution with probability at least $1/q$.

Concerning the computation of $\mathbb{E}_{\mathfrak{A}}[\boldsymbol{Z}^2]$, because the set of solutions is either empty or a translation of the kernel, we obtain

$$\begin{aligned}\mathbb{E}_{\mathfrak{A}}[\boldsymbol{Z}^2] &= \sum_{\sigma,\tau\in\mathbb{F}_q^n} \mathbb{P}_{\mathfrak{A}}\left[\mathbb{A}\sigma = \mathbb{A}\tau = \boldsymbol{y}\right] = \sum_{\sigma,\tau\in\mathbb{F}_q^n} \mathbb{P}_{\mathfrak{A}}\left[\mathbb{A}\sigma = \boldsymbol{y}\right] \mathbb{P}_{\mathfrak{A}}\left[\sigma - \tau \in \ker\mathbb{A}\right] \\ &= \mathbb{E}_{\mathfrak{A}}\left[\boldsymbol{Z}\right] \mathbb{E}_{\mathfrak{A}}\left[|\ker\mathbb{A}|\right].\end{aligned} \tag{2.2}$$

Hence, we are left to calculate $\mathbb{E}_{\mathfrak{A}}\left[|\ker\mathbb{A}|\right]$.

Unlike in the case (2.1) of the first moment of $\boldsymbol{Z}$, the probability of belonging to the kernel of $\mathbb{A}$ is not the same for all $x \in \mathbb{F}_q^n$. Indeed, as an extreme example, the zero vector always belongs to the kernel. By contrast, depending on $\boldsymbol{d}, \boldsymbol{k}$ and q there may be vectors that cannot possibly belong to the kernel for divisibility reasons; e.g., if $\boldsymbol{k} = 3$ and $q = 2$, then the all-ones vector cannot lie in $\ker\mathbb{A}$.

Hence, we need to tread carefully. In order to calculate the expected size of the kernel we need to discriminate vectors x according to the number $n_\ell(s)$ of variable nodes of a given degree ℓ that take a specific value $s \in \mathbb{F}_q$. Further, given the $n_\ell(s)$ we need to know the numbers $m_{\chi_1,\ldots,\chi_\ell}(s_1,\ldots,s_\ell)$ of rows with non-zero entries $\chi_1,\ldots,\chi_\ell$ whose neighbouring variable nodes in $\mathbb{G}$ receive values $s_1,\ldots,s_\ell$. Since given $\mathfrak{A}$ the matching of the variable and check nodes remains random given their degrees, the ensuing contribution to the first moment works out to be

$$\begin{aligned}\Xi(n_\ell(s), m_{\chi_1,\ldots,\chi_\ell}(s_1,\ldots,s_\ell))_{\ell,s,s_1,\ldots,s_\ell} &= \sum_{s\in\mathbb{F}_q} \mathbb{E}\left[(\boldsymbol{d}-1)n_{\boldsymbol{d}}(s)\log\frac{n_{\boldsymbol{d}}(s)}{n}\right] \\ -\frac{d}{k}\mathbb{E}&\left[\sum_{\sigma_1,\ldots,\sigma_{\boldsymbol{k}}\in\mathbb{F}_q} \mathbb{1}\left\{\boldsymbol{\chi}\perp\sigma\right\} m_{\boldsymbol{\chi}_{1,1},\ldots,\boldsymbol{\chi}_{1,\boldsymbol{k}}}(\sigma_1,\ldots,\sigma_{\boldsymbol{k}}) \log\frac{m_{\boldsymbol{\chi}_{1,1},\ldots,\boldsymbol{\chi}_{1,\boldsymbol{k}}}(\sigma_1,\ldots,\sigma_{\boldsymbol{k}})}{\boldsymbol{m}}\right].\end{aligned} \tag{2.3}$$

Then

$$\mathbb{E}_{\mathfrak{A}}|\ker\mathbb{A}| = \exp\left[\max_{n_\ell(s),m_{\chi_1,\ldots,\chi_\ell}(s_1,\ldots,s_\ell)} \Xi(n_\ell(s), m_{\chi_1,\ldots,\chi_\ell}(s_1,\ldots,s_\ell)) + o(n)\right]. \tag{2.4}$$

Hence, in order to compute the expected kernel size we should maximise the fairly impressive formula (2.3) over a potentially *very* large range of parameters $n_\ell(s)$, $m_{\chi_1,\ldots,\chi_\ell}(s_1,\ldots,s_\ell)$. The choice of these parameters is subject to the constraint that for every value $s \in \mathbb{F}_q$ the number of occurrences of s counted from the side of the check nodes must equal the number of occurrences viewd from the variable side. This leads to the equations

$$\mathbb{E}[\boldsymbol{d}n_{\boldsymbol{d}}(s)] = \mathbb{E}\left[\sum_{\sigma_1,\ldots,\sigma_{\boldsymbol{k}}\in\mathbb{F}_q} \boldsymbol{k}\mathbb{1}\left\{\sigma_1 = s\right\}\mathbb{1}\left\{\boldsymbol{\chi}\perp\sigma\right\} m_{\boldsymbol{\chi}_{1,1},\ldots,\boldsymbol{\chi}_{1,\boldsymbol{k}}}(\sigma_1,\ldots,\sigma_{\boldsymbol{k}})\right] \forall s \in \mathbb{F}_q.$$

Taking these constraints into account, we can transform (2.4) into a Lagrangian optimisation problem whose only variables are the $n_\ell(s)$, $s \in \mathbb{F}_q$, $\ell \in \mathrm{supp}\boldsymbol{d}$. A somewhat delicate application of the Laplace method then shows that $\mathbb{E}_{\mathfrak{A}}[|\ker\mathbb{A}|] = O(q^{n-\boldsymbol{m}})$, i.e., that the second moment method "nearly works", if and only if the maximum of (2.3) is attained at the "equitable" solution

$$n_\ell(s) \sim n\mathbb{P}\left[\boldsymbol{d} = \ell\right]/q \qquad \text{for all } s \in \mathbb{F}_q,\ \ell \in \mathrm{supp}\boldsymbol{d}. \tag{2.5}$$

Unfortunately, we have no idea how to solve the optimisation problem (2.4) in any generality. Worse, even if we knew how to tackle this optimisation task, that would still not suffice to prove Theorem 2. Indeed, the plain moment calculation fails even for random 3-XORSAT, i.e., the case $q = 2$, $\boldsymbol{k} = 3$ constant and $\boldsymbol{d} = \mathrm{Po}(d)$. In this case the second moment calculation reduces to the one-dimensional optimisation problem

$$\log \mathbb{E}_{\mathfrak{A}}|\ker\mathbb{A}| \sim n \cdot \max_{z\in[0,1]} -z\log z - (1-z)\log(1-z) + \frac{\boldsymbol{m}}{n}\log\frac{1+(1-2z)^3}{2} \quad \text{(cf. [18])}. \tag{2.6}$$

At $z = 1/2$ the r.h.s. of (2.6) simplifies to $(n - \boldsymbol{m})\log 2$, and thus $\mathbb{E}_{\mathfrak{A}}|\ker\mathbb{A}| = 2^{n-\boldsymbol{m}}$ matches the first moment (2.1). But if the maximum (2.6) is attained at $z \neq 1/2$, then $\mathbb{E}_{\mathfrak{A}}|\ker\mathbb{A}| \gg 2^{n-\boldsymbol{m}}$ and the second moment method fails. Figure 2 displays (2.6) for $d = 2.5$ and $d = 2.7$. While for $d = 2.5$ the function takes its maximum at $z = 1/2$, for $d = 2.7$ the maximum is attained at $z \approx 0.085$. However, the actual random 3-XORSAT threshold is $d \approx 2.75$ [18]. Thus, method of moments fails short of the real threshold.

The reason for this is that rare events are apt to boost the *expected* number of vectors in the kernel. This is precisely what happens in random k-XORSAT. The rare event in question is a fluctuation of the density of the *2-core* $\mathbb{G}^{(2)}$ of the Tanner graph, which is obtained by iteratively removing any variable nodes of degree at most one along with their unique adjacent check node if the variable degree equals one. Dubois and Mandler therefore pinpointed the 3-XORSAT threshold by applying the second moment method to the minor $\mathbb{A}^{(2)}$ induced by $\mathbb{G}^{(2)}$ while conditioning on the 2-core having its typical size and density. However, even in random k-XORSAT with $k > 3$ the ensuing optimisation problem (2.3) is anything but straightforward, as witnessed by the work of Pittel and Sorkin [31]. Furthermore, increasing the size of the field to $q > 2$ boosts the number of variables involved, which adds further significant challenges to the optimisation problem; even the case $q = 3$ turns out to be essentially intractable [21]. Finally, for general $\boldsymbol{d}, \boldsymbol{k}$ and q it is far from clear what the "relevant" variables would be that are responsible for any large deviations effects. Inspecting a few examples of degree distributions $\boldsymbol{d}, \boldsymbol{k}$ reveals that conditioning on the size and density of the 2-core will not generally suffice.

The upshot is that the second moment method hardly seems like a promising path towards Theorem 2. But we learned that we basically need to get a handle on the typical size of the kernel of $\mathbb{A}$. Specifically, if we could prove that typical vectors in the kernel are nearly equitable in the sense that all elements $s \in \mathbb{F}_q$ occur about n/q times, then we could conceivably derive the desired bound $|\ker\mathbb{A}| \leq q^{n-\boldsymbol{m}}$ w.h.p.

▶ **Example 9** (failure of the moment method). To underscore the issue with the method of moments, consider the random variables $\boldsymbol{d}, \boldsymbol{k}$ with generating functions $D(z) = 0.889z^3 + 0.111z^{21}$ and $K(z) = z^5$ and set $q = 101$. The resulting function $\Phi(z)$ (just barely) attains its unique maximum at $z = 1$. Hence, Theorem 2 shows that $\mathbb{A}$ has full row rank w.h.p. However, the moment formula (2.4) fails to attain its global maximum at the uniform solution; hence, the method of moments provably fails on this example, even though the 2-core $\mathbb{G}^{(2)}$ coincides with the entire original Tanner graph $\mathbb{G}$. Indeed, Figure 2 displays $\Phi(z)$ (middle) along with a numerical *lower* bound on the moment formula (right). The parameter on the horizontal axis of the right plot corresponds to the fraction variable occurrences set to zero. Hence, a necessary condition for the method of moments to succeed is that the maximum value be attained at $1/q$, which clearly is not the case.

2.2 Quenched analysis

Informed by this discussion, we are thus going to seize upon a different set of techniques to show that typical kernel vectors are essentially equitable. To be precise, let $\boldsymbol{x}_{\mathbb{A}} = (\boldsymbol{x}_{\mathbb{A},i})_{i\in[n]} \in \mathbb{F}_q^n$ be a random vector from the kernel of $\mathbb{A}$. We would like to show that for a given random matrix $\mathbb{A}$, such a random vector $\boldsymbol{x}_{\mathbb{A}} \in \ker \mathbb{A}$ is equitable w.h.p. In physics jargon, such a direct investigation of random solutions to a typical random combinatorial problem instance (in contrast to a moment calculation) is termed a *quenched analysis*. The fundamental merit of such a conditional (or quenched) analysis is that we may condition on the matrix $\mathbb{A}$ being *typical*; hence, we do not need to take very unlikely outcomes of $\mathbb{A}$ into consideration. By contrast, in the moment computations that we sketched in Section 2.1 we average over *all* possible outcomes of $\mathbb{A}$, including pathological cases that for some reason possess excessively large kernels.

The cornerstone of the quenched analysis will be to prove that w.h.p. over the choice of $\mathbb{A}$ the event

$$\mathfrak{O} = \left\{ \sum_{\sigma,\tau\in\mathbb{F}_q} \sum_{i,j=1}^{n} \left|\mathbb{P}\left[\boldsymbol{x}_{\mathbb{A},i} = \sigma,\ \boldsymbol{x}_{\mathbb{A},j} = \tau \mid \mathbb{A}\right] - q^{-2}\right| = o(n^2) \right\} \tag{2.7}$$

occurs. In words, $\mathfrak{O}$ asks that for any two field elements $\sigma, \tau \in \mathbb{F}_q$ for most pairs $1 \le i, j \le n$ the probability that the i-th entry $\boldsymbol{x}_{\mathbb{A},i}$ of a random kernel vector $\boldsymbol{x}_{\mathbb{A}}$ equals σ while the j-th entry $\boldsymbol{x}_{\mathbb{A},j}$ equals τ is about q^{-2}. Thus, for most choices of the indices i, j the pair $(\boldsymbol{x}_{\mathbb{A},i}, \boldsymbol{x}_{\mathbb{A},j}) \in \mathbb{F}_q^2$ is approximately uniformly distributed. Together with Chebyshev's inequality, this implies that a random vector $\boldsymbol{x}_{\mathbb{A}} \in \ker \mathbb{A}$ is equitable w.h.p. In fact, if $\mathfrak{O}$ occurs then even the degree-weighted empirical distribution of the entries of a typical $\boldsymbol{x}_{\mathbb{A}}$ is asymptotically uniform w.h.p., i.e., w.h.p. over the choice of $\boldsymbol{x}_{\mathbb{A}}$ we have

$$\sum_{i=1}^{n} \boldsymbol{d}_i \mathbb{1}\{\boldsymbol{x}_{\mathbb{A}} = \tau\} \sim q^{-1} \sum_{i=1}^{n} \boldsymbol{d}_i \qquad \text{for all } \tau \in \mathbb{F}_q. \tag{2.8}$$

Thus, the thrust behind considering the event $\mathfrak{O}$ is to accomplish just what we failed to accomplish via the moment computation: to show that the dominant contribution to the kernel comes from approximately equitable vectors.

Apart from showing that $\mathbb{A} \in \mathfrak{O}$ w.h.p., the following proposition also shows that the first moment formula (2.1) remains true on $\mathfrak{O}$.

▶ **Proposition 10.** *Under the assumptions of Theorem 2 we have* $\mathbb{P}\left[\mathbb{A} \in \mathfrak{O}\right] \sim 1$ *and*

$$\mathbb{E}_{\mathfrak{A}}\left[\boldsymbol{Z} \cdot \mathbb{1}\left\{\mathbb{A} \in \mathfrak{O}\right\}\right] \sim \mathbb{E}_{\mathfrak{A}}\left[\boldsymbol{Z}\right] \sim q^{n-\boldsymbol{m}}. \tag{2.9}$$

Before we elaborate on the proof of Proposition 10 in Section 2.3, we remark that the second moment method "works" once we condition on the event $\mathfrak{O}$. Indeed, the estimate (2.8), which is valid on $\mathfrak{O}$ w.h.p., demonstrates that once we condition on $\mathfrak{O}$, the dominant contribution to (2.3) comes from approximately uniform choices of $n_{\boldsymbol{d}}(s)$ as in (2.5). Due to the concavity of the entropy function, (2.5) implies that the optimal choices of the check variables $m_{\chi_1,\ldots,\chi_\ell}$ are asymptotically uniform as well, subject to the obvious linear constraint. Explicitly, the optimal $m_{\chi_1,\ldots,\chi_\ell}$ read

$$m_{\chi_1,\ldots,\chi_\ell}(s_1,\ldots,s_\ell) \sim \mathbb{1}\{s_1\chi_1+\cdots+s_\ell\chi_\ell=0\}q^{1-\ell}\boldsymbol{m}\mathbb{P}\left[\boldsymbol{k}=\ell\right]\prod_{i=1}^{\ell}\mathbb{P}\left[\boldsymbol{\chi}=\chi_i\right]. \tag{2.10}$$

Expanding (2.3) around (2.5) and (2.10), one could derive the bound $\mathbb{E}_{\mathfrak{A}}\left[\boldsymbol{Z}^2\cdot\mathbb{1}\left\{\mathbb{A}\in\mathfrak{O}\right\}\right] = O(\mathbb{E}_{\mathfrak{A}}\left[\boldsymbol{Z}\right])^2$ via a routine application of the Laplace method. However, to prove Theorem 2 we actually require the following more precise estimate.

▶ **Proposition 11.** *Under the assumptions of Theorem 2 we have*

$$\mathbb{E}_{\mathfrak{A}}\left[\boldsymbol{Z}^2\cdot\mathbb{1}\left\{\mathbb{A}\in\mathfrak{O}\right\}\right] \sim \mathbb{E}_{\mathfrak{A}}\left[\boldsymbol{Z}\right]^2. \tag{2.11}$$

The key challenge towards the proof the (2.11) is to obtain asymptotic equality, rather than the weaker bound $\mathbb{E}_{\mathfrak{A}}\left[\boldsymbol{Z}^2\cdot\mathbb{1}\left\{\mathbb{A}\in\mathfrak{O}\right\}\right] = O(\mathbb{E}_{\mathfrak{A}}\left[\boldsymbol{Z}\right]^2)$. This requires a meticulous expansion of the second moment around the equitable solution, which involves the detailed analysis of the lattices generated by integer vectors that encode conceivable values of the variables from (2.3). We are going to outline this analysis in Section 2.4. But first let us observe that Theorem 2 follows from Propositions 10 and 11 easily.

Proof of Theorem 2. The assumption (1.3) implies that $1-d/k=\Phi(0)>\Phi(1)=0$. Since $\boldsymbol{m}=\mathrm{Po}(dn/k)$, we thus obtain $n-\boldsymbol{m}=\Omega(n)$ w.h.p. Therefore, (2.9) implies that $\mathbb{E}_{\mathfrak{A}}\left[\boldsymbol{Z}\cdot\mathbb{1}\left\{\mathbb{A}\in\mathfrak{O}\right\}\right]\sim q^{n-\boldsymbol{m}}=q^{\Omega(n)}$ w.h.p. Hence, (2.11) implies together with Chebyshev's inequality that $\boldsymbol{Z}\geq\boldsymbol{Z}\mathbb{1}\{\mathbb{A}\in\mathfrak{O}\}=q^{\Omega(n)}$ w.h.p. Consequently, the random linear system $\mathbb{A}x=\boldsymbol{y}$ has a solution w.h.p., which implies that $\mathrm{rk}\,\mathbb{A}=\boldsymbol{m}$ w.h.p. ◀

2.3 Proof of Proposition 10: typical kernel vectors

The asymptotic rank formula (1.4) provides our point of departure toward the proof of Proposition 10. The basic idea is to show that (1.4) could not possibly be correct unless $\mathbb{A}\in\mathfrak{O}$ w.h.p. However, at closer inspection it turns out we cannot just apply (1.4) as is. Instead, we need to derive the analogue of (1.4) for a slightly enhanced random matrix from scratch.

Specifically, for an integer $t\geq 0$ obtain $\mathbb{A}_{[t]}$ from $\mathbb{A}$ by adding t more rows that each contain precisely three non-zero entries. The positions of these non-zero entries are chosen uniformly, mutually independently and independently of $\mathbb{A}$. The non-zero entries themselves are independent copies of $\boldsymbol{\chi}$. For this enhanced matrix we derive the following upper bound on its asymptotic rank.

▶ **Proposition 12.** *If* (1.3) *is satisfied then there exists* $\delta_0=\delta_0(\boldsymbol{d},\boldsymbol{k})>0$ *such that for all* $0<\delta<\delta_0$ *we have*

$$\limsup_{n\to\infty}\frac{1}{n}\mathbb{E}[\mathrm{nul}\,\mathbb{A}_{[\lfloor\delta n\rfloor]}]\leq 1-\frac{d}{k}-\delta. \tag{2.12}$$

The proof of Proposition 12 relies on the so-called “Aizenman-Sims-Starr scheme” [4], a coupling argument inspired by spin glass theory that also constituted the cornerstone of the derivation of (1.4) in [9]. That said, a subtle modification of this argument is necessary to accommodate the additional ternary equations. A vital assumption towards the proof of Proposition 12 is that the function Φ from (1.2) attains its *unique* global max at $z = 0$. In fact, the proof of Proposition 12 is the only place where the uniqueness of the maximiser is required.

How does Proposition 10 follow from Proposition 12? Assuming (1.3), we obtain from (1.4) that

$$\frac{1}{n}\operatorname{nul}\mathbb{A} \sim 1 - \frac{d}{k} \qquad \text{w.h.p.} \tag{2.13}$$

Now suppose that we add $\lfloor \delta n \rfloor$ extra ternary rows to $\mathbb{A}$ to obtain $\mathbb{A}_{[\lfloor \delta n \rfloor]}$. Comparing (2.12) and (2.13), we conclude that all but $o(n)$ of these extra rows decrease the nullity by one. Indeed, adding a single row cannot decrease the nullity by more than one, and routine arguments show that $\operatorname{nul}\mathbb{A}_{[\lfloor \delta n \rfloor]}$ concentrates about its expectation.

But a drop in nullity of $\delta n + o(n)$ w.h.p. is conceivable only if $\mathbb{A} \in \mathfrak{O}$ w.h.p. To see this, let us contemplate the kernel of a general $M \times N$ matrix A over $\mathbb{F}_q$ for a brief moment. Draw $\boldsymbol{x}_A = (\boldsymbol{x}_{A,i})_{i \in [N]} \in \ker A$ uniformly at random. For any given coordinate $\boldsymbol{x}_{A,i}$, $i \in [N]$ there are two possible scenarios: either $\boldsymbol{x}_{A,i} = 0$ with probability one, or $\boldsymbol{x}_{A,i}$ is uniformly distributed over $\mathbb{F}_q$. To see this, consider a basis $\zeta_1, \ldots, \zeta_h$ of the kernel of A. Then we can sample $\boldsymbol{x}_A$ by just multiplying each ζ_j with a random scalar $\boldsymbol{z}_j \in \mathbb{F}_q$ and summing up: $\boldsymbol{x}_A = \boldsymbol{z}_1\zeta_1 + \cdots + \boldsymbol{z}_h\zeta_h$. If the i-th coordinate of all ζ_j is zero, then $\boldsymbol{x}_{A,i} = 0$ deterministically; otherwise $\boldsymbol{x}_{A,i}$ is a sum of uniformly random elements of $\mathbb{F}_q$, and thus uniformly random itself. It therefore makes sense to call coordinate i *frozen* if $x_i = 0$ for all $x \in \ker A$, and unfrozen otherwise. Let $\mathfrak{F}(A)$ be the set of frozen coordinates.

If $\mathbb{A}$ had many frozen coordinates then adding an extra random row with three non-zero entries could hardly decrease the nullity w.h.p. For if all three non-zero coordinates fall into the frozen set, then we get the new equation “for free”, i.e., $\operatorname{nul}\mathbb{A}_{[1]} = \operatorname{nul}\mathbb{A}$. Thus, Proposition 12 implies that $|\mathfrak{F}(\mathbb{A})| = o(n)$ w.h.p. We conclude that $\boldsymbol{x}_{\mathbb{A},i}$ is uniformly distributed over $\mathbb{F}_q$ for all but $o(n)$ coordinates $i \in [n]$. However, this does not yet imply that $\boldsymbol{x}_{\mathbb{A},i}$, $\boldsymbol{x}_{\mathbb{A},j}$ are independent for most i, j, as required by $\mathfrak{O}$. Yet a slightly more careful deliberation based on linear algebra and the “pinning lemma” [9, Proposition 2.4] shows that $\mathbb{A} \in \mathfrak{O}$ w.h.p.

2.4 Proof of Proposition 11: expansion around the equitable solution

We prove Proposition 11 by way of expanding (2.3) carefully around the uniform distribution (2.5). Recall that once the $n_\ell(s)$ are set to the equitable solution (2.5), the optimal check variables $m_{\chi_1,\ldots,\chi_\ell}(s_1, \ldots, s_\ell)$ are given by (2.10). This observation by itself now suffices to conclude without (much) further ado that

$$\mathbb{E}_{\mathfrak{A}}[\boldsymbol{Z}^2 \cdot \mathbb{1}\{\mathbb{A} \in \mathfrak{O}\}] = O\left(\mathbb{E}_{\mathfrak{A}}[\boldsymbol{Z} \cdot \mathbb{1}\{\mathbb{A} \in \mathfrak{O}\}]^2\right). \tag{2.14}$$

The challenge is to sharpen this estimate so as to obtain the asymptotic equality claimed in (2.11). In his work on adjacency matrices of random regular graphs, Huang [22] actually faced a similar issue (with $\boldsymbol{d} = \boldsymbol{k}$ constant and q a prime number). To prove Proposition 11 we need to cope with the (significantly) more general situation of arbitrary $\boldsymbol{d}, \boldsymbol{k}$ and prime powers q. This improvement actually constitutes one of the main technical obstacles that we need to surmount toward the proof of Theorem 2.

The issue is that in order to eliminate the constant factor hidden in the $O(\,\cdot\,)$ in (2.14) we need to carefully consider divisibility properties that make it possible or impossible for a vector $x \in \mathbb{F}_q^n$ to belong to the kernel. These questions depend not only on the degree distributions $\boldsymbol{d}, \boldsymbol{k}$ but also on q and the distribution $\boldsymbol{\chi}$ of the non-zero entries. Hence, to estimate the kernel size precisely we need to crystallise the conceivable frequencies of field elements that may lead to solutions. Specifically, for an integer $\ell \geq 3$ and $\chi_1, \ldots, \chi_\ell \in \mathbb{F}_q \setminus \{0\}$ let

$$\mathcal{S}_q(\chi_1, \ldots, \chi_\ell) = \left\{ \sigma \in \mathbb{F}_q^\ell : \sum_{i=1}^{\ell} \chi_i \sigma_i = 0 \right\} \tag{2.15}$$

comprise all solutions to a linear equation with coefficients $\chi_1, \ldots, \chi_{k_0} \in \mathbb{F}_q$. Furthermore, for each $\sigma \in \mathcal{S}_q(\chi_1, \ldots, \chi_\ell)$ we define the vector

$$\hat{\sigma} = \left(\sum_{i=1}^{\ell} \mathbb{1}\left\{ \sigma_i = s \right\} \right)_{s \in \mathbb{F}_q \setminus \{0\}} \in \mathbb{Z}^{\mathbb{F}_q \setminus \{0\}} \tag{2.16}$$

to track the frequencies with which the various non-zero field elements appear. Moreover, let

$$\mathfrak{M}_q(\chi_1, \ldots, \chi_\ell) \subseteq \mathbb{Z}^{\mathbb{F}_q \setminus \{0\}}$$

be the $\mathbb{Z}$-module generated by the frequency vectors $\hat{\sigma}$ for $\sigma \in \mathcal{S}_q(\chi_1, \ldots, \chi_\ell)$. Thus, $\mathfrak{M}_q(\chi_1, \ldots, \chi_\ell) \subseteq \mathbb{Z}^{\mathbb{F}_q \setminus \{0\}}$ captures all conceivable frequency vectors of solutions σ to $\sum_{i=1}^{\ell} \chi_i \sigma_i$.

Depending on the coefficients $\chi_1, \ldots, \chi_\ell$, the module $\mathfrak{M}_q(\chi_1, \ldots, \chi_\ell)$ may be a proper submodule of the integer lattice $\mathbb{Z}^{\mathbb{F}_q \setminus \{0\}}$. For example, in the case $q = \ell = 3$ and $\chi_1 = \chi_2 = \chi_3 = 1$ the module $\mathfrak{M}_3(1,1,1)$ constitutes the sub-lattice spanned by $\binom{1}{1}$ and $\binom{0}{3}$, which is a proper sub-lattice of $\mathbb{Z}^2$. The following proposition characterises the lattice spanned by the frequency vectors for general $\chi_1, \ldots, \chi_\ell$. The determinant formula that the proposition provides shows that $\mathfrak{M}_q(\chi_1, \ldots, \chi_\ell)$ is a proper sub-module iff all the coefficients $\chi_1, \ldots, \chi_\ell$ coincide.

▶ **Proposition 13.** *Let $q \geq 2$ be a prime power, $\ell \geq 3$ and let $\chi_1, \ldots, \chi_\ell \in \mathbb{F}_q \setminus \{0\}$. Then $\mathfrak{M}_q(\chi_1, \ldots, \chi_\ell)$ has a basis $\mathfrak{b}_1, \ldots, \mathfrak{b}_{q-1}$ of non-negative integer vectors with $\|\mathfrak{b}_i\|_1 \leq 3$ for all $1 \leq i \leq q-1$ such that*

$$\det\left(\mathfrak{b}_1 \ \cdots \ \mathfrak{b}_{q-1} \right) = q^{\mathbb{1}\{\chi_1 = \cdots = \chi_\ell\}}.$$

A vital feature of Proposition 13 is that the module basis consists of non-negative integer vectors with small ℓ_1-norm. In effect, the basis vectors are "combinatorially meaningful" towards our purpose of counting solutions. Perhaps surprisingly, the proof of Proposition 13 turns out to be rather delicate, with details depending on whether q is a prime or a prime power, among other things.

In addition to the subgrid constraints imposed by the linear equations themselves, we need to take another divisibility condition into account. Indeed, for any assignment $\sigma \in \mathbb{F}_q^n$ of values to variables the frequencies of the various field elements $s \in \mathbb{F}_q$ are divisible by the g.c.d. $\mathfrak{d}$ of $\operatorname{supp}(\boldsymbol{d})$, i.e.

$$\mathfrak{d} \mid \sum_{i=1}^{n} \boldsymbol{d}_i \mathbb{1}\left\{ \sigma_i = s \right\} \qquad \text{for all } s \in \mathbb{F}_q. \tag{2.17}$$

Thus, to compute the expected kernel size we need to study the intersection of the subgrid (2.17) with the grid spanned by the frequency vectors $\hat{\sigma}$ for $\sigma \in \mathcal{S}_q(\boldsymbol{\chi}_{1,1}, \ldots, \boldsymbol{\chi}_{1,\boldsymbol{k}})$. Specifically, in order to derive Proposition 11 from Proposition 13 we need to estimate the number of vectors $\sigma \in \mathbb{F}_q^n$ represented by each grid point and calculate the ensuing satisfiability probability. This argument combines the Laplace method with local limit techniques.

3 Discussion

While there is a substantial body of work on dense random matrices where the average number of non-zero entries per row/column diverges or even is linear in the size of the matrix (e.g., [6, 7, 14, 15, 23, 24, 33, 34]), far less is known about sparse random matrices. The aim of this paper has been to determine sufficient (as well as necessary) conditions for a sparse random matrix to have *full* row rank. To this end we drew upon some of the elements of prior work on the *asymptotic* rank of random matrices [5, 9], specifically the formula (1.4). In particular, the proof of Proposition 12 adapts and extends the Aizenman-Sims-Starr scheme from [9]. Additionally, the expansion around the centre employs some of the techniques developed in the study of satisfiability thresholds, particularly the extensive use of local limit theorems [12, 11]. These also played a role in prior work on the adjacency matrices of random d-regular graphs [22, 29].

A principal new proof ingredient is the asymptotically precise analysis of the moment formula (2.3) for general $\boldsymbol{d}, \boldsymbol{k}, q$ around the equitable solution by means of the study of the sub-grids of the integer lattice induced by the constraints. This issue that was absent in the prior literature on variations on random k-XORSAT [5, 9, 13] and on other random constraint satisfaction problems [12, 11]. That said, in the study of the random regular matrix from Example 6 Huang [22] faced a similar issue in the special case $\boldsymbol{d} = \boldsymbol{k}$ constant and $\boldsymbol{\chi} = 1$ deterministically. Proposition 13, whose proof is based on a combinatorial investigation of lattices in the general case, constitutes a considerable generalisation of this case. A further new feature of the proof of Proposition 13 is the explicit ℓ_1-bound on the basis vectors, which greatly facilitates the proof of Theorem 2.

Satisfiability thresholds of random constraint satisfaction problems have been studied extensively in the statistical physics literature via a non-rigorous technique called the “cavity method”. The cavity method comes in two installments: the simpler “replica symmetric ansatz” associated with the Belief Propagation message passing scheme, and the more intricate “replica symmetry breaking ansatz”. The proof of Theorem 2 demonstrates that the former renders the correct prediction as to the satisfiability threshold of random linear equations. By contrast, in quite a few problems, notoriously random k-SAT, replica symmetry breaking occurs [10, 17], requiring a substantially different proof strategy.

A natural question is whether the methods presented in this work can be extended to the adjacency matrices of random graphs. Apart from the aforementioned works regarding the regular case [22, 29] and the work of Bordenave, Lelarge and Salez [8], an exciting recent contribution by Glasgow, Kwan, Sah and Sawhney deals with the precise connection between the matching number and the rank [20]. By contrast to the present work, these contributions rely on local weak convergence and/or Littlewood-Offord techniques; see also [19]. Furthermore, recently the methods from [9] were extended to obtain a rank formula for the adjacency matrices of Erdös-Rényi graphs over arbitrary fields [35]. In fact, the consideration of general fields reveals new phenomena, as was already discovered in some of the earlier literature [6, 7, 25, 26, 27, 28].

References

1 D. Achlioptas and F. McSherry. Fast computation of low-rank matrix approximations. *J. ACM*, 54(2):9, 2007.

2 D. Achlioptas and C. Moore. Random k-sat: Two moments suffice to cross a sharp threshold. *SIAM J. Comput.*, 36(3):740–762, 2006.

3 D. Achlioptas, A. Naor, and Y. Peres. Rigorous location of phase transitions in hard optimization problems. *Nature*, 435(7043):759–764, June 2005.

4 M. Aizenman, R. Sims, and S. L. Starr. Extended variational principle for the sherrington-kirkpatrick spin-glass model. *Phys. Rev. B*, 68:214403, 2003.

5 P. J. Ayre, A. Coja-Oghlan, P. Gao, and N. Müller. The satisfiability threshold for random linear equations. *Comb.*, 40(2):179–235, 2020.

6 G. V. Balakin. The distribution of the rank of random matrices over a finite field. *Theory of Probability & Its Applications*, 13(4):594–605, 1968.

7 J. Blömer, R. Karp, and E. Welzl. The rank of sparse random matrices over finite fields. *Random Struct. Algorithms*, 10(4):407–419, 1997.

8 C. Bordenave, M. Lelarge, and J. Salez. The rank of diluted random graphs. *The Annals of Probability*, 39(3):1097–1121, 2011.

9 A. Coja-Oghlan, A. A. Ergür, P. Gao, S. Hetterich, and Rolvien M. The rank of sparse random matrices. *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020*, pages 579–591, 2020.

10 A. Coja-Oghlan, N. Müller, and J.B. Ravelomanana. Belief propagation on the random k-sat model. *CoRR*, abs/2011.02303, 2020. arXiv:2011.02303.

11 A. Coja-Oghlan and K. Panagiotou. Catching the k-naesat threshold. *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 899–908, 2012.

12 A. Coja-Oghlan and K. Panagiotou. The asymptotic k-sat threshold. *Advances in Mathematics*, 288:985–1068, 2016.

13 C. Cooper, A. M. Frieze, and W. Pegden. On the rank of a random binary matrix. *Electron. J. Comb.*, 26(4):4, 2019.

14 K. P. Costello and V. H. Vu. The rank of random graphs. *Random Struct. Algorithms*, 33(3):269–285, 2008.

15 K. P. Costello and V. H. Vu. On the rank of random sparse matrices. *Comb. Probab. Comput.*, 19(3):321–342, 2010.

16 M. Dietzfelbinger, A. Goerdt, M. Mitzenmacher, A. Montanari, R. Pagh, and M. Rink. Tight thresholds for cuckoo hashing via XORSAT. *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP 2010)*, 6198:213–225, 2010.

17 J. Ding, A. Sly, and N. Sun. Proof of the satisfiability conjecture for large k. *Proceedings of the 47th Annual ACM on Symposium on Theory of Computing (STOC 2015)*, pages 59–68, 2015.

18 O. Dubois and J. Mandler. The 3-xorsat threshold. *43rd Symposium on Foundations of Computer Science (FOCS 2002)*, pages 769–778, 2002.

19 A. Ferber, M. Kwan, A. Sah, and M. Sawhney. Singularity of the k-core of a random graph. *Duke Mathematical Journal*, 172(7):1293–1332, 2023.

20 M. Glasgow, M. Kwan, A. Sah, and M. Sawhney. The exact rank of sparse random graphs. *arXiv preprint*, 2023. arXiv:2303.05435.

21 A. Goerdt and L. Falke. Satisfiability thresholds beyond k- xorsat. *Proceedings of the 7th International Computer Science Symposium in Russia (CSR 2012)*, pages 148–159, 2012.

22 J. Huang. Invertibility of adjacency matrices for random d-regular graphs. *Duke Mathematical Journal*, 170(18):3977–4032, 2021.

23 J. Kahn, J. Komlós, and E. Szemerédi. On the probability that a random±1-matrix is singular. *Journal of the American Mathematical Society*, 8(1):223–240, 1995.

24 J. Komlós. On the determinant of (0-1) matrices. *Studia Scientiarium Mathematicarum Hungarica*, 2:7–21, 1967.

25 I. Kovalenko. On the limit distribution of the number of solutions of a random system of linear equations in the class of boolean functions. *Theory of Probability & Its Applications*, 12(1):47–56, 1967.

26 I. Kovalenko, A.A. Levitskaya, and MN Savchuk. Selected problems in probabilistic combinatorics. *Naukova Dumka, Kiev*, 1986.

27 A. A. Levitskaya. Invariance theorems for a system of random linear equations over an arbitrary finite ring. *Doklady Akademii Nauk*, 263(2):289–291, 1982.

28 A. A. Levitskaya. The probability of consistency of a system of random linear equations over an arbitrary finite ring. *Theory of Probability & Its Applications*, 30(2):364–375, 1986.

29 A. Mészáros. The distribution of sandpile groups of random regular graphs. *Transactions of the American Mathematical Society*, 373(9):6529–6594, 2020.

30 G. Miller and G. D. Cohen. The rate of regular LDPC codes. *IEEE Trans. Inf. Theory*, 49(11):2989–2992, 2003.

31 B. Pittel and G. Sorkin. The satisfiability threshold for k-xorsat. *Combinatorics, Probability and Computing*, 25(2):236–268, 2016.

32 T. Richardson and R. Urbanke. *Modern Coding Theory.* Cambridge University Press, 2008.

33 T. Tao and V. H. Vu. On the singularity probability of random bernoulli matrices. *Journal of the American Mathematical Society*, 20, February 2005.

34 K. Tikhomirov. Singularity of random bernoulli matrices. *Annals of Mathematics*, 191:593, March 2020.

35 R. van der Hofstad, N. Müller, and H. Zhu. The rank of sparse symmetric matrices over arbitrary fields. *arXiv preprint*, 2023. `arXiv:2301.12978`.

36 M. J. Wainwright, E. N. Maneva, and E. Martinian. Lossy source compression using low-density generator matrix codes: analysis and algorithms. *IEEE Trans. Inf. Theory*, 56(3):1351–1368, 2010.

Tighter $\mathrm{MA}/1$ Circuit Lower Bounds from Verifier Efficient PCPs for PSPACE

Joshua Cook
Department of Computer Science, University of Texas Austin, TX, USA

Dana Moshkovitz
Department of Computer Science, University of Texas Austin, TX, USA

Abstract

We prove that for some constant $a > 1$, for all $k \leq a$,

$$\mathbf{MATIME}[n^{k+o(1)}]/1 \not\subset \mathbf{SIZE}[O(n^k)],$$

for some specific $o(1)$ function. This is a super linear polynomial circuit lower bound.

Previously, Santhanam [29] showed that there exists a constant $c > 1$ such that for all $k > 1$:

$$\mathbf{MATIME}[n^{ck}]/1 \not\subset \mathbf{SIZE}[O(n^k)].$$

Inherently to Santhanam's proof, c is a large constant and there is no upper bound on c. Using ideas from Murray and Williams [26], one can show for all $k > 1$:

$$\mathbf{MATIME}[n^{10k^2}]/1 \not\subset \mathbf{SIZE}[O(n^k)].$$

To prove this result, we construct the first **PCP** for $\mathbf{SPACE}[n]$ with quasi-linear verifier time: our **PCP** has a $\tilde{O}(n)$ time verifier, $\tilde{O}(n)$ space prover, $O(\log(n))$ queries, and polynomial alphabet size. Prior to this work, **PCP**s for $\mathbf{SPACE}[O(n)]$ had verifiers that run in $\Omega(n^2)$ time. This **PCP** also proves that **NE** has **MIP** verifiers which run in time $\tilde{O}(n)$.

2012 ACM Subject Classification Theory of computation → Complexity classes; Theory of computation → Circuit complexity; Theory of computation → Interactive proof systems

Keywords and phrases MA, PCP, Circuit Complexity

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.55

Category RANDOM

Related Version *Full Version*: https://eccc.weizmann.ac.il/report/2022/014/

Funding This material is based upon work supported by the National Science Foundation under grant number 1705028.

1 Introduction

Some of the most fundamental problems in complexity theory are proving circuit lower bounds for uniform complexity classes. One such conjecture is that **NP** does not have polynomial size circuits, which is a strong version of $\mathbf{P} \neq \mathbf{NP}$. Very little is known on such lower bounds. In particular, there are no known proofs that **NEXP** does not have polynomial sized circuits! However, there are some closely related results that could be loosely seen as relaxations.

One can strengthen **NP** slightly by giving the non-deterministic algorithm access to randomness, as well as an extra bit of trusted advice. This gives the complexity class $\mathbf{MA}/1$. We can weaken polynomial sized circuits to circuits of fixed polynomial size: $\mathbf{SIZE}[n^k]$ for constant k.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 55; pp. 55:1–55:22
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Santhanam [29] proved that for any constant k, $\mathbf{MA}/1 \nsubseteq \mathbf{SIZE}[n^k]$. The $\mathbf{MA}/1$ algorithm runs in time n^{ck} for a large $c > 1$. In fact, inherently to Santhanam's proof, there is no upper bound on c (We will explain why when we describe Santhanam's proof in Section 1.2.1). One can use ideas from Murray and Williams [26] to get, for some explicit c with $2 < c < 10$, the result $\mathbf{MATIME}[n^{ck^2}]/1 \not\subset \mathbf{SIZE}[n^k]$.

The goal of this paper is to prove a fine grained separation of $\mathbf{MA}/1$ from fixed polynomial size circuits, namely,

$$\mathbf{MATIME}[n^{k+o(1)}]/1 \not\subset \mathbf{SIZE}[n^k].$$

We believe that the gold standard for separations should be fine grained separations. Fine grained separations are necessary for key results in complexity theory, e.g., Williams' program (See, e.g., [33]) and optimal derandomization [14].

Some fine grained separations are known, namely, hierarchy theorems that show that giving algorithms more time allows them to solve more problems [18, 11]. Hierarchy theorems are known for many complexity classes. While no hierarchy theorems are known for $\mathbf{MA}$, they are known for $\mathbf{MA}/1$. Fortnow, Santhanam, and Trevisan showed that $\mathbf{MA}$ with a small amount of advice can solve more problems when given more time [16]. Van Melkebeek and Pervyshev showed that for any $1 < b < d$, $\mathbf{MATIME}[n^b]/1 \subsetneq \mathbf{MATIME}[n^d]/1$ [32]. In particular, they imply that even $\mathbf{MATIME}[n^{2k}]/1$ is much larger than $\mathbf{MATIME}[n^{k+o(1)}]/1$.

1.1 Results

In this work, we give a fine grained separation for $\mathbf{MA}/1$ and $\mathbf{SIZE}[n^k]$. We show that for at least some $k > 1$, there is an $\mathbf{MA}$ protocol with one bit of advice whose verifier has time almost n^k such that any circuit solving the same problem also requires size n^k. Formally:

▶ **Theorem 1** (Fine Grained $\mathbf{MA}$ Lower Bound)**.** *There exists a constant $a > 1$, such that for all $k < a$, for some $f(n) = o(1)$,*

$$\mathbf{MATIME}[O(n^{k+f(n)})]/1 \not\subset \mathbf{SIZE}[O(n^k)].$$

We stress that we give **super linear** polynomial lower bounds. Our result holds for some k *strictly* greater than 1, even though we don't know which k. This result removes the large polynomial factor in the gap between the $\mathbf{MA}/1$ time and the circuit size in Santhanam's result. It may be the case that a is small, like $a = 1.0001$. But in that case, we get the following result for all k:

▶ **Theorem 2** ($\mathbf{MA}$ Lower Bound for Small a)**.** *If the a from Theorem 1 is finite, then for all $k > 0$, for some $f(n) = o(1)$,*

$$\mathbf{MATIME}[O(n^{ak+f(n)})]/1 \not\subset \mathbf{SIZE}[O(n^k)].$$

This gives us a win-win scenario: if a is large, we get a strong result for a large range of k, but if a is small we get a similar result for all k.

When we describe our proof we will explain why we only get separations for $k < a$ for an (unknown) $a > 1$ and not for all $k > 1$. For now we would like to stress that: (1) under plausible complexity assumptions the upper bound a is in fact super-constant in n; (2) even the case of a constant $a > 1$ as promised in our theorem is highly interesting, since it is unknown how to prove that $\mathbf{NP} \nsubseteq \mathbf{SIZE}[n^k]$ for *any* $k > 1$.

Santhanam's original proof uses an interactive protocol for $\mathbf{PSPACE}$. To prove our circuit lower bound, we replace the interactive protocol with a new, more efficient $\mathbf{PCP}$.

To get our fine grained results, we need a **PCP** for space $S = O(n)$ and time $T = 2^{O(n)}$ algorithms, where the verifier simultaneously has $\tilde{O}(n)$ time and $\mathbf{poly}(\log(n))$ many queries. Further, the **PCP** needs a prover that can compute any bit of the proof in $\tilde{O}(n)$ space. Notably, we do not need any bounds on the proof length.

The **PCP** given by Babai, Fortnow, and Lund in their proof that **MIP** = **NEXP** [4] required $\Omega(\log(T))$ queries, while we want $O(\log(\log(T)))$ queries.

Holmgren and Rothblum in their work on delegated computation [20] improved on the BFL **PCP** in several ways that can[1] be used to give a **PCP** with verifier time $\tilde{O}(n+\log(T))$. Unfortunately, it still requires $\Omega(\log(T))$ queries.

Ben-Sasson, Goldreich, Harsha, Sudan, and Vadhan [5] gave a **PCP** that uses a constant number of queries, but has verifier time $\mathbf{poly}(\log(T))$, while we need $\tilde{O}(n+\log(T))$ verifier time. Similar results were given by subsequent work [23, 8, 6].

The small space requirement for the prover is achieved by Holmgren and Rothblum [20]. In some **PCP**s, like the **PCP** in Ben-Sasson, Chiesa, Genkin, and Tromer's work on the concrete efficiency of **PCP**s [6], the prover requires space $\Omega(T)$. In contrast, our result needs prover space $\tilde{O}(S+n)$.

A sufficiently efficient **PCP** was not known, so we construct a new **PCP**.

▶ **Theorem 3** (Verifier Efficient **PCP**). *Let $S, T = \Omega(n)$ be functions, and L be any language computed by a simultaneous time T and space S algorithm. Let $\delta \in (0, 1/2)$ be a constant. Then there is a* **PCP** *for L with:*

1. *Verifier time $\tilde{O}(n+\log(T))$.*
2. *Query time $\tilde{O}(\log(T))$.*
3. *$O(\log(n)+\log(\log(T)))$ queries.*
4. *Alphabet Σ with $\log(|\Sigma|) = O(\log(\log(T)))$.*
5. *Log of proof length $\tilde{O}(\log(T))$.*
6. *Prover space $\tilde{O}(S)$.*
7. *Perfect completeness and soundness δ.*

We believe we can achieve a similar verifier time, query time and prover space while also achieving constant number of queries and alphabet size. We do not need these improvements for our main result, so we only prove this simpler result.

Only our prover requires the space bound for its efficient computation. If we remove this space limitation, we get a similar **PCP** for nondeterministic algorithms.

▶ **Theorem 4** (Verifier Efficient **PCP** for Nondeterministic Algorithms). *Let $T = \Omega(n)$, $\delta \in (0, 1/2)$ be a constant, and $L \in \mathbf{NTIME}[T]$. Then there is a* **PCP** *for L with:*

1. *Verifier time $\tilde{O}(n+\log(T))$.*
2. *Query time $\tilde{O}(\log(T))$.*
3. *$O(\log(n)+\log(\log(T)))$ queries.*
4. *Alphabet Σ with $\log(|\Sigma|) = O(\log(\log(T)))$.*
5. *Log of proof length $\tilde{O}(\log(T))$.*
6. *Perfect completeness and soundness δ.*

An immediate corollary of Theorem 4 is a more fine grained equivalence between **MIP** and **NEXP**.

[1] The **PCP** constructed by Holmgren and Rothblum was built to have no signalling soundness and has many steps that take longer than $\tilde{O}(\log(T))$ time to compute. Still, the basic elements of of their **PCP** needed for a standard **PCP** are computable in $\tilde{O}(n+\log(T))$ time.

▶ **Corollary 5** (Fine Grained Equivalence of $\mathbf{MIP} = \mathbf{NEXP}$)**.** *For any time constructable function $p(n) = \Omega(n)$, language $L \in \mathbf{NTIME}[2^{\tilde{O}(p(n))}]$ if and only if there is a two prover, one round* **MIP** *protocol for L whose verifier runs in time $\tilde{O}(p(n))$.*

Note this equivalence implies a hierarchy theorem for **MIP** since there are hierarchy theorems for **NTIME** [11, 30, 34, 15].

A special case is **MIP** protocols for **NE**.

▶ **Corollary 6** (**NE** Has Quasi-linear Time Verifiers)**.** *For any language $L \in$* **NE***, there is a two prover, one round* **MIP** *protocol for L whose verifier runs in time $\tilde{O}(n)$.*
Note this verifier time is nearly optimal since the verifier requires linear time to read its entire input.

All previous **PCP**s fail to achieve such an efficient **MIP** verifier. If the original **PCP** makes $\Omega(n)$ queries of size $\Omega(n)$, then it takes $\Omega(n^2)$ time to send the queries even if we allow more provers. And all previous **PCP**s with fewer queries require verifier time $\Omega(n^2)$ to either verify the response or compute the queries.

1.2 Proof Idea

1.2.1 MA Lower Bounds Using PCP

We first review Santhanam's original proof.

Santhanam's original result uses the fact that if $\mathbf{PSPACE} \subset \mathbf{P/poly}$, then $\mathbf{PSPACE} = \mathbf{MA}$. This follows from the famous result that $\mathbf{IP} = \mathbf{PSPACE}$ [31, 22]. The idea is that if $\mathbf{PSPACE} \subset \mathbf{P/poly}$, then an **MA** protocol can guess a circuit computing any problem in **PSPACE**. The prover in the interactive protocol for **PSPACE** is also computable in **PSPACE**. So to solve any **PSPACE** problem in **MA**, the **MA** protocol first guesses the circuit for a prover, then simulates the verifier using the circuit we guessed as the prover.

Using this, Santhanam's original proof then considered two cases: either $\mathbf{PSPACE} \subset \mathbf{P/poly}$, or $\mathbf{PSPACE} \not\subset \mathbf{P/poly}$.

If $\mathbf{PSPACE} \subset \mathbf{P/poly}$, then we already know $\mathbf{PSPACE} = \mathbf{MA}$. Now we just need a problem not computable by a size n^k circuit. But there is a straightforward algorithm that exhaustively finds a circuit of size larger than n^k that computes a function that cannot be computed by a smaller circuit. In fact, such an algorithm only requires space $\tilde{O}(n^k)$. So $\mathbf{PSPACE} \not\subset \mathbf{SIZE}[n^k]$. In this case, $\mathbf{PSPACE} = \mathbf{MA}$, so $\mathbf{MA} \not\subset \mathbf{SIZE}[n^k]$.

If $\mathbf{PSPACE} \not\subset \mathbf{P/poly}$, then we know a hard problem that is not in $\mathbf{SIZE}[n^k]$, namely any **PSPACE** complete problem. Let us take a **PSPACE** complete, downward self reducible language, Y. Now Y may be too hard for **MA** to solve, but if we give it enough padding, eventually the padded version of Y will be computable by size n^k circuits. But for this amount of padding, **MA** can pull the same trick it does in the $\mathbf{PSPACE} \subset \mathbf{P/poly}$ case. Namely, guess a circuit for Y and then simulate the **IP** protocol for Y. For some **PSPACE** complete Y, the language itself is its proof and this works. The trick is to use just the right amount of padding so it requires circuits of at least size n^k, but not much larger. Santhanam uses the single bit of advice in a clever way to figure out when there is just the right amount of padding.

In either case, the time of this protocol is roughly the time of the verifier in the **IP** protocol, plus the size of the prover circuit times the number of times the prover is queried.

There are two reasons the **MA** protocol could take polynomially more time than the size of the circuits it wants to compute in the case $\mathbf{PSPACE} \subset \mathbf{P/poly}$. One is that the **IP** from the original Santhanam result has polynomial verifier time and a polynomial time interaction with the prover, making the verifier in the $\mathbf{MA}/1$ protocol take polynomially longer than the circuit complexity of the problem being solved. By using a **PCP**, we get better results. The other is that the prover circuit complexity could be large, depending on the circuit size required for **PSPACE** (could be any polynomial when **PSPACE** $\subset$ **P/poly**). This is the reason there is no upper bound on the polynomial run time of the **MA**/1 protocol in Santhanam's proof. To avoid this issue we consider a finer case analysis.

We break the problem into three cases. For some **SPACE**$[O(n)]$ complete language, X, we have one[2] of the following:

1. $X \notin$ **P/poly**.
2. $X \in$ **SIZE**$[n^{1+o(1)}]$.
3. $X \in$ **SIZE**$[n^{a+o(1)}] \setminus$ **SIZE**$[n^{a-o(1)}]$ for some $a > 1$.

The original proof only used the two cases $X \notin$ **P/poly** and $X \in$ **P/poly**. The case where $X \notin$ **P/poly** is completely unchanged. Note that this is the plausible case, and here there is no constant upper bound a on k.

If $X \in$ **P/poly**, we use our efficient **PCP**, Theorem 3, instead of the **IP** Santhanam uses. With this substitution, the case where $X \in$ **SIZE**$[n^{1+o(1)}]$ is almost unchanged from the original proof. By separating this into it's own case, we get tight bounds for all k in this case.

If $X \in$ **SIZE**$[n^{a+o(1)}] \setminus$ **SIZE**$[n^{a-o(1)}]$ for some $a > 1$, then we use the same padding technique we use if $X \notin$ **P/poly**, just using our new **PCP**. In this case, we can only do this if for some $k < a$, we are trying to show **MATIME**$[n^{k+o(1)}]/1 \not\subset$ **SIZE**$[n^{k-o(1)}]$. This is the case where a is finite, but in this case, we can use Santhanam's argument using our **PCP** to get Theorem 2.

To see why $k > a$ poses a difficulty, suppose **SPACE**$[O(n)] \not\subseteq$ **SIZE**$[o(n^2)]$, but **SPACE**$[O(n^2)] \subseteq$ **SIZE**$[O(n^2)]$. Then to get a language requiring size n^3 circuits, we need to use a space n^3 algorithm. But the prover for a space n^3 language is a language running on an input with length n^3, and using space linear in its input length. Thus we may need a size $(n^3)^2 = n^6$ circuit for our prover. So the verifier takes time at least n^6 to even read the prover circuit, thus can't run in time n^3. See Item 2 in our open problems for further explanation.

▶ Remark 7. We note our verifier in Theorem 1 is a RAM machine, *not* a standard Turing Machine. This is because we know how to efficiently simulate a circuit on a RAM machine, but not on a standard Turing Machine.

1.2.2 Verifier Efficient PCP

Now we explain the **PCP** we actually use in the **MA** protocol. We start with a **PCP** similar to [20] and [4] that we refer to as our base **PCP**. This **PCP** has a verifier that runs in time $\tilde{O}(n + \log(T))$ and uses $O(\log(T))$ queries. To reduce the number of queries, we use **PCP** composition [3, 7, 13, 25, 12].

To perform **PCP** composition, we need a robust **PCP**. Loosely, a robust **PCP** is a **PCP** so that when $x \notin L$, for any proof, most sets of queries to that proof return not only a rejected response, but a response that is far from any accepted response. To make our base **PCP** robust, we use the aggregation through curves technique [2]. Now we briefly explain how to use aggregation through curves to convert our base **PCP** into a robust **PCP**.

An honest proof for our base **PCP** is a single low degree polynomial. Suppose our base **PCP** has q queries. To make our **PCP** robust, we first choose the randomness for the base **PCP**, and another random point in the **PCP** proof. Then we find the degree q curve that

[2] This is a trichotomy in an asymptotic sense: for every constant a, either $X \in$ **SIZE**$[O(n^a)]$ or it is not. See Section 3.5 for details.

goes through all these points. Then we check if the proof, restricted to this curve, is a low degree polynomial, and whether the base **PCP** would have accepted on this input. Since a low degree polynomial is an error correcting code, this gives robustness.

One concern one might have with this robust **PCP** is that it actually requires $\Omega(\log(T)^2)$ queries. We don't need to actually calculate all of these query locations. Since we reduce the actual number of queries with **PCP** composition, we only need to be able to calculate any individual query location quickly. To find these query locations requires us to compute a point on the degree q curve going through each of our q points our base **PCP** queries plus a random point. In our base **PCP**, $q = O(\log(T))$ and our proof has dimension $O(\log(T))$. So the naive way to compute this curve is to calculate each coordinate independently, which would take time $\tilde{O}(\log(T)^2)$.

To efficiently compute low degree curves through points, or to extrapolate a function going through those points, we introduce the concept of time extrapolatable functions.

▶ **Definition 8** (Extrapolatability). *For any $n, q, t > 0$, and field $\mathbb{F}$, we call $Q : [q] \to \mathbb{F}^n$ "t extrapolatable" (or time t extrapolatable) if there is a time t algorithm taking any $v \in \mathbb{F}^q$, that outputs*

$$\sum_{i \in [q]} v_i Q(i).$$

Equivalently, if we think of Q as outputting the columns of a matrix, then we say Q is time t extrapolatable if one can multiply a vector with it in time t. An important property of extrapolatable functions is that an extrapolation of an extrapolatable function can be computed efficiently. This is where it gets its name.

Our base **PCP** is just a sum check and a few point checks. Each of these are time $\tilde{O}(\log(T))$ extrapolatable. Our robust **PCP** only queries locations easily computable given the extrapolation of our base **PCP** query locations. Extrapolations of extrapolatable functions are easy to compute, so we can easily compute the query locations of the robust **PCP**.

We also introduce the concept an extrapolatable **PCP** (**ePCP**) as one where an honest proof is a low degree polynomial, and the query locations after fixing a choice of randomness are extrapolatable. We show that any **ePCP** can be extended into a robust **PCP** where the query locations of that robust **PCP** can be computed efficiently.

1.3 Generalization And Sharpness

We actually prove a stronger result than Theorem 1 that is sharp. First, our **MA** protocol is input oblivious: the message from Merlin is just a program for computing a **PSPACE** complete language and doesn't depend on the specific input, just its length. Second, the hardness is against the model used in Merlin's message. We used circuits, but we can describe a randomized algorithm directly to save some polynomial factors.

We define input oblivious Merlin-Arthur time, **OMATIME**, the same way as Fortnow, Santhanam, and Williams [17]. Input oblivious Merlin-Arthur are languages solvable with untrusted advice, where the advice only depends on the input length. In our case, Merlin gets to send a long, untrusted message for every input length, and Arthur also gets a single bit of trusted advice. Note that Santhanam's original proof implicitly also uses input oblivious **MA**.

The main property of circuits we use is that a randomized algorithm can efficiently simulate it. We can instead use $\mathbf{BPTIME}[n^k]/n^k$, that is, randomized algorithms running in time n^k with description length n^k. This uses the same model of computation as our verifier, allowing it to more efficiently simulate **OMATIME**.

Using **OMATIME** instead of **MATIME** and **BPTIME** instead of **SIZE**, we can follow the same proof as our main result to show:

▶ **Theorem 9** (**OMATIME** Lower Bound Against **BPTIME**). *There exists constant $a > 1$, such that for all $k < a$, for some $f(n) = o(1)$,*

$$\mathbf{OMATIME}[O(n^{k+f(n)})]/1 \not\subset \mathbf{BPTIME}[O(n^k)]/O(n^k).$$

This result is tight in the sense that for any function $f(n)$, we have

$$\mathbf{OMATIME}[f(n)]/1 \subseteq \mathbf{BPTIME}[O(f(n))]/(f(n)+1).$$

To get stronger results, we need to use nondeterminism that depends on the input. So one could say our result is less about the power of nondeterminism, and more about the power of trusted versus untrusted advice. Specifically: trusting advice doesn't always buy (much) time in the randomized setting, as long as we have *some* trusted advice.

Let us briefly outline what would need to change in our main proof to prove Theorem 9, and justify why those changes would work.

First we need to make a class of randomized programs that act more like circuits. Consider the class of programs, $\mathcal{C}$, that contain randomized algorithms that work only a specific input length. For any $C \in \mathcal{C}$, we say $C(x)$ is the random variable that simulates C on input x for time $|C|$, and outputs what C does if C terminates in time $|C|$, and outputs 0 otherwise. See that $\mathcal{C}$ behaves like circuits in the following important ways:

1. Given a program $C \in \mathcal{C}$, a randomized algorithm can calculate the random variable $C(x)$ in time $O(|C|)$.
2. For any function $f(n)$ and language $L \in \mathbf{BPTIME}[f(n)]/f(n)$, for every n, there is a $C_n \in \mathcal{C}$ such that $|C_n| = O(f(n))$ and with high probability $C_n(x) = 1_{x \in L}$.

A few notes on using $\mathcal{C}$ in our proof, as opposed to circuits.

1. First, see that $\mathbf{SPACE}[O(n^k)] \not\subseteq \mathbf{BPTIME}[o(n^k)]/o(n^k)$. This follows using the same exhaustive search type algorithm used for $\mathbf{SIZE}[o(n^k)]$.
   For any polynomial n^k, there is a deterministic program, A, with length $O(n^k)$ running in time $O(n^k)$, but is not computable with high probability by any program $C \in \mathcal{C}$ with length $o(n^k)$. This follows from a simple counting argument: length n^k deterministic programs contain more than $2^{\alpha n^k}$ functions for some constant α (just use some lookup table), while there are only $2^{\alpha n^k}$ length αn^k programs.
   Such an A can still be found by exhaustive search in space $O(n^k)$, since given $C \in \mathcal{C}$, we can space efficiently check every choice of randomness and calculate majority. This gives us that
   $$\mathbf{SPACE}[O(n^k)] \not\subseteq \mathbf{BPTIME}[o(n^k)]/o(n^k).$$
2. Given that $L \in \mathbf{BPTIME}[f(n)]/f(n)$, then for any n, there is some advice (notably, a program $C_n \in \mathcal{C}$ with size $|C_n| \leq f(n)$) such that a randomized algorithm given that advice can compute whether $x \in L$ with probability $1 - \epsilon$ in time $O(f(n)\log(\frac{1}{\epsilon}))$.
   This allows our verifier to efficiently compute a $\mathbf{SPACE}[O(n)]$ complete problem, L, in time nearly $f(n)$ if $L \in \mathbf{BPTIME}[f(n)]/f(n)$, given correct advice.
3. We also note that for some programs $C \in \mathcal{C}$, for some inputs x, our program C evaluated on x may answer one or zero with very close to half probability. That is, the syntax of $\mathcal{C}$ does not only give bounded error randomized algorithms, just a randomized algorithm. This is not an issue, because in the completeness case, there will be a program C that does have bounded error the prover should provide. And in the soundness case, the soundness of our **PCP** holds against any multi-prover strategy, even a randomized strategy. So no program provided will convince the verifier with high probability.

Finally, see that in all cases of our proof, Arthur only asks Merlin for a program computing some $\mathbf{SPACE}[O(n)]$ complete problem. This advice does not depend on the specific input, only on the input size.

2 Preliminaries

We assume some familiarity with basic complexity theory. See Arora and Barak's book for background [1]. In this paper, by algorithm, we mean algorithm on a RAM machine, and by circuit, we mean a fan in 2 circuit with unbounded depth. A randomized algorithm is a deterministic algorithm with an extra input for randomness. We will assume in this paper that all time and space bounds for algorithms are sufficiently easily computable.

Now recall that **MA** is the complexity class of problems with polynomial sized certificates that can be verified with bounded error by a randomized, polynomial time algorithm. This is like **NP** with a randomized verifier. Then we define **MATIME** in an analogous way to **NTIME**. Our results have perfect completeness, so we only define **MATIME** with perfect completeness.

▶ **Definition 10** ($\mathbf{MATIME}/1$). *For any function $f : \mathbb{N} \to \mathbb{N}$, define $\mathbf{MATIME}[f(n)]/1$ as the set of languages, L, such that there is a function $b : \mathbb{N} \to \{0,1\}$ and a time $f(n)$ randomized algorithm M taking four inputs, an input x, a random input r, a witness w, and an advice bit such that*

Completeness *If $x \in L$ and $n = |x|$, then there exists w with $|w| \leq f(n)$ such that*

$$\Pr_r[M(x, r, w, b(n)) = 1] = 1.$$

Soundness *If $x \notin L$ and $n = |x|$, then for every w,*

$$\Pr_r[M(x, r, w, b(n)) = 1] < 1/2.$$

We let **SIZE** denote the class of languages with circuits of a given size.

▶ **Definition 11** (**SIZE**). *For any function $f : \mathbb{N} \to \mathbb{N}$, $\mathbf{SIZE}[f(n)]$ is the class of languages, L, where for each input length n, there is a circuit of size $f(n)$ with n inputs computing L for inputs of length n.*

Further, $\mathbf{SIZE}[O(f(n))]$ is the class of languages, L, such that for some $g(n) = O(f(n))$, we have $L \in \mathbf{SIZE}[g(n)]$. Similarly for $\mathbf{SIZE}[o(f(n))]$.

In this paper, we will focus on time and space efficient, non-adaptive **PCP**s with perfect completeness. Because we need to pay close attention to the amount of time it takes to make a single query to the proof, we separate the algorithm for producing queries, Q, from the algorithm for verifying the response, V. We also separate the function that gives all the query locations for a choice of randomness, I, from the algorithm that gives a single one of those query locations, Q.

So at a high level, a **PCP** protocol does the following:

1. Chooses a common random string, r.
2. Runs query function Q with randomness r for $q(n)$ many times to get all query locations, I.
3. Looks up all query locations, I, into a provided proof, π, to get proof window π_I.
4. Runs verifier V with randomness r and proof window π_I and outputs if V accepts.

Now we formally define a **PCP**.

▶ **Definition 12** (**PCP**). *We say that a language L has a non-adaptive* **PCP**, *A, with perfect completeness if there exists verifier V, prover P, index function I, and query function Q, such that, for some alphabet Σ, $\delta \in [0,1]$, and functions $r, l, q : \mathbb{N} \to \mathbb{N}$:*

1. *I takes 2 inputs, an input of length n and randomness of length $r(n)$, and outputs an element of $[l(n)]^{q(n)}$. That is, I outputs $q(n)$ indexes in a length $l(n)$ string,*
2. *Q is an algorithm with three inputs, an input x of length n, randomness r of length $r(n)$, and an index $i \in [q(n)]$ and outputs an element of $[l(n)]$ such that $Q(x, r, i) = I(x, r)_i$.*
3. *V is an algorithm with three inputs, an input of length n, randomness of length $r(n)$, and $q(n)$ symbols from Σ, and outputs either accept or reject.*
4. *P is an algorithm that takes two inputs, an input of length n, and an index $i \in [l(n)]$, and outputs a symbol from Σ.*

Completeness *If $x \in L$ and $n = |x|$, then there exists $\pi^x \in \Sigma^{l(n)}$ such that*

$$\Pr_r[V(x, r, \pi^x_{I(x,r)}) = 1] = 1,$$

and for every $i \in [l(n)]$, $P(x, i) = \pi^x_i$.
Soundness *If $x \notin L$ then for every π',*

$$\Pr_r[V(x, r, \pi'_{I(x,r)}) = 1] \leq \delta.$$

Then we also say:

1. *A has proof length $l(n)$.*
2. *A has alphabet Σ.*
3. *A has soundness δ.*
4. *A uses $q(n)$ queries.*
5. *A uses $r(n)$ bits of randomness.*
6. *If V runs in time $t(n)$, A has verifier time $t(n)$.*
7. *If V runs in space $s(n)$, A has verifier space $s(n)$.*
8. *If P runs in space $s'(n)$, A has prover space $s'(n)$.*
9. *If Q is computable in time $t'(n)$, A has query time $t'(n)$.*

3 Efficient PCP To Fine Grained Lower Bounds

We only give a detailed sketch here. The full version of our paper is available at [9].

Our analysis depends on the circuit complexity of some **PSPACE** complete problem. So we start by choosing a **SPACE**$[O(n)]$ complete problem. We use a version of `SPACE TMSAT` (on page 83 of [1]).

▶ **Definition 13** (Specific Problem). *`SPACE TMSAT` is the language*

$$\{(M, x, 1^n, 0^*) : \text{Turing machine } M \text{ accepts } x \text{ using at most } n \text{ space.}\}$$

Note: `SPACE TMSAT` $\in$ **SPACE**$[O(n)]$ and `SPACE TMSAT` is **SPACE**$[O(n)]$ complete. The 0^* is just there to make it explicit the language is paddable. In particular, this means that the circuit complexity of `SPACE TMSAT` is non-decreasing.

▶ **Lemma 14** (`SPACE TMSAT` Circuit Complexity is Non-Decreasing). *If $A'(n)$ is the size of the minimum circuit solving `SPACE TMSAT` for inputs of length n, then $A'(n)$ is non-decreasing.*

Proof. Let C be the circuit of size $A'(n+1)$ solving `SPACE TMSAT` for length $n+1$ inputs. Then to get a circuit for length n inputs, use C with an extra 0 hard coded into the last input. The resulting circuit will be at most the size of C and solve length n inputs. Thus $A'(n+1) \geq A'(n)$. ◀

Then using Theorem 3, we can get a **PCP** for `SPACE TMSAT` by setting $T = 2^{O(n)}$ and $S = O(n)$. This can be turned into a **PCP** with a binary alphabet by replacing every query for a symbol in Σ with $O(\log(n))$ queries to the individual bits of that symbol.

▶ **Corollary 15** (**PCP** for `SPACE TMSAT`)**.** *There is a* **PCP** *for* `SPACE TMSAT` *with:*

1. *Verifier time* $\tilde{O}(n)$*.*
2. *Query time* $\tilde{O}(n)$*.*
3. $\mathbf{poly}(\log(n))$ *queries.*
4. *Binary alphabet.*
5. *Log of proof length* $\tilde{O}(n)$*.*
6. *Prover space* $\tilde{O}(n)$*.*
7. *Soundness* $1/2$ *and perfect completeness.*

We prove three different **MATIME**/1 lower bounds that are based on three different hard problems. Different ones work better in different parameter regimes. After constructing them all, we show we always fall into some range of parameters so that we can get the lower bounds of Theorem 1.

3.1 Implicitly Encoding Advice in Input Length

In each of our cases, we will use advice to find the size of some prover circuit. To do this, we implicitly encode a number in the input length. If that implicitly encoded number describes the size, our advice bit will be 1. Otherwise, the advice bit is 0.

For any input length $n \in \mathbb{N}$, for some $l \in \mathbb{N}$, we have $n \in [2^l, 2^{l+1})$. For such an l, there is some $m \in \mathbb{N}$ such that $n = 2^l + m$. This m, or equivalently this l, is our implicitly encoded number. Because we will use this decomposition a lot, we will explicitly define some functions that perform this decomposition.

▶ **Definition 16** (Implicit Encoding In Input)**.** *For natural* $n \geq 1$*, let* $l \geq 0$ *be an integer so that* $n \in [2^l, 2^{l+1})$*, and* $m \geq 0$ *be an integer so that* $n = 2^l + m$*. Then define* $\mu(n) = m$ *and* $\rho(n) = l$*.*

There is a simple interpretation of this $m = \mu(n)$ and $l = \rho(n)$ in terms of the binary representation of n. You can think of l as the length of the binary number, and m the binary number after the top bit is removed.

3.2 SPACE TMSAT not in P/poly

In this case, we follow the proof in the original work [29] where **PSPACE** $\not\subset$ **P/poly**. We present the same arguments here with more precise parameters.

When **PSPACE** $\not\subset$ **P/poly**, the circuit complexity of `SPACE TMSAT` for different input sizes could change drastically and in a way that may be hard to analyze. This is an issue because the **PCP** for `SPACE TMSAT` needs a prover with a longer input than the input being verified, thus might require a much larger circuit.

Instead, we use a downward self reducible **PSPACE** complete language. Specifically, a language that has a sound interactive protocol with queries the same length as its input and whose prover is the language itself. We cite the result from Lemma 11 in [29]:

▶ **Lemma 17** (Same Size, Self Proving **PSPACE** Complete Language)**.** *There is a* **PSPACE**-*complete language Y and a probabilistic polynomial-time oracle Turing machine M such that for any input x:*

1. *M only asks its oracle queries of length $|x|$.*
2. *If M is given Y as oracle and $x \in Y$, then M accepts with probability 1.*
3. *If $x \notin Y$, then irrespective of the oracle given to M, M rejects with probability at least $1/2$.*

The important feature of language Y is that for an input x, the prover for x is the same language Y, and queries to the prover have the same length as x. This means Y, and the prover for Y, have the same circuit. Then if $\mathbf{PSPACE} \not\subset \mathbf{P/poly}$, then the polynomial overhead of the proof protocol grows much more slowly than the difficulty of Y. So a padded version of Y can be used as our language in this case. A proof sketch is in Appendix A, full proofs are in the full version [9].

▶ **Lemma 18** (Bound if **PSPACE** does not have Polynomial Sized Circuits)**.** *If* `SPACE TMSAT` $\notin$ **P/poly***, then for any $k > 0$, and some $f(n) = o(1)$:*

$$\mathbf{MATIME}[O(n^{k+f(n)})]/1 \not\subset \mathbf{SIZE}[O(n^k)].$$

3.3 SPACE TMSAT in Almost Linear Size

The idea in this case is to use a brute force, small space algorithm that finds a problem not in a fixed polynomial size. In particular, for circuit size $S(n)$, the brute force algorithm uses space $O(S(n))$ to compute some function with minimum circuit size $\Theta(S(n))$. Then we want to simulate the **PCP** from Corollary 15 to prove the output of this algorithm. Since the **PCP** is efficient, the prover for this algorithm does not use much more space than the brute force algorithm itself.

If `SPACE TMSAT` has almost linear sized circuits, the prover doesn't require much larger circuits than the space of the prover. Finally, our **PCP** is efficient, so the time of the **MA** verifier isn't much more than the size of the prover circuit. So the **MA** protocol doesn't require much more time then the size of the circuit it proves the output of.

If `SPACE TMSAT` requires larger circuits, say quadratic circuits, then the size of the prover circuits would be quadratically larger than the input length of the prover. That is, the prover circuit would be quadratically larger than the circuit it is trying to prove. This would give quadratic overhead for the **MA** verifier time over the size of the circuit it verifies. So this construction only works well enough when `SPACE TMSAT` has almost linear sized circuits.

So this gives a proof when `SPACE TMSAT` has almost linear sized circuits. A proof sketch is in Appendix A, full proofs are in the full version [9].

▶ **Lemma 19** (Bound if `SPACE TMSAT` has Size $n^{1+o(1)}$)**.** *If for some $g(n) = o(1)$ and some non-decreasing function $A(n) = n^{1+g(n)}$ we have* `SPACE TMSAT` $\in \mathbf{SIZE}[O(A(n))]$*, then for any $k > 0$, there is an $f(n) = o(1)$ such that:*

$$\mathbf{MATIME}[O(n^{k+f(n)})]/1 \not\subset \mathbf{SIZE}[O(n^k)].$$

3.4 SPACE TMSAT in Polynomial Size, but not Almost Linear

This is the "bad" case, where we can't prove the result for every constant k, only for $k < a$. This is the most complicated case, requiring us to both pad the input to get the correct problem difficulty, and use advice to get the size of the circuits for the prover. The idea is to solve `SPACE TMSAT` on a padded version of the input using our **PCP**. So we need the advice to tell us three things:

1. Some m so that SPACE TMSAT on length m inputs requires circuits of size $\omega(n^k)$.
2. Further, we need SPACE TMSAT on length m inputs to require circuits of size near m^a. This keeps the prover from requiring circuits too much larger than SPACE TMSAT on length m inputs does.
3. How big the circuit for the prover in Corollary 15 needs to be.

Similar to the previous cases, this advice will come implicitly from the input length, and the single advice bit will be 1 if and only if the input length encodes valid advice. A skectch is in Appendix A. For full details, see the full version [9].

▶ **Lemma 20** (Bound if SPACE TMSAT has size $n^{a+o(1)}$). *Suppose for some function $h(n)$ with $|h(n)| = o(1)$ and for some constant $a > 1$, for some function $A(n)$ we have $A(n) = n^{a+h(n)}$. Then if $A(n)$ is non-decreasing and we have SPACE TMSAT $\in \mathbf{SIZE}[O(A(n))] \setminus \mathbf{SIZE}[o(A(n))]$, then for any $k < a$, for some $f(n) = o(1)$,*

$$\mathbf{MATIME}[O(n^{k+f(n)})]/1 \not\subset \mathbf{SIZE}[O(n^k)].$$

3.5 Altogether

Altogether, these three cases imply Theorem 1. More details can be found in the full version [9].

▶ **Theorem 1** (Fine Grained **MA** Lower Bound). *There exists a constant $a > 1$, such that for all $k < a$, for some $f(n) = o(1)$,*

$$\mathbf{MATIME}[O(n^{k+f(n)})]/1 \not\subset \mathbf{SIZE}[O(n^k)].$$

Proof. First, we will find the best polynomial approximation of the circuit complexity of SPACE TMSAT. So define set

$$S = \{a \in \mathbb{R} : \texttt{SPACE TMSAT} \in \mathbf{SIZE}[O(n^a)]\}.$$

If $S = \emptyset$, then there is no constant a such that SPACE TMSAT $\in \mathbf{SIZE}[O(n^a)]$. Then SPACE TMSAT $\notin \mathbf{P/poly}$, so we use Lemma 18.

So suppose $S \neq \emptyset$. One can show that SPACE TMSAT requires linear size circuits since it depends on all inputs. Thus for any $a < 1$, we know that SPACE TMSAT $\notin \mathbf{SIZE}[O(n^a)]$, that is $a \notin S$. So 1 is a lower bound for S.

Then the set S is nonempty and has a lower bound. So S has an infimum, a, so that for any constant $\epsilon > 0$, we have SPACE TMSAT $\in \mathbf{SIZE}[O(n^{a+\epsilon})]$, but SPACE TMSAT $\notin \mathbf{SIZE}[O(n^{a-\epsilon})]$. This implies that for some $h(n)$ with $|h(n)| = o(1)$ and $A(n) = n^{a+h(n)}$ that SPACE TMSAT $\in \mathbf{SIZE}[O(A(n))] \setminus \mathbf{SIZE}[o(A(n))]$.

If $a = 1$, we use Lemma 19. If $a > 1$, we use Lemma 20. ◀

To prove Theorem 2, we use a proof similar to Lemma 19. See the full paper [9] for details.

4 PCP Sketch

We give a brief overview of the **PCP** proof. A full proof is available at [9].

Our overall protocol constructs an extrapolatable **PCP** by observing that a simple BFL style **PCP** is extrapolatable. Then this can be turned into an **rPCP** with an efficient query function. Finally we can compose this with a decodable, BFL style **PCP** with a very fast verifier.

4.1 Extrapolatable PCPs

We start by showing some lemmas about extrapolatability that make the definition easy to work with. Notably, if 2 functions are extrapolatable, so is their concatenation. This allows us to show components of a **PCP** are extrapolatable individually to get an extrapolatable **PCP**.

▶ **Lemma 21** (Extrapolatability Combination). *For integers $n, q, q', t, t' > 0$, and field $\mathbb{F}$, if $p : [q] \to \mathbb{F}^n$ is t extrapolatable, and $p' : [q'] \to \mathbb{F}^n$ is t' extrapolatable, then $g : [q + q'] \to \mathbb{F}^n$ is $O(t + t' + n\log(\mathbb{F}))$ extrapolatable where*

$$g(i) = \begin{cases} p(i) & i \leq q \\ p'(i-q) & i > q \end{cases}.$$

Of course, extrapolatability's main purpose is to allow efficient extrapolation.

▶ **Lemma 22** (Efficient Polynomials From Extrapolatability). *For any $n, q, t > 0$, field $\mathbb{F}$ where $|\mathbb{F}| > q$, and t extrapolatable $Q : [q] \to \mathbb{F}^n$, there is a time $O(t + q\,\mathbf{polylog}(|\mathbb{F}|))$ algorithm computing the value of a degree $q - 1$ polynomial, g, such that for all $i \in [q]$ we have $g(i) = Q(i)$.*

Now we introduce extrapolatable **PCP**s and show they imply robust **PCP**s with fast query functions. Extrapolatable **PCP**s are **PCP**s whose proofs are low degree functions, only require soundness against proofs that are low degree functions, and whose query location function is extrapolatable.

In our notation, the query function Q returns a single query location of the verifier V on a choice of randomness, whereas I returns every query location for a choice of randomness. Query function, Q, needs to be extrapolatable, but index function, I, is more convenient for defining completeness and soundness. See standard **PCP**s, Definition 12, for reference.

▶ **Definition 23** (Extrapolatable **PCP**). *We say a non-adaptive **PCP**, A, for language L with verifier V, prover P, individual query function Q, and index function I is an extrapolatable **PCP** (**ePCP**) if for some m and d:*

1. *For some field $\mathbb{F}$, A uses alphabet $\mathbb{F}$.*
2. *The proof length is $|\mathbb{F}|^m$.*
   That is, any proof, π, can be viewed as a function $\pi : \mathbb{F}^m \to \mathbb{F}$.

Low Degree Completeness *If $x \in L$ and $n = |x|$, then there exists a polynomial $\pi^x : \mathbb{F}^n \to \mathbb{F}$ of degree at most d such that $\Pr_r[V(x, r, \pi^x(I(x, r))) = 1] = 1$, and for every $i \in [l(n)]$, we have $P(x, i) = \pi^x_i$.*

Low Degree Soundness *If $x \notin L$ then for every polynomial $\pi' : \mathbb{F}^n \to \mathbb{F}$ of degree at most d has $\Pr_r[V(x, r, \pi'(I(x, r))) = 1] \leq \delta$.*

Further, we say A has:

1. *Extrapolation time $t(n)$ if for any x, r, the function $Q_{x,r}(i) = Q(x, r, i)$ is time $t(n)$ extrapolatable.*
2. *Degree d and m variables.*
3. *Low degree soundness δ.*
4. *Perfect low degree completeness.*

The main application of an **ePCP** is a convenient primitive in constructing efficient robust **PCP**s (**rPCP**). Robust **PCP**s are a standard primitive in the literature and are commonly used in **PCP** composition [3, 7, 13, 25, 12]. This conversion of **ePCP** to **rPCP** is where all the low degree testing is performed. But low degree test queries are simple lines so are easy to compute. Note that our **rPCP**s constructed from an **ePCP** uses the same proof, so if the **ePCP** prover is efficient, so is the **rPCP** prover. See the full paper [9] for a proof.

▶ **Theorem 24** (**ePCP** gives efficient **rPCP**). *For any language L with an* **ePCP**, *A, with*

1. *Verifier time $t(n)$.*
2. *Verifier space $s(n)$.*
3. *Extrapolation time $t'(n)$.*
4. *Randomness $r(n)$.*
5. *Degree $d(n)$ and $m(n)$ variables.*
6. *$q(n)$ queries.*
7. *Alphabet $\mathbb{F}$ where $|\mathbb{F}| > 10q(n)d(n)$.*
8. *Prover P.*
9. *Low degree soundness 0.1.*
10. *Perfect low degree completeness.*

Language L has an **rPCP**, *B, with*

1. *Verifier time $O(t(n) + |\mathbb{F}|^3\mathbf{polylog}(|\mathbb{F}|))$.*
2. *Verifier space $O(s(n) + \log(|\mathbb{F}|))$.*
3. *Randomness $r(n) + O(m(n)\log(|\mathbb{F}|))$.*
4. *Query time $O(t'(n) + (q(n) + m(n))\mathbf{polylog}(|\mathbb{F}|))$.*
5. *$O(|\mathbb{F}|)$ queries.*
6. *Prover P with perfect completeness.*
7. *Soundness at most 0.99.*

4.2 Constructing our Extrapolatable PCP

Our **ePCP** is a BFL [4] style **PCP**. We start by converting our algorithm into a cellular automata so that it has very uniform, local constraints, similar to the Cook-Levin theorem [10, 21], or what was done in [20]. In particular, the constraints can be described by a simple polynomial of the computation history of the cellular automata.

Then our **ePCP** will ask for a multilinear extension of the computation history, a low degree polynomial encoding the constraint function of that computation history, and a sum check [22] style proof verifying that constraint is always satisfied. Then the verifier just needs to check the computation history is consistent with the input, the constraint polynomial is consistent with the computation history polynomial, and the sum check of the constraint polynomial succeeds. All low degree testing is handled in the conversion from an **ePCP** to an **rPCP**.

Now we verify that a variation on standard sum check [22, 4] is extrapolatable. Since **ePCP**s only need soundness against low degree proofs, our base **ePCP** only needs to perform constantly many queries besides the sum check. So the check sum is the most difficult thing to prove the extrapolatability of.

▶ **Lemma 25** (Sum Check Queries Are Extrapolatable). *For any variable number $n \in \mathbb{N}$, degree $d \in \mathbb{N}$, field $\mathbb{F}$ with $|\mathbb{F}| > \max\{d, n\} + 1$, and randomness r, the query location function, Q_r, used in the sum check protocol for a degree d and n variable function are $O(nd\mathbf{polylog}(|\mathbb{F}|))$ extrapolatable.*

The proof is straightforward and comes from a close observation of the specific sum check query locations.

We need a low space prover to use in our **MA** lower bounds. Creating the computation history itself takes about as much space as the original algorithm. Multilinear extensions can be done straightforwardly in small space if one only needs one symbol, or does not care about time.

▶ **Lemma 26** (Multilinear Extensions Require Low Space). *Suppose function $G : \{0,1\}^n \to \{0,1\}$ is computable in space S. Then the multilinear function g consistent with G on Boolean inputs is computable in space $O(n \log(|\mathbb{F}|) + S)$.*

Similarly, individual symbols from the constraint polynomial can be computed in small space from the multilinear extension of the computation history. The sum check proof is essentially partial linearizations of the constraint polynomial, so can also be done in small space.

4.3 Decodable PCP

After that, we use a similar **PCP** for the decodable inner **PCP** (**dPCP**), and apply a standard robust, decodable **PCP** composition [12]. Essentially only two changes need to be made to get our **dPCP**. First we need to only check consistency of the computation history with the explicit input (and *not* the implicit input). Second, we need to do an additional query to an implicit input to decode from it. This does not require any new tricks.

Finally, we do a standard **PCP** composition of a robust **PCP** with a decodable **PCP** to do query reduction. We note that we can not do the standard trick of having the outer **rPCP** output a circuit that the inner **dPCP** proves the output of as that circuit description would be too large for a verifier to compute. Instead this must be done implicitly, similar to [5]. We also note that to keep the space of the inner **PCP** prover small requires the space of the outer **PCP** prover, the space of the outer query function, *and* the space of the outer **PCP** verifier to be small.

4.4 Fine Grained MIP = NEXP

We note that the low space and deterministic algorithm requirements were only to keep the prover space low. Dropping the prover space requirement and using the same proof gives Theorem 4. Then using Theorem 4, one can use a standard technique to get $\frac{1}{\mathbf{polylog}(n)}$ error, 2 query **PCP**. Combining this with a repetition theorem [28, 19, 27] gives the 2 query, constant error **MIP** stated in Corollary 5.

5 Open Problems

There are several ways we would like to improve the circuit lower bounds.

1. Remove the advice bit.
   We still had to use advice, a limitation from the original Santhanam result. It would be nice if we could get lower bounds on **MA** with no non-uniformity.
2. Prove tight bounds for all k.
   Another limitation of our circuit lower bound is that it does not prove this tight bound for all $k > 1$, just for some k.

The major barrier is in the case that **SPACE**$[n]$ algorithms may require super linear, but polynomial, sized circuits. Then the circuit size required for any given space may change in a strange way. For example, suppose for some $a > 1$

$$\mathbf{SPACE}[n] \subseteq \mathbf{SIZE}[O(n^a)] \setminus \mathbf{SIZE}[o(n^a)].$$

What we would like, but this does not obviously imply, is that for all $b > 1$:

$$\mathbf{SPACE}[n^b] \subseteq \mathbf{SIZE}[O(n^{ab})] \setminus \mathbf{SIZE}[o(n^{ab})].$$

While a padding argument shows $\mathbf{SPACE}[n^b] \subseteq \mathbf{SIZE}[O(n^{ab})]$, it does not show that $\mathbf{SPACE}[n^b] \not\subseteq \mathbf{SIZE}[o(n^{ab})]$.

3. Make lower bound more frequent.
   We only prove that infinitely often this lower bound occurs, but it may have even super exponential gaps between input sizes where the circuit lower bound holds. Murray and Williams [26] gave a refinement of the Santhanam circuit lower bounds, that is incomparable to ours, which gives circuit lower bounds that holds for input lengths only polynomially far apart.
4. Prove exponential lower bounds for **MAEXP**.
   A similar problem is to prove exponential circuit lower bounds for the exponential version of **MA**, known as **MAEXP**. The best circuit lower bounds known for **MAEXP** are “half-exponential” by Miltersen, Vinodchandran, and Watanabe [24]. Loosely, a function is half exponential if that function composed with itself is exponential.

Another open problem is to give a **PCP** whose verifier time matches ours with only quasilinear proof length. Known **PCP**s with proof length $\tilde{O}(T)$ have verifier time $\Omega(n + \log(T)^2)$. We suspect ideas from theorem 2.6 in [5][3] (which extends the results of [7] from **NP** to **NEXP**) may give a **PCP** with verifier time near $\tilde{O}(n + \log(T))$ while giving a proof of length $T^{1+o(1)}$. But it would not have proof length $\tilde{O}(T) = T\mathbf{polylog}(T)$. We stress that close analysis of the verifier time for the **PCP** of [5] has not been performed, and that **PCP** is much more complex than ours.

References

1 Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, USA, 1st edition, 2009.

2 Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, May 1998. doi:10.1145/278298.278306.

3 Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of np. *J. ACM*, 45(1):70–122, January 1998. doi:10.1145/273865.273901.

4 L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, pages 16–25 vol.1, 1990. doi:10.1109/FSCS.1990.89520.

5 E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Short pcps verifiable in polylogarithmic time. In *20th Annual IEEE Conference on Computational Complexity (CCC'05)*, pages 120–134, 2005. doi:10.1109/CCC.2005.27.

[3] We emphasize this is the theorem labeled “Efficient PCPPs with small query complexity”, which does not have quasi-linear proof length.

6 Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. On the concrete efficiency of probabilistically-checkable proofs. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 585–594. Association for Computing Machinery, 2013. doi:10.1145/2488608.2488681.

7 Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust pcps of proximity, shorter pcps and applications to coding. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '04, pages 1–10. Association for Computing Machinery, 2004. doi:10.1145/1007352.1007361.

8 Eli Ben-Sasson and Emanuele Viola. Short pcps with projection queries. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *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 163–173. Springer, 2014. doi:10.1007/978-3-662-43948-7_14.

9 Joshua Cook and Dana Moshkovitz. Tighter ma/1 circuit lower bounds from verifier efficient pcps for pspace, 2022. URL: https://eccc.weizmann.ac.il/report/2022/014/.

10 Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. Association for Computing Machinery. doi:10.1145/800157.805047.

11 Stephen A. Cook. A hierarchy for nondeterministic time complexity. In *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing*, STOC '72, pages 187–192. Association for Computing Machinery, 1972. doi:10.1145/800152.804913.

12 Irit Dinur and Prahladh Harsha. Composition of low-error 2-query pcps using decodable pcps. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 472–481, 2009. doi:10.1109/FOCS.2009.8.

13 Irit Dinur and Omer Reingold. Assignment testers: towards a combinatorial proof of the pcp-theorem. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 155–164, 2004. doi:10.1109/FOCS.2004.16.

14 Dean Doron, Dana Moshkovitz, Justin Oh, and David Zuckerman. Nearly optimal pseudorandomness from hardness. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1057–1068. IEEE, 2020.

15 Lance Fortnow and Rahul Santhanam. Robust simulations and significant separations. In *Proceedings of the 38th International Colloquim Conference on Automata, Languages and Programming - Volume Part I*, ICALP'11, pages 569–580. Springer-Verlag, 2011.

16 Lance Fortnow, Rahul Santhanam, and Luca Trevisan. Hierarchies for semantic classes. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 348–355. Association for Computing Machinery, 2005. doi:10.1145/1060590.1060642.

17 Lance Fortnow, Rahul Santhanam, and Ryan Williams. Fixed-polynomial size circuit bounds. In *2009 24th Annual IEEE Conference on Computational Complexity*, pages 19–26, 2009. doi:10.1109/CCC.2009.21.

18 J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965. URL: http://www.jstor.org/stable/1994208.

19 Thomas Holenstein. Parallel repetition: Simplifications and the no-signaling case. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 411–419. Association for Computing Machinery, 2007. doi:10.1145/1250790.1250852.

20 Justin Holmgren and Ron Rothblum. Delegating computations with (almost) minimal time and space overhead. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 124–135, 2018. doi:10.1109/FOCS.2018.00021.

21 Leonid Levin. Universal'nyie perebornyie zadachi (universal search problems, in russian). *Problemy Peredachi Informatsii*, 9:265–266, 1973.

22 Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, October 1992. doi:10.1145/146585.146605.

23 Or Meir. Combinatorial pcps with efficient verifiers. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 463–471, 2009. doi:10.1109/FOCS.2009.10.

24 Peter Bro Miltersen, N. V. Vinodchandran, and Osamu Watanabe. Super-polynomial versus half-exponential circuit size in the exponential hierarchy. In *Proceedings of the 5th Annual International Conference on Computing and Combinatorics*, COCOON'99, pages 210–220. Springer-Verlag, 1999.

25 Dana Moshkovitz and Ran Raz. Two query pcp with sub-constant error. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 314–323, 2008. doi:10.1109/FOCS.2008.60.

26 Cody Murray and Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime: An easy witness lemma for np and nqp. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 890–901. Association for Computing Machinery, 2018. doi:10.1145/3188745.3188910.

27 Anup Rao. Parallel repetition in projection games and a concentration bound. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 1–10. Association for Computing Machinery, 2008. doi:10.1145/1374376.1374378.

28 Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998. doi:10.1137/S0097539795280895.

29 Rahul Santhanam. Circuit lower bounds for merlin-arthur classes. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 275–283. Association for Computing Machinery, 2007. doi:10.1145/1250790.1250832.

30 Joel I. Seiferas, Michael J. Fischer, and Albert R. Meyer. Separating nondeterministic time complexity classes. *J. ACM*, 25(1):146–167, 1978. doi:10.1145/322047.322061.

31 Adi Shamir. Ip = pspace. *J. ACM*, 39(4):869–877, October 1992. doi:10.1145/146585.146609.

32 D. van Melkebeek and K. Pervyshev. A generic time hierarchy for semantic models with one bit of advice. In *21st Annual IEEE Conference on Computational Complexity (CCC'06)*, pages 14 pp.–144, 2006. doi:10.1109/CCC.2006.7.

33 Ryan Williams. Non-uniform acc circuit lower bounds. In *2011 IEEE 26th Annual Conference on Computational Complexity*, pages 115–125, 2011. doi:10.1109/CCC.2011.36.

34 Stanislav Žák. A turing machine time hierarchy. *Theoretical Computer Science*, 26(3):327–333, 1983. doi:10.1016/0304-3975(83)90015-4.

A MA Circuit Lower Bound Proofs

In this section, we sketch the proofs of the main lemmas of Section 3. Full proofs are available in the full version [9].

▶ **Lemma 18** (Bound if **PSPACE** does not have Polynomial Sized Circuits). *If SPACE TMSAT $\notin$ $\mathbf{P/poly}$, then for any $k > 0$, and some $f(n) = o(1)$:*

$$\mathbf{MATIME}[O(n^{k+f(n)})]/1 \not\subset \mathbf{SIZE}[O(n^k)].$$

Proof. The idea is to take length m inputs to Y, pad them up to length n inputs so that Y on length m inputs requires, and has, circuits of size just slightly larger than n^k. Then as long as n grows much faster than m, the overhead of the interactive protocol verifier will be small and we only need to focus on prover overhead, which is around n^k since the circuit for Y is the prover for Y.

First, I claim that, since $Y \notin \mathbf{P/poly}$, for some $g(n) = \omega(1)$, we have $Y \notin \mathbf{SIZE}[O(n^{g(n)})]$. If this was not true, then Y must have circuits of size n^k for some k upper bounding the best choice of $g(n)$.

Now we define our language, W, in $\mathbf{MATIME}[n^{k+f(n)}]/1$ but not in $\mathbf{SIZE}[O(n^k)]$. For any input size, n, using Definition 16, let $m = \mu(n)$ and $l = \rho(n)$. Let our advice bit be 1 if both:

1. Y on length m inputs has circuits with size $n^k \log(n)$. This makes sure there is a fast prover for Y.
2. And Y on length m inputs does not have circuits of size $m^{g(m)}$. This makes sure m grows much more slowly than n.

Then $x \in W$ for some x with $|x| = n$ if and only if the advice bit is 1 and for some $y \in Y$ with $|y| = m$ we have $x = y1^{n-m}$.

Let $a > 0$ be the constant so that the verifier (M in Lemma 17) for Y's interactive protocol runs in time $O(n^a)$. Define the $\mathbf{MA}$ protocol as follows. If the trusted advice bit is one, Merlin gives Arthur a length $n^k \log(n)$ circuit. Then Arthur simulates the interactive protocol for Y on y using the circuit from Merlin as the prover. This protocol runs in time $O(m^a n^k \log(n))$. By assumption we have $m^{g(m)} < n^k \log(n)$, which one can show implies $m < n^{f'(n)}$ for some $f'(n) = o(1)$. Thus the $\mathbf{MA}/1$ protocol runs in time $O(n^{f'(n)} n^k \log(n)^2) = n^{k+f(n)}$ for some $f(n) = o(1)$.

By choice of $g(m)$, for infinitely many m, the second condition is satisfied and $g(m) > k+1$. For such m, the smallest choice of $n = 2^l + m$ where Y on length m inputs has circuits of size $n^k \log(n)$ can not have circuits of size $(n/2)^k \log(n/2)$. For such n, language W does not have circuits of size $O(n^k)$, but does have an $\mathbf{MA}$ protocol running in time $n^{k+f(n)}$. ◀

▶ **Lemma 19** (Bound if `SPACE TMSAT` has Size $n^{1+o(1)}$). *If for some $g(n) = o(1)$ and some non-decreasing function $A(n) = n^{1+g(n)}$ we have `SPACE TMSAT` $\in \mathbf{SIZE}[O(A(n))]$, then for any $k > 0$, there is an $f(n) = o(1)$ such that:*

$$\mathbf{MATIME}[O(n^{k+f(n)})]/1 \not\subset \mathbf{SIZE}[O(n^k)].$$

Proof. The proof proceeds in five steps.

1. Find a language $L \in \mathbf{SPACE}[n^k \log(n)^2] \setminus \mathbf{SIZE}[n^k \log(n)/10]$. In particular, for every input length n, language L has circuits of size $n^k \log(n)$ but not $n^k \log(n)/10$.
2. Reduce L to `SPACE TMSAT` and use Corollary 15. In particular, find a circuit, C_n, for the prover in an $\mathbf{MA}$ protocol for L on length n inputs.
3. Define our advice bit to implicitly give an upper bound for the size of C_m for some m within a factor of 2 of n. Then we define W to be length m elements of L, padded to length n.
4. Show that infinitely often the advice bit is 1 and W does not have small circuits.
5. Show that W has an efficient $\mathbf{MA}$ protocol.

With that outline in mind, let us begin the proof.

1. Find a language $L \in \mathbf{SPACE}[n^k \log(n)^2] \setminus \mathbf{SIZE}[n^k \log(n)/10]$.
   From the non-uniform hierarchy (in Arora and Barak [1, Theorem 6.22]), there is a language $L \in \mathbf{SIZE}[n^k \log(n)] \setminus \mathbf{SIZE}[n^k \log(n)/10]$. In particular, for every n, language L on length n has circuits size $n^k \log(n)$ but not size $n^k \log(n)/10$. A brute force search can then find and simulate such a circuit using space $O(n^k \log(n)^2)$.
2. Reduce L to `SPACE TMSAT` and use Corollary 15.
   Since M only uses space $g(n) = \tilde{O}(n^k)$, we know $x \in L$ if and only if $(M, x, 1^{g(n)}, 0) \in$ `SPACE TMSAT`. We know `SPACE TMSAT` on length $\tilde{O}(n^k)$ inputs has a $\mathbf{PCP}$ protocol from Corollary 15 that uses $\mathbf{polylog}(n)$ many length $\tilde{O}(n^k)$ queries to a space $\tilde{O}(n^k)$ prover, P, where each query can be calculated by a time $\tilde{O}(n^k)$ algorithm, Q, and the results from P are verified by a time $\tilde{O}(n^k)$ verifier, V.

Now we reduce the prover P to `SPACE TMSAT` so we can use that `SPACE TMSAT` $\in$ $\mathbf{SIZE}[O(n^{1+g(n)})]$ to get a circuit for P. A length $\tilde{O}(n^k)$ query, q, to P can be converted into a length $\tilde{O}(n^k)$ input, q', for `SPACE TMSAT` by providing the algorithm for P and $\tilde{O}(n^k)$ 1s. Call the circuit for `SPACE TMSAT` on length $|q'|$ inputs C_n. Since `SPACE TMSAT` $\in \mathbf{SIZE}[O(n^{1+g(n)})]$, we know C_n has size $(\tilde{O}(n^k))^{1+g(n)} = n^{k+g'(n)}$ for some $g'(n) = o(1)$.

3. Define our advice bit.

Now an **MA** protocol can guess C_n, but we may not be able to compute how large C_n needs to be. The function $g'(n)$ may be hard to compute. So we use advice.

Let $l = \rho(n)$, $m = 2^l$ and $t = \mu(n)$ so that $n = m + t$. Then let the advice bit be 1 if

a. Circuit C_m has size $m^k 2^t$.

b. For any natural t' less than t, circuit C_m does not have size $m^k 2^{t'}$.

This condition allows us to use the smallest t possible for a given m.

Then $x \in W$ for some x with $|x| = n$ if and only if the advice bit is 1 and for some $y \in$ `SPACE TMSAT` with $|y| = m$ we have $x = y1^{n-m}$.

4. Show W does not have small circuits.

First see that for every large enough l, for $m = 2^l$, there will be one t such that this advice bit is 1. That is for some $t < m$, C_m has size $m^k 2^t$. Otherwise, C_m would have exponential size, but it only has almost linear size. Then for the minimum such t, the advice bit will be one. So infinitely often, the advice bit will be 1.

When the advice bit is 1, the language W on length $n = m + t$ inputs is equal to L on length m inputs. Language L on length m inputs does not have circuits of size $m^k \log(m)/10$. See by choice of m that $2m > n$, so $n^k = o(m^k \log(m)/10)$. Thus infinitely often, W does not have size n^k circuits. Thus $W \notin \mathbf{SIZE}[O(n^k]$.

5. Show W has an efficient **MA** protocol.

If the advice bit is 0, this is trivially true. For $n = 2^l + t$ so that the advice bit is 1 and $m = 2^l$, either

$t = 0$**:** Then C_m has size $n^k = O(m^{k+g'(m)})$.

$t \geq 1$**:** Then C_m has size $m^k 2^t$ but not $m^k 2^{t-1}$. Since C_m does have circuits of size $m^{k+g'(m)}$, we have $m^k 2^t = O(m^{k+g'(m)})$.

In either case, an **MA** protocol can guess C_m with a circuit with size $O(m^{k+g'(m)})$.

Then an **MA** protocol for $x = y1^{n-m}$ and an advice bit of 1 can verify if $y \in L$ by first guessing a circuit for C_m, then using it as the prover in the **PCP** protocol from Corollary 15.

The **MA** verifier needs to calculate $\mathbf{polylog}(m)$ queries with Q, run C_m on each of those queries, and run V on those results. Since C_m has size $O(m^{k+g'(m)})$, and Q and V run in time $\tilde{O}(m^{k+g'(m)})$, calculating all query locations, running C_m on each of those locations, and V on those outputs takes time

$$\begin{aligned} &\mathbf{polylog}(m)(\tilde{O}(m^k) + O(m^{k+g'(m)})) + \tilde{O}(m^k) \\ =&\tilde{O}(m^{k+g'(m)}). \end{aligned}$$

Finally, since $m < n$, for some $f(n) = o(1)$, the **MA** verifier runs in time $n^{k+f(n)}$.

The **MA** protocol is complete and sound since the **PCP** is. Thus

$$W \in \mathbf{MATIME}[O(n^{k+f(n)})]/1. \qquad \blacktriangleleft$$

▶ **Lemma 20** (Bound if `SPACE TMSAT` has size $n^{a+o(1)}$)**.** *Suppose for some function $h(n)$ with $|h(n)| = o(1)$ and for some constant $a > 1$, for some function $A(n)$ we have $A(n) = n^{a+h(n)}$. Then if $A(n)$ is non-decreasing and we have `SPACE TMSAT` $\in \mathbf{SIZE}[O(A(n))] \setminus \mathbf{SIZE}[o(A(n))]$, then for any $k < a$, for some $f(n) = o(1)$,*

$$\mathbf{MATIME}[O(n^{k+f(n)})]/1 \not\subset \mathbf{SIZE}[O(n^k)].$$

Proof. We want to solve a smaller instance of `SPACE TMSAT` that requires circuits of size $n^k \log(n)$, and we also need advice to tell us the size of circuits needed to prove `SPACE TMSAT`. The advice for this will come implicitly from the input length.

For input x of length n, (using ρ and μ from Definition 16) let $l = \rho(n)$, $l' = \rho(\mu(n))$, and $t = \mu(\mu(n))$ so that $n = 2^l + 2^{l'} + t$. We want to solve `SPACE TMSAT` on length $2^{l'}$ inputs, so we let $m := 2^{l'}$. Then $n = 2^l + m + t$ and our language will solve length m inputs for `SPACE TMSAT` using prover circuits of size $(2^l)^k 2^t$. Our advice bit will tell us when 2^l, m and t are appropriate.

So then m is the input length to `SPACE TMSAT` we want to solve, 2^l is how much padding is needed to make length m problems the right difficulty, and $(2^l)^k 2^t$ is the size of the circuits needed for our **PCP** prover.

The proof proceeds in 4 steps.

1. Define circuits C_m that prove `SPACE TMSAT` for length m inputs using our **PCP** and our theorem assumptions on circuits for `SPACE TMSAT`.
2. Define when the advice bit should be 1.
3. Show infinitely often the advice bit is 1 and $W \notin \mathbf{SIZE}[O(n^k)]$.
4. Show that W has an efficient **MA** protocol.

Now following this outline:

1. Define circuits C_m that prove `SPACE TMSAT` for length m inputs.
   Then `SPACE TMSAT` on length m inputs has a **PCP** protocol with verifier time $\tilde{O}(m)$, log of proof length $\tilde{O}(m)$, and prover space $\tilde{O}(m)$. Then the prover for `SPACE TMSAT` on length m inputs can be reduced to a circuit for `SPACE TMSAT` with length $\tilde{O}(m)$ inputs. Then `SPACE TMSAT` on length $\tilde{O}(m)$ inputs has a circuit, C_m, of size at most $\tilde{O}(m)^{a+h(m)}$. So for some $h'(m) = o(1)$, we have $|C_m| \le m^{a+h'(m)}$.
2. Define when the advice bit should be 1.
   Since `SPACE TMSAT` $\notin \mathbf{SIZE}[o(A(n))]$, for some $c_1 > 0$, for some infinite set, U', for all $n' \in U'$, language `SPACE TMSAT` on length n' inputs does not have circuits with size $c_1 A(n')$.
   Let the advice bit be 1 if and only if each of the following hold:
   a. `SPACE TMSAT` on length m inputs does not have circuits with size at most $c_1 A(m/2)$. This restricts us to m where the circuits for `SPACE TMSAT` require size near our upper bound. This limits how much bigger C_m needs to be than the circuits for `SPACE TMSAT` on length m inputs.
   b. `SPACE TMSAT` on length m inputs does not have a circuit with size $(l-1)(2^{l-1})^k$.
   c. `SPACE TMSAT` on length m inputs does have a circuit with size $l(2^l)^k$.
   d. Circuit C_m has size $(2^l)^k 2^t$.
   e. Either $t = 0$, or C_m does not have size $(2^l)^k 2^{t-1}$.
   Then $x \in W$ for some x with $|x| = n$ if and only the advice bit is 1 and for some $y \in$ `SPACE TMSAT` with $|y| = m$ we have $x = y1^{n-m}$.
3. Now we will argue that infinitely often the advice bit is 1 and W does not have circuits with size $O(n^k)$. See the full version [9] for full details. We do this in a few steps:

- First restrict our focus to m large enough and where `SPACE TMSAT` on length m inputs requires circuits of size at least $c_1 A(m/2)$. This will be the set of input lengths, U. Since m can be any power of 2 and A is monotone, as long as A is tight, we can always find an infinite set of $m = 2^{l'}$ such that `SPACE TMSAT` on length m inputs requires circuits of size $A(m/2)$.
- For each $m \in U$, find appropriate l and t.
  Take the smallest l so that `SPACE TMSAT` on length m inputs does have a circuit of size $l(2^l)^k$. Note that $l > l' = \log(m)$, since `SPACE TMSAT` on length m inputs does not have circuits with size $m^k \log(m)$ since $k < a$.
  Let t be the smallest t such that C_m has size $(2^l)^k 2^t$. Since we know C_m has poly sized circuits, we know that some $t < m$ achieves this.
- Now for $n = 2^l + m + t$, the advice bit is 1 and language `SPACE TMSAT` on length n inputs does not have circuits with size $O(n^k)$.

4. Show that

$$W \in \mathbf{MATIME}\left[n^{k+f(n)}\right]/1.$$

If the advice bit is 0, this is trivial. Otherwise, assume for n the advice bit is 1.
When the advice bit is 1, we know C_m has size at most $(2^l)^k 2^t$ and either

$t = 0$: Then C_m has size $(2^l)^k = O(n^k)$.

$t \geq 1$: Then C_m does not have size $(2^l)^k 2^{t-1}$ by choice of t. Circuit C_m has size $m^{a+h'(m)}$. Further, `SPACE TMSAT` on length m inputs does not have circuits with size $c_1 A(m/2)$ since the advice bit is 1, but it does have circuits with size $l(2^l)^k$. Together this implies that

$$\begin{aligned} c_1 A(m/2)\frac{2^{t-1}}{l} &< m^{a+h'(m)} \\ 2^t &< \frac{2}{c_1}\frac{l m^{a+h'(m)}}{A(m/2)}. \end{aligned}$$

Then one can show that $2^t = m^{o(1)}$. So for some $f'(n) = o(1)$ we have $(2^l)^k 2^t = n^{k+f'(n)}$.

The verifier for `SPACE TMSAT` can be simulated in time $\tilde{O}(m)$, and the $\mathbf{poly}(\log(m))$ queries to the prover can be simulated in time

$$\mathbf{poly}(\log(m))S(2^l)2^t = n^{k+f(n)}$$

for some $f(n) = o(1)$. Thus

$$W \in \mathbf{MATIME}\left[n^{k+f(n)}\right]/1.$$

◀

Robustness for Space-Bounded Statistical Zero Knowledge

Eric Allender

Rutgers University, Piscataway, NJ, USA

Jacob Gray

University of Massachusetts, Amherst, MA, USA

Saachi Mutreja

University of California, Berkeley, CA, USA

Harsha Tirumala

Rutgers University, Piscataway, NJ, USA

Pengxiang Wang

University of Michigan, Ann Arbor, MI, USA

Abstract

We show that the space-bounded Statistical Zero Knowledge classes $\mathsf{SZK}_{\mathsf{L}}$ and $\mathsf{NISZK}_{\mathsf{L}}$ are surprisingly robust, in that the power of the verifier and simulator can be strengthened or weakened without affecting the resulting class. Coupled with other recent characterizations of these classes [4], this can be viewed as lending support to the conjecture that these classes may coincide with the non-space-bounded classes SZK and NISZK, respectively.

2012 ACM Subject Classification Theory of computation → Complexity classes

Keywords and phrases Interactive Proofs

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.56

Category RANDOM

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2022/138/` [2]

Funding *Eric Allender*: Supported in part by NSF Grants CCF-1909216 and CCF-1909683.
Jacob Gray: Supported in part by NSF grants CNS-215018 and CCF-1852215
Saachi Mutreja: Supported in part by NSF grants CNS-215018 and CCF-1852215
Harsha Tirumala: Supported in part by NSF Grants CCF-1909216 and CCF-1909683.
Pengxiang Wang: Supported in part by NSF grants CNS-215018 and CCF-1852215

Acknowledgements This work was done in part while EA and HT were visiting the Simons Institute for the Theory of Computing. This work was carried out while JG, SM, and PW were participants in the 2022 DIMACS REU program at Rutgers University. We thank Yuval Ishai for helpful conversations about SREN, and we thank Markus Lohrey, Sam Buss, and Dave Barrington for useful discussions about Lemma 34. We also thank the anonymous referees for helpful comments.

1 Introduction

The complexity class SZK (Statistical Zero Knowledge) and its "non-interactive" subclass NISZK have been studied intensively by the research communities in cryptography and computational complexity theory. In [12], a space-bounded version of SZK, denoted $\mathsf{SZK}_{\mathsf{L}}$ was introduced, primarily as a tool for understanding the complexity of estimating the entropy of distributions represented by very simple computational models (such as low-degree polynomials, and NC^0 circuits). There, it was shown that $\mathsf{SZK}_{\mathsf{L}}$ contains many important problems previously known to lie in SZK, such as Graph Isomorphism, Discrete Log, and

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 56; pp. 56:1–56:21
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Decisional Diffie-Hellman. The corresponding "non-interactive" subclass of SZK_L, denoted $\mathsf{NISZK}_\mathsf{L}$, was subsequently introduced in [1], primarily as a tool for clarifying the complexity of computing time-bounded Kolmogorov complexity under very restrictive reducibilities (such as projections). Just as every problem in $\mathsf{SZK} \leq_{\mathrm{tt}}^{\mathsf{AC}^0}$ reduces to problems in NISZK [14], so also every problem in $\mathsf{SZK}_\mathsf{L}{\leq}_{\mathrm{tt}}^{\mathsf{AC}^0}$ reduces to problems in $\mathsf{NISZK}_\mathsf{L}$, and thus $\mathsf{NISZK}_\mathsf{L}$ contains intractable problems if and only if SZK_L does.

Very recently, all of these classes were given surprising new characterizations, in terms of efficient reducibility to the Kolmogorov random strings. Let $\widetilde{R}_K$ be the (undecidable) promise problem $(Y_{\widetilde{R}_K}, N_{\widetilde{R}_K})$ where $Y_{\widetilde{R}_K}$ contains all strings y such that $K(y) \geq |y|/2$ and the NO instances $N_{\widetilde{R}_K}$ consists of those strings y where $K(y) \leq |y|/2 - e(|y|)$ for some approximation error term $e(n)$, where $e(n) = \omega(\log n)$ and $e(n) = n^{o(1)}$.

▶ **Theorem 1** ([4]). *Let A be a decidable promise problem. Then*

- *$A \in \mathsf{NISZK}$ if and only if A is reducible to $\widetilde{R}_K$ by randomized polynomial time reductions.*
- *$A \in \mathsf{NISZK}_L$ if and only if A is reducible to $\widetilde{R}_K$ by randomized AC^0 or logspace reductions.*
- *$A \in \mathsf{SZK}$ if and only if A is reducible to $\widetilde{R}_K$ by randomized polynomial time "Boolean formula" reductions.*
- *$A \in \mathsf{SZK}_L$ if and only if A is reducible to $\widetilde{R}_K$ by randomized logspace "Boolean formula" reductions.*

In all cases, the randomized reductions are restricted to be "honest", so that on inputs of length n all queries are of length $\geq n^\epsilon$.

There are very few natural examples of computational problems A where the class of problems reducible to A via polynomial-time reductions differs (or is conjectured to differ) from the class or problems reducible to A via AC^0 reductions. For example the natural complete problems for NISZK under $\leq_{\mathrm{m}}^{\mathsf{P}}$ reductions remain complete under AC^0 reductions. Thus Theorem 1 gives rise to speculation that NISZK and $\mathsf{NISZK}_\mathsf{L}$ might be equal. (This would also imply that $\mathsf{SZK} = \mathsf{SZK}_\mathsf{L}$.)

This motivates a closer examination of SZK_L and $\mathsf{NISZK}_\mathsf{L}$, to answer questions that have not been addressed by earlier work on these classes.

Our main results are:

1. **The verifier and simulator may be very weak.** $\mathsf{NISZK}_\mathsf{L}$ and SZK_L are defined in terms of three algorithms: (1) A logspace-bounded *verifier*, who interacts with (2) a computationally-unbounded *prover*, following the usual rules of an interactive proof, and (3) a logspace-bounded *simulator*, who ensures the zero-knowledge aspects of the protocol. (More formal definitions are to be found in Section 2.) We show that the verifier and simulator can be restricted to lie in AC^0. Let us explain why this is surprising.

   The proof presented in [1], showing that $\mathsf{EA}_{\mathsf{NC}^0}$ is complete for $\mathsf{NISZK}_\mathsf{L}$, makes it clear that the verifier and simulator can be restricted to lie in $\mathsf{AC}^0[\oplus]$ (as was observed in [24]). But the proof in [1] (and a similar argument in [14]) relies heavily on hashing, and it is known that, although there are families of universal hash functions in $\mathsf{AC}^0[\oplus]$, no such families lie in AC^0 [19]. We provide an alternative construction, which avoids hashing, and allows the verifier and simulator to be very weak indeed.
2. **The verifier and simulator may be somewhat stronger.** The proof presented in [1], showing that $\mathsf{EA}_{\mathsf{NC}^0}$ is complete for $\mathsf{NISZK}_\mathsf{L}$, also makes it clear that the verifier and simulator can be as powerful as $\oplus\mathsf{L}$, without leaving $\mathsf{NISZK}_\mathsf{L}$. This is because the proof relies on the fact that logspace computation lies in the complexity class PREN of functions that have *perfect randomized encodings* [7], and $\oplus\mathsf{L}$ also lies in PREN. Applebaum, Ishai, and Kushilevitz defined PREN and the somewhat larger class SREN (for *statistical*

randomized encodings), in proving that there are one-way functions in SREN if and only if there are one-way functions in NC^0. They also showed that other important classes of functions, such as NL and GapL, are contained in SREN.[1] We initially suspected that $\mathsf{NISZK_L}$ could be characterized using verifiers and simulators computable in GapL (or even in the slightly larger class DET, consisting of problems that are $\leq_T^{\mathsf{NC}^1}$ reducible to GapL), since DET is known to be contained in $\mathsf{NISZK_L}$ [1].[2] However, we were unable to reach that goal.

We were, however, able to show that the simulator and verifier can be as powerful as NL, without making use of the properties of SREN. In fact, we go further in that direction. We define the class PM, consisting of those problems that are $\leq_T^{\mathsf{L}}$-reducible to the Perfect Matching problem. PM contains NL [18], and is not known to lie in (uniform) NC (and it is not known to be contained in SREN). We show that statistical zero knowledge protocols defined using simulators and verifiers that are computable in PM yield only problems in $\mathsf{NISZK_L}$.

3. **The complexity of the simulator is key.** As part of our attempt to characterize $\mathsf{NISZK_L}$ using simulators and verifiers computable in DET, we considered varying the complexity of the simulator and the verifier separately. Among other things, we show that the verifier can be as complex as DET if the simulator is logspace-computable. In most cases of interest, the NISZK class defined with verifier and simulator lying in some complexity class remains unchanged if the rules are changed so that the verifier is significantly stronger or weaker.

We also establish some additional closure properties of $\mathsf{NISZK_L}$ and $\mathsf{SZK_L}$, some of which are required for the characterizations given in [4].

The rest of the paper is organized as follows: Section 3 will show how $\mathsf{NISZK_L}$ can be defined equivalently using an AC^0 verifier and simulator. Section 4 will show that increasing the power of the verifier and simulator to lie in PM does not increase the size of $\mathsf{NISZK_L}$ (where PM is the class of problems (containing NL) that are logspace Turing reducible to Perfect Matching). Section 5 expands the list of problems known to lie in $\mathsf{NISZK_L}$. McKenzie and Cook [20] studied different formulations of the problem of solving linear congruences. These problems are not known to lie in DET, which is the largest well-studied subclass of P known to be contained in $\mathsf{NISZK_L}$. However, these problems are randomly logspace-reducible to DET [8]. We show that $\mathsf{NISZK_L}$ is closed under randomized logspace reductions, and hence show that these problems also reside in $\mathsf{NISZK_L}$. Section 6 shows that the complexity of the simulator is more important than the complexity of the verifier, in non-interactive zero-knowledge protocols. In particular, the verifier can be as powerful as DET, while still defining only problems in $\mathsf{NISZK_L}$. Finally Section 7 will show that $\mathsf{SZK_L}$ is closed under logspace Boolean formula truth-table reductions.

2 Preliminaries

We assume familiarity with basic complexity classes $\mathsf{L}, \mathsf{NL}, \oplus\mathsf{L}$ and P, and circuit complexity classes NC^0 and AC^0. We assume knowledge of m-reducibility (many-one-reducibility) and Turing-reducibility. $\#\mathsf{L}$ is the class of functions that count the number of accepting paths of NL machines, and $\mathsf{GapL} = \{f - g : f, g \in \#\mathsf{L}\}$. The determinant is complete for GapL, and the complexity class DET is the class of languages NC^1-Turing reducible to functions in GapL.

[1] This is not stated explicitly for GapL, but it follows from [17, Theorem 1]. See also [11, Section 4.2].

[2] More precisely, as observed in [3], the Rigid Graph (non-) Isomorphism problem is hard for DET [26], and the Rigid Graph Non-Isomorphism problem is in $\mathsf{NISZK_L}$ [1, Corollary 23].

Many of the problems we consider deal with entropy (also known as Shannon entropy). The *entropy* of a distribution X (denoted $H(X)$) is the expected value of $\log(1/\Pr[X=x])$. Given two distributions X and Y, the *statistical difference* between the two is denoted $\Delta(X,Y)$ and is equal to $\sum_\alpha \big|\Pr[X=\alpha]-\Pr[Y=\alpha]\big|/2$. Equivalently, for finite domains D, $\Delta(X,Y)=\max_{S\subseteq D}\{\big|\Pr_X[S]-\Pr_Y[S]\big|\}$. This quantity is also known as the *total variation distance* between X and Y. The *support* of X, denoted $\mathrm{supp}(X)$, is $\{x : \Pr[X=x]>0\}$.

▶ **Definition 2.** *Promise Problem: a promise problem Π is a pair of disjoint sets (Π_Y, Π_N) (the "YES" and "NO" instances, respectively). A* solution *for Π is any set S such that $\Pi_Y \subseteq S$, and $S \cap \Pi_n = \emptyset$.*

▶ **Definition 3.** *A* branching program *is a directed acyclic graph with a single source and two sinks labeled 1 and 0, respectively. Each non-sink node in the graph is labeled with a variable in $\{x_1,\ldots,x_n\}$ and has two edges leading out of it: one labeled 1 and one labeled 0. A branching program computes a Boolean function f on input $x=x_1\ldots x_n$ by first placing a pebble on the source node. At any time when the pebble is on a node v labeled x_i, the pebble is moved to the (unique) vertex u that is reached by the edge labeled 1 if $x_i=1$ (or by the edge labeled 0 if $x_i=0$). If the pebble eventually reaches the sink labeled b, then $f(x)=b$. Branching programs can also be used to compute functions $f:\{0,1\}^m\to\{0,1\}^n$, by concatenating n branching programs $p_1,\ldots,p_n$, where p_i computes the function $f_i(x)=$ the i-th bit of $f(x)$. For more information on the definitions, backgrounds, and nuances of these complexity classes, circuits, and branching programs, see the text by Vollmer [27].*

▶ **Definition 4** (Non-interactive zero-knowledge proof (NISZK), adapted from [1, 14])**.** *A non-interactive statistical zero-knowledge proof system for a promise problem Π is defined by a pair of deterministic polynomial time machines*[3] *(V,S) (the* verifier *and* simulator*, respectively) and a probabilistic routine P (the* prover*) that is computationally unbounded, together with a polynomial $r(n)$ (which will give the size of the random reference string σ), such that:*

1. *(Completeness): For all $x\in\Pi_Y$, the probability (over random σ, and over the random choices of P) that $V(x,\sigma,P(x,\sigma))$ accepts is at least $1-2^{-O(|x|)}$.*
2. *(Soundness): For all $x\in\Pi_N$, and for every possible prover P', the probability that $V(x,\sigma,P'(x,\sigma))$ accepts is at most $2^{-O(|x|)}$. (Note P' here can be malicious, meaning it can try to fool the verifier)*
3. *(Zero Knowledge): For all $x\in\Pi_Y$, the statistical distance between the following two distributions is bounded by $2^{-|x|}$:*
   a. *Choose $\sigma\leftarrow\{0,1\}^{r(|x|)}$ uniformly random, $p\leftarrow P(x,\sigma)$, and output (p,σ).*
   b. *$S(x,r)$ (where the coins r for S are chosen uniformly at random).*

It is known that changing the definition, to have the error probability in the soundness and completeness conditions and in the simulator's deviation be $\frac{1}{n^{\omega(1)}}$ results in an equivalent definition [1, 14]. (See the comments after [1, Claim 39].) We will occasionally make use of this equivalent formulation, when it is convenient.

NISZK *is the class of promise problems for which there is a non-interactive statistical zero knowledge proof system.*

$\mathsf{NISZK}_{\mathcal{C}}$ *denotes the class of problems in* NISZK *where the verifier V and simulator S lie in complexity class $\mathcal{C}$.*

[3] In prior work on NISZK [14, 1], the verifier and simulator were said to be probabilistic machines. We prefer to be explicit about the random input sequences provided to each machine, and thus the machines can be viewed as deterministic machines taking a sequence of random bits as input.

▶ **Definition 5** (EA and $\mathsf{EA}_{\mathsf{NC}^0}$, [1, 14]). *Consider Boolean circuits $C_X : \{0,1\}^m \to \{0,1\}^n$ representing distribution X. The promise problem* EA *is given by:*

$$\mathsf{EA}_Y := \{(C_X, k) : H(X) > k+1\}$$

$$\mathsf{EA}_N := \{(C_X, k) : H(X) < k-1\}$$

$\mathsf{EA}_{\mathsf{NC}^0}$ *is the variant of* EA *where the distribution C_x is an* NC^0 *circuit with each output bit depending on at most 4 input bits.*

▶ **Definition 6** (SDU and $\mathsf{SDU}_{\mathsf{NC}^0}$). *Consider Boolean circuits $C_X : \{0,1\}^m \to \{0,1\}^n$ representing distributions X. The promise problem* $\mathsf{SDU} = (\mathsf{SDU}_Y, \mathsf{SDU}_N)$ *is given by:*

$$\mathsf{SDU}_Y := \{C_X : \Delta(X, U_n) < 1/n\}$$

$$\mathsf{SDU}_N := \{C_X : \Delta(X, U_n) > 1 - 1/n\}.$$

$\mathsf{SDU}_{\mathsf{NC}^0}$ *is the analogous problem, where the distributions X are represented by* NC^0 *circuits where no output bit depends on more than* four *input bits.*

▶ **Theorem 7** ([1, 4]). $\mathsf{EA}_{\mathsf{NC}^0}$ *and* $\mathsf{SDU}_{\mathsf{NC}^0}$ *are complete for* $\mathsf{NISZK}_\mathsf{L}$. $\mathsf{EA}_{\mathsf{NC}^0}$ *remains complete, even if k is fixed to $k = n-3$.*

▶ **Definition 8** (SD and $\mathsf{SD}_{\mathsf{BP}}$, [12, 25]). *Consider a pair of Boolean circuits $C_1, C_2 : \{0,1\}^m \to \{0,1\}^n$ representing distributions X_1, X_2. The promise problem* SD *is given by:*

$$\mathsf{SD}_Y := \{(C_1, C_2) : \Delta(X_1, X_2) > 2/3\}$$

$$\mathsf{SD}_N := \{(C_1, C_2) : \Delta(X_1, X_2) < 1/3\}.$$

$\mathsf{SD}_{\mathsf{BP}}$ *is the variant of* SD *where the distributions X_1, X_2 are represented by branching programs.*

2.1 Perfect Randomized Encodings

We will make use of the machinery of *perfect randomized encodings* [7].

▶ **Definition 9.** *Let $f : \{0,1\}^n \to \{0,1\}^\ell$ be a function. We say that $\hat{f} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^s$ is a perfect randomized encoding of f with blowup b if it is:*

- **Input independent:** *for every $x, x' \in \{0,1\}^n$ such that $f(x) = f(x')$, the random variables $\hat{f}(x, U_m)$ and $\hat{f}(x', U_m)$ are identically distributed.*
- **Output Disjoint:** *for every $x, x' \in \{0,1\}^n$ such that $f(x) \neq f(x')$, $\mathrm{supp}(\hat{f}(x, U_m)) \cap \mathrm{supp}(\hat{f}(x', U_m)) = \emptyset$.*
- **Uniform:** *for every $x \in \{0,1\}^n$ the random variable $\hat{f}(x, U_m)$ is uniform over the set $\mathrm{supp}(\hat{f}(x, U_m))$.*
- **Balanced:** *for every $x, x' \in \{0,1\}^n$ $|\mathrm{supp}(\hat{f}(x, U_m))| = |\mathrm{supp}(\hat{f}(x', U_m))| = b$*

The following property of perfect randomized encodings is established in [12].

▶ **Lemma 10.** *Let $f : \{0,1\}^n \to \{0,1\}^\ell$ be a function and let $\hat{f} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^s$ be a perfect randomized encoding of f with blowup b. Then $H(\hat{f}(U_n, U_m)) = H(f(U_n)) + \log b$.*

3 Simulators and Verifiers in AC^0

In this section, we show that $\mathsf{NISZK_L}$ can be defined equivalently using verifiers and simulators that are computable in AC^0. The standard complete problems for NISZK and $\mathsf{NISZK_L}$ take a circuit C as input, where the circuit is viewed as representing a probability distribution X; the goal is to approximate the entropy of X, or to estimate how far X is from the uniform distribution. Earlier work [15, 1, 24] that had presented non-interactive zero-knowledge protocols for these problems had made use of the fact that the verifier could compute hash functions, and thereby convert low-entropy distributions to distributions with small support. But an AC^0 verifier cannot compute hash functions [19].

Our approach is to "delegate" the problem of computing hash functions to a logspace verifier, and then to make use of the uniform encoding of this verifier to obtain the desired distributions via an AC^0 reduction. To this end, we begin by defining a suitably restricted version of $\mathsf{SDU_{NC^0}}$ and show that this restricted version remains complete for $\mathsf{NISZK_L}$ under AC^0 reductions (and even under projections).

With this new complete problem in hand, we provide a $\mathsf{NISZK_{AC^0}}$ protocol for the complete problem, to conclude $\mathsf{NISZK_L} = \mathsf{NISZK_{AC^0}}$.

▶ **Definition 11.** *Consider an* NC^0 *circuit* $C : \{0,1\}^m \to \{0,1\}^n$ *and the probability distribution* X *on* $\{0,1\}^n$ *defined as* $C(U_m)$ *- where* U_m *denotes* m *uniformly random bits. For some fixed* $\epsilon > 0$ *(chosen later in Remark 16), we define:*

$$\mathsf{SDU'}_{\mathsf{NC^0},Y} = \{X : \Delta(C, U_n) < \frac{1}{2^{n^\epsilon}}\}$$

$$\mathsf{SDU'}_{\mathsf{NC^0},N} = \{X : |\operatorname{supp}(X)| \le 2^{n-n^\epsilon}\}$$

We will show that $\mathsf{SDU'_{NC^0}}$ is complete for $\mathsf{NISZK_L}$ under uniform $\le_m^{\mathsf{proj}}$ reductions. In order to do so, we first show that $\mathsf{SDU'_{NC^0}}$ is in $\mathsf{NISZK_L}$ by providing a reduction to $\mathsf{SDU_{NC^0}}$.

▷ Claim 12. $\mathsf{SDU'_{NC^0}} \le_m^{\mathsf{proj}} \mathsf{SDU_{NC^0}}$, and thus $\mathsf{SDU'_{NC^0}} \in \mathsf{NISZK_L}$.

Proof. On a given probability distribution X defined on $\{0,1\}^n$ for $\mathsf{SDU'_{NC^0}}$, we claim that the identity function $f(X) = X$ is a reduction of $\mathsf{SDU'_{NC^0}}$ to $\mathsf{SDU_{NC^0}}$. If X is a YES instance for $\mathsf{SDU'_{NC^0}}$, then $\Delta(X, U_n) < \frac{1}{2^{n^\epsilon}}$, which clearly is a YES instance of $\mathsf{SDU_{NC^0}}$. If X is a NO instance for $\mathsf{SDU'_{NC^0}}$, then $|\operatorname{supp}(X)| \le 2^{n-n^\epsilon}$. Thus, if we let T be the complement of $\operatorname{supp}(X)$, we have that, under the uniform distribution, a string α is in T with probability $\ge 1 - \frac{1}{2^{n^\epsilon}}$, whereas this event has probability zero under X. Thus $\Delta(X, U_n) \ge 1 - \frac{1}{2^{n^\epsilon}}$, easily making it a NO instance of $\mathsf{SDU_{NC^0}}$. ◁

3.1 Hardness for $\mathsf{SDU'_{NC^0}}$

▶ **Theorem 13.** $\mathsf{SDU'_{NC^0}}$ *is hard for* $\mathsf{NISZK_L}$ *under* $\le_m^{\mathsf{proj}}$ *reductions.*

Proof. In order to show that $\mathsf{SDU'_{NC^0}}$ is hard for $\mathsf{NISZK_L}$, we will show that the reduction given in [1] proving the hardness of $\mathsf{SDU_{NC^0}}$ for $\mathsf{NISZK_L}$ actually produces an instance of $\mathsf{SDU'_{NC^0}}$.

Let Π be an arbitrary promise problem in $\mathsf{NISZK_L}$ with proof system (P, V) and simulator S. Let x be an instance of Π. Let $M_x(r)$ denote a machine that simulates $S(x)$ with randomness r to obtain a transcript (σ, p) - if $V(x, \sigma, p)$ accepts then $M_x(r)$ outputs σ; else it outputs $0^{|\sigma|}$. We will assume without loss of generality that $|\sigma| = n^k$ for some constant k.

It was shown in [15, Lemma 3.1] that for the promise problem EA, there is an NISZK protocol with completeness error, soundness error and simulator deviation all bounded from above by 2^{-m} for inputs of length m. Furthermore, as noted in the paragraph before Claim 38 in [1], the proof carries over to show that $\mathsf{EA}_{\mathsf{BP}}$ has an $\mathsf{NISZK}_{\mathsf{L}}$ protocol with the same parameters. Thus, any problem in $\mathsf{NISZK}_{\mathsf{L}}$ can be recognized with exponentially small error parameters by reducing the problem to $\mathsf{EA}_{\mathsf{BP}}$ and then running the above protocol for $\mathsf{EA}_{\mathsf{BP}}$ on that instance. In particular, this holds for $\mathsf{EA}_{\mathsf{NC}^0}$. In what follows, let M_x be the distribution described in the preceding paragraph, assuming that the simulator S and verifier V yield a protocol with these exponentially small error parameters.

▷ Claim 14. If $x \in \Pi_{YES}$ then $\Delta(M_x(r), U_{n^k}) \leq 1/2^{n-1}$. And if $x \in \Pi_{NO}$ then $|\operatorname{supp}(M_x(r))| \leq 2^{n^k - n^{\epsilon k}}$ for $\epsilon < \frac{1}{k}$.

Refer to Appendix A.1 for the proof.

The above claim talks about the distribution $M_x(r)$ where M is a logspace machine. We will instead consider an NC^0 distribution with similar properties that can be constructed using projections. This distribution (denoted by C_x) is a perfect randomized encoding of $M_x(r)$. We make use of the following construction:

▶ **Lemma 15** ([1, Lemma 35]). *There is a function computable in AC^0 (in fact, it can be a projection) that takes as input a branching program Q of size l computing a function f and produces as output a list p_i of NC^0 circuits, where p_i computes the i-th bit of a function $\hat{f}$ that is a perfect randomized encoding of f that has blowup $b = 2^{(\binom{l}{2}-1)2((l-1)^2-1)}$ (and thus the length of $\hat{f}(r) = \log b + |f(r)|$). Each p_i depends on at most four input bits from (x, r) (where r is the sequence of random bits in the randomized encoding).*

The properties of perfect randomized encodings (see Definition 9) imply that the range of $\hat{f}$ (and thus also the range of C_x) can be partitioned into equal sized pieces corresponding to each value of $f(r)$. Thus, let $\alpha_1, \alpha_2, .., \alpha_z$ be the range of $f(r)$, and let $[\alpha] = \{\hat{f}(r, s) : f(r) = \alpha\}$. It follows that $|[\alpha]| = b$. For a given α, and for a given β of length $\log b$ we denote by $\alpha\beta$ the β-th element of $[\alpha]$. Since the simulator S runs in logspace, each bit of $M_x(r)$ can be simulated with a branching program Q_x. Furthermore, it is straightforward to see that there is an AC^0-computable function that takes x as input and produces an encoding of Q_x as output, and it can even be seen that this function can be a projection. Let the list of NC^0 circuits produced from Q_x by the construction of Lemma 15 be denoted C_x.

We show that this distribution C_x is an instance of $\mathsf{SDU'}_{\mathsf{NC}^0}$ if $x \in \Pi$. For $x \in \Pi_{YES}$, we have $\Delta(M_x(r), U_{n^k}) \leq 1/2^{n-1}$, and we want to show $\Delta(C_x(r), U_{\log b + n^k}) \leq 1/2^{n-1}$. Thus it will suffice to observe that $\Delta(M_x(r), U_{n^k}) = \Delta(C_x(r), U_{\log b + n^k}) \leq 1/2^{n-1}$.

To see this, note that

$$\Delta(C_x(r), U_{\log b + n^k}) = \sum_{\alpha\beta} \Big| \Pr[C_x = \alpha\beta] - \frac{1}{2^{n^k + b}} \Big|/2 = \sum_{\beta} \sum_{\alpha} \Big| \Pr[M_x = \alpha] \frac{1}{2^b} - \frac{1}{2^b} \frac{1}{2^{n^k}} \Big|/2$$

$$= \sum_{\alpha} \Big| \Pr[M_x = \alpha] - \frac{1}{2^{n^k}} \Big|/2 = \Delta(M_x(r), \mathcal{U}_{n^k}).$$

Thus, for $x \in \Pi_{YES}$, C_x is a YES instance for $\mathsf{SDU'}_{\mathsf{NC}^0}$.

For $x \in \Pi_{NO}$, Claim 14 shows that $|\operatorname{supp}(M_x(r))| \leq 2^{n^k - n}$. Since the NC^0 circuit C_x is a perfect randomized encoding of $M_x(r)$, we have that the support of C_x for $x \in \Pi_{NO}$ is bounded from above by $b \times 2^{n^k - n}$ Note that $\log b$ is polynomial in n; let $q(n) = \log b$. Let $r(n)$ denote the length of the output of C; $r(n) = q(n) + n^k$. Thus the size of $\operatorname{supp}(C_x) \leq 2^{n^k - n + q(n)} = 2^{r(n) - n} < 2^{r(n) - r(n)^{\epsilon}}$ (if $1/\epsilon$ is chosen to be greater than the degree of r), and hence C_x is a NO instance for $\mathsf{SDU'}_{\mathsf{NC}^0}$. ◀

▶ Remark 16. Here is how we pick ϵ in the definition of $\mathsf{SDU'}_{\mathsf{NC}^0}$. $\mathsf{SDU}_{\mathsf{NC}^0}$ is in $\mathsf{NISZK}_\mathsf{L}$ via some simulator and verifier, where the error parameters are exponentially small, and the shared reference strings σ have length n^k on inputs of length n. Now we pick $\epsilon > 0$ so that $\epsilon < 1/k$ (as in Claim 14) and also $1/\epsilon$ is greater than the degree of r (as in the last sentence of the proof of Theorem 13).

3.2 $\mathsf{NISZK}_{\mathsf{AC}^0}$ protocol for $\mathsf{SDU'}_{\mathsf{NC}^0}$ on input X represented by circuit C

3.2.1 Non Interactive proof system

1. Let C take inputs of length m and produce outputs of length n, and let σ be the reference string of length n.
2. If there is no r such that $C(r) = \sigma$, then the prover sends $\perp$. Otherwise, the prover picks an element r uniformly at random from $p \sim \{r | C(r) = \sigma\}$ and sends it to the verifier.
3. V accepts iff $C(r) = \sigma$. (Since C is an NC^0 circuit, this can be accomplished in AC^0 – this step can not be accomplished in NC^0 since it depends on all of the bits of σ.)

3.2.2 Simulator for $\mathsf{SDU'}_{\mathsf{NC}^0}$ proof system, on input X represented by circuit C

1. Pick a random s of length m and compute $\gamma = C(s)$.
2. Output (s, γ).

3.3 Proofs of Zero Knowledge, Completeness and Soundness

- **Completenss:** Suppose $X \in \mathsf{SDU'}_{\mathsf{NC}^0,Y}$, then $\Delta(X, U_n) < \frac{1}{2^{n^\epsilon}}$. This implies $|\operatorname{supp}(X)| > 2^n(1 - \frac{1}{2^{n^\epsilon}})$, which immediately implies that the verifier accepts with high probability.
- **Soundness:** Suppose $X \in \mathsf{SDU'}_{\mathsf{NC}^0,N}$, we note that whenever $\sigma \notin \operatorname{supp}(X)$, no prover can make the verifier accept. If $X \in \mathsf{SDU'}_{\mathsf{NC}^0,N}$, the probability that $\sigma \notin \operatorname{supp}(X)$ is greater than $1 - \frac{1}{2^{n^\epsilon}}$.
- **Statistical Zero-Knowledge:** Refer to Appendix A.2 for proof. ⌟

4 Simulator and Verifier in PM

In this section, we show that $\mathsf{NISZK}_\mathsf{L}$ can be defined equivalently using verifiers and simulators that lie in the class PM of problems that logspace-Turing reduce to Perfect Matching. (PM is not known to lie in (uniform) NC.) That is, we can increase the computational power of the simulator and the verifier from L to PM without affecting the power of noninteractive statistical zero knowledge protocols.

The Perfect Matching problem is the well-known problem of deciding, given an undirected graph G with $2n$ vertices, if there is a set of n edges covering all of the vertices. We define a corresponding complexity class PM as follows:

$$\mathsf{PM} := \{A : A \leq_T^L \text{Perfect Matching}\}$$

It is known that $\mathsf{NL} \subseteq \mathsf{PM}$ [18].

Our argument proceeds by first observing[4] that $\mathsf{NISZK}_\mathsf{L} = \mathsf{NISZK}_{\oplus\mathsf{L}}$, and then making use of the details of the argument that Perfect Matching is in $\oplus\mathsf{L/poly}$ [6].

[4] This equality was previously observed in [24].

▶ **Proposition 17.** $\mathsf{NISZK}_{\oplus\mathsf{L}} = \mathsf{NISZK}_{\mathsf{L}}$

Proof. It suffices to show $\mathsf{NISZK}_{\oplus\mathsf{L}} \subseteq \mathsf{NISZK}_{\mathsf{L}}$. We do this by showing that the problem $\mathsf{EA}_{\mathsf{NC}^0}$ is hard for $\mathsf{NISZK}_{\oplus\mathsf{L}}$; this suffices since $\mathsf{EA}_{\mathsf{NC}^0}$ is complete for $\mathsf{NISZK}_{\mathsf{L}}$. The proof of [1, Theorem 26] (showing that $\mathsf{EA}_{\mathsf{NC}^0}$ is complete for $\mathsf{NISZK}_{\mathsf{L}}$ involves (a) building a branching program to simulate a logspace computation called M_x that is constructed from a logspace-computable simulator and verifier, and (b) constructing an NC^0-computable perfect randomized encoding of M_x, using the fact that $\mathsf{L} \subset \mathcal{PREN}$, where $\mathcal{PREN}$ is the class defined in [7], consisting of all problems with perfect randomized encodings. But Theorem 4.18 in [7] shows the stronger result that $\oplus\mathsf{L}$ lies in $\mathcal{PREN}$, and hence the argument of [1, Theorem 26] carries over immediately, to reduce any problem in $\mathsf{NISZK}_{\oplus\mathsf{L}}$ to $\mathsf{EA}_{\mathsf{NC}^0}$ (by modifying step (a), to build a *parity* branching program for M_x that is constructed from a $\oplus\mathsf{L}$ simulator and verifier). ◀

We also rely on the following lemma:

▶ **Lemma 18** (Adapted from [6, Section 3] and [21, Section 4]). *Let $W = (w_1, w_2, \cdots, w_{n^{k+3}})$ be a sequence of n^{k+3} weight functions, where each $w_i : [\binom{n}{2}] \to [4n^2]$ is a distinct weight assignment to edges in n-vertex graphs. Let (G, w_i) denote the result of weighting the edges of G using weight assignment w_i. Then there is a function f in* GapL*, such that, if (G, w_i) has a unique perfect matching of weight j, then $f(G, W, i, j) \in \{1, -1\}$, and if G has no perfect matching, then for* every *(W, i, j), it holds that $f(G, W, i, j) = 0$. Furthermore, if W is chosen uniformly at random, then with probability $\geq 1 - 2^{-n^k}$, for* <u>*each*</u> *n-vertex graph G:*

- *If G has no perfect matching then $\forall i \forall j$ $f(G, W, i, j) = 0$.*
- *If G has a perfect matching then $\exists i$ such that (G, w_i) has a unique minimum-weight matching, and hence $\exists i \exists j$ $f(G, W, i, j) \in \{1, -1\}$.*

Thus if we define $g(G, W)$ to be $1 - \Pi_{i,j}(1 - f(G, W, i, j)^2)$, we have that $g \in$ GapL *and with probability $\geq 1 - 2^{-n^k}$ (for randomly-chosen W), $g(G, W) = 1$ if G has a perfect matching, and $g(G, W) = 0$ otherwise.*

Note that this lemma is saying that most W constitute a good "advice string", in the sense that $g(G, W)$ provides the correct answer to the question "Does G have a perfect matching?" for every graph G with n vertices.

▶ **Corollary 19.** *For every language $A \in$* PM *there is a language $B \in \oplus\mathsf{L}$ such that, if $x \in A$, then $\Pr_{W \leftarrow [4n^2]^{n^5}}[(x, W) \in B] \geq 1 - 2^{-n^2}$, and if $x \notin A$, then $\Pr_{W \leftarrow [4n^2]^{n^5}}[(x, W) \in B] \leq 2^{-n^2}$.*

Refer to Appendix A.3 for proof.

▶ **Theorem 20.** $\mathsf{NISZK}_{\mathsf{L}} = \mathsf{NISZK}_{\mathsf{PM}}$

Proof. We show that $\mathsf{NISZK}_{\mathsf{PM}} \subseteq \mathsf{NISZK}_{\oplus\mathsf{L}}$, and then appeal to Proposition 17.

Let Π be an arbitrary problem in $\mathsf{NISZK}_{\mathsf{PM}}$, and let (S, P, V) be the PM simulator, prover, and verifier for Π, respectively. Let S' and V' be the $\oplus\mathsf{L}$ languages that are probabilistic realizations of S, V, respectively, guaranteed by Corollary 19. We now define a $\mathsf{NISZK}_{\mathsf{L}}$ protocol (S'', P'', V'') for Π.

On input x with shared randomness σW, the prover P'' sends the same message $p = P(x, \sigma)$ as the original prover sends. The verifier V'', returns the value of $V'((x, \sigma, p), W)$, which with high probability is equal to $V(x, \sigma, p)$. The simulator S'', given as input x and random sequence rW, executes $S'((x, r, i), W)$ for each bit position i to obtain a bit that (with high probability) is equal to the i^{th} bit of $S(x, r)$, which is a string of the form (σ, p), and outputs $(\sigma W, p)$.

Now we will analyze the properties of (S'', P'', V''):

- **Completeness:** Suppose $x \in \Pi_Y$, then $\Pr_\sigma[V(x, \sigma, P(x, \sigma)) = 1] \geq 1 - 2^{-O(n)}$. Since $\forall y \in \{0,1\}^n : \Pr_W[V(y) = V'(y, W)] \geq 1 - 2^{-n^k}$ we have:

$$\Pr_{\sigma W}[V'((x, \sigma, P''(x, \sigma)), W) = 1] \geq [1 - 2^{-O(n)}][1 - 2^{-n^k}] = 1 - 2^{-O(n)}$$

- **Soundness:** Suppose $x \in \Pi_N$, then $\Pr_\sigma[\forall p : V(x, \sigma, p) = 0] \geq 1 - 2^{-O(n)}$. Since $\forall y \in \{0,1\}^n : \Pr_W[V(y) = V'(y, W)] \geq 1 - 2^{-n^k}$, we have:

$$\Pr_{\sigma W}[\forall p : V'((x, \sigma, p), W) = 0] \geq [1 - 2^{-O(n)}][1 - 2^{-n^k}] = 1 - 2^{-O(n)}$$

- **Statistical Zero-Knowledge:** Suppose $x \in \Pi_Y$. Let S^* denote the distribution on strings of the form (σ, p) that $S(x, r)$ produces, where r is uniformly generated, and let P^* denote the distribution on strings given by $(\sigma, P(x, \sigma))$ where σ is chosen uniformly at random. Similarly, let S''^* denote the distribution on strings of the form $(\sigma W, p)$ that $S''(x, rW)$ produces, where r and W are chosen uniformly, and let P''^* be the distribution given by $(\sigma W, P''(x, \sigma W))$. Let $A = \{(\sigma W, p) : \exists i \exists r\ S(x, r)_i \neq S'((x, r, i), W)\}$.
Since $\Pr_W[\forall i \forall r : S(x, r)_i = S'((x, r, i), W)] \geq 1 - 2^{-O(n)}$ we have:

$$\begin{aligned}
\Delta(S''^*, P''^*) &= \frac{1}{2} \sum_{(\sigma W, p)} \left| \Pr[S''^* = (\sigma W, p)] - \Pr[P''^* = (\sigma W, p)] \right| \\
&\leq \frac{1}{2}(2^{-O(n)} + \sum_{(\sigma W, p) \in \overline{A}} \left| \Pr[S''^* = (\sigma W, p)] - \Pr[P''^* = (\sigma W, p)]) \right| \\
&= \frac{1}{2}(2^{-O(n)} + \sum_{(\sigma W, p) \in \overline{A}} \left| \Pr[S^* = (\sigma, p)] - \Pr[P^* = (\sigma, p)] \right| \Pr[W]) \\
&\leq 2^{-O(n)} + \sum_W \Pr[W] \frac{1}{2} \sum_{(\sigma, p)} \left| \Pr[S^* = (\sigma, p)] - \Pr[P^* = (\sigma, p)] \right| \\
&= 2^{-O(n)} + \Delta(S^*, P^*) = 2^{-O(n)}
\end{aligned}$$

Therefore (S'', P'', V'') is a $\mathsf{NISZK}_{\oplus\mathsf{L}}$ protocol deciding Π. ◀

5 Additional problems in $\mathsf{NISZK_L}$

In this section, we give additional examples of problems in P that lie in $\mathsf{NISZK_L}$. These problems are not known to lie in (uniform) NC. Our main tool is to show that $\mathsf{NISZK_L}$ is closed under a class of randomized reductions.

The following definition is from [4]:

▶ **Definition 21.** *A promise problem $A = (Y, N)$ is $\leq_m^{\mathsf{BPL}}$-reducible to $B = (Y', N')$ with threshold θ if there is a logspace-computable function f and there is a polynomial p such that*

- *$x \in Y$ implies $\Pr_{r \in \{0,1\}^{p(|x|)}}[f(x, r) \in Y'] \geq \theta$.*
- *$x \in N$ implies $\Pr_{r \in \{0,1\}^{p(|x|)}}[f(x, r) \in N'] \geq \theta$.*

Note, in particular, that the logspace machine computing the reduction has two-way access to the random bits r; this is consistent with the model of probabilistic logspace that is used in defining $\mathsf{NISZK_L}$.

▶ **Theorem 22.** $\mathsf{NISZK_L}$ *is closed under $\leq_m^{\mathsf{BPL}}$ reductions with threshold $1 - \frac{1}{n^{\omega(1)}}$.*

Proof. Let $\Pi \leq_{\mathrm{m}}^{\mathsf{BPL}} \mathsf{EA}_{\mathsf{NC}^0}$, via logspace-computable function f. Let (S_1, V_1, P_1) be the $\mathsf{NISZK_L}$ proof system for $\mathsf{EA}_{\mathsf{NC}^0}$.

Algorithm 1 Simulator $S(x, r\sigma')$.

```
(σ, p) ← S_1(f(x, σ'), r);
return ((σ, σ'), p);
```

Algorithm 2 Verifier $V(x, (\sigma, \sigma'), p)$.

```
return V_1((f(x, σ'), σ, p))
```

Algorithm 3 Prover $P(x, (\sigma, \sigma'))$.

```
return P_1((f(x, σ'), σ));
```

We now claim that (S, P, V) is a $\mathsf{NISZK_L}$ protocol for Π.

It is apparent that S and V are computable in logspace. We just need to go through completeness, soundness, and statistical zero-knowledge of this protocol.

- **Completeness:** Suppose x is YES instance of Π. Then with probability $1 - \frac{1}{n^{\omega(1)}}$ (over randomness of σ'): $f(x, \sigma')$ is a YES instance of $\mathsf{EA}_{\mathsf{NC}^0}$. Thus for a randomly chosen σ:

$$\Pr[V_1(f(x,\sigma'), \sigma, P_1(f(x,\sigma'), \sigma)) = 1] \geq 1 - \frac{1}{n^{\omega(1)}}$$

- **Soundness:** Suppose x is NO instance of Π. Then with probability $1 - \frac{1}{n^{\omega(1)}}$ (over randomness of σ'): $f(x, \sigma')$ is a NO instance of $\mathsf{EA}_{\mathsf{NC}^0}$. Thus for a randomly chosen σ:

$$\Pr[V_1(f(x,\sigma'), \sigma, P_1(f(x,\sigma'), \sigma)) = 0] \geq 1 - \frac{1}{n^{\omega(1)}}$$

- **Statistical Zero-Knowledge:** If x is a YES instance, $f(x, \sigma')$ is a YES instance of $\mathsf{EA}_{\mathsf{NC}^0}$ with probability close to 1. For any YES instance y of $\mathsf{EA}_{\mathsf{NC}^0}$, the distribution given by S_1 on input y is exponentially close the the distribution on transcripts (σ, p) induced by (V_1, P_1) on input y. Thus the distribution on $(\sigma\sigma', p)$ induced by (V, P) has distance at most $\frac{1}{n^{\omega(1)}}$ from the distribution produced by S on input x. The claim now follows by the comments regarding error probabilities in Definition 4. ◀

McKenzie and Cook [20] defined and studied the problems LCON, LCONX and $\mathsf{LCONNULL}$. LCON is the problem of determining if a system of linear congruences over the integers mod q has a solution. LCONX is the problem of finding a solution, if one exists, and $\mathsf{LCONNULL}$ is the problem of computing a spanning set for the null space of the system.

These problems are known to lie in uniform NC^3 [20], but are not known to lie in uniform NC^2, although Arvind and Vijayaraghavan showed that there is a set B in $\mathsf{L}^{\mathsf{GapL}} \subseteq \mathsf{DET} \subseteq \mathsf{NC}^2$ such that $x \in \mathsf{LCON}$ if and only if $(x, W) \in B$, where W is a randomly-chosen weight function [8]. (The probability of error is exponentially small.) The mapping $x \mapsto (x, W)$ is clearly a $\leq_{\mathrm{m}}^{\mathsf{BPL}}$ reduction. Since $\mathsf{DET} \subseteq \mathsf{NISZK_L}$ [1], it follows that

$$\mathsf{LCON} \in \mathsf{NISZK_L}$$

The arguments in [8] carry over to LCONX and $\mathsf{LCONNULL}$ as well.

▶ **Corollary 23.** $\mathsf{LCON} \in \mathsf{NISZK_L}$. $\mathsf{LCONX} \in \mathsf{NISZK_L}$. $\mathsf{LCONNULL} \in \mathsf{NISZK_L}$.

6 Varying the Power of the Verifier

In this section, we show that the computational complexity of the simulator is more important than the computational complexity of the verifier, in non-interactive protocols. The results in this section were motivated by our attempts to show that $\mathsf{NISZK}_\mathsf{L} = \mathsf{NISZK}_\mathsf{DET}$. Although we were unable to reach this goal, we were able to show that the verifier could be as powerful as DET, if the simulator was restricted to be no more powerful than NL. The general approach here is to replace a powerful verifier with a weaker verifier, by requiring the prover to provide a proof to convince a weak verifier that the more powerful verifier would accept.

We define $\mathsf{NISZK}_{A,B}$ as the class of problems with a NISZK protocol where the simulator is in A and the verifier is in B (and hence $\mathsf{NISZK}_A = \mathsf{NISZK}_{A,A}$). We will consider the case where $A \subseteq B \subseteq \mathsf{NISZK}_A$ and A, B are both classes of functions that are closed under composition.

▶ **Theorem 24.** $\mathsf{NISZK}_{A,B} = \mathsf{NISZK}_A$

Proof. Let Π be an arbitrary promise problem in $\mathsf{NISZK}_{A,B}$ with (S_1, V_1, P_1) being the A simulator, B verifier, and prover for Π's proof system, where the reference string has length $p_1(|x|)$ and the prover's messages have length $q_1(|x|)$. Since $V_1 \in B \subseteq \mathsf{NISZK}_A$, $L(V_1)$ has a proof system (S_2, V_2, P_2), where the reference string has length $p_2(|x|)$ and the prover's messages have length $q_2(|x|)$.

Then Π has the following NISZK_A proof system:

Algorithm 4 Simulator $S(x, r_1, r_2)$.

```
Data: $x \in \Pi_{Yes} \cup \Pi_{No}$
$(\sigma, p) \leftarrow S_1(x, r_1)$;
$(\sigma', p') \leftarrow S_2((x, \sigma, p), r_2)$;
return $((\sigma, \sigma'), (p, p'))$;
```

Algorithm 5 Verifier $V(x, (\sigma, \sigma'), (p, p'))$.

```
return $V_2((x, \sigma, p), \sigma', p')$
```

Algorithm 6 Prover $P(x, \sigma\sigma')$.

```
Data: $x \in \Pi_{Yes} \cup \Pi_{No}, \sigma \in \{0,1\}^{p_1(|x|)}, \sigma' \in \{0,1\}^{p_2(|x|)}$
if $x \in \Pi_{Yes}$ then
    $p \leftarrow P_1(x, \sigma)$;
    $p' \leftarrow P_2((x, \sigma, p), \sigma')$;
    return $(p, p')$;
else
    return $\perp, \perp$;
end
```

- **Correctness:** Suppose $x \in \Pi_{Yes}$, then given random σ, with probability $(1 - \frac{1}{2^{O(|x|)}})$: $(x, \sigma, P_1(x, \sigma)) \in L(V_1)$ which means with probability $(1 - \frac{1}{2^{O(|x|+p_1(|x|)+|p|)}})$ it holds that $((x, \sigma, p), \sigma', P_2(x, \sigma, P_1(x, \sigma)) \in L(V_2)$. So the probability that V accepts is at least:
$$(1 - \frac{1}{2^{O(|x|)}})(1 - \frac{1}{2^{O(|x|+p_1(|x|)+q_1(|x|))}}) = 1 - \frac{1}{2^{O(|x|)}}$$
- **Soundness:** Suppose $x \in \Pi_N$. When given a random σ, we have that with probability less than $\frac{1}{2^{O(|x|)}}$: $\exists p$ such that $(x, \sigma, p) \in L(V_1)$. For $(x, \sigma, p) \notin L(V_1)$, the probability that there is a p such that $((x, \sigma, p), \sigma', p') \in L(V_2)$ is at most $\frac{1}{2^{O(|x|+p_1(|x|)+|p|)}}$ (given random σ'). So the probability that V rejects is at least:
$$(1 - \frac{1}{2^{O(|x|)}})(1 - \frac{1}{2^{O(|x|+p(|x|)+|p|)}}) = 1 - \frac{1}{2^{O(|x|)}}$$

- **Statistical Zero-Knowledge:** Refer to the full version [2]. ◀

▶ **Corollary 25.** $\mathsf{NISZK_L} = \mathsf{NISZK_{AC^0}} = \mathsf{NISZK_{AC^0,DET}} = \mathsf{NISZK_{NL,DET}}$

The proof of Theorem 24 did not make use of the condition that the verifier is at least as powerful as the simulator. Thus, maintaining the condition that $A \subseteq B \subseteq \mathsf{NISZK}_A$, we also have the following corollary:

▶ **Corollary 26.** $\mathsf{NISZK}_B = \mathsf{NISZK}_{B,A}$

▶ **Corollary 27.** $\mathsf{NISZK}_{A,B} \subseteq \mathsf{NISZK}_{B,A}$

▶ **Corollary 28.** $\mathsf{NISZK_{DET}} = \mathsf{NISZK_{DET,AC^0}}$

7 $\mathsf{SZK_L}$ closure under $\leq^{\mathsf{L}}_{\mathrm{bf-tt}}$ reductions

Although our focus in this paper has been on $\mathsf{NISZK_L}$, in this section we report on a closure property of the closely-related class $\mathsf{SZK_L}$.

The authors of [12], after defining the class $\mathsf{SZK_L}$, wrote:

> We also mention that all the known closure and equivalence properties of SZK (e.g. closure under complement [22], equivalence between honest and dishonest verifiers [15], and equivalence between public and private coins [22]) also hold for the class $\mathsf{SZK_L}$.

In this section, we consider a variant of a closure property of SZK (closure under $\leq^{\mathsf{P}}_{\mathrm{bf-tt}}$ [25]), and show that it also holds[5] for $\mathsf{SZK_L}$. Although our proof follows the general approach of the proof of [25, Theorem 4.9], there are some technicalities with showing that certain computations can be accomplished in logspace (and for dealing with distributions represented by branching programs instead of circuits) that require proof. (The characterization of $\mathsf{SZK_L}$ in terms of reducibility to the Kolmogorov-random strings presented in [4] relies on this closure property.)

▶ **Definition 29** (From [25, Definition 4.7]). *For a promise problem* Π*, the characteristic function of* Π *is the map* $\mathcal{X}_\Pi : \{0,1\}^* \to \{0,1,*\}$ *given by*

$$\mathcal{X}_\Pi(x) = \begin{cases} 1 & \textit{if } x \in \Pi_{Yes}, \\ 0 & \textit{if } x \in \Pi_{No}, \\ * & \textit{otherwise.} \end{cases}$$

▶ **Definition 30.** *Logspace Boolean formula truth-table reduction* ($\leq^{\mathsf{L}}_{\mathrm{bf-tt}}$ *reduction): We say a promise problem* Π ***logspace Boolean formula truth-table reduces*** *to* Γ *if there exists a logspace-computable function* f*, which on input* x *produces a tuple* $(y_1, \ldots, y_m)$ *and a Boolean formula* ϕ *(with* m *input gates) such that:*

$$x \in \Pi_{Yes} \implies \phi(\mathcal{X}_\Gamma(y_1), \ldots, \mathcal{X}_\Gamma(y_m)) = 1$$

$$x \in \Pi_{No} \implies \phi(\mathcal{X}_\Gamma(y_1), \ldots, \mathcal{X}_\Gamma(y_m)) = 0$$

[5] We observe that open questions about closure properties of NISZK also translate to open questions about $\mathsf{NISZK_L}$. NISZK is not known to be closed under union [23], and neither is $\mathsf{NISZK_L}$. Neither is known to be closed under complementation. Both are closed under conjunctive logspace-truth-table reductions.

We begin by proving a logspace analogue of a result from [25], used to make statistically close pairs of distributions closer and statistically far pairs of distributions farther.

▶ **Lemma 31** (Polarization Lemma, adapted from [25, Lemma 3.3]). *There is a logspace-computable function that takes a triple $(P_1, P_2, 1^k)$, where P_1 and P_2 are branching programs, and outputs a pair of branching programs (Q_1, Q_2) such that:*

$$\Delta(P_1, P_2) < \frac{1}{3} \implies \Delta(Q_1, Q_2) < 2^{-k}$$

$$\Delta(P_1, P_2) > \frac{2}{3} \implies \Delta(Q_1, Q_2) > 1 - 2^{-k}$$

To prove this, we adapt the same method as in [25] and alternate two different procedures, one to drive pairs with large statistical distance closer to 1, and one to drive distributions with small statistical distance closer to 0. The following lemma will do the former:

▶ **Lemma 32** (Direct Product Lemma, from [25, Lemma 3.4]). *Let X and Y be distributions such that $\Delta(X, Y) = \epsilon$. Then for all k,*

$$k\epsilon \geq \Delta(\otimes^k X, \otimes^k Y) \geq 1 - 2\exp(-k\epsilon^2/2)$$

The proof of this statement follows from [25]. To use this for Lemma 31, we note that a branching program for $\otimes^k P$ can easily be created in logspace from a branching program P by simply copying and concatenating k independent copies of P together.

We now introduce a lemma to push close distributions closer:

▶ **Lemma 33** (XOR Lemma, adapted from [25, Lemma 3.5]). *There is a logspace-computable function that maps a triple $(P_0, P_1, 1^k)$, where P_0 and P_1 are branching programs, to a pair of branching programs (Q_0, Q_1) such that $\Delta(Q_0, Q_1) = \Delta(P_0, P_1)^k$. Specifically, Q_0 and Q_1 are defined as follows:*

$$Q_0 = \bigotimes_{i \in [k]} P_{y_i} : y \leftarrow_R \{y \in \{0,1\}^k : \oplus_{i \in [k]} y_i = 0\}$$

$$Q_1 = \bigotimes_{i \in [k]} P_{y_i} : y \leftarrow_R \{y \in \{0,1\}^k : \oplus_{i \in [k]} y_i = 1\}$$

Refer to Appendix A.4 for proof. We now have the tools to prove Lemma 31.

Proof of Lemma 31. From [25, Section 3.2], we know that we can polarize $(P_0, P_1, 1^k)$ by:

- Letting $l = \lceil \log_{4/3} 6k \rceil$, $j = 3^{l-1}$
- Applying Lemma 33 to $(P_0, P_1, 1^l)$ to get (P_0', P_1')
- Applying Lemma 32: $P_0'' = \otimes^j P_0'$, $P_1'' = \otimes^j P_1'$
- Applying Lemma 33 to $(P_0'', P_1'', 1^k)$ to get (Q_0, Q_1)

Each step is computable in logspace, and since logspace is closed under composition, this completes our proof. ◀

We also mention the following lemma, which will be useful in evaluating the Boolean formula given by the $\leq^{\mathsf{L}}_{\mathrm{bf-tt}}$ reduction.

▶ **Lemma 34.** *There is a function in NC^1 that takes as input a Boolean formula ϕ (with m input bits) and produces as output an equivalent formula ψ with the following properties:*

1. *The depth of ψ is $O(\log m)$.*
2. *ψ is a tree with alternating levels of AND and OR gates.*

3. *The tree's non-leaf structure is always the same for a fixed input length.*
4. *All NOT gates are located just before the leaves.*

Refer to Appendix A.5 for proof.

▶ **Theorem 35.** $\mathsf{SZK_L}$ *is closed under* $\leq^{\mathsf{L}}_{\mathrm{bf-tt}}$ *reductions.*

To begin the proof of this theorem, we first note that as in the proof of [25, Lemma 4.10], given two $\mathsf{SD_{BP}}$ pairs, we can create a new pair which is in $\mathsf{SD}_{\mathsf{BP},No}$ if both of the original two pairs are (which we will use to compute ANDs of queries.) We can also compute in logspace the OR query for two queries by creating a pair $(P_1 \otimes S_1, P_2 \otimes S_2)$. We prove that these operations produce an output with the correct statistical difference with the following two claims:

▷ Claim 36. $\{(y_1, y_2) | \mathcal{X}_{\mathsf{SD_{BP}}}(y_1) \vee \mathcal{X}_{\mathsf{SD_{BP}}}(y_2) = 1\} \leq^{\mathsf{L}}_{\mathrm{m}} \mathsf{SD_{BP}}$.

▷ Claim 37. $\{(y_1, y_2) | \mathcal{X}_{\mathsf{SD_{BP}}}(y_1) \wedge \mathcal{X}_{\mathsf{SD_{BP}}}(y_2) = 1\} \leq^{\mathsf{L}}_{\mathrm{m}} \mathsf{SD_{BP}}$.

Refer to Appendix A.6 and Appendix A.7 for the construction and proof. Crucially we note that the construction still retains a 2/3 completeness and 1/3 soundness bound.

Proof of Theorem 35. Now suppose that we are given a promise problem Π such that $\Pi \leq^{\mathsf{L}}_{\mathrm{bf-tt}} \mathsf{SD_{BP}}$. We want to show $\Pi \leq^{\mathsf{L}}_{\mathrm{m}} \mathsf{SD_{BP}}$, which by $\mathsf{SZK_L}$'s closure under $\leq^{\mathsf{L}}_{\mathrm{m}}$ reductions implies $\Pi \in \mathsf{SZK_L}$.

We follow the steps below on input x to create an $\mathsf{SD_{BP}}$ instance (F_0, F_1) which is in $\mathsf{SD}_{\mathsf{BP},Y}$ if $x \in \Pi_Y$:

1. Run the L machine for the $\leq^{\mathsf{L}}_{\mathrm{bf-tt}}$ reduction on x to get queries $(q_1, \ldots, q_m)$ and the formula ϕ.
2. Build ψ from ϕ using Lemma 34. Replace negated queries $\neg q_i$ with the query produced by the reduction from $\mathsf{SD}_{\mathsf{BP},Y}$ to $\mathsf{SD}_{\mathsf{BP},N}$ on q_i, and then apply Lemma 31 (the Polarization Lemma) with $k = n$ on these queries to get $(y_1, \ldots, y_k)$. Pad the output bits of each branching program so each branching program has m output bits.
3. Build the template tree T. At the leaf level, for each variable in ψ, we will plug in the corresponding query y_i. By Lemma 34 the tree is full.
4. Given x and designated output position j of F_0 or F_1, there is a logspace computation which finds the original output bit from $y_1 \ldots y_m$ that bit j was copied from. This machine traverses down the template tree from the output bit and records the following:
   - The node that the computation is currently at on the template tree, with the path taken depending on j.
   - The position of the random bits used to decide which path to take when we reach nodes corresponding to AND.

   This takes $O(\log m)$ space. We can use this algorithm to copy and compute each output bit of F_0 and F_1, creating (F_0, F_1) in logspace.

For step 4, we give an algorithm $\mathsf{Eval}(x, j, \psi, y_1, \ldots, y_m)$ to compute the jth output bit of F_0 or F_1 on x, for a formula ψ satisfying the properties of Lemma 34, a list of $\mathsf{SD_{BP}}$ queries $(y_1, \ldots, y_m)$, and j. Without loss of generality, we lay out the algorithm to compute only $F_0(x)$.

Outline of $\mathsf{Eval}(x, j, \psi, y_1, \ldots, y_m)$:

The idea is to compute the jth output bit of F_0 by recursively calculating which query output bit it was copied from. To do this, first notice that the AND and OR operations produce branching programs where each output bit is copied from exactly one output bit of one of the query branching programs, so composing these operations together tells us that every output bit in F_0 is copied from exactly one output bit from one query. By Lemma 34 and our AND and OR operations preserving the number of output bits, we also have that if every BP has l output bits, F_0 will have $2^a l = |\psi| l$ output bits, where a is the depth of ψ. This can be used to recursively calculate which query the jth bit is from: for an OR gate, divide the output bits into fourths, and decide which fourth the jth bit falls into (with each fourth corresponding to one BP, or two fourths corresponding to a subtree.) For an AND gate, divide the output into fourths, decide which fourth the jth bit falls into, and then use the 4 random bits for the XOR operation to compute which fourth corresponds to which branching programs (2 fourths will correspond to 1 BP or subtree, and the other 2 fourths will correspond to the 2 BPs from the other subtree.) If j is updated recursively, then at the query level, we can directly return the j'th output bit. This can be done in logspace, requiring a logspace path of "lefts" and "rights" to track the current gate, logspace to record and update j', logspace to compute $2^a l$ at each level, and logspace to compute which subtree/query the output bit comes from at each level.

The resulting BP will be two distributions that will be in $\mathsf{SD}_{\mathsf{BP},Y} \iff x \in \Pi_Y$. By this process $\Pi \leq^{\mathsf{L}}_{\mathrm{m}} \mathsf{SD}_{\mathsf{BP}}$. ◀

References

1 Eric Allender, John Gouwar, Shuichi Hirahara, and Caleb Robelle. Cryptographic hardness under projections for time-bounded Kolmogorov complexity. *Theoretical Computer Science*, 940:206–224, 2023. doi:10.1016/j.tcs.2022.10.040.

2 Eric Allender, Jacob Gray, Saachi Mutreja, Harsha Tirumala, and Pengxiang Wang. Robustness for space-bounded statistical zero knowledge. *Electron. Colloquium Comput. Complex.*, TR22-138, 2022. URL: https://eccc.weizmann.ac.il/report/2022/138.

3 Eric Allender and Shuichi Hirahara. New insights on the (non-) hardness of circuit minimization and related problems. *ACM Transactions on Computation Theory (TOCT)*, 11(4):1–27, 2019.

4 Eric Allender, Shuichi Hirahara, and Harsha Tirumala. Kolmogorov complexity characterizes statistical zero knowledge. In *14th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 251 of *LIPIcs*, pages 3:1–3:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ITCS.2023.3.

5 Eric Allender and Ian Mertz. Complexity of regular functions. *Journal of Computer and System Sciences*, 104:5–16, 2019. Language and Automata Theory and Applications - LATA 2015. doi:10.1016/j.jcss.2016.10.005.

6 Eric Allender, Klaus Reinhardt, and Shiyu Zhou. Isolation, matching, and counting uniform and nonuniform upper bounds. *Journal of Computer and System Sciences*, 59(2):164–181, 1999. doi:10.1006/jcss.1999.1646.

7 Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM Journal on Computing*, 36(4):845–888, 2006. doi:10.1137/S0097539705446950.

8 V. Arvind and T. C. Vijayaraghavan. Classifying problems on linear congruences and abelian permutation groups using logspace counting classes. *computational complexity*, 19(1):57–98, November 2009. doi:10.1007/s00037-009-0280-6.

9 Samuel R. Buss. The Boolean formula value problem is in ALOGTIME. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 123–131. ACM, 1987. doi:10.1145/28395.28409.

10 Samuel R Buss. Algorithms for Boolean formula evaluation and for tree contraction. *Arithmetic, Proof Theory, and Computational Complexity*, 23:96–115, 1993.

11 Ronald Cramer, Serge Fehr, Yuval Ishai, and Eyal Kushilevitz. Efficient multi-party computation over rings. In *Proc. International Conference on the Theory and Applications of Cryptographic Techniques; Advances in Cryptology (EUROCRYPT)*, volume 2656 of *Lecture Notes in Computer Science*, pages 596–613. Springer, 2003. doi:10.1007/3-540-39200-9_37.

12 Zeev Dvir, Dan Gutfreund, Guy N Rothblum, and Salil P Vadhan. On approximating the entropy of polynomial mappings. In *Second Symposium on Innovations in Computer Science*, pages 460–475. Tsinghua University Press, 2011.

13 Moses Ganardi and Markus Lohrey. A universal tree balancing theorem. *ACM Transactions on Computation Theory*, 11(1):1:1–1:25, 2019. doi:10.1145/3278158.

14 Oded Goldreich, Amit Sahai, and Salil Vadhan. Can statistical zero knowledge be made non-interactive? or On the relationship of SZK and NISZK. In *Annual International Cryptology Conference*, pages 467–484. Springer, 1999. doi:10.1007/3-540-48405-1_30.

15 Oded Goldreich, Amit Sahai, and Salil P. Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 399–408. ACM, 1998. doi:10.1145/276698.276852.

16 Ulrich Hertrampf, Steffen Reith, and Heribert Vollmer. A note on closure properties of logspace MOD classes. *Information Processing Letters*, 75(3):91–93, 2000. doi:10.1016/S0020-0190(00)00091-0.

17 Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *Proc. International Conference on Automata, Languages, and Programming (ICALP)*, volume 2380 of *Lecture Notes in Computer Science*, pages 244–256. Springer, 2002. doi:10.1007/3-540-45465-9_22.

18 Richard M. Karp, Eli Upfal, and Avi Wigderson. Constructing a perfect matching is in random NC. *Combinatorica*, 6(1):35–48, 1986. doi:10.1007/BF02579407.

19 Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, Fourier transform, and learnability. *J. ACM*, 40(3):607–620, 1993. doi:10.1145/174130.174138.

20 Pierre McKenzie and Stephen A. Cook. The parallel complexity of Abelian permutation group problems. *SIAM Journal on Computing*, 16(5):880–909, 1987. doi:10.1137/0216058.

21 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 345–354. ACM, 1987. doi:10.1145/28395.383347.

22 Tatsuaki Okamoto. On relationships between statistical zero-knowledge proofs. *Journal of Computer and System Sciences*, 60(1):47–108, 2000. doi:10.1006/jcss.1999.1664.

23 Chris Peikert and Vinod Vaikuntanathan. Noninteractive statistical zero-knowledge proofs for lattice problems. In *Proc. Advances in Cryptology: 28th Annual International Cryptology Conference (CRYPTO)*, volume 5157 of *Lecture Notes in Computer Science*, pages 536–553. Springer, 2008. doi:10.1007/978-3-540-85174-5_30.

24 Vishal Ramesh, Sasha Sami, and Noah Singer. Simple reductions to circuit minimization: DIMACS REU report. Technical report, DIMACS, Rutgers University, 2021. Internal document.

25 Amit Sahai and Salil P. Vadhan. A complete problem for statistical zero knowledge. *J. ACM*, 50(2):196–249, 2003. doi:10.1145/636865.636868.

26 Jacobo Torán. On the hardness of graph isomorphism. *SIAM Journal on Computing*, 33(5):1093–1108, 2004. doi:10.1137/S009753970241096X.

27 Heribert Vollmer. *Introduction to circuit complexity: a uniform approach*. Springer Science & Business Media, 1999. doi:10.1007/978-3-662-03927-4.

A Appendix

This appendix contains some proofs that were moved from the main part of the paper, due to space limitations.

A.1 Proof of Claim 14

Proof. For $x \in \Pi_{YES}$, claim 38 of [1] shows that $\Delta(M_x(r), U_{n^k}) \leq 1/2^{n-1}$, establishing the first part of the claim.

For $x \in \Pi_{NO}$, from the soundness guarantee of the $\mathsf{NISZK_L}$ protocol for $\mathsf{EA_{NC^0}}$, we know that, for at least a $1 - \frac{1}{2^n}$ fraction of the shared reference strings $\sigma \in \{0,1\}^{n^k}$, there is no message p that the prover can send that will cause V to accept. Thus there are at most $2^{n^k - n}$ outputs of $M_x(r)$ other than 0^{n^k}. For $\epsilon < \frac{1}{k}$, we have $|\operatorname{supp}(M_x(r))| \leq 2^{n^k - n^{\epsilon k}}$. $\lhd$

A.2 Proof of Statistical Zero-Knowledge in Section 3.3

Proof. Suppose $X \in \mathsf{SDU'}_{\mathsf{NC^0},Y}$. Recall that $\sigma \sim \{0,1\}^n$, $s \sim \{0,1\}^m$, $p \sim \{r : C(r) = \sigma\}$ and $\gamma = C(s)$. In order to provide an upper bound on $\Delta((p,\sigma),(s,\gamma))$, we consider the element wise probability of each distribution and show that for $X \in \mathsf{SDU'}_{\mathsf{NC^0},Y}$ the claim holds. For $a \in \{0,1\}^m$ and $b \in \{0,1\}^n$ we have:

$$\Delta((p,\sigma),(s,\gamma)) = \sum_{(a,b)} \frac{1}{2} \left|\Pr[(p,\sigma) = (a,b)] - \Pr[(s,\gamma) = (a,b)]\right|$$

Let us consider an element $b \in \{0,1\}^n$. Let $A_b = \{a_1, a_2, .., a_{k_b}\}$ be the pre-images of b under C i.e. for $1 \leq i \leq k_b$ it holds that $C(a_i) = b$. Let $\beta_b = \Pr_{y \sim U_m}[C(y) = b]$. Then $k_b 2^{-m} = \beta_b$ (since exactly k_b elements of $\{0,1\}^m$ are mapped to b under C). Let $B = \{b | \neg\exists y : C(y) = b\}$. Since $\Delta(C(U_m), U_n) \leq \frac{1}{2^{n^\epsilon}}$, it follows that $\frac{|B|}{2^m} \leq \frac{1}{2^{n^\epsilon}}$. We have:

$$\begin{aligned}
\Delta((p,\sigma),(s,\gamma)) &= \sum_{(a,b)} \frac{1}{2}(|\Pr[(p,\sigma) = (a,b)] - \Pr[(s,\gamma) = (a,b)]|) \\
&= \frac{1}{2} \sum_{(a,b):b \in B} |\Pr[(p,\sigma) = (a,b)] - \Pr[(s,\gamma) = (a,b)]| \\
&\quad + \frac{1}{2} \sum_{(a,b):b \notin B} |\Pr[(p,\sigma) = (a,b)] - \Pr[(s,\gamma) = (a,b)]|
\end{aligned}$$

For (a,b) satisfying $b \in B$, we have $\Pr[(s,\gamma) = (a,b)] = \Pr[(p,\sigma) = (a,b)] = 0$. For $b \notin B$ and a satisfying $C(a) \neq b$ we again have $\Pr[(s,\gamma) = (a,b)] = \Pr[(p,\sigma) = (a,b)] = 0$. For $(a,b) : C(a) = b$ we have $\Pr[(s,\gamma) = (a,b)] = 2^{-m}$ since $s \sim U_m$ and picking s fixes b. We also have $\Pr[(p,\sigma) = (a,b)] = \frac{2^{-n}}{k_b}$ since $\sigma \sim U_n$ and then the prover picks p uniformly from A_b. This gives us

$$\begin{aligned}
\Delta((p,\sigma),(s,\gamma)) &= \frac{1}{2} \sum_{(a,b):C(a)=b} \left|2^{-m} - \frac{2^{-n}}{k_b}\right| \\
&= \frac{1}{2} \sum_{(a,b):C(a)=b} \left|2^{-m} - \frac{2^{-m-n}}{\beta_b}\right| \\
&= \frac{1}{2} \sum_{(a,b):C(a)=b} \frac{2^{-m}}{\beta_b} \left|\beta_b - 2^{-n}\right| \\
&\leq \frac{1}{2} \sum_{(a,b):C(a)=b} \left|\beta_b - 2^{-n}\right| = \Delta(C(U_m), U_n) \leq \frac{1}{2^{n^\epsilon}}
\end{aligned}$$

where the first inequality holds since $\beta_b \geq 2^{-m}$ whenever $\beta_b \neq 0$. Thus we have :

$$\Delta((p,\sigma),(s,\gamma)) = O(\frac{1}{2^{n^\epsilon}}).$$ ◀

A.3 Proof of Corollary 19

Proof. Let A be in PM, where there is a logspace oracle machine M accepting A with an oracle P for Perfect Matching. We may assume without loss of generality that all queries made by M on inputs of length n have the same number of vertices $p(n)$. This is because G has a perfect matching iff $G \cup \{x_1 - y_1, x_2 - y_2, ..., x_k - y_k\}$ has a perfect matching. (I.e., we can "pad" the queries, to make them all the same length.)

Let $C = \{(G, W) : g(G, W) \equiv 1 \bmod 2\}$, where g is the function from Lemma 18. Clearly, $C \in \oplus\mathsf{L}$. Now, a logspace oracle machine with input (x, W) and oracle C can simulate the computation of M^P on x; each time M poses the query "Is $G \in P$", instead we ask if $(G, W) \in C$. Then with high probability (over the random choice of W) all of the queries will be answered correctly and hence this routine will accept if and only if $x \in A$, by Lemma 18. Let B be the language accepted by this logspace oracle machine. We see that $B \in \mathsf{L}^C \subseteq \mathsf{L}^{\oplus\mathsf{L}} = \oplus\mathsf{L}$, where the last equality is from [16]. ◀

A.4 Proof of Lemma 33

Proof. The proof that $\Delta(Q_0, Q_1) = \Delta(P_0, P_1)^k$ follows from [25, Proposition 3.6]. To finish proving this lemma, we show a logspace-computable mapping between $(P_0, P_1, 1^k)$ and (Q_0, Q_1).

Let ℓ and w be the max length and width between P_0 and P_1. We describe the structure of Q_0, with Q_1 differing in a small step: to begin with, Q_0 reads the $k-1$ random bits $y_1, \ldots, y_{k-1}$. For each of the random bits, it can pick the correct of two different branches, one having P_0 built in at the end and the other having P_1. We will read y_1, branch to P_0 or P_1 (and output the distribution accordingly), then unconditionally branch to reading y_2 and repeat until we reach y_{k-1} and branch to P_0 or P_1. We then unconditionally branch to y_1 and start computing the parity, and at the end we will be able to decide the value of y_k which will allow us to branch to the final copy of P_0 or P_1.

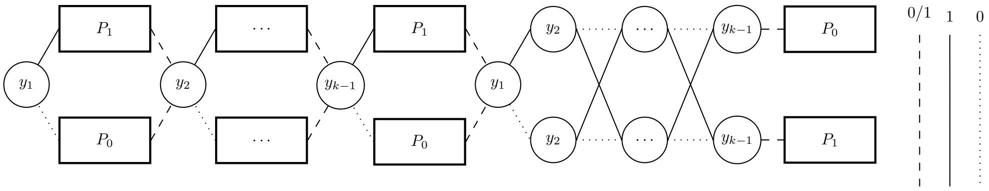


Figure 1 Branching program for Q_0 of Lemma 33.

Creating (Q_0, Q_1) can be done in logspace, requiring logspace to create the section to compute y_k and logspace to copy the independent copies of P_0 and P_1. ◀

A.5 Proof of Lemma 34

Proof. Although this lemma does not seem to have appeared explicitly in the literature, it is known to researchers, and is closely related to results in [13] (see Theorems 5.6 and 6.3, and Lemma 3.3) and in [5] (see Lemma 5). Alternatively, one can derive this by using the fact that the Boolean formula evaluation problem lies in NC^1 [9, 10], and thus there is an alternating Turing machine M running in $O(\log n)$ time that takes as input a Boolean formula ψ and an assignment α to the variables of ψ, and returns $\psi(\alpha)$. We may assume without loss of generality that M alternates between existential and universal states at each step, and that M runs for exactly $c \log n$ steps on each path (for some constant c), and that M accesses its input (via the address tape that is part of the alternating Turing machine model) only at a halting step, and that M records the sequence of states that it has visited along the current path in the current configuration. Thus the configuration graph of M, on inputs of length n, corresponds to a formula of $O(\log n)$ depth having the desired structure, and this formula can be constructed in NC^1. Given a formula ϕ, an NC^1 machine can thus build this formula, and hardwire in the bits that correspond to the description of ϕ, and identify the remaining input variables (corresponding to M reading the bits of α) with the variables of ϕ. The resulting formula is equivalent to ϕ and satisfies the conditions of the lemma. ◀

A.6 Proof of Claim 36

Proof. Let $y_1 = (A_1, B_1)$ and $y_2 = (A_2, B_2)$. Let $p > 0$ be a parameter, where we are guaranteed that:

$$(A_i, B_i) \in \mathsf{SD}_{\mathsf{BP},Y} \implies \Delta(A_i, B_i) > 1 - p$$

$$(A_i, B_i) \in \mathsf{SD}_{\mathsf{BP},N} \implies \Delta(A_i, B_i) < p$$

Then consider:

$$y = (A_1 \otimes A_2, B_1 \otimes B_2)$$

Let us analyze the Yes and No instance of $\mathcal{X}_{\mathsf{SD}_{\mathsf{BP}}}(y_1) \vee \mathcal{X}_{\mathsf{SD}_{\mathsf{BP}}}(y_2)$:

- YES: $\Delta(A_1 \otimes A_2, B_1 \otimes B_2) \geq \max\{\Delta(A_1 \otimes B_2, B_1 \otimes B_2), \Delta(B_1 \otimes A_2, B_1 \otimes B_2)\} = \max\{\Delta(A_1, B_1), \Delta(A_2, B_2)\} > 1 - p$.
- NO: $\Delta(A_1 \otimes A_2, B_1 \otimes B_2) \leq \Delta(A_1, B_1) + \Delta(A_2, B_2) < 2p$.

The second equality is from [25, Fact 2.3]. ◁

In our Boolean formula, we will have only $d = O(\log m)$ depth, so we have this OR operation for at most $\frac{d+1}{2}$ levels (and the soundness gap doubles at every level). Since $p = \frac{1}{2^m}$ at the beginning, the gap (for NO instance) will be upper bounded at the end by:

$$< 2^{\frac{d+1}{2}} \frac{1}{2^m} = \frac{m^{O(1)}}{2^m} < 1/3.$$

A.7 Proof of Claim 37

Proof. Let $y_1 = (A_1, B_1)$ and $y_2 = (A_2, B_2)$. Let $p > 0$ be a parameter, where we are guaranteed that:

$$(A_i, B_i) \in \mathsf{SD}_{\mathsf{BP},Y} \implies \Delta(A_i, B_i) > 1 - p$$

$$(A_i, B_i) \in \mathsf{SD}_{\mathsf{BP},N} \implies \Delta(A_i, B_i) < p$$

We can construct a pair of BPs $y = (A, B)$ whose statistical difference is exactly

$$\Delta(A_1, B_1) \cdot \Delta(A_2, B_2)$$

The pair (A, B) we construct is analogous to (Q_0, Q_1) in Lemma 33, and can be created in logspace with 2 random bits b_0, b_1. We have $A = (A_1, A_2)$ if $b_0 = 0$ and $A = (B_1, B_2)$ if $b_0 = 1$, while $B = (A_1, B_2)$ if b_2 is 0 and (A_2, B_1) if $b_1 = 1$.

Let us analyze the Yes and No instance of $\mathcal{X}_{\mathsf{SD}_{\mathsf{BP}}}(y_1) \wedge \mathcal{X}_{\mathsf{SD}_{\mathsf{BP}}}(y_2)$:

- YES: $\Delta(A_1, B_1) \cdot \Delta(A_2, B_2) > (1-p)^2$.
- NO: $\Delta(A_1, B_1) \cdot \Delta(A_2, B_2) \leq \max\{\Delta(A_1, B_1), \Delta(A_2, B_2)\} < p$. ◁

In our Boolean formula we will have only $d = O(\log m)$ depth, so we have this AND operation for at most $\frac{d+1}{2}$ levels (and the completeness gap squares itself at every level). Since $p = \frac{1}{2^m}$ at the beginning, the gap (for YES instance) will be lower bounded at the end by:

$$> (1 - \frac{1}{2^m})^{2^{\frac{d+1}{2}}} = (1 - \frac{1}{2^m})^{m^{O(1)}} > (1 - \frac{1}{2^m})^{2^m/m} \approx (\frac{1}{e})^{1/m} > \frac{2}{3}.$$

On Constructing Spanners from Random Gaussian Projections

Sepehr Assadi
Department of Computer Science, Rutgers University, New Brunswick, NJ, USA

Michael Kapralov
School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland

Huacheng Yu
Department of Computer Science, Princeton University, NJ, USA

Abstract

Graph sketching is a powerful paradigm for analyzing graph structure via linear measurements introduced by Ahn, Guha, and McGregor (SODA'12) that has since found numerous applications in streaming, distributed computing, and massively parallel algorithms, among others. Graph sketching has proven to be quite successful for various problems such as connectivity, minimum spanning trees, edge or vertex connectivity, and cut or spectral sparsifiers. Yet, the problem of approximating shortest path metric of a graph, and specifically computing a spanner, is notably missing from the list of successes. This has turned the status of this fundamental problem into one of the most longstanding open questions in this area.

We present a partial explanation of this lack of success by proving a strong lower bound for a *large family of graph sketching algorithms* that encompasses prior work on spanners and many (but importantly not also all) related cut-based problems mentioned above. Our lower bound matches the algorithmic bounds of the recent result of Filtser, Kapralov, and Nouri (SODA'21), up to lower order terms, for constructing spanners via the same graph sketching family. This establishes near-optimality of these bounds, at least restricted to this family of graph sketching techniques, and makes progress on a conjecture posed in this latter work.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Sparsification and spanners

Keywords and phrases sketching algorithm, lower bound, graph spanner

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.57

Category RANDOM

Related Version *Full Version*: https://arxiv.org/abs/2209.14775

1 Introduction

Analyzing structure of different objects via random linear projections, also known as *sketching*, is a fundamental paradigm that arise in various contexts. Canonical examples of this approach include dimensionality reduction results such as Johnson-Lindenstrauss lemma [25], sparse recovery results in compressed sensing [16], approximation algorithms for large matrices [40, 41], or various sketches for statistical estimation such as AMS sketch [4], count sketch [12], or count-min sketch [14] in data streams.

A pioneering work of [1] initiated *graph sketching* that considers this paradigm for graphs. A graph sketching algorithm samples a sketching matrix A from a fixed distribution, independent of the input graph G, and compute $A \cdot R(G)$ where R is a suitable representation of G chosen by the algorithm designer, say, its adjacency matrix, Laplacian, or (signed) edge-incidence matrix. The algorithm then uses $A \cdot R(G)$, referred to as the *sketch*, to (approximately) discover properties of G with no further access to G, e.g., to determine

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 57; pp. 57:1–57:18
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

whether or not G is connected. Assuming one can design a sketching matrix A with "few" rows, this approach leads to sketches that can be stored and updated efficiently and be used to recover fundamental properties of G.

The linearity of the sketches and the natural "composability" guarantee that comes with it makes graph sketching a versatile tool in many applications. For instance, graph sketching is the de facto method of algorithm design for *dynamic streaming* algorithms that process a graph specified via a sequence of edge insertions and deletions; see, e.g. [1, 2, 30, 6, 31, 21]. Graph sketching works seamlessly in this model as linearity of sketches allows one to update them easily after each update in the stream. It is even known that this method is universal for dynamic streams under certain (strong) assumptions on length of the stream [35, 3] (see also [28] for necessity of these assumptions). Another model that has benefited greatly from graph sketching is that of *distributed sketching* (a.k.a. simultaneous communication model or broadcast congested clique) wherein every vertex is a processor that sees only edges incident on the vertex and its task is to communicate a small message, simultaneously with other vertices, that allows a referee to solve the problem; see, e.g. [10, 1, 2, 11, 37, 7, 42]. Finally, graph sketching has also been a powerful tool for designing distributed or massively parallel algorithms; see, e.g. [1, 2, 24, 22, 38, 27, 20, 21].

All these considerations have turned graph sketching into a highly attractive solution concept in the last decade since their introduction in [1]. We now have efficient sketches, that often match existentially optimal bounds up to poly-log factors[1], for various fundamental problems such as connectivity [1], minimum spanning trees [1], edge connectivity [1], vertex connectivity [23], cut sparsifiers [2], spectral sparsifiers [30, 31], graph coloring [5], densest subgraph [36], and others.

Graph sketching for spanners

We study graph sketching for the problem of computing *spanners* that (approximately) preserve the shortest path metric of the input graph. Formally,

▶ **Definition 1.** *A subgraph H of a graph $G = (V, E)$ is a d-spanner of G for some integer $d \geq 1$, called the* <u>*stretch*</u> *of the spanner, if for every pair $u, v \in V$ one has*

$$dist_G(u, v) \leq dist_H(u, v) \leq d \cdot dist_G(u, v),$$

where $dist_*(\cdot, \cdot)$ stands for the shortest path metric of the corresponding graph.

For every integer $k \geq 1$, every n-vertex graph $G = (V, E)$ admits a $(2k - 1)$-spanner with only $O(n^{1+1/k})$ edges which is also existentially optimal under the widely-believed Erdős Girth Conjecture. For instance, every graph admits an $O(\log n)$-spanner on $O(n)$ edges.

Spanners are notably absent from the list of successes in graph sketching. Indeed, despite the significant attention given to sketching spanners, see, e.g., [2, 34, 20, 21, 17], until very recently, it was not even known whether an $o(n)$-spanner can be recovered via sketches of $O(n)$ size. The work of [21] made the first progress on this problem in nearly a decade by presenting an $O(n^{2/3})$-spanner using sketches of $\widetilde{O}(n)$ size[2], or more generally a d-spanner using sketches of size $\widetilde{O}(n^2/d^{3/2})$. But such bounds are still quite far from existential bounds on spanners dictated by the girth conjecture. Yet, no non-trivial lower bounds are known for

[1] For instance, sketches of size $O(n \log^3 n)$ for spanning forests of n-vertex graphs [1] compared to existential bound of $\Omega(n \log n)$ bits to store the spanning forest.

[2] We use $\widetilde{O}(\cdot)$ and $\widetilde{\Omega}(\cdot)$ notation to hide poly-log factors.

this problem[3], beside the work of [37] (see also [42]) that proves that finding *any* spanning tree requires sketches of size $\Omega(n \log^3 n)$ bits (namely, a lower bound for any spanner of finite stretch).

The lack of progress on understanding graph sketching for spanners have also been consequential in other computing models that use graph sketching as their primary tool, most notably, the dynamic streaming model. Indeed, complexity of spanners has been a tantalizing open question in the dynamic streaming model already since its introduction in [1] (for insertion-only streams, optimal algorithms that essentially match existential bounds under Erdős girth conjecture have been known since the introduction of the model itself in [18]; see, also [9, 19]).

This state-of-affairs raises the following question: *What is the best stretch-vs-size tradeoff possible for constructing spanners via graph sketching?* We make progress on this longstanding open question by proving a nearly-tight lower bound for a large family of graph sketching algorithms that encompasses prior work on spanners in [21] and most other closely related problems.

1.1 Our Contribution

We prove **a lower bound on the size of a special case yet general family of sketches for graph spanners**. This family, that shall be defined shortly, contains the prior sketching algorithm of [21] for graph spanners – our lower bound matches their bound up to lower order terms and is thus **nearly-optimal**. In addition, this family also contains many prior sketching algorithms for "cut-based" problems such as connectivity [1], vertex connectivity [23], and spectral sparsifiers [30] (and thus also cut sparsifiers). We now elaborate more on our results, starting with the definition of our sketches, which we call *random Gaussian sketches.*

Random Gaussian sketches

To date, the main success of graph sketching has been for *cut-based* problems [1, 2, 30, 23]. These sketches all work by encoding a graph G as its $\binom{n}{2} \times n$ *signed edge-incidence matrix* $B(G)$ (see Section 3.1) and then apply a sketching matrix A with few rows on the left to obtain the sketch $A \cdot B(G)$. The power of these sketches comes from surprisingly powerful cancellations that the use of the signed edge incidence matrix enables. In addition, the sketching matrix A of in these approaches implements a *sparse recovery* scheme on carefully chosen random subgraphs of the input graphs (e.g. uniformly random subgraphs of the input graph in the case of connectivity [1], cut sparsifiers [2], or spectral sparsifiers [30], and sampled vertex induced subgraphs in the case of spanners [21]).

To give a concrete example, let us consider the AGM sketches [1] for finding spanning forests. For any graph $G = (V, E)$ and any set of vertices $S \subseteq V$, adding up the columns of $B(G)$ corresponding to vertices in S, i.e., $\sum_{v \in S} B(G)^v$ gives us a vector with non-zero entries corresponding to edges of the cut $(S, V \setminus S)$. The linearity of matrix A then allows us to obtain

$$A \cdot \left(\sum_{v \in S} B(G)^v \right) = \sum_{v \in S} A \cdot B(G)^v,$$

[3] This state-of-affairs is in sharp contrast with another widely-studied problem of finding large matchings which is also absent from the list of successes in graph sketching; for the matching problem, *asymptotically tight* lower bounds which are much stronger than existential bounds are known; see [6, 15, 8].

for a cut S specified in the recovery phase. The sketching matrix A itself is an ℓ_0-sampler sketch that samples a non-zero entry of a vector v given $A \cdot v$ (see [26, 32]). An ℓ_0-sampler sketch is typically implemented via a simple sparse recovery sketch combined with a sampling matrix that samples the edges of the graph at $O(\log n)$ geometrically decreasing rates. Combined with the above approach, we can thus sample an edge from any cut of the graph specified in the recovery phase. The algorithm of [1] heavily builds on this subroutine by implementing Borůvka's algorithm for growing connected components via using these sketches to find an outgoing edge from each component in each step.

In this paper, we focus on this family of sketches where the sparse recovery scheme is implemented using *random Gaussian projections*. This means that each *row* of the sketching matrix is of the type $g \cdot S$ where S is an $\binom{n}{2} \times \binom{n}{2}$-dimensional diagonal sampling matrix – where $S_{(u,v),(u,v)} = 1$ iff (u,v) is sampled – and g is an $\binom{n}{2}$-dimensional vector of independent Gaussian variables:

$$\begin{bmatrix} \quad g \quad \end{bmatrix}_{1\times\binom{n}{2}} \times \begin{bmatrix} \\ \quad S \quad \\ \\ \end{bmatrix}_{\binom{n}{2}\times\binom{n}{2}} \times \begin{bmatrix} \\ B(G) \\ \\ \end{bmatrix}_{\binom{n}{2}\times n} = \begin{bmatrix} \; g \cdot S \cdot B(G) \; \end{bmatrix}_{1\times n}.$$

The entire sketch is obtained by taking s such rows where sampling matrices can be correlated but Gaussian vectors are independent. The recovery algorithm is given sampling matrices and the sketch but *not* Gaussian variables. We refer to s as the *dimension* of the sketch (thus size of the sketch is $O(s \cdot n)$). See Section 3.1 for formal definitions.

General "power" of random Gaussian sketches? In Appendix A, we show this family of sketches can implement many (but importantly *not* all) prior cut-based sketching algorithms in [1, 30, 23], and most importantly the spanner sketch of [21]. But we also point out that these sketches are *not* universal and one can easily construct problems where the power of these sketches does not match general sketching algorithms[4]. Perhaps more importantly, we assume that the recovery algorithm of these sketches is *oblivious* to the Gaussian vectors used in the sketching matrix which means that the recovery algorithm has a *partial* knowledge of the sketching matrix. A particular shortcoming of this is that while these sketches handle the "main" source of cancelations enabled by edge-incidence matrix, they do *not* handle a "secondary" source of cancelation: to obtain sketches of subgraphs of the input by generating the sketching matrix again at the recovery phase, apply it on some recovered part of the input, and subtract it from the original sketch (this approach is used in the edge connectivity and cut sparsifier sketch of [2] – although we note that random Gaussian sketches can recover a cut sparsifier by instead implementing the algorithm of [30]). We thus see the merit of study of this family as arguably the "most natural" candidate for finding spanners, given their past successes for closely related problems.

Our result

We prove a near-optimal lower bound on the dimension of random Gaussian sketches for constructing spanners, or even returning the distance of two fixed vertices (see also Theorem 5).

[4] Consider recovering the induced subgraph of the input on the first $\sqrt{n}$ vertices. A sparse recovery algorithm that spends $O(\sqrt{n})$ bits per each of these $\sqrt{n}$ vertices gives a sketch of size $O(n)$ for this problem. However, any random Gaussian sketch requires a dimension of $\Theta(\sqrt{n})$ that *cannot* be amortized over all vertices, leading to a sketch of size $O(n^{3/2})$ instead.

▶ **Result 1.** *Any random Gaussian sketch for constructing a d-spanner with constant probability of success requires dimension $\Omega(n^{1-o(1)}/d^{3/2})$, or put differently, any random Gaussian sketch of dimension s can only achieve a stretch of $\Omega((n/s)^{2/3-o(1)})$. The lower bound applies even to the problem of approximating the distance of two fixed vertices.*

Our lower bounds in Result 1 matches algorithmic bounds of [21] up to the $n^{o(1)}$ term for computing spanners via graph sketching (whose sketches fit the framework of random Gaussian sketches) for all stretch d. This establishes the optimality of these bounds at least among this popular family of graph sketching algorithms. We note that [21] conjectured optimality of their algorithmic bounds among *all* graph sketching techniques; our bounds in Result 1 makes partial progress towards settling this conjecture.

Before moving on, we note that for the case when dimension $s = \text{polylog}(n)$, corresponding to sketches of size $\widetilde{O}(n)$, Result 1 implies a lower bound of $n^{2/3-o(1)}$ on the stretch; this should be contrasted with the $O(\log n)$ bound of existential results on the stretch of spanners with $O(n)$ edges, suggesting that computing spanner is much harder using graph sketching (specifically via random Gaussian sketches) compared to existential bounds and arbitrary algorithms. Finally, the lower bound holds even for the algorithmically easier problem of simply estimating distance of two fixed vertices in the graph, as opposed to recovering the entire shortest path metric via a spanner.

Our techniques

We consider a family of hard instances that form a random chain of cliques of size (n/d) with diameter d, and a single edge e^* that connects two vertices at distance $\Theta(d)$ together (see Figure 1). It is easy to see that such e^* should belong to every $o(d)$-spanner of the graph and we prove that no random Gaussian sketch of "small" dimension can recover e^*. The proof is through analyzing how much a *single* random Gaussian projection can reveal information about e^*, or a bit more formally, the KL-divergence between the resulting sketches of two neighboring graphs that only differ on e^*. The rest follows by summing up this information across the s projections.

To prove the bound for a single projection, we use properties of Gaussian variables and KL-divergence to bound the information revealed about the edge e^* by the *effective resistance* of the *sampled* subgraph of the input after applying the sampling matrix. We prove that the distribution of our input, combined with a *hierarchical expander decomposition* of all edges of sampling matrix, implies that the sampled subgraph of the input form a chain of *expanders* (with proper lower bounds on both expansion and minimum degree). This step requires analyzing expansion of vertex-sampled subgraphs of an expander which can be of independent interest. Lastly, we bound the effective resistance of the edge e^* in this chain of expanders by exhibiting a proper *electrical flow* in the graph using properties of expanders.

Related work

In a recent independent work Chen, Khanna and Li [13] showed, similarly to our work, a lower bound matching the sketching dimension of [21] for linear sketches that can support *continuous* weight updates (as opposed to sketches that are only required to work for unweighted graphs). Thus, from the perspective of the ultimate result, the lower bound of [13] is incomparable to ours. Their lower bound works for more general sketches than ours (although still not universal), but assumes that these sketches work in the continuous weight update model; our lower bound assumes a special sketch structure, but works in the mode standard setting of unweighted graphs. There is quite a bit of overlap in techniques: both papers use expander decompositions and prove that expanders are preserved under vertex sampling (but the actual proofs of the corresponding lemmas are different).

2 Preliminaries

Notation

We use $\mathcal{N}(\mu, \sigma)$ to denote the Gaussian distribution with mean μ and variance σ^2. For any distributions P and Q, $\mathbb{D}(P \mid\mid Q)$ denotes the *KL-divergence* of P from Q and $\|P - Q\|_{\text{tvd}}$ is the *total variation distance* between P and Q.

For a graph $G = (V, E)$ on n vertices, we use $d_1, \ldots, d_n$ to denote the degrees of vertices in G. For any sets of vertices $S, T \subseteq V$, $E(S, T)$ denotes the set of edges between S and T and $\text{vol}_G(S) := \sum_{v \in S} d_v$ denotes the *volume* of S (we drop the subscript when clear). The *conductance* of G is defined as

$$\varphi(G) := \min_{S \subseteq V} \frac{|E(S, V \setminus S)|}{\min\{\text{vol}(S), \text{vol}(V \setminus S)\}}.$$

We say that G is a *φ-expander* if its conductance is at least φ.

For a graph G, $\mathbf{A}$ is the *adjacency matrix*, $\mathbf{D}$ is the *degree diagonal matrix*, $\mathbf{B}$ is the *signed edge-incidence matrix*, $\mathbf{L}$ is the *Laplacian matrix*, and $\widetilde{\mathbf{L}}$ is the *normalized Laplacian matrix*. The *spectral gap* of G is defined as the second smallest eigenvalue of $\widetilde{\mathbf{L}}$ which is related to the conductance via Cheeger's inequality. Finally, $R^G_{\text{eff}}(u, v)$ denotes the *effective resistance* between u, v when treating edges of G as resistors with unit resistance.

We also use the following (variant of) expander decomposition that bounds the minimum degree of resulting expanders. The proof is a simple modification of standard decompositions, e.g. in [29, 39].

▶ **Proposition 2.** *Let $G = (V, E)$ be any graph on n vertices and m edges, and $\varepsilon \in (0, 1/2)$ and $d_{\min} \geq 1$ be parameters. The vertices of G can be partitioned into subgraphs $H_1, \ldots, H_k$ such that:*

(i) *Each H_i is an ε-expander with minimum degree $d_{\min}$;*

(ii) *At most $8\varepsilon \cdot m \log n + n \cdot d_{\min}$ edges E_0 of G do not belong to any subgraph $\{H_i\}_{i \in [k]}$.*

3 Main Result

We formalize Result 1 in this section. We start by defining the sketching model, using random Gaussian projections, that we study. We then present our lower bound for constructing spanners (and in general preserving shortest path metric) using these sketches. Finally, we give the proof outline of this result here and postpone the proof of its main ingredients to the subsequent sections.

3.1 Random Gaussian Projections and Sketches

For an n-vertex graph $G = (V, E)$, its **signed edge-incidence** matrix is an $\binom{n}{2} \times n$-dimensional matrix $\mathbf{B} = \mathbf{B}(G)$ defined as follows:

- Each column corresponds to a vertex v and each row corresponds to a pair of vertices (u, w);
- The entry $\mathbf{B}_{(u,w),v}$ is either $+1$ if (u, w) is an edge in G and $v = u$, -1 if (u, w) is an edge in G and $v = w$, and 0 otherwise.

Note that for any edge $e = (u, v)$ of G, the corresponding row (u, v) in $\mathbf{B}$ has exactly one $+1$ at column u, one -1 at column v, and is otherwise 0. A row (u, v) of $\mathbf{B}$ which does not have a corresponding edge in G is the all-0 row.

Our sketches are based on taking random Gaussian projections of matrix $\mathbf{B}$, which roughly speaking correspond to sampling edge of G (using any sampling scheme oblivious to the graph), and multiply a Gaussian vector with signed edge-incidence matrix of the resulting graph. Formally,

▶ **Definition 3.** *Let $G = (V, E)$ be an n-vertex graph and consider the following:*

(i) **Sampling matrix:** *Let $\mathbf{S}$ be a $\binom{n}{2} \times \binom{n}{2}$-dimensional* diagonal *matrix with 0-1-values on the diagonal. Notice that the matrix $\mathbf{S} \cdot \mathbf{B}(G)$ is the edge-incidence matrix of the subgraph of G obtained by picking only those edges of G that their corresponding (diagonal) value in $\mathbf{S}$ is 1.*

(ii) **Gaussian projection:** *Let $\mathbf{g}$ be a $\binom{n}{2}$-dimensional vector of Gaussian random variables, where each entry is sampled independently from $\mathcal{N}(0, 1)$.*

A **random Gaussian projection** *of G with respect to $\mathbf{S}$ is an n-dimensional vector obtained by sampling $\mathbf{g} \sim \mathcal{N}(0, 1)^{\binom{n}{2}}$, and returning $\boldsymbol{p} := \mathbf{g} \cdot \mathbf{S} \cdot \mathbf{B}(G)$.*

Using Definition 3, we can define the sketches we focus on as follows.

▶ **Definition 4.** *Let Π be a problem defined on n-vertex graphs $G = (V, E)$. A* ***random Gaussian sketch*** *for Π is defined via the following pair:*

(i) **Sketching matrices:** *A distribution $\mathcal{D}^{\mathtt{smpl}}$ on s-tuples of sampling matrices for some $s \geq 1$.*

(ii) **Recovery algorithm:** *An algorithm that given s sampling matrices $\mathcal{S} = (\mathbf{S}_1, \ldots, \mathbf{S}_s) \sim \mathcal{D}^{\mathtt{smpl}}$ and s random Gaussian projection $\mathcal{P} = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_s)$ of any graph G with respect to these sampling matrices, returns a solution to $\Pi(G)$.*

We refer to s as the dimension of the sketch (note that a sketch of dimension s has size $O(s \cdot n)$).

A random Gaussian sketch for a graph G then consists of sampling the sketching matrices $\mathcal{S}$ from $\mathcal{D}^{\mathtt{smpl}}$ (independent of G), receiving random Gaussian projections $\mathcal{P}$, and running the recovery algorithm on $(\mathcal{S}, \mathcal{P})$ to return the solution.

We emphasize that in Definition 4, the recovery algorithm is given the sketching matrices used for random Gaussian projections explicitly, but is *not* given the Gaussian vectors themselves.

We note that our formalization of random Gaussian sketches is new to this paper, albeit it has been used implicitly in prior algorithmic results for in graph sketching literature. In Appendix A, we elaborate more on this connection and point out that how these sketches can be used to solve many of the canonical problems in graph sketching literature such as connectivity, minimum spanning tree, cut or spectral sparsifiers, and most closely related to ours, spanners. But we also emphasize that these sketches are *not* universal – see the discussion on the power of these sketches in Section 1.1.

3.2 The Lower Bound

The following is the formalization of Result 1 that we prove.

▶ **Theorem 5.** *For any absolute constant $\delta \in (0, 1)$, and integers $n \geq 1$ and $1 \leq d \leq n^{2/3-\delta}$, any random Gaussian sketch (Definition 4) that outputs a d-spanner of every given n-vertex graph G with probability at least $2/3$ has dimension (i.e., number of rows)*

$$\Omega(\frac{n^{1-\delta}}{d^{3/2}}).$$

Moreover, the lower bound continues to hold even if the algorithm is only required to answer the shortest path distance *between two* prespecified *vertices up to a factor of d.*

Theorem 5 can alternatively be seen as proving that any random Gaussian sketch of dimension s can only achieve a stretch of

$$\Omega((\frac{n}{s})^{2/3-\delta}),$$

for any constant $\delta > 0$. In light of the result of [21], the bounds obtained in Theorem 5 are optimal, up to $n^{o(1)}$-factors, for the entire range of dimension s or stretch d. In particular, Theorem 5 implies that to obtain a $n^{2/3-\Omega(1)}$-spanner, one needs random Gaussian sketches of dimension $n^{\Omega(1)}$. This makes progress on a conjecture of [21] that stated the same bounds for *arbitrary* sketches.

Finally, we also mention that Theorem 5 works even for the problem wherein we are given two vertices a and b of the graph, and our goal is to simply determine the distance of a and b in the graph using the sketches. This problem is algorithmically easier than finding a spanner of the graph in that firstly, we do not need to pick subset of edges of the graph G and can preserve the shortest path metric in any desired way, and secondly that we only need to maintain the distance between two vertices and not all pairs. Yet, effectively the entirety of our effort is to prove the result for spanners already and we get this stronger lower bound almost for free using standard ideas.

In the rest of this section, we first present a hard input distribution used to establish Theorem 5. We then state our main technical lemma that bounds the information revealed by a single random Gaussian projection on the graphs sampled from this distribution and show how this lemma easily implies the theorem. The next subsection then includes the proof outline of this technical lemma, whose main ingredients are postponed to the next sections.

3.3 A Hard Input Distribution

For any sufficiently large $n, d > 0$, we define a hard distribution $\mu = \mu(n, d)$ over n-vertex graphs. For simplicity, we prove the lower bound for $(d/2)$-spanners instead – reparameterizing d then implies the same asymptotic lower bound for exact d-spanners as well (see Figure 1).

Distribution $\boldsymbol{\mu(n, d)}$. A hard input distribution for $(d/2)$-spanners of n-vertex graphs.

1. Partition the vertices V into d groups $V_1, \ldots, V_d$: each $v \in V$ is sent to one of the groups chosen uniformly at random.
2. Let G be a graph obtained by placing a clique on each $V_i \cup V_{i+1}$ for $i \in [d-1]$.
3. Sample a pair of vertices $(u^*, v^*) \in \binom{V}{2}$ independently and return the graph $G + e^*$ for $e^* = (u^*, v^*)$.

In the following, we use $\mathbf{B} = \mathbf{B}(G)$ to the denote the signed edge-incidence matrix of G; we also use $\mathbf{B}(e^*)$ as the edge-incidence matrix of the n-vertex graph consisting of the single edge $e^* = (u^*, v^*)$. We emphasize that the final graph output by the distribution is $G + e^*$ (this notation will be make the latter parts of the proof cleaner).

We first establish a straightforward property of graphs sampled from μ in context of spanners.

▶ **Lemma 6.** *With constant probability over the choice of $(G, e^*) \sim \mu(n, d)$, every $(d/2)$-spanner of $G + e^*$ contains the edge $e^* = (u^*, v^*)$.*

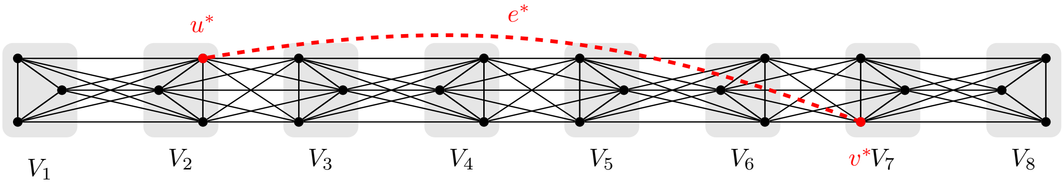


Figure 1 An illustration of $\mu = \mu(n, d)$ for $n = 24$ and $d = 8$. Any 4-spanner of G contains e^*.

Proof. For a graph G and pairs (u^*, v^*) sampled from $\mu(n, d)$,

$$\Pr_{u^*,v^*}\left(dist_G(u^*, v^*) > d/2\right) = \frac{1}{d^2} \cdot \left(O(d) + \sum_{i=1}^{d} |d/2 - i|\right) > \frac{1}{5},$$

where the equality holds since when $u^* \in V_1$, v^* can be in $V_{d/2+2}, \ldots, V_d$, when $u^* \in V_2$, v^* can be in $V_{d/2+3}, \ldots, V_d$, and so on ($O(d)$ handles the differences of even or odd choices of d and $d/2$).

Moreover, whenever the distance of u^*, v^* in G is more than $d/2$, any $(d/2)$-spanner of $G + e^*$ should contain the edge e^*, as otherwise the distance between u^* and v^* in the spanner will be more than $d/2$ times their distance in $G + e^*$, violating the bound on the stretch of the spanner. ◀

The following lemma is the main technical contribution of our work. Roughly speaking, this lemma bounds the "information" that can be learned about the edge e^* in μ using a *single* sub-sampled Gaussian projection of a graph sampled from μ.

▶ **Lemma 7.** *Let* $\mathbf{S}$ *be any sampling matrix and consider a single random Gaussian projection with respect to* $\mathbf{S}$*. For* (G, e^*) *sampled from* $\mu = \mu(n, d)$*,*

$$\mathop{\mathbb{E}}_{G,e^*}\left[\min\{1, \mathbb{D}_{\mathbf{g}}(\mathbf{g}\cdot\mathbf{S}\cdot\mathbf{B}(G) \mid\mid \mathbf{g}\cdot\mathbf{S}\cdot(\mathbf{B}(G) + \mathbf{B}(e^*)))\}\right] = O(\frac{d^{3/2}}{n^{1-\delta}}),$$

for any constant $\delta > 0$*, where the KL-divergence is taken only over the Gaussian variables.*

Before getting to the proof of Lemma 7, we show that it implies Theorem 5 immediately.

Proof of Theorem 5 (assuming Lemma 7). Let $(\mathcal{D}^{\mathtt{smpl}}, \mathcal{A})$ be any sub-sampled Gaussian sketch of dimension $s \geq 1$ for recovering a $(d/2)$-spanner. Consider a distribution μ' on n-vertex graphs defined as follows:

- Distribution μ': Sample (G, e^*) from μ and $\theta \in \{0, 1\}$ uniformly at random; if $\theta = 0$, return G, otherwise return $G + e^*$.

Let G' be a graph sampled from μ'. Suppose we sample $\mathcal{S} = (\mathbf{S}_1, \ldots, \mathbf{S}_s)$ from $\mathcal{D}^{\mathtt{smpl}}$ and receive sub-sampled Gaussian projections $\mathcal{P} = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_s)$ where for every $i \in [s]$, $\boldsymbol{p}_i = \mathbf{g}_i \cdot \mathbf{S}_i \cdot \mathbf{B}(G')$ for a Gaussian vector $\mathbf{g}_i$. Additionally, suppose we are even given $(\mathcal{S}, G, e^*)$, and thus the only unknown information is whether or not $e^* \in G'$ also, i.e., whether $\theta = 1$ or not. This way, we can run $\mathcal{A}$, using $\mathcal{S}, \mathcal{P}$ as input, to obtain a $(d/2)$-spanner of G': if e^* belongs to this spanner, we declare e^* is in G' and otherwise we say it is not. By Lemma 6, we are going to be able to determine the value of θ with probability $1/2 + \Theta(1)$. This implies that over the distribution μ',

$$\|[(\mathcal{S}, G, e^*, \mathcal{P}) \mid \theta = 0] - [(\mathcal{S}, G, e^*, \mathcal{P}) \mid \theta = 1]\|_{\mathrm{tvd}} = \Omega(1), \tag{1}$$

as otherwise, we cannot estimate the value of θ with probability better than $1/2 + o(1)$ given our input $(\mathcal{S}, G, e^*, \mathcal{P})$ which is sampled from either $\mu' \mid \theta = 0$ or $\mu' \mid \theta = 1$. We now have,

$$\text{LHS of Eq (1)} \leq \mathop{\mathbb{E}}_{(\mathcal{S},G,e^*)} \|[\mathcal{P} \mid \theta = 0, \mathcal{S}, G, e^*] - [\mathcal{P} \mid \theta = 1, \mathcal{S}, G, e^*]\|_{\text{tvd}}$$

(as the distribution of $(\mathcal{S}, G, e^*)$ is the same under both $\theta = 0$ and $\theta = 1$)

$$\leq \sqrt{\mathop{\mathbb{E}}_{(\mathcal{S},G,e^*)} \min\{1, \mathbb{D}(\mathcal{P} \mid \theta = 0, \mathcal{S}, G, e^* \mid\mid \mathcal{P} \mid \theta = 1, \mathcal{S}, G, e^*)\}}$$

(by Pinsker's inequality, the fact that TVD is bounded by 1, and concavity of $\sqrt{\cdot}$)

$$= \sqrt{\mathop{\mathbb{E}}_{(\mathcal{S},G,e^*)} \left[\min\{1, \sum_{i=1}^{s} \mathbb{D}(\boldsymbol{p}_i \mid \theta = 0, \mathcal{S}, G, e^* \mid\mid \boldsymbol{p}_i \mid \theta = 1, \mathcal{S}, G, e^*)\}\right]}$$

(by chain rule of KL-divergence as $\boldsymbol{p}_i$'s are now only function of $\mathbf{g}_i$'s and so are independent)

$$\leq \sqrt{\sum_{i=1}^{s} \mathop{\mathbb{E}}_{(\mathcal{S},G,e^*)} \left[\min\{1, \mathbb{D}(\boldsymbol{p}_i \mid \theta = 0, \mathcal{S}, G, e^* \mid\mid \boldsymbol{p}_i \mid \theta = 1, \mathcal{S}, G, e^*)\}\right]}$$

(we can take min inside the summation to get an upper bound)

$$= \sqrt{\sum_{i=1}^{s} \mathop{\mathbb{E}}_{(\mathbf{S}_i,G,e^*)} \left[\min\{1, \mathbb{D}(\boldsymbol{p}_i \mid \theta = 0, \mathbf{S}_i, G, e^* \mid\mid \boldsymbol{p}_i \mid \theta = 1, \mathbf{S}_i, G, e^*)\}\right]}$$

(as $\boldsymbol{p}_i$ is only a function of $\mathbf{S}_i$ conditioned on G, e^*)

$$= \sqrt{\sum_{i=1}^{s} \mathop{\mathbb{E}}_{(\mathbf{S}_i,G,e^*)\sim\mu} \min\{1, \mathbb{D}_{\mathbf{g}_i}(\mathbf{g}_i \cdot \mathbf{S}_i \cdot \mathbf{B}(G) \mid\mid \mathbf{g}_i \cdot \mathbf{S}_i \cdot (\mathbf{B}(G) + \mathbf{B}(e^*)))\}},$$

where the last equality is because input graph G' in μ' is G when $\theta = 0$ and $G + e^*$ when $\theta = 1$, and distribution of $(\mathbf{S}_i, G, e^*, \mathbf{g}_i)$ is the same under μ and μ'.

Now given that $\mathbf{S}_i \perp (G, e^*)$ in μ, each term in the RHS above is the same quantity upper bounded in Lemma 7. Thus, combining Eq (1), the above equation, and Lemma 7, we get that

$$\Omega(1) = \|[(\mathcal{S}, G, e^*, \mathcal{P}) \mid \theta = 0] - [(\mathcal{S}, G, e^*, \mathcal{P}) \mid \theta = 1]\|_{\text{tvd}} \leq \sqrt{s \cdot O(\frac{d^{3/2}}{n^{1-\delta}})},$$

which implies that $s = \Omega(n^{1-\delta}/d^{3/2})$ as desired. This implies the first part of Theorem 5.

The proof of the second part follows almost immediately from the above argument as follows. Consider the following distribution:

- Distribution μ'': Sample (G', e^*, θ) from μ'. Add two new vertices a and b to the graph and add edges (a, u^*) and (v^*, b) to the graph as well.

Let G'' be a graph sampled from μ''. Consider the distance between a and b in G'': if $\theta = 1$ in the sampled G', distance of a and b is 3, otherwise, if $\theta = 0$, by the same argument as Lemma 6, the distance between a and b is more than $(d/2)$ with constant probability. This means that if our algorithm could simply estimate the distance of a and b to within a factor of $(d/6)$, it can determine the value of θ with probability $1/2 + \Theta(1)$.

Now if we further give u^*, v^*, and the Gaussian variables on all edges incident to a or b to the recovery algorithm, what the algorithm knows becomes the sketches of $G'' \setminus \{a, b\}$ (by simply subtracting the corresponding Gaussians). Since the Gaussians revealed are independent of the sketch of $G'' \setminus \{a, b\}$, the same exact argument as the first part now implies that the same lower bound of $s = \Omega(d^{3/2}/n^{1-\delta})$ on the sketch dimension. Given that the number of vertices in G'' is $n + 2$, and by re-parameterizing d with a constant factor, we obtain the desired lower bound. This concludes the proof of Theorem 5. ◀

3.4 Proof Outline of Lemma 7

We now present the proof outline of Lemma 7, postponing the proof of its two main ingredients to the next two sections. For convenience, we restate Lemma 7 below.

▶ **Lemma** (Restatement of Lemma 7). *Let* $\mathbf{S}$ *be any sampling matrix and consider a single random Gaussian projection with respect to* $\mathbf{S}$*. For* (G, e^*) *sampled from* $\mu = \mu(n, d)$*,*

$$\mathop{\mathbb{E}}_{G,e^*}\left[\min\{1, \mathbb{D}_{\mathbf{g}}(\mathbf{g}\cdot\mathbf{S}\cdot\mathbf{B}(G) \mid\mid \mathbf{g}\cdot\mathbf{S}\cdot(\mathbf{B}(G)+\mathbf{B}(e^*)))\}\right] = O(\frac{d^{3/2}}{n^{1-\delta}}),$$

for any constant $\delta > 0$*, where the KL-divergence is taken only over the Gaussian variables.*

To continue, we define the following notation for the sampling matrix $\mathbf{S}$:

- *graph*$(\mathbf{S})$: the graph on V containing all edges $e \in \binom{V}{2}$ where $\mathbf{S}_{e,e} = 1$.
- $m(\mathbf{S})$: the number of *edges* in *graph*$(\mathbf{S})$.
- $G(\mathbf{S})$: the subgraph of G on edges that belong to *graph*$(\mathbf{S})$, i.e., $G(\mathbf{S}) := G \cap \mathit{graph}(\mathbf{S})$. Note that this way we have, $\mathbf{B}(G(\mathbf{S})) = \mathbf{S}\cdot\mathbf{B}(G)$ and $\mathbf{B}(G(\mathbf{S})+e^*) = \mathbf{S}\cdot(\mathbf{B}(G)+\mathbf{B}(e^*))$.

Ingredient one: from KL-divergence to effective resistances

The first key step of the proof of Lemma 7 is to relate the KL-divergence term of Lemma 7 to *effective resistance* of the edge e^* in the underlying sampled graph. Formally,

▶ **Lemma 8.** *For any sampling matrix* $\mathbf{S}$*, any fixed* G*, and any pair of vertices* $e = (u, v) \in \binom{V}{2}$*,*

$$\min\{1, \mathbb{D}_{\mathbf{g}}(\mathbf{g}\cdot\mathbf{S}\cdot\mathbf{B}(G) \mid\mid \mathbf{g}\cdot\mathbf{S}\cdot(\mathbf{B}(G)+\mathbf{B}(e)))\} \leq 2\cdot R_{\text{eff}}^{G(\mathbf{S})+e}(u, v).$$

We will apply Lemma 8 to the choice of edge $e^* = e$ whenever e^* belongs to *graph*$(\mathbf{S})$, i.e., when e^* is sampled by the sampling matrix $\mathbf{S}$. To prove Lemma 8, we first calculate the KL-divergence between two high-dimensional Gaussians in terms of their covariance matrices. Then we observe that the covariance matrix of $\mathbf{g}\cdot\mathbf{S}\cdot\mathbf{B}(G)$ is simply the *Laplacian matrix* of $G(\mathbf{S})$. The lemma is proved by plugging in the Laplacian matrices of $G(\mathbf{S})$ and $G(\mathbf{S}) + e$, and applying the connection between effective resistance and Laplacian matrix. The proof is provided in the full version.

Ingredient two: bounding effective resistances via expanders

Our strategy is now to bound the effective resistance of the edge e^* in $G(\mathbf{S}) + e^*$. To do so, we will identify a "good"-*expander* subgraph H of the *graph*$(\mathbf{S})$ that contains the edge e^*, and then primarily focus on the edges of H that appear in $G(\mathbf{S})$ to bound the effective resistance of e^* also. The following lemma is the heart of the proof.

▶ **Lemma 9.** *For any sampling matrix* $\mathbf{S}$*, suppose* H *is any subgraph of graph*$(\mathbf{S})$ *which is an* ε*-expander with min-degree* D *for some* $\varepsilon > 0$ *and* $D \geq (\varepsilon^{-8}\cdot n^{\delta})\cdot d$ *for a constant* $\delta > 0$*. For any edge* $e = (u, v) \in H$*,*

$$\mathop{\mathbb{E}}_{G}\left[R_{\text{eff}}^{G(\mathbf{S})+e}(u, v)\right] = O(\varepsilon^{-4})\cdot\left(\frac{d}{D} + \frac{d^3}{D^2}\right).$$

We will use a hierarchical expander decomposition of *graph*$(\mathbf{S})$ to identify an expander that contains the edge e^* and then apply Lemma 9 to this expander and edge e^*. To prove Lemma 9, we first observe that adding edges to a graph could only decrease the effective resistance, and thus, it suffices to study the effective resistance of (u, v) in $G \cap H$. Note that $G \cap H$ randomly partitions the vertices into d sets, and only keeps the edges of H with endpoints in the same set or adjacent sets. We then show that because H is an expander with large min-degree, $G \cap H$ restricted to any two adjacent sets must also be an expander with large min-degree with high probability. Hence, $G \cap H$ looks like a "chain of expanders" (which we call *a balanced path of expanders*), where adjacent expanders have a constant fraction overlap. Finally, we show that since $G \cap H$ overall is well-connected, if we place a unit electric flow from u to v, the flow will be well-spread across the graph. Most edges have a small current, i.e., a low potential difference. Therefore, it allows us to argue that the potential difference between u and v is also small, i.e., the effective resistance between u and v is small. The detailed proof is provided in the full version.

Putting everything together

We now put these two ingredients together to prove Lemma 7. In order to be able to apply our second tool in Lemma 9, we need a *hierarchical* expander decomposition of *graph*$(\mathbf{S})$, which shows that the edge e^* is "more likely" to land in "better" expanders of *graph*$(\mathbf{S})$ for the purpose of Lemma 9 – here, "better" means an expander with a higher minimum degree (the parameter in Lemma 9 that governs the final bound).

▶ **Lemma 10.** *For every $t \geq 1$, we can partition edges of graph$(\mathbf{S})$ into t sets $E_1(\mathbf{S}), \ldots, E_t(\mathbf{S})$ such that:*

(i) *For any $i \leq t$, define $m_i(\mathbf{S}) := |E_i(\mathbf{S})|$; then, $m_1(\mathbf{S}) \leq m(\mathbf{S})$ and $m_{i+1}(\mathbf{S}) \leq m_i(\mathbf{S})/2$.*

(ii) *For any $i < t$, there is some $k_i \geq 1$ such that edges in $E_i(\mathbf{S})$ can be partitioned into ε-expanders $H^i_1, \ldots, H^i_{k_i}$ with minimum degree at least D_i for parameters*[5]

$$\varepsilon := \frac{1}{36 \log n} \quad \textit{and} \quad D_i \geq \frac{m_i(\mathbf{S})}{36n}.$$

Proof. For simplicity of exposition, we drop $(\mathbf{S})$ when denoting $E_i(\mathbf{S})$'s in the following. We construct $E_1, \ldots, E_t$ inductively using an auxiliary set of edges $F_0, \ldots, F_t$. Start with F_0 being the set of all edges in *graph*$(\mathbf{S})$ and for $i = 1$ to t do:

1) Apply the expander decomposition of Proposition 2 to F_{i-1} with parameters

$$\varepsilon = \frac{1}{36 \log n} \quad \text{and} \quad d_{\min} = D_i = \frac{|F_{i-1}|}{36n},$$

to get ε-expanders $H^i_1, \ldots, H^i_{k_i}$ each with minimum degree at least D_i.

2) Let E_i be the union of edges assigned to the expanders in the decomposition of Proposition 2 in the previous step, and F_i be the leftover edges. Continue to iteration $i + 1$.

We argue that $|F_i| \leq |F_{i-1}|/4$ for all $i \leq t$. For $i > 0$, we have that $|F_i|$ is the number of leftover edges of the decomposition and thus by Proposition 2,

$$|F_i| \leq 8\varepsilon \cdot |F_{i-1}| \cdot \log n + n \cdot D_i = \frac{8\,|F_{i-1}|}{36} + \frac{|F_{i-1}|}{36} = \frac{|F_{i-1}|}{4}.$$

Now firstly, $E_i = F_{i-1} \setminus F_i$ and so by the above bound, $|E_i| \geq 2\,|F_i|$. At the same time, $E_{i+1} \subseteq F_i$ for and thus $|E_i| \geq 2\,|E_{i+1}|$. This proves the first part.

Secondly, we get property (ii) of the lemma by the choice of $\varepsilon = 1/36 \log n$ in the decomposition and since $D_i = |F_{i-1}|/36n \geq |E_i|/36n$ as $E_i \subseteq F_{i-1}$. ◀

[5] Notice that the edges $E_t(\mathbf{S})$ admit no such type of expander decomposition in our partitioning.

We now have all the tools needed to prove Lemma 7. For the rest of the proof, we fix a partitioning $(E_1(\mathbf{S}), \ldots, E_t(\mathbf{S}))$ of $graph(\mathbf{S})$ using Lemma 10 for some $t \geq 1$ such that:

$$t \text{ is the } \underline{\text{largest}} \text{ index where:} \quad m_{t-1}(\mathbf{S}) \geq n^{1+\delta} \cdot d^{3/2}, \tag{2}$$

where $\delta > 0$ is the absolute constant in Lemma 7. This means that for every $e \in E_i(\mathbf{S})$ for $i < t$, the edge e belongs to some ε-expander H^i_j for $j \in [k_i]$ with min-degree D_i such that,

$$\varepsilon = \frac{1}{36 \log n} \qquad \text{and} \qquad D_i \geq \frac{m_i(\mathbf{S})}{36n}. \tag{3}$$

This also implies that $D_i \geq (1/36) \cdot d^{3/2} \cdot n^{\delta} \geq (\varepsilon^{-10} \cdot n^{\delta'}) \cdot d$ for some absolute constant $\delta' > 0$ which allows us to apply Lemma 9 to each expander H^i_j in the proof.

We now have,

$$\begin{aligned}
\text{LHS of Lemma 7} &= \mathop{\mathbb{E}}_{G,e^*}\Big[\min\{1, \mathbb{D}_{\mathbf{g}}(\mathbf{g}\cdot\mathbf{S}\cdot\mathbf{B}(G) \mid\mid \mathbf{g}\cdot\mathbf{S}\cdot(\mathbf{B}(G)+\mathbf{B}(e^*)))\}\Big] \\
&= \sum_{e \in graph(\mathbf{S})} \Pr(e^* = e) \cdot \mathop{\mathbb{E}}_{G \mid e^* = e}\Big[\min\{1, \mathbb{D}_{\mathbf{g}}(\mathbf{g}\cdot\mathbf{S}\cdot\mathbf{B}(G) \mid\mid \mathbf{g}\cdot\mathbf{S}\cdot(\mathbf{B}(G)+\mathbf{B}(e^*)))\}\Big] \\
&\qquad \text{(whenever } e^* \notin graph(\mathbf{S})\text{, both terms of the KL-divergence will be the same and thus it will be 0)} \\
&= \sum_{i=1}^{t} \sum_{e \in E_i(\mathbf{S})} \Pr(e^* = e) \cdot \mathop{\mathbb{E}}_{G}\Big[\min\{1, \mathbb{D}_{\mathbf{g}}(\mathbf{g}\cdot\mathbf{S}\cdot\mathbf{B}(G) \mid\mid \mathbf{g}\cdot\mathbf{S}\cdot(\mathbf{B}(G)+\mathbf{B}(e^*)))\}\Big] \\
&\qquad \text{(by the partitioning of edges of } graph(\mathbf{S}) \text{ and since } G \perp e^* \text{ in } \mu) \\
&= \frac{1}{\binom{n}{2}} \sum_{i=1}^{t} \sum_{e \in E_i(\mathbf{S})} \Big[\min\{1, \mathbb{D}_{\mathbf{g}}(\mathbf{g}\cdot\mathbf{S}\cdot\mathbf{B}(G) \mid\mid \mathbf{g}\cdot\mathbf{S}\cdot(\mathbf{B}(G)+\mathbf{B}(e)))\}\Big] \\
&\qquad \text{(as the marginal distribution of } e^* \text{ is uniform over } \tbinom{V}{2} \text{ and we conditioned on } e^* = e) \\
&\leq \frac{m_t(\mathbf{S})}{\binom{n}{2}} + \frac{1}{\binom{n}{2}} \cdot \sum_{i=1}^{t-1} \sum_{e=(u,v) \in E_i(\mathbf{S})} \mathop{\mathbb{E}}_{G}\Big[2 \cdot R_{\text{eff}}^{G(\mathbf{S})+e}(u,v)\Big] \\
&\qquad \text{(using the trivial upper bound of 1 for } E_t(\mathbf{S}) \text{ and Lemma 8 for } E_1(\mathbf{S}), \ldots, E_{t-1}(\mathbf{S})) \\
&= \frac{m_t(\mathbf{S})}{\binom{n}{2}} + \frac{2}{\binom{n}{2}} \cdot \sum_{i=1}^{t-1} \sum_{j=1}^{k_i} \sum_{e=(u,v) \in H^i_j} \mathop{\mathbb{E}}_{G}\Big[R_{\text{eff}}^{G(\mathbf{S})+e}(u,v)\Big] \\
&\qquad \text{(as each } E_i(\mathbf{S}) \text{ for } i < t \text{ is partitioned into expanders } H^i_1, \ldots, H^i_{k_i} \text{ by Lemma 10)} \\
&= \frac{m_t(\mathbf{S})}{\binom{n}{2}} + \frac{2}{\binom{n}{2}} \cdot \sum_{i=1}^{t-1} \sum_{j=1}^{k_i} \sum_{e=(u,v) \in H^i_j} O(\varepsilon^{-4}) \cdot \Big(\frac{d}{D_i} + \frac{d^3}{D_i^2}\Big) \\
&\qquad \text{(by Lemma 9 as each } H^i_j \text{ is an } \varepsilon\text{-expander with min-degree } D_i \text{ (and by Eq (3) we can use the lemma))} \\
&= \frac{m_t(\mathbf{S})}{\binom{n}{2}} + \frac{2}{\binom{n}{2}} \cdot \sum_{i=1}^{t-1} m_i(\mathbf{S}) \cdot O(\log^4 n) \cdot \Big(\frac{d \cdot n}{m_i(\mathbf{S})} + \frac{d^3 \cdot n^2}{m_i(\mathbf{S})^2}\Big) \\
&\qquad \text{(as } \varepsilon = \Theta(1/\log n) \text{ and } D_i \geq m_i(\mathbf{S})/12n \text{ by Eq (3) and } H^i_1, \ldots, H^i_{k_i} \text{ have } m_i(\mathbf{S}) \text{ edges in total)} \\
&= \frac{m_t(\mathbf{S})}{\binom{n}{2}} + O(\log^5 n \cdot \frac{d}{n}) + O(\log^4 n) \cdot \frac{d^3}{m_{t-1}(\mathbf{S})} \\
&\qquad \text{(as } m_i(\mathbf{S})\text{'s decrease (at least) by a geometric series and } t = O(\log n) \text{ by Lemma 10)} \\
&\leq \frac{n^{1+\delta} \cdot d^{3/2}}{\binom{n}{2}} + O(\log^5 n \cdot \frac{d}{n}) + O(\log^4 n) \cdot \frac{d^3}{n^{1+\delta} \cdot d^{3/2}} \qquad \text{(by the choice of } t \text{ in Eq (2))} \\
&= O(\frac{d^{3/2}}{n^{1-\delta}}).
\end{aligned}$$

This concludes the proof of Lemma 7. Due to page limits, we omit the proof of Lemma 8 and Lemma 9 here. They can be found in the full version.

References

1 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 459–467, 2012.

2 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 5–14, 2012.

3 Yuqing Ai, Wei Hu, Yi Li, and David P. Woodruff. New characterizations in turnstile streams with applications. In Ran Raz, editor, *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, volume 50 of *LIPIcs*, pages 20:1–20:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016.

4 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 20–29. ACM, 1996.

5 Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear algorithms for $(\Delta + 1)$ vertex coloring. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 767–786. SIAM, 2019.

6 Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1345–1364, 2016.

7 Sepehr Assadi, Gillat Kol, and Rotem Oshman. Lower bounds for distributed sketching of maximal matchings and maximal independent sets. In Yuval Emek and Christian Cachin, editors, *PODC '20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020*, pages 79–88. ACM, 2020.

8 Sepehr Assadi and Vihan Shah. An asymptotically optimal algorithm for maximum matching in dynamic streams. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 – February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 9:1–9:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

9 Surender Baswana and Sandeep Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Struct. Algorithms*, 30(4):532–563, 2007.

10 Florent Becker, Martín Matamala, Nicolas Nisse, Ivan Rapaport, Karol Suchan, and Ioan Todinca. Adding a referee to an interconnection network: What can(not) be computed in one round. In *25th IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2011, Anchorage, Alaska, USA, 16-20 May, 2011 – Conference Proceedings*, pages 508–514. IEEE, 2011.

11 Florent Becker, Pedro Montealegre, Ivan Rapaport, and Ioan Todinca. The simultaneous number-in-hand communication model for networks: Private coins, public coins and determinism. In Magnús M. Halldórsson, editor, *Structural Information and Communication Complexity – 21st International Colloquium, SIROCCO 2014, Takayama, Japan, July 23-25, 2014. Proceedings*, volume 8576 of *Lecture Notes in Computer Science*, pages 83–95. Springer, 2014.

12 Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. Finding frequent items in data streams. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan J. Eidenbenz, and Ricardo Conejo, editors, *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings*, volume 2380 of *Lecture Notes in Computer Science*, pages 693–703. Springer, 2002.

13 Yu Chen, Sanjeev Khanna, and Huan Li. On weighted graph sparsification by linear sketching. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 474–485. IEEE, 2022.

14 Graham Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. 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 29–38. Springer, 2004.

15 Jacques Dark and Christian Konrad. Optimal lower bounds for matching and vertex cover in dynamic graph streams. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 30:1–30:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

16 David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

17 Michael Elkin and Chhaya Trehan. $(1+\epsilon)$-approximate shortest paths in dynamic streams. *CoRR*, abs/2107.13309, 2021. arXiv:2107.13309.

18 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. In Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, *Automata, Languages and Programming: 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings*, volume 3142 of *Lecture Notes in Computer Science*, pages 531–543. Springer, 2004.

19 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the data-stream model. *SIAM J. Comput.*, 38(5):1709–1727, 2008.

20 Manuel Fernandez, David P. Woodruff, and Taisuke Yasuda. Graph spanners in the message-passing model. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 77:1–77:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

21 Arnold Filtser, Michael Kapralov, and Navid Nouri. Graph spanners by sketching in dynamic streams and the simultaneous communication model. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10–13, 2021*, pages 1894–1913. SIAM, 2021.

22 Mohsen Ghaffari and Merav Parter. MST in log-star rounds of congested clique. In George Giakkoupis, editor, *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*, pages 19–28. ACM, 2016.

23 Sudipto Guha, Andrew McGregor, and David Tench. Vertex and hyperedge connectivity in dynamic graph streams. In Tova Milo and Diego Calvanese, editors, *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31 – June 4, 2015*, pages 241–247. ACM, 2015.

24 James W. Hegeman, Gopal Pandurangan, Sriram V. Pemmaraju, Vivek B. Sardeshmukh, and Michele Scquizzato. Toward optimal bounds in the congested clique: Graph connectivity and MST. In Chryssis Georgiou and Paul G. Spirakis, editors, *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21–23, 2015*, pages 91–100. ACM, 2015.

25 William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space 26. *Contemporary mathematics*, 26:28, 1984.

26 Hossein Jowhari, Mert Saglam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In Maurizio Lenzerini and Thomas Schwentick, editors, *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, pages 49–58. ACM, 2011.

27 Tomasz Jurdzinski and Krzysztof Nowicki. MST in $O(1)$ rounds of congested clique. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2620–2632. SIAM, 2018.

28 John Kallaugher and Eric Price. Separations and equivalences between turnstile streaming and linear sketching. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1223–1236. ACM, 2020.

29 Ravi Kannan, Santosh S. Vempala, and Adrian Vetta. On clusterings – good, bad and spectral. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 367–377. IEEE Computer Society, 2000.

30 Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 561–570. IEEE Computer Society, 2014.

31 Michael Kapralov, Aida Mousavifar, Cameron Musco, Christopher Musco, Navid Nouri, Aaron Sidford, and Jakab Tardos. Fast and space efficient spectral sparsification in dynamic streams. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1814–1833. SIAM, 2020.

32 Michael Kapralov, Jelani Nelson, Jakub Pachocki, Zhengyu Wang, David P. Woodruff, and Mobin Yahyazadeh. Optimal lower bounds for universal relation, and for samplers and finding duplicates in streams. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 475–486. IEEE Computer Society, 2017.

33 Michael Kapralov, Navid Nouri, Aaron Sidford, and Jakab Tardos. Dynamic streaming spectral sparsification in nearly linear time and space. *CoRR*, abs/1903.12150, 2019. `arXiv:1903.12150`.

34 Michael Kapralov and David P. Woodruff. Spanners and sparsifiers in dynamic streams. In Magnús M. Halldórsson and Shlomi Dolev, editors, *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, pages 272–281. ACM, 2014.

35 Yi Li, Huy L. Nguyen, and David P. Woodruff. Turnstile streaming algorithms might as well be linear sketches. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 – June 03, 2014*, pages 174–183. ACM, 2014.

36 Andrew McGregor, David Tench, Sofya Vorotnikova, and Hoa T. Vu. Densest subgraph in dynamic graph streams. In Giuseppe F. Italiano, Giovanni Pighizzini, and Donald Sannella, editors, *Mathematical Foundations of Computer Science 2015 – 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part II*, volume 9235 of *Lecture Notes in Computer Science*, pages 472–482. Springer, 2015.

37 Jelani Nelson and Huacheng Yu. Optimal lower bounds for distributed and streaming spanning forest computation. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1844–1860. SIAM, 2019.

38 Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. Fast distributed algorithms for connectivity and MST in large graphs. *ACM Trans. Parallel Comput.*, 5(1):4:1–4:22, 2018.

39 Thatchaphol Saranurak and Di Wang. Expander decomposition and pruning: Faster, stronger, and simpler. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2616–2635. SIAM, 2019.

40 Tamás Sarlós. Improved approximation algorithms for large matrices via random projections. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 143–152. IEEE Computer Society, 2006.

41 David P. Woodruff. Sketching as a tool for numerical linear algebra. *Found. Trends Theor. Comput. Sci.*, 10(1-2):1–157, 2014.

42 Huacheng Yu. Tight distributed sketching lower bound for connectivity. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10–13, 2021*, pages 1856–1873. SIAM, 2021.

A Implementing Prior Work Via Random Gaussian Sketches

We now outline implementations of existing works on graph sketching in our model.

A.1 ℓ_0-samplers and connectivity sketches

Recall that in the ℓ_0-sampling problem one needs to design a sketching matrix A such that for every $x \in \mathbb{R}^n$ one can recover a uniformly random element of x from Ax (to within total variation distance δ) or output FAIL (with failure probability bounded by δ)[6]. We outline a construction of an ℓ_0-sampler in our model, i.e. where every rows of the sketch A is of the form $g \cdot S$, where S is an arbitrary matrix with zeros and ones on the diagonal and zeros on off-diagonal entries (S is known to the decoder) and g is a vector with i.i.d. unit variance Gaussian entries (g is not known to the decoder). Note that our ℓ_0-sampler only needs to work for vectors x whose entries are in $\{-1, 0, +1\}$, as this is the case in all applications of graph sketching.

We first recall the construction of a basic ℓ_0 sampler (see [26] for a space-optimal construction). For integer j between 0 and $\lceil \log_2 n \rceil$ let $x^j \in \mathbb{R}^n$ denote the restriction of x^j to elements of a subset of the universe $[n]$ that includes every element independently with probability 2^{-j}. There exists j^* such that with constant probability x^j contains exactly one nonzero. To determine the value of j^* or conclude that such an index does not exist, it suffices to estimate the ℓ_2^2 norm of x to within a $1 \pm 1/3$ factor, for example (since nonzero entries of x equal 1 in absolute value). The latter can be achieved (with at most inverse polynomial failure probability) by averaging squared dot products of $O(\log n)$ independent Gaussian vectors with x^j, which is allowed by our model. Note that here the decoder indeed does not need to know the Gaussian vectors, as required. If j^* exists, one must recover the identity of the nonzero element. The typical way to do it is to compute the dot product of x^{j^*} with the vector whose i-th coordinate equals i, for every $i \in [n]$. This is not available in our model. To replace this approach, for every $j = 0, \ldots, \lceil \log_2 n \rceil$ and $b = 0, \ldots, \lceil \log_2 n \rceil$ approximate the ℓ_2^2 norm of the vector x^j restricted to the set of elements in $[n]$ that have 1 in the b-th position in their binary representation using $O(\log n)$ dot products with i.i.d. Gaussians. This allows one to read off the binary representation of the nonzero in x^{j^*}, and therefore yields an ℓ_0 sampler.

Graph connectivity and spanning trees

Since an ℓ_0-sampler is the only sketch used by the connectivity sketch of [1], it follows that a spanning forest of the input graph can be recovered by a sketch that fits our model and has a polylogarithmic number of rows.

[6] Note that these two parameters appear differently in the space complexity of ℓ_0-sampling, and are therefore treated separately in works that obtain optimal space bounds for ℓ_0-samplers [1]. We set both parameters to δ for simplicity.

Approximate vertex connectivity

The result of [23] uses the spanning tree sketch of [1] black box (the sketch is applied to random vertex induced subgraphs) to approximate vertex connectivity. Since the sketch of [1] can be implemented in our model, as described above, the result of [23] also can.

A.2 ℓ_2-heavy hitters, spectral sparsifiers and spanners

Recall that in the φ-heavy hitters problem in ℓ_2 one needs to design a sketching matrix A such that for every $x \in \mathbb{R}^n$ one can recover a list of elements $L \subseteq [n]$ such that every $i \in [n]$ satisfying $x_i^2 \geq \varphi\|x\|_2^2$ belongs to L and no $i \in [n]$ with $x_i^2 < c\varphi\|x\|_2^2$ for a constant $c > 0$ belongs to L.

A basic ℓ_2 heavy hitters sketch works by first hashing elements of $[n]$ to $B \approx 1/\varphi$ buckets, i.e. effectively defining x^b for $b \in [B]$ to be the restriction of x to bucket b, and computing the sum of elements of x^b with random signs. In our model we can replace the random signs with random Gaussians, so that the resulting dot product is Gaussian with variance $\|x^b\|_2^2$. Fixing any $j \in [n]$ and letting b denote the bucket that j hashes to we get that a single hashing can be used to obtain an estimate of its absolute value that is correct up to constant factor and an additive $O(1/\sqrt{B})\|x\|_2$ term with probability[7] at least 9/10. We can now repeat the estimator $O(\log n)$ times and include in L elements that are estimated as larger than a $c'\varphi\|x\|_2$ for a sufficiently small constant $c' > 0$ in absolute value. Therefore, setting $B = O(1/\varphi)$ achieves the required bounds. This yields an ℓ_2-heavy hitters sketch with decoding time nearly linear in the size n of the universe. The decoding time can be improved to $(1/\varphi) \cdot \mathrm{poly}(\log n)$ using a bit-encoding approach similar to the one from Section A.1 above.

Spectral sparsifiers and spanners

Spectral sparsification sketches [30, 33, 31] require graph connectivity sketches, which we already implemented in Section A.1, as well as ℓ_2-heavy hitters sketches, and therefore can also be implemented in our model. Non-adaptive sketching algorithms for spanner construction [21] rely on spectral sparsification sketches that are applied to vertex-induced subgraphs of the input graph. Thus, these sketches can also be implemented in our model with at most a polylogarithmic loss in the number of rows.

[7] We use the fact that the dot product of x^b with a random Gaussian vector will be distributed as $g_j x_j + N(0, \|x^b_{-j}\|_2^2)$, and $|g_j|$ is at least a constant with probability at least 9/10. Here x_j^b stands for the vector obtained from x^b by zeroing out entry j.

Evaluating Stability in Massive Social Networks: Efficient Streaming Algorithms for Structural Balance

Vikrant Ashvinkumar
Department of Computer Science, Rutgers University, New Brunswick, NJ, USA

Sepehr Assadi
Department of Computer Science, Rutgers University, New Brunswick, NJ, USA
Cheriton School of Computer Science, University of Waterloo, Canada

Chengyuan Deng
Department of Computer Science, Rutgers University, New Brunswick, NJ, USA

Jie Gao
Department of Computer Science, Rutgers University, New Brunswick, NJ, USA

Chen Wang
Department of Computer Science, Rutgers University, New Brunswick, NJ, USA

Abstract

Structural balance theory studies stability in networks. Given a n-vertex complete graph $G = (V, E)$ whose edges are labeled positive or negative, the graph is considered *balanced* if every triangle either consists of three positive edges (three mutual "friends"), or one positive edge and two negative edges (two "friends" with a common "enemy"). From a computational perspective, structural balance turns out to be a special case of correlation clustering with the number of clusters at most two. The two main algorithmic problems of interest are: (i) detecting whether a given graph is balanced, or (ii) finding a partition that approximates the *frustration index*, i.e., the minimum number of edge flips that turn the graph balanced.

We study these problems in the streaming model where edges are given one by one and focus on *memory efficiency*. We provide randomized single-pass algorithms for: (i) determining whether an input graph is balanced with $O(\log n)$ memory, and (ii) finding a partition that induces a $(1+\varepsilon)$-approximation to the frustration index with $O(n \cdot \operatorname{polylog}(n))$ memory. We further provide several new lower bounds, complementing different aspects of our algorithms such as the need for randomization or approximation.

To obtain our main results, we develop a method using pseudorandom generators (PRGs) to sample edges between independently-chosen *vertices* in graph streaming. Furthermore, our algorithm that approximates the frustration index improves the running time of the state-of-the-art correlation clustering with two clusters (Giotis-Guruswami algorithm [SODA 2006]) from $n^{O(1/\varepsilon^2)}$ to $O(n^2 \log^3 n/\varepsilon^2 + n \log n \cdot (1/\varepsilon)^{O(1/\varepsilon^4)})$ time for $(1+\varepsilon)$-approximation. These results may be of independent interest.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Streaming algorithms, structural balance, pseudo-randomness generator

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.58

Category RANDOM

Related Version *Full Version*: `https://arxiv.org/abs/2306.00668`

Funding *Sepehr Assadi*: Assadi is supported in part by a Alfred P. Sloan Research Fellowship, and Assadi and Wang are supported in part by a NSF CAREER Grant CCF-2047061, a gift from Google Research, and a Rutgers Research Council Fulcrum Award.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 58; pp. 58:1–58:23
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Jie Gao: Ashvinkumar, Deng and Gao have been supported by NSF Grant CCF-2118953, CRCNS-2207440, CCF-2208663, and CNS-2137245.

Acknowledgements We thank Karthik C.S. for some preliminary discussions.

1 Introduction

Structural balance theory [21, 30, 35, 36] arises in the study of social relationships with positive and negative relations. Positive links describe friendship/agreement and negative links describe antagonism/disagreement. The theory dates back to work by Heider [35]. It describes the stability of relations among three individuals – only two kinds of triangles are stable: the ones where all three ties are positive, indicating mutual friendship; and the ones with two negative ties and one positive tie, describing the folklore that "the enemy of your enemy is your friend." A network that is far from being balanced (i.e., with many unstable triangles) accumulates stress which may lead to major re-arrangements of edges.

The structural balance theory appears in many application areas such as international relations [7, 43], biological networks [29, 38], portfolio analysis in financial networks [33], Ising model [14] in statistical physics, and online social media and opinion formation, dynamics, and evolution [6, 48–50] which bears itself on, for example, Facebook [47] and Twitter [40, 51].

One fundamental question is to understand whether a network is close to being balanced or not. In a *complete signed* graph where each pair of vertices is labeled positive or negative – the focus of our study – , the *Cartwright-Harary Theorem* [21, 32] states that every balanced network must have the nodes partitioned into (at most) two "camps", inside each of which the edges are all positive and between them the edges are negative. For graphs that are not balanced, a natural measure to characterize its distance from total balance is the *frustration index*, defined as the minimum number of edges whose negation of signs results in balance [1, 8, 34, 53]. These questions can be distilled into the following algorithmic problems (see Problem 1 and Problem 2 for the formal definitions):

- **Structural Balance Testing**: Given a complete signed graph G, decide whether or not it is balanced, namely, does not contain any imbalanced triangle.
- **Frustration-minimizing Partition**: Given a complete signed graph G, find a partition of vertices into two camps such that the minimum number of sign flips on the edges is required for the resulting graph to be balanced.

The problems we consider are closely related to (min-disagreement) *correlation clustering* where the goal is to partition the graph into clusters, so as to minimize the total number of negative edges inside clusters and positive edges across the clusters [5, 13, 24, 25, 28, 31], except that structural balance enforces the number of clusters to be two. Structural balance testing is straightforward to solve in $O(n^2)$ time on n-vertex graphs: place an arbitrary vertex and all its positive neighbors on one side of the bi-partition L, and its negative neighbors on the other side R, and then verify. On the other hand, the classical work by Giotis and Guruswami [31] on correlation clustering with two clusters implies the NP-hardness of the frustration-minimizing partition. It further provides a PTAS with running time $n^{O(1/\varepsilon^2)} + n \cdot (1/\varepsilon)^{O(1/\varepsilon^4)}$ for $(1+\varepsilon)$-approximation of this problem. To our knowledge, this is the state of the art for structural balance testing and frustration-minimizing partition.

Prior work primarily focused on *running time* of algorithms and assume *unrestricted access* to the entire graph. In many modern applications of large-scale networks, however, there are many other considerations to take into account. For instance, the algorithms may only have **limited access** to and a **memory** much smaller than the input. One of the most popular models capturing the above scenario is the **graph streaming model**. In this model, the edges arrive one after another and the algorithm needs to process this stream "on-the-fly" with limited memory. The algorithm is allowed to make one or multiple passes over the stream, and the memory is usually substianlly smaller than the (worst-case) input size, i.e. $\Theta(n^2)$. We focus on the single-pass setting and ask the following motivating question: *How well can we solve structural balance testing and frustration-minimizing partition problems in the single-pass graph streaming setting?*

The golden spots for streaming algorithms are (i) the polylog(n) memory regime, which is polynomial to the memory that is necessary to represent a single edge, and (ii) the $\widetilde{O}(n) := O(n \cdot \text{polylog}(n))$ memory regime – often referred to as the *semi-streaming* model. We study structural balance in the graph streaming model in these parameter regimes.

1.1 Our Contributions

Structural balance testing. One way of testing structural balance of G is to check whether the 2-lift of G associated with its edge signing is connected. Computing the 2-lift can be done on-the-fly, and testing connectivity in the streaming model can be done with $O(n \log n)$ space. This is already a quadratic improvement over the trivial algorithm with $O(n^2)$ space, and it is also known that graph connectivity requires $\Omega(n \log n)$ space in the streaming model. Is this also the limit for structural balance testing? Our first result is a simple algebraic algorithm, which stems from testing for complete bipartiteness of negative edges, that uses *exponentially improved* space complexity. In the full version, we also provide a companion algorithm that, while more complicated to describe, is combinatorial and introduces a technique that partially derandomizes a vertex sketch we use where hash functions with limited-independence alone do not suffice.

▶ **Result 1** (Informal statement of Theorem 2)**.** *There is a single-pass randomized algorithm that given a complete signed graph G in a graph stream, tests whether G is balanced with probability at least $\frac{99}{100}$ using $O(\log n)$ space.*

Result 1 shows that structural balance testing can be done with high space efficiency – note that $\Theta(\log n)$ space is necessary to simply write down a single edge. As we elaborate later, our algorithms in Result 1 are *sketching-based* algorithms that are randomized and crucially use the fact that each edge of the graph appears precisely once in the stream. We further complement our algorithms with two new lower bounds (Propositions 10 and 11) that show the necessity of both conditions for obtaining any $o(n)$-space algorithm for this problem. Additionally, a standard $\Omega(n \log n)$ space lower bound for input graphs that are not necessarily complete via a reduction from bipartiteness testing can be shown (see the full version); the generalized problem and the $o(n)$ space regime are thus incompatible.

Frustration-minimizing partition. Outputting the solution to frustration-minimizing partition already requires $\Omega(n)$ space, thus, as is standard, we focus on semi-streaming algorithms for this problem. One can use *cut sparsifiers* and a variant of an argument by [3] (for correlation clustering) to obtain an algorithm with $\widetilde{O}(n/\varepsilon^2)$ space that can return the "frustration value" of any bi-partition of vertices to within a $(1 \pm \varepsilon)$ factor. By enumerating over all bi-partitions, then, we can find the frustration-minimizing one up to a $(1 + \varepsilon)$-approximation. The problem with this approach is that the resulting algorithm takes exponential time (to enumerate all bi-partitions), which is quite prohibitive, especially in large graphs[1].

[1] This scenario is not uncommon in the streaming model. For instance, the Tournament Feedback Arc Set problem admits an "easy" exponential-time $(1 + \varepsilon)$-approximation semi-streaming algorithm [22] that was improved very recently by [16] to a PTAS albeit with $O(\log n)$ passes over the stream; see [15] for another example.

To bypass this challenge, we present a "streamified" version of the correlation clustering algorithm with two clusters of [31] (henceforth, the Giotis-Guruswami algorithm); namely, an approach that allows us to collect just enough information from the stream to *weakly* simulate (meaning not entirely faithfully) the Giotis-Guruswami algorithm in polynomial time at the end of the stream.

▶ **Result 2** (Informal statement of Theorem 7)**.** *There is a single-pass randomized algorithm that given a complete signed graph in a graph stream, with high probability*[a] *finds a partition of vertices with frustration value at most* $(1+\varepsilon)$ *factor of the frustration index using* $\widetilde{O}(n/\varepsilon^2)$ *space and polynomial (in* n *but not* ε*) time.*

[a] Here, and throughout, with high probability means with probability at least $1 - 1/n$.

Result 2 gives an efficient semi-streaming algorithm for $(1 + \varepsilon)$-approximation of the frustration-minimizing partition problem, which is NP-hard to solve exactly. We further show (Proposition 13) that even if we allow the streaming algorithm to use exponential (or more) time, solving this problem exactly requires $\Omega(n^2)$ space (the same as storing the entire input). This fully rules out any non-trivial streaming algorithms for solving this problem exactly.

There is still one missing piece in obtaining "truly efficient" semi-streaming algorithms for our problem. As stated earlier, the Giotis-Guruswami algorithm that Result 2 builds on has running time (roughly) $n^{O(1/\varepsilon^2)}$ even ignoring any streaming aspects[2], which makes this algorithm quite impractical. Our final contribution remedies this state of affairs. By building on our weak simulation of the Giotis-Guruswami algorithm in Result 2, we design an improved offline (non-streaming) algorithm for frustration-minimizing partition with nearly-linear running time for any fixed $\varepsilon > 0$ (note that the input is of size $\Theta(n^2)$ in this problem over a complete signed graph).

▶ **Result 3** (Informal statement of Theorem 9)**.** *There is a randomized (classical) algorithm that given a complete signed graph, with high probability, finds a partition of vertices with frustration value at most* $(1+\varepsilon)$ *factor of the frustration index in* $\widetilde{O}(n^2/\varepsilon^2 + n\cdot(1/\varepsilon)^{O(1/\varepsilon^4)})$ *time.*

Result 3 presents the first improvement after two decades over the Giotis-Guruswami algorithm for frustration-minimizing partition and correlation clustering with two fixed clusters *outside of graph streaming models*, which can be of independent interest. Moreover, the algorithm in Result 3 can also be used in Result 2, improving the running time to nearly-linear for any constant $\varepsilon > 0$ in the semi-streaming model.

Our algorithmic results combined with our new lower bounds (in Section 5) collectively complete the picture of streaming and efficient algorithms for structural balance testing and frustration-minimizing partition problems.

1.2 Our Techniques

Structural balance testing. The lower bounds in Propositions 10 and 11 show that any $o(n)$-space algorithm for testing structural balance has to be randomized and, more importantly, uses the fact that it sees the sign between every pair of vertices *exactly once*. This motivates

[2] A simple modification of the Giotis-Guruswami algorithm and a slightly more careful analysis actually reduces the running time of their algorithm to roughly $O(n^{100} + n \cdot (1/\varepsilon)^{O(1/\varepsilon^4)})$ (see the analysis in Section 4). While this is faster than the original $n^{O(1/\varepsilon^2)}$ bound, it is still quite far from being practical.

us to consider *sketching-based* algorithms that are based on collecting aggregate statistics of large subsets of edges in the graph when seeing them in the stream (without having to store explicit information). We proceed by proving the following structural result (stated informally): (i) In any balanced graph, for any set S of *odd* size, there is an *even* number of pairs of vertices in S with a negative sign (easy direction); (ii) Conversely, in any unbalanced graph, a constant fraction of odd-size sets S (of certain cardinality) have an *odd* number of negatively-signed pairs inside (hard direction).

This result naturally suggests the following algorithm (stated with some oversimplification): sample an odd-size set S of vertices uniformly at random at the beginning of the stream and count the number of negative edges in the stream with both endpoints in S. In the balanced case, this number will be even, while in the unbalanced case, it has a non-trivial chance of being odd.

There is a serious challenge in making this strategy work: determining S and therefrom which edges should be counted requires $\Omega(n)$ space to *store the random bits.* We surmount this by observing that the function for the parity of "$-$" edges in a sampled set S can be viewed as a *degree-2 polynomial* with one variable per vertex. As such, we turn from sampling vertices to using PRGs for low-degree polynomials based on sums of small bias generators so that $O(\log(n))$ truly random bits and $O(\log(n))$ extra space suffice to generate pseudorandom bits whose 0-set on the polynomial is in proportion with that of truly random bits. In the full version, we also give an alternative combinatorial algorithm (in contrast with the abovestated algebraic algorithm) which is interesting in its own right, especially from a techniques standpoint.

Frustration-minimizing partition. Our algorithm in Result 2 is obtained via "streamifying" the Giotis-Guruswami algorithm [31]. Their algorithm considers the high versus low frustration index cases separately where the high frustration index scenario means at least a constant fraction of all pairs needs to flip sign. Among these, the first part has a simple solution in [31] which already lends itself to a semi-streaming algorithm immediately. Our main technical ingredient in Result 2 lies in addressing the low frustration index case. For simplicity of exposition, in the following, we assume ε is a constant.

The Giotis-Guruswami algorithm in the low frustration index case is as follows. Sample $O(\log n)$ vertices S and enumerate all $n^{O(1)}$ bi-partitions of S. Then, for every bi-partition (S_L, S_R), perform a *merging* operation followed by a *switching* operation: In merging, every vertex $v \in V \setminus S$ is assigned in parallel to the side of (S_L, S_R) that creates less frustration for v, i.e., the number of negative edges of v to this side plus its positive edges to the opposing side is minimized. Let (L, R) be the resulting bi-partition of vertices. In switching, each vertex $v \in V$ in parallel is re-assigned to the side of (L, R) with the least frustration. The analysis of [31] is as follows: for the bi-partition (S_L, S_R) of S that is consistent with the optimal solution, (i) after merging, most vertices are in the "correct" side of (L, R) already, namely, they are consistent with the optimal bi-partition, and (ii) after switching, the vertices already consistent with the optimal solution do not change side, and the rest of vertices are moved to the side that only induce a small additive cost. The key reasoning behind both steps is that in the low frustration-index case, most vertices have a "clear" preference toward which side of the optimal bi-partition they should reside.

To simulate the Giotis-Guruswami algorithm via a semi-streaming algorithm, we can sample the set S at the beginning of the stream and store all $\widetilde{O}(n)$ edges incident on it during the stream. The merging phase can now be easily implemented as it relies only on the edges connected to S. The switching phase, on the other hand, requires checking *every* edge in the graph and thus cannot be faithfully implemented in $o(n^2)$ space. We instead develop a way to *approximately* implement this step, by sampling $O(\log n)$ edges per vertex and using them to estimate the frustration of each vertex in the switching phase. This allows us to still classify the majority of vertices the same as that of the Giotis-Guruswami algorithm except for the ones with no clear preference between bi-partitions of the optimal solution. An extra argument here ensures that this step does not increase the cost too much and this new solution is still a $(1+\varepsilon)$-approximation, leading to our semi-streaming algorithm.

The ideas above also allow us to significantly improve the *running time* of the Giotis-Guruswami algorithm even outside of graph streaming models. The most time-consuming step of that algorithm is the need to enumerate all bi-partitions of the sampled set S of size $O(\log n)$. We preface the sampling step of the set S by sampling a smaller set T of size only $O(\log \log n)$, enumerate only over $(\log n)^{O(1)}$ bi-partitions of T, and use those to find an *approximate* optimal bi-partition of the set S. We then follow a similar strategy as above to simulate the Giotis-Guruswami algorithm, by showing that even though the bi-partition of the set S is no longer truly optimal, the extra error occurred by the approximation, does *not* propagate too much in the merging and switching step – in other words, these operations are "robust" enough to handle even an approximately optimal bi-partition of S, not a truly optimal one.

Lower Bounds. Most of our lower bounds are based on reductions from known problems in communication complexity such as Index or Equality. The exception is our lower bound in Proposition 11 which shows that, perhaps counter-intuitively, when a stream contains copies of an edge more than once, solving structural balance testing becomes impossible in $o(n)$ space. We show this lower bound by presenting a new *3-party* communication problem which is a mixture of the Index and Set Intersection problems. Roughly speaking, in this new problem, we also provide the information about the hidden index of the Index problem to *all* players, but in a way that to find this index, they will need to solve an instance of the Set Intersection problem. We then borrow an idea from [9, 10] that allows us to argue that a low-communication protocol cannot even change the distribution of the hidden index by much. We then combine this with standard information-theoretic arguments to show that the Index problem the players need to solve remains hard as the distribution of its hidden index has not been altered too much.

1.3 Related Work

By slightly relaxing structural balance to allow triangles with three negative ties as in [30], the frustration-minimizing partitions becomes fully equivalent to the correlation clustering problem (without any constraint on the number of clusters). More recently, motivated by similar considerations as in our paper, there has been a flurry of results on this problem in modern models of computation such as sublinear-time, streaming, or massively parallel algorithms [3, 11, 17, 18, 26, 27]. In particular, for the single-pass streaming setting, the very recent state-of-the-art result by [23] obtains a $(3+\varepsilon)$-approximation correlation clustering in $O(n/\varepsilon)$ space and polynomial time. Our result shows that we can obtain a better approximation when the number of clusters is two. Furthermore, [18] independently observes a $(1+\varepsilon)$-approximation algorithm for correlation clustering in exponential time using cut sparsifiers, which is similar to our observation in Lemma 8.

In a general graph (not necessarily complete), structural balance means that all cycles must contain an even number of negative edges. The problem of minimizing frustration index, phrased in the literature as the Balanced Subgraph problem [37], is MaxSNP-hard [45, 52] and even NP-hard to approximate within any constant factor assuming Khot's Unique Games Conjecture [39]. On the positive side, this problem admits a polynomial time $O(\sqrt{\log n})$-approximation [2] and an $O(\log k)$-approximation [12] when the frustration index is k. Readers can refer to [37] for further details about applications in practice. None of these algorithms consider space constraints or the streaming setting.

2 Preliminaries

Notation. Throughout, we use $G = (V, E = E^+ \cup E^-)$ to denote a complete signed graph with $|V| = n$ vertices, where E^+ is the set of "+" edges and E^- is the set of "−" edges. We further use $G^+ = (V, E^+)$ and $G^- = (V, E^-)$ to denote the graph restricted to the "+" and "−" edges. Note that E^+ and E^- are always disjoint and $|E^+ \cup E^-| = \binom{n}{2}$. For any set $S \subseteq V$ of vertices, $G[S]$ denotes the induced subgraph of G on S.

For two sets of vertices S, $T \subset V$, we use $E(S,T)$ to denote the set of edges (both "+" and "−") with one endpoint in S and the other in T. Analogously, we use $E(v,S)$ to denote the set of edges (both "+" and "−") with one endpoint v and the other in S. Furthermore, $E^+(S,T)$ (resp. $E^-(S,T)$) refers to the set of "+" (resp. "−") edges with one endpoint in S and the other in T, and $E^+(v,S)$ (resp. $E^-(v,S)$) refers to the set of "+" (resp. "−") edges with one endpoint v and the other in S. In particular, $N^+(S) = E^+(S, V \setminus S)$ and similarly, $N^+(v) = E^+(v,V)$ ($N^-(S)$ and $N^-(v)$ defined in the same manner).

When the context involves more than one graph (say G and H), we add a subscript to make the notation clear as to which graph is being referred to (for example, $E_G(S,T)$ and $E_H(S,T)$).

2.1 Problem Definition

In this paper, we consider (strong) structural balance. A complete signed graph $G = (V, E^+ \cup E^-)$ is said to exhibit structural balance property (in short, is balanced) if every triangle has an even number of "−" edges. An equivalent characterization of structural balance shown in [21] is that there exists a bi-partition of the vertices into S and T, $S \cap T = \emptyset$ and $S \cup T = V$, such that every edge with both endpoints in S (or T) is labelled "+", and every edge connecting S to T is labelled "−". Inspired by this, the notion of frustration index [1, 34, 53] captures how far a graph is from structural balance.

▶ **Definition 1** (Frustration Index). *Let $G = (V, E^+ \cup E^-)$ be a complete signed graph, and let (L, R) be a bi-partition of V. Then the* frustration *of G with respect to L, R is*

$$\mathsf{frust}(G,\ L, R) = \left|E^+(L,R)\right| + \left|E^-(L)\right| + \left|E^-(R)\right|.$$

When the context is clear, we use the notation $\mathsf{frust}(G,\ L)$ instead of $\mathsf{frust}(G,\ L, R)$. The frustration index *$\mathsf{frust}(G)$ is the minimum of $\mathsf{frust}(G,\ L, R)$ over all possible bi-partitions (L, R) of V.*

Frustration can be equivalently defined using "disagreement" notation:

$$\mathsf{frust}(G,\ L, R) = \frac{1}{2} \sum_{v \in V} \mathsf{Dis}(v, L, R).$$

We use $\mathsf{Dis}(v, S, T)$ to count the disagreement of edges incident to v with respect to S, T. For example, when $v \in S$, we have $\mathsf{Dis}(v, S, T) = |E^-(v, S)| + |E^+(v, T)|$. When $T = V \setminus S$, we may write $\mathsf{Dis}(v, S)$ instead of $\mathsf{Dis}(v, S, V \setminus S)$.

Now we are ready to formally define our problems.

▶ **Problem 1** (Structural Balance Testing). *Given a complete signed graph $G = (V, E^+ \cup E^-)$, decide if there is a partition (L, R) of V such that*

- *If $(u, v) \in E^+$, then u and v are both contained in L or both contained in R;*
- *If $(u, v) \in E^-$, then either u is in L and v is in R, or vice versa.*

In other words, the answer is "YES" if and only if the graph is balanced.

▶ **Problem 2** (Frustration-minimizing Partition). *Given a complete signed graph $G = (V, E^+ \cup E^-)$, find a partition (L^*, R^*) of V such that*

$$(L^*, R^*) = \arg\min_{\substack{(L,R): \\ L \cup R = V \\ L \cap R = \{\}}} \mathsf{frust}(G,\, L, R).$$

In other words, the partition (L^*, R^*) minimizes the frustration.

3 Structural Balance Testing in Logarithmic Space

We formally present our main algorithm for structural balance testing (Problem 1) – a randomized algorithm that uses only $O(\log n)$ bits and returns whether the graph is balanced with high (constant) probability.

▶ **Theorem 2** (Formalization of Result 1). *There is a randomized single-pass algorithm* BalanceTester *that solves Problem 1 with $O(\log n)$ bits of space and the following guarantees:*

- *If input graph G is balanced,* BalanceTester *always outputs* Balanced*;*
- *If input graph G is not balanced,* BalanceTester *outputs* NotBalanced *with probability at least $\frac{99}{100}$.*

A few remarks on Theorem 2 are in order. First, it is not hard to design a deterministic algorithm that uses $\tilde{O}(n)$ space to test whether the graph is balanced – for completeness, such an algorithm can be found in the full version of this paper. However, by our lower bound in Proposition 10, any single-pass streaming algorithm for structural balance testing with $o(n)$ memory has to be randomized. Furthermore, we remark that the $O(\log n)$ memory is asymptotically optimal since it is the number of bits that is necessary to store even a single edge. Finally, since the error is one-sided, we can boost the success probability to $1 - \frac{1}{n}$ by running the algorithm $O(\log n)$ times, which results in $O(\log^2 n)$ overall space complexity.

We now proceed to the design and analysis of our algorithm. In what follows, we assume our graph contains at least $n \geq 3$ vertices, as the structural balance is always satisfied otherwise.

3.1 A Sample-and-Test Lemma for "$-$" Edges

The idea behind our algorithm starts as follows. If we focus on "$-$" edges, our task can be framed as testing if the graph $G^- = (V, E^-)$ is a *complete bipartite graph*. To this end, observe that if we sample a set S of an odd number of vertices, and count the number of "$-$" edges in $G[S]$, the parity will always be even when G^- is complete bipartite. With a slightly more involved analysis, we can show that if G^- is not complete bipartite, then by sampling S uniformly at random, the number of "$-$" edges in $G[S]$ is odd with constant probability. As such, we can use the parity of such a counter as a signal of the structural balance of the graph.

We now formalize the above intuition. In particular, in the following lemma, we design the S-sampler and state its properties.

▶ **Lemma 3** (S-sampler Lemma). *For a given complete signed graph $G = (V, E)$, let $S \subseteq V$ be a subset of vertices sampled with the following $\underline{S\text{-sampler}}$:*

1. *Pick a fixed vertex v_n arbitrarily.*
2. *Sample each vertex in $v \in V \setminus \{v_n\}$ independently with probability $\frac{1}{2}$ to get S'.*
3. *If $|S'|$ is odd, let $S = S'$; otherwise, let $S = S' \cup \{v_n\}$.*

Then, the following statements are true:

1. *If G is balanced, the induced subgraph $G[S]$ always has an even number of "$-$" edges;*
2. *If G is not balanced, the induced subgraph $G[S]$ has an odd number of "$-$" edges with probability at least $\frac{1}{4}$, over the randomness of sampling S.*

The main idea to prove Lemma 3 is by observing that the parity of "$-$" edges counted by our S-sampler is in fact captured by a *degree-2* polynomial over $\mathbb{F}_2$. The observation can be formalized as the following lemma.

▶ **Lemma 4.** *Let $X_i \in \{0, 1\}$ be the indicator random variable for whether vertex v_i is sampled by S-sampler. Define the following polynomial with $(n-1)$ variables over $\mathbb{F}_2$:*

$$P^*(X) = \sum_{\substack{i,j<n \\ (v_i,v_j)\in E^-}} X_i X_j + \sum_{\substack{i<n \\ (v_i,v_n)\in E^-}} X_i \left(1 + \sum_{j=1}^{n-1} X_j\right). \tag{1}$$

Then, the polynomial $P^*(X) = |E^-(G[S])|$ over $\mathbb{F}_2$, which is the parity of number of negative edges induced by the sample set S.

Proof. Note that Eq (1) is essentially obtained by substituting $X_n = 1 + \sum_{j=1}^{n-1} X_j$, and counting a "$-$" edge (v_i, v_j) if both v_i and v_j are sampled. If the number of sampled vertices other than v_n is odd, we have $\sum_{j=1}^{n-1} X_j = 1$, which implies $X_n = 0$ over $\mathbb{F}_2$; otherwise, if the number of sample vertices other than v_n is even, we have $\sum_{j=1}^{n-1} X_j = 0$, which implies $X_n = 1$ over $\mathbb{F}_2$. Therefore, the polynomial P^* exactly captures the sampling process of the S-sampler. Finally, if the total number of "$-$" edges is odd, the polynomial P^* evaluates to 1, and vice versa. ◀

As we will see shortly, Lemma 4 is also crucial to run our *streaming* algorithm for the S-sampler by storing limited number of random bits. To show that the S-sampler gives a useful signal for whether the graph is balanced, we show that the polynomial P^* in Eq (1) gives a useful signal. We first show that P^* is identically 0 if and only if G is balanced.

▶ **Lemma 5.** *The polynomial P^* in Eq (1) is identically 0 if and only if G is balanced.*

Proof. Suppose first that G is balanced. Then either G^- is empty, or it is a complete bipartite graph. Let S be the sample from S-sampler. In the first case, the number of "$-$" edges in $G[S]$ is always 0. In the second case, let L and R be the bi-partition of G^-, and let $A = L \cap S$ and $B = R \cap S$. The number of "$-$" edges in $G[S]$ is $|A| \cdot |B|$, which is an even number since one of $|A|$ and $|B|$ is even (recall that $|S|$ is odd). By Lemma 4, P^* is thus identically 0 when G is balanced.

Suppose, on the other hand, that G is not balanced. Then there is a triangle (v_i, v_j, v_k) with an odd number of "$-$" edges. If $i, j, k \neq n$, setting $X_i = X_j = X_k = 1$ and $X_\ell = 0$ for $\ell \neq i, j, k$ yields $P^*(X) = 1$. If on the other hand, say, $k = n$, then setting $X_i = X_j = 1$ and $X_\ell = 0$ for $\ell \neq i, j$ yields $P^*(X) = 1$. Hence P^* is not identically 0 when G is not balanced. ◀

Next, we give a lemma on the *fraction* of assignments that are non-zero on the polynomial P^*, which gives us a lower bound for the success probability to capture an odd number of "$-$" edges when G is imbalanced. We note that the lemma is standard in the coding theory community (e.g. [46]), but we state a proof here for completeness.

▶ **Lemma 6.** *Let $P(x_1, x_2, \ldots, x_n)$ be a polynomial over $\mathbb{F}_2$ with degree at most 2. If P is not identically 0, then there is a set X, with cardinality at least 2^{n-2}, such that $x \in X$ implies $P(x) = 1$.*

Proof. We may assume that P is multilinear since $\bar{P}(x) = P(x)$ for all $x \in \{0,1\}^n$, where $\bar{P}$ is the linearization of P (i.e. all monomials of the form x_i^2 are replaced with x_i).

If P is of degree 1 or less, the claim holds by using the Schwartz-Zippel lemma.Suppose then that P is of degree 2. Up to a reordering of variables, $x_{n-1}x_n$ is a term of P with coefficient 1. Let a be a fixed assignment of x_k to $\{0,1\}$ for all $k \in [n-2]$. Then $P(a, x_{n-1}, x_n)$ is a multilinear degree 2 polynomial $Q(x_{n-1}, x_n)$ with the term $x_{n-1}x_n$ having coefficient 1. Checking over all 8 possible polynomials that Q may take, we see that each of them has at least one assignment a' of x_{n-1}, x_n to $\{0,1\}$ such that $Q(a') = 1$. This completes the proof since there are 2^{n-2} possible assignments a. ◀

Finalizing the proof of Lemma 3. By Lemma 4 and Lemma 5, if G is balanced, the polynomial P^* is identically 0, which means the S-sampler always finds an even number of "$-$" edges. Otherwise, if G is not balanced, P^* is not 0. Therefore, by Lemma 6, the S-sampler finds an odd number of "$-$" edges with probability at least $\frac{1}{4}$. ◀

By Lemma 3, to test the structural balance of a complete signed graph, it suffices to implement the S-sampler and count the number of negative edges. However, while the counter takes $O(\log n)$ space, it is not immediately clear how the S-sampler can be implemented with a small space in the streaming setting. In particular, since we need to sample each vertex *independently*, the trivial solution requires $\Omega(n)$ memory to store the random bits (for each vertex). We address this issue in the next step.

3.2 Simulating the S-sampler in Streaming with $O(\log n)$ Space

We tackle this issue by using the fact that our S-sampler is a degree-2 polynomial (Lemma 4). We may consequently use a PRG that fools degree-2 polynomials (see [20, 41, 44] for constructions). More concretely, there exists a PRG that takes $O(\log n)$ truly random bits, and generates n pseudorandom bits with $O(\log n)$ extra space per bit so that a degree-2 polynomial cannot distinguish a truly random input from a pseudorandom input to within a small constant error.

Using this, we design an $O(\log n)$-memory algorithm as follows.

A streaming algorithm to test structural balance.

- Run 100 independent copies of the following S-sampler simulation:
  1. Maintain $O(\log n)$ truly random bits and a counter.
  2. For each arriving "$-$" edge $(u, v) \in E^-$:
     a. Generate the pseudorandom bits for vertices u and v (name the bits X_u and X_v) with a PRG for degree 2 polynomials with $\varepsilon = \frac{1}{20}$ (see [20, 41, 44]).
     b. If both $X_u = 1$ and $X_v = 1$, increase the counter by 1.
  3. By the end of the stream, if the parity of the counter is odd, return 1; otherwise, return 0.
- If any of the copies of S-sampler simulation returns 1, report Not Balanced; otherwise, report Balanced.

Analysis of the space complexity. The truly random bits and the counter take $O(\log n)$ bits of space. Furthermore, for each "$-$" edge, the PRG generates the pseudorandom bits for X_u and X_v with $O(\log n)$ bits of extra space since $\varepsilon = \frac{1}{20}$. The space for this purpose can be re-used across different edges. Each copy of the S-sampler simulation therefore takes $O(\log n)$ space to implement. Finally, since we run 100 independent copies in parallel, the final algorithm take $O(\log n)$ space.

Analysis of correctness. By Lemma 3, if the graph is balanced, the S-sampler simulation always returns 0. On the other hand, if the graph is imbalanced and we sample with truly random bits U_{n-1}, it holds from Lemma 3 that $\Pr(P^*(U_{n-1}) = 1) \geq \frac{1}{4}$ since P^* is a valid implementation of the S-sampler. By the guarantees of the PRG $g : \{0,1\}^{O(\log n) \to (n-1)}$, we have

$$|\Pr(P^*(g(U_\ell)) = 1) - \Pr(P^*(U_{n-1}) = 1)| \leq \frac{1}{20},$$

where U_ℓ are $O(\log n)$ truly random bits, the seed for the PRG. We hence have $\Pr(P^*(g(U_l)) = 1) \geq \frac{1}{4} - \frac{1}{20} = \frac{1}{5}$. As such, the probability for no copy to return 1 is at most $(\frac{4}{5})^{100} < \frac{1}{100}$, as desired.

4 Semi-streaming Frustration-minimizing Partition in Poly-Time

In this section, we consider Problem 2 of finding, in the semi-streaming model, a partition of a complete signed graph that is as close to structurally balanced as possible. More specifically, we want to divide the set of vertices into two parts and minimize the number of "$-$" edges contained in either part and "$+$" edges connecting the two parts.

Giotis and Guruswami [31] showed that Problem 2 is NP-hard. In the streaming setting, we show that any algorithm that gives the exact optimal frustration-minimizing partition (or computes the optimal frustration index) requires $\Omega(n^2)$ memory (see Proposition 13). In view of this, we settle for finding a partition that approximates one which is as close to structurally balanced as possible. This section shows how to adapt the correlation clustering algorithm with two clusters in [31] to the semi-streaming model. We leave the technical details to the full version of the paper and, here, highlight the ingredients that make up the following result.

▶ **Theorem 7** (Formalization of Result 2). *There is an algorithm that, given a complete signed graph $G = (V, E^+ \cup E^-)$ and $\varepsilon \in (0,1)$, with high probability finds a partition (L, R) of V where*

$$\mathsf{frust}(G,\, L) \leq (1+\varepsilon) \cdot \mathsf{frust}(G).$$

Moreover, the algorithm uses $O\Big(n \cdot \frac{\log^4(n/\varepsilon)}{\varepsilon^6}\Big)$ space, a single pass over the stream and has running time of $O\Big((\frac{n}{\varepsilon})^2 \cdot \log^3 n + n \log n \cdot (\frac{1}{\varepsilon})^{O(\varepsilon^{-4})}\Big)$.

4.1 Evaluating Frustration

The first key ingredient is a tool for determining $\mathsf{frust}(G,\, L)$ for an arbitrary $L \subseteq V$, subject to the memory constraints of the semi-streaming setting. We turn to the idea of cut sparsifiers (see [4, 19, 42]), which store a sparse representation of G which approximately answers the cut queries for all $S \subseteq V$: what is the size of $E(S, V \setminus S)$?

▶ **Lemma 8.** *Let $G = (V, E^+ \cup E^-)$ be a complete signed graph. There is a single-pass streaming algorithm using $O\big(n \log^3 n/\varepsilon^2\big)$ space and $O\big(n^2 \log^2 n\big)$ time that, for any $\varepsilon \in (0,1)$, with high probability produces an oracle that, when given $L \subseteq V$, returns $\mathsf{frust}_\varepsilon(G,\, L)$ in $O\big(n \log n/\varepsilon^2\big)$ time where*

$$(1-\varepsilon)\mathsf{frust}(G,\, L) \leq \mathsf{frust}_\varepsilon(G,\, L) \leq (1+\varepsilon)\mathsf{frust}(G,\, L).$$

Our construction of the "frustration sparsifier" in Lemma 8 follows from a standard application of cut sparsifiers, and the fact that the portion of frustration contributed to by the "−" edges in each side can be estimated indirectly by looking at "+" edges crossing the cut.

Frustration Sparsifier for $G = (V, E^+ \cup E^-)$ and $\varepsilon \in (0,1)$:

- Let $H = (V, E_H, w_H)$ be an $\frac{\varepsilon}{2}$ cut sparsifier for $G' = (V, E^+)$.
- Return $\mathsf{frust}_\varepsilon(G,\, \cdot)$ which, when given $L \subseteq V$, computes

$$\underbrace{w_H(L, V \setminus L)}_{\approx |E^+(L, V \setminus L)|} + \underbrace{|E^-| - (|L| \cdot |V \setminus L| - w_H(L, V \setminus L))}_{\approx |E^-(L)| + |E^-(V \setminus L)|}.$$

The proof of Lemma 8 can be found in the full version of this paper.

Observe as an aside that frustration sparsifiers alone give us a single-pass algorithm that uses $\widetilde{O}(n)$ space which, however, runs in exponential time: construct a frustration sparsifier with parameter ε, enumerate over all partitions of V, and return the one which gives the smallest estimated frustration.

4.2 A Polynomial Time Algorithm: Streamification of Giotis-Guruswami Algorithm

The next key ingredient is a simulation of the Giotis-Guruswami algorithm [31]. For the remainder of this section, let $\mathsf{frust}(G) = \gamma n^2$ where γ is not necessarily a constant. We are particularly interested in the case when $\gamma \leq \varepsilon/100^4$, which is the case of the Giotis-Guruswami algorithm which is non-trivial to simulate.

When there is no constraint on space usage, the Giotis-Guruswami algorithm works by sampling $\Theta(\log n)$ vertices (called the sample S) and trying every partition (S_L, S_R) of S. S_L and S_R are taken to be subsets of L and R respectively, where (L, R) is a partition of V. For each such partition (S_L, S_R), there should be a clear local choice for whether a vertex v should be assigned to L or to R by choosing the smaller of $\mathsf{Dis}(v, S_L + v, S_R)$[3] and $\mathsf{Dis}(v, S_L, S_R + v)$ respectively; we do this for all unsampled vertices, and refer to this step of the algorithm as MERGING. The algorithm then tries to further refine the solution quality by marking each vertex that improves the frustration when moved from L to R or vice versa, keeping every other vertex fixed. All marked vertices are then moved to the other part. The process of marking vertices and moving them after all have been marked is referred to as the SWITCHING step.

As it turns out, the correctness of the Giotis-Guruswami algorithm still holds with a small error introduced by graph sparsification. This is crucial as it allows us to design space efficient streaming algorithms by simply maintaining a cut sparsifier during the stream, and performing the Giotis-Guruswami procedure in the end. More concretely, only once the sparse representations of the input stream have been obtained (i.e. a single pass over the stream has been completed), do we then start iterating over every partition (S_L, S_R) of S, and in each iteration MERGING, SWITCHING, and storing the best resulting (L, R) seen so far. MERGING is easily ported to the streaming setting; we just need to store the $\Theta(n \log n)$ edges incident to S. The way SWITCHING is adapted, on the other hand, is less trivial; we store a sparse representation of the input graph by sampling $\Theta\big((1/\varepsilon^2)\log n\big)$ neighbors for each vertex. Finally, determining the (approximate) best partition seen through all iterations can be done by using a frustration sparsifier.

```
**Streamification of [31]** for $G = (V, E^+ \cup E^-)$ and $\varepsilon \in (0,1)$, when $\gamma \lessapprox \varepsilon$:
1. Sample $100 \log n$ vertices uniformly at random. Call this set $S$.
2. Sample $100^3 \cdot \frac{\log n}{\varepsilon^2}$ vertices uniformly at random for each vertex $v$. Call these sets $N_v$.
3. Read through the stream of (signed) edges and
   - Store edges incident to $S$. Call this graph $G_S$.
   - Store edges joining $v$ and $N_v$ for all $v$. Call this graph $G_N$.
   - Store $\mathsf{frust}_{\varepsilon/10}(G, \cdot)$, a frustration sparsifier of $G$.
4. For every partition $(S_L, S_R)$ of $S$
   (Merging)
      For every $v \in V \setminus S$, assign $v$ to $L$ if $\mathsf{Dis}_{G_S}(v, S_L + v, S_R) < \mathsf{Dis}_{G_S}(v, S_L, S_R + v)$, and $R$ otherwise.
      $S_L$ is assigned to $L$ and $S_R$ is assigned to $R$.
   (Switching)
      Mark $v \in L$ if $\mathsf{Dis}_{G_N}(v, L + v, R - v) > \mathsf{Dis}_{G_N}(v, L - v, R + v)$ and mark $v \in R$ similarly (flip the inequality).
      Switch assignments of all marked vertices from $L$ to $R$ or vice versa.
    Keep $(L, R)$ if $\mathsf{frust}_{\varepsilon/10}(G, L)$ is the smallest seen so far.
5. Return $(L, R)$.
```

[3] In this section, we use the following notation: For any set S and any vertex v, $S + v$ denotes $S \cup \{v\}$ and $S - v$ denotes $S \setminus \{v\}$.

The roles of MERGING and SWITCHING are roughly explained as follows. When (S_L, S_R) is congruent with how S would be partitioned in an optimal solution, MERGING assigns most vertices to L and R in the same way they would have been assigned in said optimal solution. These vertices do not change assignment when SWITCHING. The remaining vertices may have lots of disagreement with their current assignment in spite of having lots of agreement with S_L or S_R. SWITCHING rectifies this, minimizing the disagreement of these vertices with a large number of vertices (the ones that stay put, mentioned earlier).

We leave the proofs to the full version of the paper. Suffice to say, the core technical work here is in the analysis showing that weakly simulating SWITCHING in the way we have still yields a correct algorithm.

4.3 Looking at Exponentially Fewer Candidate Partitions

Observe that the first two ingredients yield an algorithm with $\Theta(n^{101})$ running time, where the inefficiency stems from having to iterate over all partitions of a sample of size $100 \log n$. The last key ingredient completes Theorem 7 by instead iterating over all partitions of a sample of size $100 \log \log n$, building from this a partition of size $O(\log n)$ via an independent sample (we call this step MINI-MERGING), and thereafter running MERGING and SWITCHING.

This gives us Theorem 7, and also Theorem 9 by running the algorithm in an offline setting regardless of it being a streaming algorithm.

▶ **Theorem 9** (Formalization of Result 3). *There is a randomized algorithm that, given a complete signed graph $G = (V, E^+ \cup E^-)$, and for any $\varepsilon \in (0, 1)$, with high probability returns a partition (L, R) of V where*

$$\mathsf{frust}(G, L) \leq (1 + \varepsilon) \cdot \mathsf{frust}(G).$$

Moreover, the algorithm has runtime $O\left(\frac{n^2 \log^3 n}{\varepsilon^2} + n \log n \cdot (1/\varepsilon)^{O\left(\varepsilon^{-4}\right)}\right)$.

To the best of our knowledge, this is the first algorithm for Problem 2 and 2-correlation clustering that is nearly-linear in the input. A little more formalism is supplied in Appendix A, but much of the details are left to the full version of the paper.

5 Space Lower Bounds

In this section, we state space lower bounds that complement our algorithmic results.

- Any p-pass deterministic streaming algorithm solving Problem 1 requires $\Omega\left(\frac{n}{p}\right)$ space.
- Any single-pass randomized algorithm solving Problem 1 on a complete signed multi-graph, where edges are allowed to repeat twice, requires $\Omega(n)$ space.
- Any single-pass randomized streaming algorithm solving Problem 2 *exactly*, or that outputs the *exact* frustration index, requires $\Omega\left(n^2\right)$ space.

Combined with our upper bounds in Theorem 2 and Theorem 7, our results convey the following message: (i). It is possible to test structural balance with a very high space efficiency by taking advantage of the arrive-once inputs and randomness, and both aspects are *necessary*; and (ii). It is possible to achieve a $(1 + \varepsilon)$-approximation in $\tilde{O}(n)$ space and polynomial time for any constant ε. In contrast, the exact solution requires almost all the input to be stored.

5.1 Structural Balance Testing by Deterministic Algorithms

We first present space lower bound for deterministic streaming algorithms. The formal statement of the lower bound is as follows.

▶ **Proposition 10.** *Any deterministic p-pass algorithm for Problem 1 requires $\Omega\left(\frac{n}{p}\right)$ bits of space. In particular, any deterministic single-pass algorithm requires $\Omega(n)$ bits of space.*

The proof of our deterministic lower bound uses a reduction from EQUALITY, which is known to require $\Omega\left(\frac{n}{p}\right)$ bits of communication over p rounds. More can be found in the full version.

5.2 Structural Balance Testing by Randomized Algorithms with Multi-edges

The second lower bound is the most interesting from a technical standpoint. It shows that we need to use $\Omega(n)$ space if the inputs are complete signed graphs with multi-edges. We assume that the signs of the multi-edges between the same pair of vertices are *consistent*: if e_1 between (u, v) is "+", a second edge e_2 between (u, v) also has to be "+". The definition of structural balance still holds by checking one of the edges between each vertex pair.

The formal statement of our lower bound is as follows.

▶ **Proposition 11.** *Any single-pass streaming algorithm that correctly tests structural balance with probability at least $\frac{99}{100}$ on complete signed multi-graphs with sign-consistent multi-edges has to use a memory of $\Omega(n)$ bits.*

On the high level, the plan to prove Proposition 11 is a reduction from INDEX. The standard INDEX problem is described as follows: Alice holds a string $x \in \{0, 1\}^N$, Bob holds an index $i^* \in [N]$, and the two players want to learn the value of x_{i^*}.

In a first attempt at a reduction, the two players are initially given $2N + 2$ vertices, with each bit x_j corresponding to two vertices u_j and v_j and two special vertices s and t. Alice creates two empty sets A and B, and for each input bit x_j, $j \in [N]$, Alice puts u_j to A and v_j to B if $x_j = 0$, and v_j to A and u_j to B if $x_j = 1$. Then, for every vertex except u_{i^*} and v_{i^*}[4], Alice adds "+" edges from s to the vertices in A and from t to vertices in B. Furthermore, for each pair of vertices *both inside* A or B, Alice adds a "+" edge; and for vertex pairs such that $u \in A$ and $v \in B$, Alice adds a "−" edge. On the other end, Bob completes the graph by adding "+" edges to (s, u_{i^*}) and (t, v_{i^*}) and "−" edges to (s, v_{i^*}) and (t, u_{i^*}). Now, if $x_{i^*} = 0$, the resulting graph is balanced; otherwise, the graph is unbalanced since v_{i^*} is in A.

While the plan sounds nice, it does *not* work. Careful readers may have already found the problem: Alice does *not* know which vertices correspond to u_{i^*} and v_{i^*}; and if she somehow infers it, the problem becomes easy as she can output x_{i^*} without any communication. As such, for the above idea to work, we need to somehow "hide" the index to the holder of the string, but give them the ability to create graphs that leave everything but x_{i^*}.

To the above end, we introduce the following version of Leave-One INDEX.

▶ **Problem 3** (Leave-One INDEX). *Consider a communication problem between 3 players, namely Alice, Bob, and Charlie. Both Alice and Bob hold the same string $x \in \{0, 1\}^N$, and Charlie holds an index $i^* \in [N]$. Furthermore, Alice holds $S \subset [N]$ and Bob holds $T \subset [N]$, such that $S \cup T = [N] \setminus \{i^*\}$. The communication goes in the order of Alice, Bob and Charlie, and the players want to output x_{i^*}.*

[4] The construction is in fact not possible as we will see in the next paragraph; it is here to explain the intuition.

The sets of S and T create some "flexibility" for Alice and Bob to infer the index. Indeed, if $S \cap T = \emptyset$, the problem becomes easy as Bob can learn Alice's bits from a message that simply encodes which bits are missing[5]. Nevertheless, we prove (see Appendix B) that the worst-case communication complexity of the Leave-One INDEX problem is still $\Omega(N)$. The formal lower bound is as follows.

▶ **Lemma 12.** *Any (possibly randomized) one-way communication protocol for the Leave-One INDEX defined in Problem 3 that outputs x_{i^*} correctly with probability at least $\frac{99}{100}$ requires $\Omega(N)$ bits of communication.*

Figure 1 shows how Alice, Bob, and Charlie can use the previously mentioned (wrong) idea to get a (now) correct reduction of Leave-One Index to testing structural balance, where the multi-edges occur from the intersection $S \cap T$ of Alice's and Bob's sets.

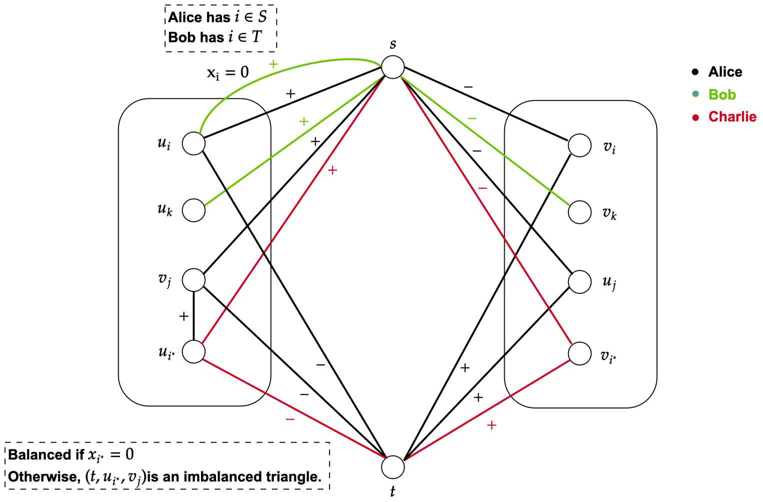


Figure 1 An illustration of the reduction from Leave-One Index.

5.3 Frustration-minimizing Partition by Exact Algorithms

Finally, we show that any single-pass streaming algorithm that solves the frustration index *exactly* has to store almost the entire graph.

▶ **Proposition 13.** *Any single-pass streaming algorithm that solves Problem 2 exactly or returns the optimal frustration index value requires $\Omega(n^2)$ bits of space.*

Proposition 13 goes through communication complexity again. This time, our reduction is from INDEX (a matrix variant thereof) which is hard under the one-way communication regime. Details are to be found in the full version.

[5] In particular, there is a simple protocol with 1 bit of communication when $S \cap T = \emptyset$: Alice takes XOR of all bits in $x[S]$, and send it to Bob. Bob takes XOR of all bits in $x[T]$ he will learn the XOR of $x[S \cup T]$. Bob then compares the XOR of x vs. the XOR of all bits in $x[S \cup T]$ to learn x_{i^*}

References

1 Robert P. Abelson and Milton J. Rosenberg. Symbolic psycho-logic: A model of attitudinal cognition. *Behavioral Science*, 3(1):1–13, 1958.

2 Amit Agarwal, Moses Charikar, Konstantin Makarychev, and Yury Makarychev. O($\sqrt{\log n}$) approximation algorithms for min UnCut, min 2CNF deletion, and directed cut problems. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing – STOC '05*, New York, New York, USA, 2005. ACM Press.

3 Kook Jin Ahn, Graham Cormode, Sudipto Guha, Andrew McGregor, and Anthony Wirth. Correlation clustering in data streams. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2237–2246. JMLR.org, 2015.

4 Kook Jin Ahn and Sudipto Guha. Graph sparsification in the semi-streaming model. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming, 36th Internatilonal Colloquium, ICALP 2009*, volume 5556 of *Lecture Notes in Computer Science*, pages 328–338. Springer, 2009. doi:10.1007/978-3-642-02930-1_27.

5 Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: ranking and clustering. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 684–693. ACM, 2005.

6 Claudio Altafini. Dynamics of opinion forming in structurally balanced social networks. *PloS one*, 7(6):e38135, 2012.

7 T Antal, P L Krapivsky, and S Redner. Social balance on networks: The dynamics of friendship and enmity, 2006.

8 Samin Aref, Andrew J Mason, and Mark C Wilson. A modeling and computational study of the frustration index in signed networks. *Networks*, 75(1):95–110, January 2020.

9 Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Polynomial pass lower bounds for graph streaming algorithms. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.*, pages 265–276, 2019.

10 Sepehr Assadi and Ran Raz. Near-quadratic lower bounds for two-pass graph streaming algorithms. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 342–353. IEEE, 2020. doi:10.1109/FOCS46700.2020.00040.

11 Sepehr Assadi and Chen Wang. Sublinear Time and Space Algorithms for Correlation Clustering via Sparse-Dense Decompositions. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, volume 215 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:20, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITCS.2022.10.

12 Adi Avidor and Michael Langberg. The multi-multiway cut problem. *Theor. Comput. Sci.*, 377(1):35–42, May 2007.

13 Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Mach. Learn.*, 56(1-3):89–113, 2004. doi:10.1023/B:MACH.0000033116.57574.95.

14 Francisco Barahona. On the computational complexity of Ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10):3241–3253, 1982.

15 MohammadHossein Bateni, Hossein Esfandiari, and Vahab S. Mirrokni. Almost optimal streaming algorithms for coverage problems. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2017*, pages 13–23, 2017.

16 Anubhav Baweja, Justin Jia, and David P. Woodruff. An efficient semi-streaming PTAS for tournament feedback arc set with few passes. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022*, volume 215 of *LIPIcs*, pages 16:1–16:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

17 Soheil Behnezhad, Moses Charikar, Weiyun Ma, and Li-Yang Tan. Almost 3-approximate correlation clustering in constant rounds. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022*, pages 720–731. IEEE, 2022. doi:10.1109/FOCS54457.2022.00074.

18 Soheil Behnezhad, Moses Charikar, Weiyun Ma, and Li-Yang Tan. Single-pass streaming algorithms for correlation clustering. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023*, pages 819–849. SIAM, 2023. doi:10.1137/1.9781611977554.ch33.

19 András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 47–55. ACM, 1996. doi:10.1145/237814.237827.

20 Andrej Bogdanov and Emanuele Viola. Pseudorandom bits for polynomials. *SIAM J. Comput.*, 39(6):2464–2486, 2010. doi:10.1137/070712109.

21 Dorwin Cartwright and Frank Harary. Structural balance: a generalization of Heider's theory. *Psychol. Rev.*, 63(5):277–293, September 1956.

22 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.

23 Sayak Chakrabarty and Konstantin Makarychev. Single-pass pivot algorithm for correlation clustering. keep it simple! *arXiv preprint arXiv:2305.13560*, 2023.

24 Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. In *Proceedings of the 44th Symposium on Foundations of Computer Science (FOCS 2003)*, pages 524–533, 2003.

25 Shuchi Chawla, Konstantin Makarychev, Tselil Schramm, and Grigory Yaroslavtsev. Near optimal LP rounding algorithm for correlation clustering on complete and complete k-partite graphs. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015*, pages 219–228. ACM, 2015.

26 Flavio Chierichetti, Nilesh N. Dalvi, and Ravi Kumar. Correlation clustering in mapreduce. In Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani, editors, *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14*, pages 641–650. ACM, 2014.

27 Vincent Cohen-Addad, Silvio Lattanzi, Slobodan Mitrovic, Ashkan Norouzi-Fard, Nikos Parotsidis, and Jakub Tarnawski. Correlation clustering in constant many parallel rounds. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, volume 139 of *Proceedings of Machine Learning Research*, pages 2069–2078. PMLR, 2021. URL: http://proceedings.mlr.press/v139/cohen-addad21b.html.

28 Vincent Cohen-Addad, Euiwoong Lee, and Alantha Newman. Correlation clustering with Sherali-Adams. *CoRR*, abs/2207.10889, 2022.

29 Bhaskar DasGupta, German Andres Enciso, Eduardo Sontag, and Yi Zhang. Algorithmic and complexity results for decompositions of biological networks into monotone subsystems. *Biosystems.*, 90(1):161–178, July 2007.

30 James A Davis. Clustering and structural balance in graphs. *Human relations*, 20(2):181–187, 1967.

31 Ioannis Giotis and Venkatesan Guruswami. Correlation clustering with a fixed number of clusters. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, pages 1167–1176, 2006.

32 Frank Harary. On the notion of balance of a signed graph. *Michigan Math. J.*, 2(2), January 1953.

33 Frank Harary. Signed graphs for portfolio analysis in risk management. *IMA Journal of Management Mathematics*, 13(3):201–210, 2002.

34 Frank Harary. On the measurement of structural balance. *Behavioral Science*, 4(4):316–323, 2007.

35 Fritz Heider. Attitudes and cognitive organization. *J. Psychol.*, 21:107–112, January 1946.

36 Fritz Heider. *The Psychology of Interpersonal Relations*. Psychology Press, 1982.

37 Falk Hüffner, Nadja Betzler, and Rolf Niedermeier. Separator-based data reduction for signed graph balancing. *J. Comb. Optim.*, 20(4):335–360, November 2010.

38 Giovanni Iacono, Fahimeh Ramezani, Nicola Soranzo, and Claudio Altafini. Determining the distance to monotonicity of a biological network: a graph-theoretical approach. *IET Syst. Biol.*, 4(3):223–235, May 2010.

39 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, STOC '02, pages 767–775, New York, NY, USA, May 2002. Association for Computing Machinery.

40 Funda Kivran-Swaine, Priya Govindan, and Mor Naaman. The impact of network structure on breaking ties in online social networks: unfollowing on twitter. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1101–1104. ACM, 2011.

41 Shachar Lovett. Unconditional pseudorandom generators for low degree polynomials. *Theory Comput.*, 5(1):69–82, 2009. `doi:10.4086/toc.2009.v005a003`.

42 Andrew McGregor. Graph stream algorithms: a survey. *SIGMOD Rec.*, 43(1):9–20, 2014. `doi:10.1145/2627692.2627694`.

43 Michael Moore. An international application of Heider's balance theory. *Eur. J. Soc. Psychol.*, 8(3):401–405, July 1978.

44 Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993. `doi:10.1137/0222053`.

45 Christos Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, STOC '88, pages 229–234, New York, NY, USA, January 1988.

46 Ron M. Roth. *Introduction to coding theory*. Cambridge University Press, 2006.

47 Christopher Sibona. Unfriending on facebook: Context collapse and unfriending behaviors. In *2014 47th Hawaii International Conference on System Sciences*, pages 1676–1685. ieeexplore.ieee.org, January 2014.

48 Jiliang Tang, Yi Chang, Charu Aggarwal, and Huan Liu. A survey of signed network mining in social media. *ACM Computing Surveys (CSUR)*, 49(3):1–37, 2016.

49 Andreia Sofia Teixeira, Francisco C Santos, and Alexandre P Francisco. Emergence of social balance in signed networks. In *Complex Networks VIII*, pages 185–192. Springer International Publishing, 2017.

50 Haotian Wang, Feng Luo, and Jie Gao. Co-evolution of opinion and social tie dynamics towards structural balance. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'22)*, pages 3362–3388, January 2022.

51 Bo Xu, Yun Huang, Haewoon Kwak, and Noshir Contractor. Structures of broken ties: Exploring unfollow behavior on twitter. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 871–876, 2013. `doi:10.1145/2441776.2441875`.

52 Mihalis Yannakakis. Edge-Deletion problems. *SIAM J. Comput.*, 10(2):297–309, May 1981.

53 Thomas Zaslavsky. Balanced decompositions of a signed graph. *J. Combin. Theory Ser. B*, 43(1):1–13, August 1987.

A Offline Solution to Problem 2 (More Details)

Here we present an offline algorithm for solving Problem 2 when $\gamma < \varepsilon/100^4$ (which is an important part leading to Theorem 9), which can be transferred to the semi-streaming setting with superficial modification.

```
Nearly-Linear Time EPTAS solving Problem 2 for $G = (V, E^+ \cup E^-)$ and $\varepsilon \in (0,1)$, when $\gamma < \varepsilon/100^4$:
1. Sample $2000 \log n$ vertices uniformly at random from $V$. Call this set $S$ and let
   $G_S = (V, E(S) \cup E(S, V \setminus S))$.
2. Sample $100 \log \log n$ vertices uniformly at random from $S$. Call this set $S'$, the
   seed-set, and let $G_{S'} = (V, E(S') \cup E(S', V \setminus S'))$.
3. Sample $100^3 \cdot \frac{\log n}{\varepsilon^2}$ vertices uniformly at random for each vertex $v$. Call these sets
   $N_v$ and let $G_N = (V, \bigcup_v E(v, N_v))$.
4. Construct $\mathsf{frust}_{\varepsilon/10}(G, \cdot)$, a frustration sparsifier of $G$.
5. For every partition $(S'_L, S'_R)$ of $S'$
   (Mini-merging on $S'$)
      For every $v \in S \backslash S'$, assign $v$ to $S_L$ if $\mathsf{Dis}_{G_{S'}}(v, S'_L + v, S'_R) < \mathsf{Dis}_{G_{S'}}(v, S'_L, S'_R + v)$,
      and $S_R$ otherwise.
      $S'_L$ is assigned to $S_L$ and $S'_R$ is assigned to $S_R$.
   (Merging on $S$)
      For every $v \in V \setminus S$, assign $v$ to $L$ if $\mathsf{Dis}_{G_S}(v, S_L + v, S_R) < \mathsf{Dis}_{G_S}(v, S_L, S_R + v)$,
      and $R$ otherwise.
      $S_L$ is assigned to $L$ and $S_R$ is assigned to $R$.
   (Switching)
      Mark $v \in L$ if $\mathsf{Dis}_{G_N}(v, L + v, R - v) > \mathsf{Dis}_{G_N}(v, L - v, R + v)$ and mark $v \in R$
      similarly (flip the inequality).
      Switch assignments of all marked vertices from $L$ to $R$ or vice versa.
    Keep $(L, R)$ if $\mathsf{frust}_{\varepsilon/10}(G, L)$ is the smallest seen so far.
6. Return $(L, R)$.
```

B Proof of Lemma 12

In the proof of the communication complexity for the standard INDEX problem, the intuition is that the first player (Alice) has no knowledge of i^* other than knowing it is uniformly distributed (in the particular hard distribution). As such, the problem can only be solved if a sufficiently large number of bits are communicated. However, in the Leave-One INDEX problem (Problem 3), the first two players (Alice and Bob) might be able to learn where exactly i^* is by comparing their sets of indices S and T. In Problem 3, by the promise that $i^* \notin T$, Bob can learn the distribution of i^* better than uniform even without communication.

Our plan to prove Lemma 12 is to conduct a case analysis on whether the communication between Alice and Bob somehow "solves" the problem. To be precise, we separate the cases based on whether Alice's message significantly changes the distribution of i^* from Bob's perspective. If such a case happens, then we can use a result from [10] to show that the communication between *Alice and Bob* has to be $\Omega(\varepsilon^2 N)$. On the other hand, if the message from Alice does *not* tell Bob a lot about the distribution of i^*, a large number of bits are still required for the communication between *Bob and Charlie* since there are constant fraction of coordinates where i^* still looks almost uniform. The latter bound cannot be obtained directly by a reduction since i^* is already not uniform from Bob's perspective. Nevertheless, we can adapt the information-theoretic proof for INDEX to get the desired lower bound in a standard manner.

We now formalize the above intuition. Let $\Pi_{A,B}$ be the random variable for the message between Alice and Bob, and $\Pi_{B,C}$ be the random variable for the message between Bob and Charlie. Furthermore, let X be the random variable for the string being held by Alice and Bob, and let I be the random variable for i^*. For the choice of the set T, we use T' to denote the realization. Conditioning on the first message $\Pi_{A,B} = \pi_{A,B}$ and Bob's local inputs X, T, we let $I^B := I \mid X, T, \Pi_{A,B}$ be the random variable for i^* from *Bob's internal perspective after message* $\pi_{A,B}$. We slightly abuse notation and let these random variables also represent their distributions.

As promised, the first case we are going to show is the communication lower bound between Alice and Bob if the distribution of i^* that Bob learns is ε-far from his *prior* knowledge. Formally, we define the notation of ε-learn as follows.

▶ **Definition 14.** *Let $\pi_{A,B}$ be the message of a randomized one-way communication protocol. We say that Bob ε-learns i^* if after the message $\pi_{A,B}$, in expectation, the distribution of i^* from Bob's perspective is more than ε-far from the distribution he knows from his own input in the total variation distance, i.e.*

$$\mathbb{E}[\|I \mid \Pi_{A,B}, X, T - I \mid X, T\|_{\mathrm{TVD}}] > \varepsilon,$$

where the expectation is over the randomness of X, T and $\Pi_{A,B}$.

The notion of ε-learn was first developed by [9,10]. As such, we can prove a communication lower bound for Bob to ε-learn i^*.

▶ **Lemma 15.** *Any such protocol $\pi_{A,B}$ for Bob to ε-learn i^* has to use $\Omega\big(\varepsilon^2 \cdot N\big)$ bits of communication between Alice and Bob.*

Proof. We prove the lemma by a direct reduction from [10], which address the set intersection problem. The definition of the problem and the communication complexity is as follows.

▶ **Proposition 16** ([10]). *The Set-Intersection is a two-player game between Alice and Bob. The two players are given $A \subseteq [m]$ and $B \subseteq [m]$, respectively, with the promise that there exists a unique element e^* such that $e^* = A \cap B$. The goal is to find the target element e^* after one-way communication. Let E be the distribution of e^*, it is known that any communication protocol that achieves*

$$\mathbb{E}[\|E \mid \Pi, B - E \mid B\|_{\mathrm{TVD}}] > \varepsilon$$

has to use a communication of $\Omega\big(\varepsilon^2 m\big)$ bits, where the expectation is over the randomness of B and the protocol's randomness Π. In particular, the communication lower bound applies to the following distribution.

A hard distribution for Set-Intersection.

- *Sample two* disjoint *sets of coordinates A' and B' of size $N/4 - 1$ each uniformly at random from $[N]$.*
- *Sample an element e^* uniformly at random from $[N] \setminus (A' \cup B')$, and let $A := A' \cup \{e^*\}$, $B := B' \cup \{e^*\}$.*

The result of Proposition 16 can be generalized to back-and-forth communication if we allow Alice *or* Bob to ε-learn the distribution. However, for the purpose of our reduction, the one-way version suffices. The reduction from Set-Intersection to the communication between Alice and Bob in Lemma 12 is as follows.

Inputs: Alice holds $A \subseteq [m]$, Bob holds $B \subseteq [m]$, $A \cap B = e^*$.

A communication protocol PROT for Lemma 12 that ε-learns the index i^* as in Lemma 15.

1. Alice creates a set $S' = [m] \setminus A$.
2. Bob creates a set $T' = [m] \setminus B$.
3. Alice runs PROT, sends to Bob. Bob uses the distribution for i^* as the distribution for e^*.

To prove the correctness of the reduction, we only need to show that a). S' and T' are valid inputs for PROT and b). $i^* = e^*$. It is straightforward to verify these conditions: we know that an element is *not* covered by $S' \cup T'$ if and only if *both* A and B cover it, which is exactly e^*. Therefore, the desired lower bound is established. ◀

Lemma 15 implies that with $o(N)$ bits of communication it is impossible for Bob to gain knowledge of the distribution of I that is significantly different from what he already knows. We now observe that with the input X and T, and conditioning on Alice's message does *not* significantly change Bob's distribution, Bob's distribution of I is close to uniform on the $[N] \setminus T$ coordinates.

▷ Claim 17. Conditioning on the event that Bob does *not* ε-learn i^*, the distribution of i^* that Bob learns from T and Alice's message $\pi_{A,B}$ is at most ε-far from uniform in expectation over the supports of $[N] \setminus T$, i.e.,

$$\mathop{\mathbb{E}}_{T,\Pi_{A,B}} [\|I \mid T, \Pi_{A,B} - U([N] \setminus T)\|_{\mathrm{TVD}}] \leq \varepsilon.$$

Proof. We first show that without Alice's message, the distribution I of Bob restricted to $[N] \setminus T$ is uniform. This is a simple observation: with the promise that $i^* \notin T$, Bob can safely discard all coordinates therein. Since the string x is *independent* of i^*, Bob does not learn anything from x. As such, Bob's distribution remains *uniform* on the $[N] \setminus T$ coordinates.

We now condition on the fact that Bob does not ε learn i^*. As such, there is

$$\begin{aligned}
&\mathop{\mathbb{E}}_{X,T,\Pi_{A,B}} [\|I \mid X, T, \Pi_{A,B} - U([N] \setminus T)\|_{\mathrm{TVD}}] \\
&\leq \mathop{\mathbb{E}}_{X,T,\Pi_{A,B}} \left[\|I \mid X, T - U([N] \setminus T)\|_{\mathrm{TVD}} + \|I^B - I \mid X, T\|_{\mathrm{TVD}}\right] \qquad \text{(triangle inequality)} \\
&\leq \mathop{\mathbb{E}}_{X,T,\Pi_{A,B}} [0 + \|I \mid \Pi_{A,B}, X, T - I \mid X, T\|_{\mathrm{TVD}}] \leq \varepsilon,
\end{aligned}$$

where the last inequality directly comes from our assumption of no ε-learning.

Finally, note that the choice of i^* is independent of X (whether or not conditioning on $\Pi_{A,B}$). Therefore, we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{X,T,\Pi_{A,B}} [\|I \mid X, T, \Pi_{A,B} - U([N] \setminus T)\|_{\mathrm{TVD}}] \\
&= \mathop{\mathbb{E}}_{T,\Pi_{A,B}} [\|I \mid T, \Pi_{A,B} - U([N] \setminus T)\|_{\mathrm{TVD}}] \\
&\leq \varepsilon,
\end{aligned}$$

as desired. ◁

We now use the result of Claim 17 to prove the communication complexity conditioning on Bob does not ε-learn i^*. To continue, we insist that the distribution T where T' is sampled from always produces size-t sets. As such, we can establish the following lemma.

▶ **Lemma 18.** *Suppose the players sample from a distribution T to ensure the size of the set given to Bob is always t. Conditioning on the event that Bob does* not *ε-learn i^*, any communication protocol such that Charlie correctly outputs x_{i^*} with probability at least $1 - \delta$ requires at least $(1 - H_2(\delta) - 2\varepsilon) \cdot (N - t)$ bits of communication.*

On a high level, the proof of Lemma 18 uses a strategy that is similar to the information-theoretic proof of INDEX, but it controls the entropy on the coordinates outside T. The sketch of the proof is as follows. By Fano's inequality, for Charlie to output x_{i^*} correctly with probability at least $1-\delta$, there has to be $\mathbb{H}(X_I \mid \Pi_{A,B}, \Pi_{B,C}, T, I) \leq H_2(\delta)$. Using Claim 17, we can prove that if Bob does *not* ε-learn i^*, the "uncertainty" of X_I – measured by the entropy – is still comparable to the entropy of X_I restricted to the $[N] \setminus T$ coordinates, i.e.,

$$\mathbb{H}(X_I \mid \Pi_{A,B}, \Pi_{B,C}, T, I) \geq \frac{1}{N-t} \cdot \mathbb{H}\big(X_{[N]\setminus T} \mid \Pi_{A,B}, \Pi_{B,C}, T\big) - 2\varepsilon.$$

Therefore, we can apply Fano's inequality to lower bound the mutual information between $X_{[N]\setminus T}$ and the information revealed by the protocol, i.e., $(\Pi_{A,B}, \Pi_{B,C})$, which in turn gives us the lower bound on the number of bits to communicate. We leave the proof of Lemma 18 to the full version.

Finalizing the proof of Lemma 12. For completeness, we construct the hard distribution of Problem 3 as follows.

A hard distribution for Problem 3.

1. Sample A and B according to the hard distribution for Set-Intersection prescribed in Proposition 16.
2. Let $S = [N] \setminus A$, and $T = [N] \setminus B$. Sample X from $\{0,1\}^N$ uniformly at random.
3. Give (X, S) to Alice, (X, T) to Bob, and i^* to Charlie.

Observe that in the above distribution, for any choice of T', there is $t = |T'| = \frac{3}{4} \cdot N$. Furthermore, it agrees with the hard distribution of Proposition 16 through the reduction in Lemma 15.

If Bob ε-learns i^* for $\varepsilon = \frac{1}{100}$, then the communication lower bound is already $\Omega\big(N^2\big)$ (by Lemma 15). Therefore, we condition on the case when Bob does *not* (1/100)-learns i^*. As such, let us conditioning on the event that $\mathbb{E}\big[\|I^B - I \mid X, T\|_{\mathrm{TVD}}\big] \leq \frac{1}{100}$; and by Lemma 18, to make up the success probability of $\frac{99}{100}$, the protocol needs to send a message of length at least $(1 - H_2(\frac{1}{100}) - 2/100) \cdot (N - t) = \Omega(N)$ bits, as desired. ◀

How to Make Your Approximation Algorithm Private: A Black-Box Differentially-Private Transformation for Tunable Approximation Algorithms of Functions with Low Sensitivity

Jeremiah Blocki ✉
Purdue University, West Lafayette, IN, USA

Elena Grigorescu ✉
Purdue University, West Lafayette, IN, USA

Tamalika Mukherjee ✉
Purdue University, West Lafayette, IN, USA

Samson Zhou ✉
University of California Berkeley, CA, USA
Rice University, Houston, TX, USA

Abstract

We develop a framework for efficiently transforming certain approximation algorithms into differentially-private variants, in a black-box manner. Specifically, our results focus on algorithms A that output an approximation to a function f of the form $(1-\alpha)f(x)-\kappa \leq A(x) \leq (1+\alpha)f(x)+\kappa$, where $\kappa \in \mathbb{R}_{\geq 0}$ denotes additive error and $\alpha \in [0,1)$ denotes multiplicative error can be"tuned" to small-enough values while incurring only a polynomial blowup in the running time/space. We show that such algorithms can be made differentially private without sacrificing accuracy, as long as the function f has small "global sensitivity". We achieve these results by applying the "smooth sensitivity" framework developed by Nissim, Raskhodnikova, and Smith (STOC 2007).

Our framework naturally applies to transform non-private FPRAS and FPTAS algorithms into ε-differentially private approximation algorithms where the former case requires an additional postprocessing step. We apply our framework in the context of sublinear-time and sublinear-space algorithms, while preserving the nature of the algorithm in meaningful ranges of the parameters. Our results include the first (to the best of our knowledge) ε-edge differentially-private sublinear-time algorithm for estimating the number of triangles, the number of connected components, and the weight of a minimum spanning tree of a graph whose accuracy holds with high probability. In the area of streaming algorithms, our results include ε-DP algorithms for estimating L_p-norms, distinct elements, and weighted minimum spanning tree for both insertion-only and turnstile streams. Our transformation also provides a private version of the smooth histogram framework, which is commonly used for converting streaming algorithms into sliding window variants, and achieves a multiplicative approximation to many problems, such as estimating L_p-norms, distinct elements, and the length of the longest increasing subsequence.

2012 ACM Subject Classification Security and privacy → Usability in security and privacy

Keywords and phrases Differential privacy, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.59

Category RANDOM

Related Version *Full Version*: https://arxiv.org/abs/2210.03831

Funding *Jeremiah Blocki*: Supported in part by NSF CCF-1910659, NSF CNS-1931443. and NSF CAREER award CNS-2047272.
Elena Grigorescu: Supported in part by NSF CCF-1910659, NSF CCF-1910411, and NSF CCF-2228814.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 59; pp. 59:1–59:24
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Tamalika Mukherjee: Supported in part by NSF CCF-1910659 and and NSF CCF-2228814.
Samson Zhou: Work done in part while at Carnegie Mellon University and supported in part by a Simons Investigator Award and by NSF CCF-1815840.

1 Introduction

Approximation algorithms are often used to efficiently approximate a function $f : \mathcal{D} \to \mathbb{R}^+$ in settings where resource constraints prevent us from computing the function exactly. For example, problems such as Knapsack are NP-Hard and, unless $\mathsf{P} = \mathsf{NP}$, do not admit a polynomial time solution. However, the Knapsack problem admits a fully polynomial time approximation scheme (FPTAS) i.e., for any $\alpha > 0$ there is a deterministic algorithm, running in time $\text{poly}(n, 1/\alpha)$, which outputs a solution that is guaranteed to be be nearly as good (up to multiplicative factor $1 \pm \alpha$) as the optimal solution. As a second example, even if the problem is computationally tractable it may still be the case that the input dataset $D \in \mathcal{D}$ is extremely large, making it infeasible to load the entire dataset into RAM, or impractical to execute a linear time algorithm. To remedy such shortcomings, models such as sublinear-space and sublinear-time algorithms have been proposed. For example, one may want to estimate frequencies of elements that appear in a stream of n elements up to a multiplicative $1 \pm \alpha$ factor, while using only poly $\left(\log n, \frac{1}{\alpha}\right)$ memory cells. Or, one may want to estimate the number of connected components of a dense graph on n vertices up to (relative) additive error κ by only inspecting $\text{poly}(\log n, \frac{1}{\kappa})$ many edges.

In addition to time and space efficiency, user privacy is another important consideration in contexts where the input to our function f is sensitive user data. Differential privacy (DP) [20, 22] is a rigorous mathematical concept that gives provable guarantees on what it means for an algorithm to preserve the privacy of individual information in the input dataset. Informally, a randomized function computed on a dataset D is *differentially private* if the distribution of the function's output does not change significantly with the presence or absence of an individual data point. Thus, a natural goal is to develop efficient differentially private algorithms to approximate functions/queries of interest.

One general way to preserve differential privacy is to add noise scaled to the global sensitivity Δ_f of our function f, i.e., the maximum amount $|f(D) - f(D')|$ that the answer could change by modifying a single record in our dataset D to obtain a new dataset D'. This general approach yields efficient and accurate approximations for f as long as we have an efficient algorithm to compute f *exactly* and the global sensitivity of f is sufficiently small. However, in some resource-constrained settings, we may need to use an approximation algorithm $\mathcal{A}_f$ instead of evaluating f exactly. Unfortunately, the mechanism that computes $\mathcal{A}_f(D)$ and then adds noise scaled to the global sensitivity Δ_f of our function f is not necessarily differentially private. In particular, even if we are guaranteed that $|\mathcal{A}_f(D) - f(D)| \le \alpha f(D)$ we might still have $|\mathcal{A}_f(D) - \mathcal{A}_f(D')| \ge 2\alpha f(D) \gg \Delta_f$ for neighboring datasets D and D', e.g., suppose $\mathcal{A}_f(D) = (1 + \alpha) f(D)$ and $\mathcal{A}_f(D') = (1 - \alpha) f(D')$. Thus, the global sensitivity of $\mathcal{A}_f$ can be quite large and adding noise proportional to $\Delta_{\mathcal{A}_f}$ would prevent us from providing meaningful accuracy guarantees. This raises a natural question: Suppose that our function f admits an accurate (but not necessarily resource-efficient) differentially private approximation algorithm and that f also admits an efficient (but not necessarily private) approximation algorithm. Is it necessarily the case that there is also an equally efficient differentially private approximation algorithm?

Unfortunately, a result of [36, 18] suggests that the answer to the previous question may be no. Suppose our dataset D consists of n users $x_1, \ldots, x_n$ with n binary attributes i.e., $x_i \in \{0, 1\}^n$. Consider the function $f(D)$ that computes all of the one-way marginals i.e.,

$f(D) = \langle \frac{1}{n} \sum_{i=1}^{n} x_i[j] \rangle_{j=1}^{n} \in \mathbb{R}^n$. In particular, there is a non-private sublinear time algorithm that samples $\mathcal{O}(\log n)$ users and (with high probability) outputs a good approximation to *all* n one-way marginals. However, if we require that our algorithm satisfy pure, i.e, ε-differential privacy (resp. approximate, i.e., (ε, δ)-differential privacy) then we need to look at *at least* $\Omega(n/\varepsilon)$ (resp. $\Omega(\sqrt{n \log(1/\delta)})$) samples [36, 18]. In light of this, we pose the following general questions:

What are sufficient conditions for an approximation algorithm to be made differentially private?

Can an approximation algorithm be made differentially private in an efficient black-box manner?

Over the years, many differentially private approximation algorithms have been developed for problems in optimization, machine learning, and distribution testing (see for e.g., [35, 1, 30, 34]), in a somewhat ad-hoc manner. Often, these results give a differentially private algorithm for that specific problem and do not easily generalize to give differentially private algorithms for a large class of problems. A general framework for developing differentially private approximation algorithms for a large class of problems is desirable as this would not only make DP approximation algorithms more easily accessible to non-DP experts, but more importantly, it would shed light on what kinds of algorithms are more amenable to differential privacy. Furthermore, a framework that uses the underlying approximation algorithm as a *black-box* is desirable as this avoids the need to (re)design, (re)analyze, and (re)implement the new differentially private versions of these approximation algorithms. We emphasize that this type of framework has been well-studied for *computing* functions privately by calibrating noise proportional to their global or smooth sensitivity [23, 41] (see Section A for more discussion).

Our work makes partial progress towards answering these general questions. In particular, we give an efficient black-box framework for converting a non-private approximation algorithm $\mathcal{A}_f$ with tunable accuracy parameters into a differentially private approximation algorithm $\mathcal{A}'_f$ with reasonable accuracy guarantees as long as the global sensitivity Δ_f of the function f being approximated is sufficiently low. For the case when $\mathcal{A}_f$ is deterministic, we achieve a pure ε-differentially private approximation algorithm via a direct transformation, and when $\mathcal{A}_f$ is randomized, i.e., has a small failure probability, we achieve ε-differential privacy by first applying a transformation that gives a (ε, δ)-differentially private algorithm and then apply a postprocessing step to achieve ε-differentially privacy. For example, suppose that for any $\alpha > 0$ our algorithm $\mathcal{A}_f$, taking α and our dataset D as input, provides the guarantee that $|\mathcal{A}_f(D) - f(D)| \leq \alpha f(D)$ e.g., any FPTAS algorithm would satisfy our tunable accuracy requirement. In such a case, for any $\alpha > 0$ we can transform our non-private algorithm $\mathcal{A}_f$ into a differentially private version with multiplicative error α and small additive error term which (necessarily) comes from the noise that we added. Intuitively, we exploit the fact that we can run $\mathcal{A}_f$ with an even smaller accuracy parameter $\rho \ll \alpha$ which can be tuned to ensure that the smooth sensitivity of our algorithm is sufficiently small. Our same general framework still applies if we allow that the approximation algorithm $\mathcal{A}_f$ has a small additive error term i.e., $|\mathcal{A}_f(D) - f(D)| \leq \alpha f(D) + \kappa$. If $\mathcal{A}_f(D)$ is only guaranteed to output a good approximation (i.e., $|\mathcal{A}_f(D) - f(D)| \leq \alpha f(D) + \kappa$) with probability $1 - \delta/2$ (e.g., an FPRAS algorithm would satisfy this requirement with additive error $\kappa = 0$) then our framework achieves ε-differential privacy by first obtaining an approximate (ε, δ)-differential privacy algorithm and then a postprocessing step. In cases where the approximation algorithm is not tunably accurate our black-box framework does not necessarily apply[1]. For example, the best known approximation algorithms for vertex cover achieve the guarantee $f(G) \leq \mathcal{A}_f(G) \leq 2f(G)$ i.e., because there is no way to control the smooth sensitivity of our approximation algorithm.

1.1 Our Contributions

We introduce a generic black-box framework for converting certain approximation algorithms for a function $f : \mathcal{D} \to \mathbb{R}^+$ into a differentially private approximation algorithm using smooth sensitivity [41]. We first introduce a definition for *tunable* approximation algorithms used throughout our paper, and then present our main results for the DP framework, and then give new differentially private algorithms for a variety of approximation algorithms obtained via this unifying framework. We refer to our full version [11] for missing details.

▶ **Definition 1** ((α, κ, δ)-approximation). *An algorithm $\mathcal{A}_f$ is a (α, κ, δ)-approximation for f if for every $D \in \mathcal{D}$ with probability at least $1 - \delta$, we have that $(1 - \alpha)f(D) - \kappa \leq \mathcal{A}_f(D) \leq (1 + \alpha)f(D) + \kappa$.*

We may abuse notation and omit the failure probability δ parameter in the above definition, if it is clear from the context. Some algorithms $\mathcal{A}_f$ may take the approximation parameters $\alpha, \kappa, \delta \geq 0$ as input[2].

▶ **Definition 2** (tunable approximation). *$\mathcal{A}_f(D, \alpha, \kappa, \delta)$ provides a tunable approximation of f if for every $\alpha, \kappa, \delta \geq 0$ the algorithm $\mathcal{A}_f(\cdot, \alpha, \kappa, \delta)$ obtained by hardcoding α, κ and δ is a (α, κ, δ)-approximation for f.*
When the parameters α, κ, δ are clear from the context, we may abuse notation and just write $\mathcal{A}_f(D)$. For a tunable approximation algorithm we will use $R(n, \alpha, \kappa, \delta)$ to denote the amount of a particular resource used by the algorithm. The resources we consider in this work include time, space and query complexity of the algorithm (depending on the model) which we denote by $T(\cdot, \cdot, \cdot, \cdot)$, $S(\cdot, \cdot, \cdot, \cdot)$, and $Q(\cdot, \cdot, \cdot, \cdot)$ respectively.

As a concrete example any FPTAS algorithm $\mathcal{A}_f$ for f would be a tunable approximation for f with running time $T(n, \alpha, \kappa, \delta) = \text{poly}(n, 1/\alpha)$ for any $\alpha > 0$ and any $\kappa, \delta \geq 0$ – an FPTAS has no additive error ($\kappa = 0$) and zero failure probability ($\delta = 0$). Similarly a FPRAS algorithm would be a tunable approximation with running time $T(n, \alpha, \kappa, \delta) = \text{poly}(n, \alpha, \log(1/\delta))$ for any $\alpha, \delta > 0$ and any $\kappa \geq 0$ – an FPRAS also has no additive error ($\kappa = 0$).

General Framework for Approximation Algorithms. Our main result gives a framework for converting any existing non-DP algorithm $\mathcal{A}_f$ that provides an (α, κ, δ)-approximation of f into an ε-DP algorithm $\mathcal{A}''_f$ in the following manner: (1) Apply Algorithm 1 to obtain an (ε, δ)-DP algorithm $\mathcal{A}'_f$ that achieves an $(\alpha', \kappa', \delta')$-approximation (see Theorem 3), (2) Apply a postprocessing step on the output of $\mathcal{A}'_f$ outlined in Theorem 5 to achieve an ε-DP algorithm $\mathcal{A}''_f$ with the same accuracy guarantees as $\mathcal{A}'_f$ barring an additive error of $o(1)$. We emphasize that $\mathcal{A}_f$ is a tunable approximation, in other words, $\mathcal{A}_f$ takes the parameters (α, κ, δ) as input.

[1] One could still apply our black-box transformation. However, the accuracy guarantees would be degraded and we would only achieve (ε, δ)-differential privacy for sufficiently large values of $\varepsilon, \delta > 0$ which depend on the approximation error parameter α.

[2] We allow that $\alpha = \kappa = \delta = 0$ in which case $\mathcal{A}_f$ can simply compute f exactly – whether or not this computation is efficient.

▶ **Theorem 3** ((ε, δ)-privacy). *Suppose that $\mathcal{A}_f$ is a tunable approximation of $f : \mathcal{D} \to \mathbb{R}^+$. Then for all $\varepsilon > 0$, $\delta = \delta(n) > 0$*[3]*, $\alpha \geq 0$ and $\kappa \geq 0$, there is an algorithm $\mathcal{A}'_f$ such that*

(1) *(Privacy) $\mathcal{A}'_f$ is (ε, δ')-differentially private where $\delta' = \delta(1 + \exp(\varepsilon/2))$.*

(2) *(Accuracy) For all $D \in \mathcal{D}$, and $0 < \gamma$, with probability $1 - \delta - \exp(-\gamma)$,*

$$(1-\alpha')f(D) - \kappa' - \frac{2\Delta_f}{\varepsilon} \cdot \gamma \leq \mathcal{A}'_f(D) \leq (1+\alpha')f(D) + \kappa' + \frac{2\Delta_f}{\varepsilon} \cdot \gamma$$

where $\alpha' = \frac{\alpha(\varepsilon+16\gamma)}{12\log(4/\delta)}$, $\kappa' = \kappa\left(\frac{2\gamma\alpha}{3\log(4/\delta)} + \frac{8\gamma}{\varepsilon} + 1\right)$, and $\Delta_f := \max_{D,D' \in \mathcal{D}, D \sim D'} \|f(D) - f(D')\|_1$.

(3) *(Resource) $\mathcal{A}'_f$ uses $R\left(n, \frac{\varepsilon\alpha}{\log(4/\delta)}, \kappa, \delta\right)$ resource, where $R(\cdot,\cdot,\cdot,\cdot)$ is the resource used by $\mathcal{A}_f$.*

We illustrate the utility of Theorem 3 with specific parameters – if we have a non-private algorithm $\mathcal{A}_f$ that guarantees an $(\alpha, 0, \delta)$-approximation, then for constant ε, $\delta = \frac{1}{n^c}$ and $\gamma = c\log(n)$, we see that the DP algorithm $\mathcal{A}_f$ achieves an $\left(\alpha(1+o(1)), \mathcal{O}\left(\frac{\Delta_f \log(n)}{\varepsilon}\right), \frac{1}{n^c}\right)$-approximation. We typically use these parameters for δ, γ in our applications for streaming and sublinear-time algorithms.

Our reduction in Theorem 3 is quite simple – we describe the associated Algorithm 1 below.

Algorithm 1 (ε, δ)-differentially private framework $\mathcal{A}'_f$ for tunable approximation algorithm $\mathcal{A}_f$.

```
Input: Input set $D$, accuracy parameters $\alpha \in (0,1)$ and $\kappa$, DP parameter $\varepsilon$, DP failure
    probability $\delta \in (0,1)$, approx. algorithm $\mathcal{A}_f$.
1: Let $x_A := \mathcal{A}_f(D, \rho, \tau, \delta/2)$, where $\rho := \left(\frac{\varepsilon\alpha}{12\log(4/\delta)}\right)$, and $\tau := \kappa$.
2: return $x_A + X$ where $X \sim \mathsf{Lap}\left(\frac{2(4\rho x_A + 4\tau + \Delta_f)}{\varepsilon}\right)$
```

Note that in Algorithm 1, we leave our additive parameter κ as is when running $\mathcal{A}_f$, but we still choose to define $\tau := \kappa$. This is because depending on the problem, and the accuracy/efficiency guarantees desired, we can set τ to be a tuned version of κ (for e.g., we set $\tau := \kappa/\log(n)$ for the problem of estimating the number of connected components).

▶ Remark 4. We also note that, even if the failure probability $\delta > 0$ of $\mathcal{A}_f$ is non-negligible, that we can always boost the success probability by running $\mathcal{A}_f(D)$ multiple times and computing the median over all outputs. Even if the error rate $0 < \delta < 1/2$ is a constant we can always reduce the failure probability to a lower target $0 < \delta' \ll \delta$ while increasing the running time by a multiplicative factor $O(\log(1/\delta'))$. In particular, we can set δ' to be a negligible function of n such as $\delta' = n^{-\log n}$ whilst only incurring a $\mathcal{O}\left(\log^2 n\right)$ blowup in our running time.

We stress that we can only apply Theorem 3 to existing non-DP algorithms $\mathcal{A}_f$ that give an approximation guarantee of the form $(1-\alpha)f(D) - \kappa \leq \mathcal{A}_f \leq (1+\alpha)f(D) + \kappa$. For example, we cannot apply Theorem 3 to obtain an (ε, δ)-DP algorithm for estimating the minimum vertex cover size in sublinear time. This is because the non-DP sublinear-time algorithm $\mathcal{A}_{vc}$ has an approximation guarantee of the form $2VC(G) - \kappa n \leq \mathcal{A}_{vc} \leq 2VC(G) + \kappa n$. On the other hand, we can use our DP framework to obtain an (ε, δ)-DP algorithm for obtaining a

[3] typically we set $\delta = \text{negl}(n)$ or $\delta = n^{-c}$ for some constant $c > 0$. In particular $\delta(n)$ may approach zero as $n \to \infty$.

$(0, \kappa n, \delta)$-approximation of the maximum matching size in sublinear time (see full version for details [11]). Intriguingly, both the minimum vertex cover size and the maximum matching size algorithms use the same underlying strategy of estimating a greedy maximal matching in a local fashion, but since they return different estimators based on the objective and we can only use our framework as a black-box, we cannot apply our framework to the former while we can still apply it to the latter.

Finally, by applying a post-processing step described below, we show how to obtain an ε-DP algorithm from the (ε, δ)-DP algorithm obtained in Theorem 3. Importantly, the accuracy guarantee of the resulting ε-DP algorithm only differs by a small additive factor of $1/(KM)$, where $M = \max_D f(D)$ is the maximum possible output value, e.g., $M \le n^3$ for triangle counting and $K > 0$. Moreover for negligible δ, the accuracy guarantee of the resulting pure DP algorithm still holds with high probability.

▶ **Theorem 5.** *Let $M = \max_D f(D)$ and let parameter $K > 0$. If $\mathcal{A}'_f(D)$ is (ε, δ)-DP algorithm with accuracy guarantee $(1-\alpha)f(D) - \kappa \le \mathcal{A}_f(D) \le (1+\alpha)f(D) + \kappa$ holding with probability $1 - \eta$ then there exists an algorithm $\mathcal{A}''_f(D)$ which is ε-DP with accuracy guarantee $(1-\alpha)f(D) - \kappa - \frac{1}{KM} \le \mathcal{A}_f(D) \le (1+\alpha)f(D) + \kappa + \frac{1}{KM}$ with probability at least $1 - \eta - p$ where $p = \frac{\delta K(M+1)}{e^\varepsilon - 1 + \delta K(M+1)}$.*

Our second result is an analogous framework for converting any existing deterministic non-DP approximation algorithm $\mathcal{A}_f$ that provides an $(\alpha, \kappa, 0)$-approximation of f into an ε-DP algorithm $\mathcal{A}'_f$.

▶ **Theorem 6** (ε-privacy)**.** *Suppose that $\mathcal{A}_f$ is a deterministic tunable approximation of $f : \mathcal{D} \to \mathbb{R}^+$.Then for all $\varepsilon > 0$, $\alpha \ge 0$ and $\kappa \ge 0$, there is an algorithm $\mathcal{A}'_f$ such that*

(1) *(Privacy) $\mathcal{A}'_f$ is ε-differentially private.*

(2) *(Accuracy) For all $D \in \mathcal{D}$, we have that with probability $\ge 9/10$,*

$$(1-\alpha')f(D) - \kappa' - \frac{7\Delta_f}{\varepsilon} \le \mathcal{A}'_f(D) \le (1+\alpha')f(D) + \kappa' + \frac{7\Delta_f}{\varepsilon}$$

where $\alpha' := \alpha C_1(\varepsilon + C_2\gamma)$, $\kappa' := \kappa C_3(\alpha + \frac{C_4}{\varepsilon})$ for some constants $C_1, C_2, C_3, C_4 > 0$ and $\Delta_f := \max_{D, D' \in \mathcal{D}, D \sim D'} \|f(D) - f(D')\|_1$.

(3) *(Resource) $\mathcal{A}'_f$ uses $R\left(n, \frac{\varepsilon\alpha}{36}, \kappa\right)$ resource, where $R(\cdot, \cdot, \cdot)$ is the resource used by $\mathcal{A}_f$.*

DP Sublinear-time Results. We use Theorem 3 in conjunction with Theorem 5 in a black-box manner to obtain pure differentially-private sublinear time algorithms for several problems (see Table 1 for a summary).

In many models of sublinear-time computation the efficiency of the algorithm is measured in the number of queries made to the input, rather than the time complexity of the algorithm. It is often the case that the two are polynomially related, but there are instances in which the actual time complexity of the algorithm may be exponentially larger than the query complexity, in terms of the approximation factor. Nevertheless, in these instances too, the literature uses time and query complexity interchangeably. This is because the sublinear-time model assumes restricted or expensive access to the input, while further computation on local machines with the answers obtained from queries is considered to be cheap. We use query complexity for the sake of clarity.

We note that in the sublinear-time literature, the approximation parameters α, κ are usually considered to be a constant, but the analyses for most of these theorems hold for $\alpha = \alpha(n), \kappa = \kappa(n) \in (0, 1)$, where n is the input size.

Here we do not explicitly define the sublinear model (or the queries allowed) for each problem.For a graph G we denote the number of vertices as n, the number of edges as m, and the average degree of the graph as $\bar{d}$.

Typically, the accuracy guarantees of the non-DP results are presented with probability at least 2/3 – in order to apply our framework, we apply the median trick (see Remark 4) to boost the probability of success to $1 - \delta$. For simplicity of comparing our results, for any constant $c > 0$, we set $\delta := 1/n^c$ in the sequel.

Table 1 Summary of Sublinear-time DP graph algorithms obtained via our black-box DP transformation. According to our notation multiplicative error α means a multiplicative factor of $(1 \pm \alpha)$.

Problem	Reference	Privacy	Mult. error	Add. error	Query Complexity
Number of Triangles	[24]	Non-Private	α	0	$\mathcal{O}\left((\frac{n}{t^{1/3}} + \frac{m^{3/2}}{t})\,\text{poly}(\log(n), \frac{1}{\alpha})\right)$
	This Work	ε-edge DP	α	$\mathcal{O}\left(\frac{n\log(n)}{\varepsilon}\right)$	$\mathcal{O}\left((\frac{n}{t^{1/3}} + \frac{m^{3/2}}{t})\,\text{poly}(\log(n), \frac{1}{\alpha\varepsilon})\right)$
Connected Components	[7]	Non-Private	0	κn	$\mathcal{O}\left(\frac{1}{\kappa^2}\log\left(\frac{1}{\kappa}\right)\log(n)\right)$
	This Work	ε-edge DP	0	$\mathcal{O}\left(\kappa n\right) + \mathcal{O}\left(\frac{\log(n)}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{\log^3(n)}{\kappa^2}\log\left(\frac{\log(n)}{\kappa}\right)\right)$
Weighted MST	[19]	Non-Private	α	0	$\mathcal{O}\left(\bar{d}w\alpha^{-2}\log\left(\frac{\bar{d}w}{\alpha}\right)\log(n)\right)$
	This Work	ε-edge DP	α	$\mathcal{O}\left(\frac{\log(n)}{\varepsilon}\right)$	$\mathcal{O}\left(\bar{d}w\frac{\log^2(n)}{\alpha^2\varepsilon^2}\log\left(\frac{\bar{d}w\log(n)}{\alpha\varepsilon}\right)\log(n)\right)$
Average Degree	[33]	Non-Private	α	0	$\mathcal{O}\left(\frac{n}{\sqrt{m}}\,\text{poly}\left(\frac{\log(n)}{\alpha}\right)\log(n)\right)$
	[10]	ε-edge DP	α	0	$\mathcal{O}\left(\sqrt{n}\,\text{poly}\left(\frac{\log(n)}{\alpha}\right)\text{poly}\left(\frac{1}{\varepsilon}\right)\log(n)\right)$ (analysis assumes $\bar{d} \geq 1$)
	This Work	ε-edge DP	α	0	$\mathcal{O}\left(\frac{n}{\sqrt{m}}\,\text{poly}\left(\frac{\log^2(n)}{\alpha\varepsilon}\right)\log(n)\right)$ for $\bar{d} = \Omega(\frac{\log(n)}{n\varepsilon})$
Maximum Matching Size	[48]	Non-Private	0	κn	$\mathcal{O}\left(d^{\mathcal{O}\left(1/\kappa^2\right)}\log(n)\right)$
	This Work	ε-node DP	0	$\mathcal{O}\left(\frac{\kappa n}{\varepsilon}\right)$	$\mathcal{O}\left(d^{\mathcal{O}\left(1/\kappa^2\right)}\log(n)\right)$
Distance to Bipartiteness	[31]	Non-Private	0	κn^2	$\mathcal{O}\left((1/\kappa^3)\log(n)\right)$
	This Work	ε-edge DP	0	$\mathcal{O}\left(\kappa n^2\right) + \mathcal{O}\left(\frac{\log(n)}{\varepsilon}\right)$	$\mathcal{O}\left((\log^4(n)/\kappa^3)\right)$

We give the first (to the best of our knowledge) ε-DP sublinear time algorithm for estimating the number of triangles, connected components, and the weight of a minimum spanning tree whose accuracy guarantees hold with high probability.

For estimating the average degree of a graph, in recent work, [10] gave a pure ε-DP algorithm that achieves an $(\alpha, 0)$-approximation – a crucial observation is that their analysis only holds under the assumption that the average degree is at least one i.e., $\bar{d} \geq 1$ (see full version [11] for details). In this work, we remove the need for this assumption in the DP setting, by directly applying our black-box DP transformation to the original algorithm of [33] which works substantially better whenever we have $m = \omega(n)$ edges.

For estimating the maximum matching size in a graph, although [10] gave an ε-DP algorithm for estimating the maximum matching size that achieves a 2-multiplicative factor and κn additive factor, they left the task of finding an $(0, \kappa n)$-approximation in the DP setting as an open problem. In this work, we partially resolve this problem by presenting an ε-DP algorithm that gives a $(0, \mathcal{O}\left(\frac{\kappa n}{\varepsilon}\right))$-approximation of the maximum matching size. Crucially, our resulting analysis cannot guarantee that the added Laplace noise will be small with high probability, but only guarantees this will be the case with *constant probability.* This problem highlights a limitation of our black-box DP framework – if the non-DP algorithm that we want to apply our DP transformation on has a time/space/query complexity that has an exponential dependence on the approximation parameters then the resulting DP algorithm that achieves a similar approximation guarantee with high probability may be highly inefficient in terms of time/space/query complexity.

We also show how to apply our DP framework to an algorithm estimating the distance to bipartiteness in dense graphs [31, 3], which is accurate with probability $1 - o(1)$. The same reduction can be similarly applied to other natural properties that enjoy the feature that they admit distance-estimation algorithms with $\mathrm{poly}(1/\kappa)$ query complexity, where κ is the additive (normalized) error. For example, in the fundamental results of [32] an efficient distance approximation algorithm for the maximum k-cut problem, and thus k-colorability is presented. [27], also based on results from [6], generalizes these properties to the notion of "semi-homogeneous partition properties" and show efficient distance estimation algorithms for properties such as Induced P_3-freeness, induced P_4-freeness, and chordality.[4]

DP Streaming Results. We also apply our framework given by Theorem 3 and Theorem 5 to obtain differentially-private streaming algorithms for many fundamental problems, i.e., see Table 2 and Table 3. We remark that while the accuracy guarantees of our resulting algorithms may be surpassed by recent works studying these problems on an individual basis, our applications are black-box reductions that avoid individual utility and privacy analysis of each non-private streaming algorithm, which can be heavily involved and quite non-trivial, e.g., [40, 9, 43, 17, 46, 13].

In the streaming model, elements of an underlying dataset arrive one-by-one and the goal is to compute or approximate some predetermined function on the dataset using space that is sublinear in the size of the dataset. Our reductions also have wide applications to various archetypes of data stream models, which we now discuss. In insertion-only streams, the updates of the stream increment the underlying dataset, such as adding edges to a graph, adding terms to a sequence, or increasing the coordinates of a frequency vector. In turnstile (or dynamic) streams, the updates of the stream can both increase and decrease (or insert and delete) elements of the underlying dataset. Finally, in the sliding window model, only the W most recent updates of the data stream define the underlying dataset. Both the turnstile streaming model and the sliding window model are generalizations of insertion-only streams, and our framework has implications in all three models.

We first show that our framework can be applied to existing non-private dynamic algorithms for weighted minimum spanning tree, L_p norm estimation for $p \geq 1$ (and also F_p moment estimation for $0 < p < 1$), and distinct elements estimation. Thus using our framework, we essentially get private dynamic algorithms for these problems for free (in terms of correctness, not optimality). Since the dynamic streaming model generalizes the insertion-only streaming model, we also obtain private streaming algorithms in the insertion-only model as well. We summarize these results in Table 2.

We then apply our framework in Theorem 3 to the sliding window model. To that end, we first recall that given a $(\alpha, 0)$-approximation algorithm for the insertion-only streaming model, the smooth histogram framework [14] provides a transformation that obtains a $(\alpha, 0)$-approximation algorithm in the sliding window model for a "smooth" function. Although there are problems that are known to not be smooth, e.g., [15, 12, 16, 25, 37], the smooth histogram framework does provide a $(\alpha, 0)$-approximation to many important problems, such as counting, longest increasing subsequence, L_p norm estimation for $p \geq 1$ (and also F_p moment estimation for $0 < p < 1$), and distinct elements estimation. We remark that

[4] In general, distance estimation is closely related to tolerant testing [42], and for dense graph properties it is known that if a property is testable with a number of queries of the form $f(\kappa)$, then they admit a distance estimator [28] with an exponential blowup in $\frac{1}{\kappa}$ in the query complexity. Hence, in its general form the query complexity of estimating the distance to "hereditary" graph properties is a tower of exponential of height $\mathrm{poly}(1/\kappa)$ [4].

Table 2 Summary of DP algorithms in the dynamic/turnstile model obtained via our black-box DP transformation. According to our notation multiplicative error α means a multiplicative factor of $(1 \pm \alpha)$.

Problem	Reference	Privacy	Mult. error	Add. error	Space Complexity
Weighted Minimum Spanning Tree	[2]	Non-Private	α	0	$\mathcal{O}\left(\frac{1}{\alpha} n \log^4 n\right)$
	This Work	ε-DP	α	$\mathcal{O}\left(\frac{M \log m}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{1}{\alpha\varepsilon} n \log^5 n\right)$
L_p-norm, $p > 2$	[29]	Non-Private	α	0	$\mathcal{O}\left(\frac{1}{\alpha^2} n^{1-\frac{2}{p}} \log^2 n + \frac{1}{\alpha^{4/p}} n^{1-\frac{2}{p}} \log^{2/p} n \log^2 n\right)$
	This Work	ε-DP	α	$\mathcal{O}\left(\frac{\log m}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{p}{\alpha^2\varepsilon^2} n^{1-2/p}\right) \cdot \text{poly}\left(\log n, \log \frac{1}{\alpha\varepsilon}\right)$
L_p-norm, $p = 2$	[5]	Non-Private	α	0	$\mathcal{O}\left(\frac{1}{\alpha^2} \log^2 n\right)$
	This Work	ε-DP	α	$\mathcal{O}\left(\frac{\log m}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{1}{\alpha^2\varepsilon^2} \log^4 n\right)$
L_p-norm, $p \in (0, 2)$	[38]	Non-Private	α	0	$\mathcal{O}\left(\frac{1}{\alpha^2} \log^2 n\right)$
	This Work	ε-DP	α	$\mathcal{O}\left(\frac{\log m}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{1}{\alpha^2\varepsilon^2} \log^4 n\right)$
L_p-norm, $p = 0$	[39]	Non-Private	α	0	$\mathcal{O}\left(\frac{1}{\alpha^2} \log^2 n \log \frac{1}{\alpha}\right)$
	This Work	ε-DP	α	$\mathcal{O}\left(\frac{\log m}{\varepsilon}\right)$	$\tilde{\mathcal{O}}\left(\frac{1}{\alpha^2\varepsilon^2} \log^4 n\right)$

if we tried to apply the non-private smooth histogram framework to a DP insertion-only streaming algorithm, this might preserve privacy by post-processing, but may significantly increase the error in terms of accuracy. On the other hand, our framework avoids these issues and achieves private analogs of these algorithms in the sliding window model without compromising utility. We summarize our results for the sliding window model in Table 3. We note that in recent work, [26] give a generalized smooth histogram approach to convert a DP continual release streaming algorithm into a sliding window algorithm in the continual release setting. We focus on the one-shot streaming setting in our work.

Table 3 Summary of DP algorithms in the sliding window model obtained via our black-box DP transformation. According to our notation multiplicative error α means a multiplicative factor of $(1 \pm \alpha)$.

Problem	Reference	Privacy	Mult. error	Add. error	Space Complexity
Longest Increasing Subsequence	[44]	Non-Private	α	0	$\mathcal{O}\left(\frac{k^2}{\alpha} \log^2 n\right)$
	This Work	ε-DP	α	$\mathcal{O}\left(\frac{\log m}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{k^2}{\alpha\varepsilon} \log^4 n\right)$
Distinct Elements	[8]	Non-Private	α	0	$\mathcal{O}\left(\frac{1}{\alpha^3} \log^2 n\right)$
	This Work	ε-DP	α	$\mathcal{O}\left(\frac{\log m}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{1}{\alpha^3\varepsilon^3} \log^5 n\right)$
L_p-norm, $p = 2$	[47]	Non-Private	α	0	$\mathcal{O}\left(\frac{1}{\alpha^2} \log^3 n \log^3 \frac{1}{\alpha}\right)$
	This Work	ε-DP	α	$\mathcal{O}\left(\frac{\log m}{\varepsilon}\right)$	$\tilde{\mathcal{O}}\left(\frac{1}{\alpha^2\varepsilon^2} \log^5 n \log^3 \frac{1}{\alpha\varepsilon}\right)$
L_p-norm, $p \in (0, 2)$	[47]	Non-Private	α	0	$\mathcal{O}\left(\frac{1}{\alpha^2} \log^3 n (\log\log n)^2 \log^3 \frac{1}{\alpha}\right)$
	This Work	ε-DP	α	$\mathcal{O}\left(\frac{\log m}{\varepsilon}\right)$	$\tilde{\mathcal{O}}\left(\frac{1}{\alpha^2\varepsilon^2} \log^5 n\right)$

1.2 Our Techniques

Given a tunable (α, κ, δ)-approximation algorithm $\mathcal{A}_f$ for the function $f : \mathcal{D} \to \mathbb{R}^+$, our goal is to obtain a differentially private approximation algorithm that achieves a target $(\alpha', \kappa', \delta')$-approximation of f where α', κ' are in terms of α, κ.

Warm-up: When $\mathcal{A}_f$ is deterministic and only has multiplicative error. For simplicity, let us first consider an $(\alpha, 0, 0)$-approximation algorithm $\mathcal{A}_f$, in other words, $\mathcal{A}_f$ *always* outputs a value such that $(1-\alpha)f(D) \leq \mathcal{A}_f(D) \leq (1+\alpha)f(D)$. Since we want to make $\mathcal{A}_f$ differentially private, intuitively, we need to add noise to the output of $\mathcal{A}_f$. The local sensitivity of $\mathcal{A}_f$ at D (i.e., $LS_{\mathcal{A}_f}(D) = \max_{D' \sim D} |\mathcal{A}_f(D) - \mathcal{A}_f(D')|$) is upper bounded by $2\alpha f(D) + \Delta_f$. Since Δ_f is small and we can tune α to be arbitrarily small, it is tempting to think that we can just add noise proportional to $2\alpha f(D) + \Delta_f$. Unfortunately, scaling noise proportional to local sensitivity is not necessarily private. On the other hand we could ensure privacy by scaling noise proportional to the global sensitivity (i.e., $\max_{D \in \mathcal{D}} LS_{\mathcal{A}_f}(D) \leq \max_{D \in \mathcal{D}} 2\alpha f(D) + \Delta_f$)

but noise will likely be too large to obtain meaningful accuracy guarantees. We adopt the strategy of adding noise proportional to the smooth sensitivity [41] of $\mathcal{A}_f$ instead. In particular, [41] observed that if we can find a "sufficiently smooth" function $S_f(D) \geq LS_{\mathcal{A}_f}(D)$ upper bounding the local sensitivity of $\mathcal{A}_f$ then we can preserve privacy by computing $\mathcal{A}_f(D)$ and adding noise scaled according to $S_f(D)$.

We can show that the function $S_f(D) = 4\alpha\mathcal{A}_f(D) + \Delta_f$ is a β-smooth upper bound on the local sensitivity of $\mathcal{A}_f$ for $\beta = 6\alpha$ where S_f is β-smooth if $S_f(D) \leq e^{\beta} S_f(D')$ for all pairs of neighboring datasets $D \sim D'$. To achieve privacy using the smooth sensitivity framework we need to ensure that β is sufficiently small relative to our privacy parameters ε and δ (if applicable). For example, we can achieve $\left(\varepsilon, \delta\left(1 + \exp\left(\frac{\varepsilon}{2}\right)\right)\right)$-differential privacy by adding Laplace Noise scaled by $\frac{2S_f(D)}{\varepsilon}$, but only if S_f is β-smooth for $\beta \leq \frac{\varepsilon}{2\ln(2/\delta)}$. For pure differential privacy we require that $\beta < \frac{\varepsilon}{2(\lambda+1)}$ where λ is a parameter of the noise distribution – smaller λ implies higher variance.

If we want to ensure that the output is accurate, we also need to ensure that the calibrated noise with $S_f(D)$ is small e.g., $o(f(D)) + \mathcal{O}(\Delta_f)$. Note that by definition since $S_f(D) = 4\alpha\mathcal{A}_f(D) + \Delta_f$, and we add noise proportional $S_f(D)$, we expect that the noise added may be $> \alpha f(D)$. Thus, in order to address this challenge, our basic strategy is to run the original (non-private) approximation algorithm $\mathcal{A}_f$ with *tuned* error factors e.g., we decrease α by a multiplicative factor of $\frac{\varepsilon}{\ln(n)}$, let $\rho := \frac{\varepsilon\alpha}{\ln(n)}$. Since we are now running $\mathcal{A}_f(D, \rho, 0, 0)$, we have that the function $S_f(D) = 4\rho\mathcal{A}_f(D) + \Delta_f$ is a β-smooth upper bound on the local sensitivity of the algorithm $\mathcal{A}_f(\cdot, \rho, 0, 0)$. Assuming the global sensitivity Δ_f is small, we can now show that w.h.p. the noise sampled proportional to $S_f(D)$ is at most $\alpha f(D) + O(\Delta_f/\varepsilon)$ thus resulting in an ε-differentially private algorithm with reasonable accuracy.

By tuning the parameter α we actually accomplish two useful properties (1. accuracy) we decrease both the local sensitivity and our smooth upper bound $S_f(D)$ which reduces the magnitude of the noise that we add, and (2. privacy) we achieve β-smoothness for increasingly small values of β so that the required condition $\beta \leq \frac{\varepsilon}{2\ln(2/\delta)}$ (or $\beta < \frac{\varepsilon}{2(\lambda+1)}$) can be satisfied if we want to scale noise according to $S_f(D)$.

Extending to deterministic $\mathcal{A}_f$ with multiplicative and additive error. More generally, if we have an $(\alpha, \kappa, 0)$-approximation algorithm $\mathcal{A}_f$ then we can show that $S_f(D) = 4\alpha\mathcal{A}_f(D) + \Delta_f + 4\tau$ is a β-smooth upper bound on the local sensitivity of $\mathcal{A}_f$ with $\beta = 6\alpha$ (see Lemma 14). In particular, note that the additive error term κ does not adversely impact smoothness. Thus, we can achieve pure differentially privacy by tuning α such that $6\alpha < \beta < \frac{\varepsilon}{2(\lambda+1)}$ and scaling our noise according to $S_f(D)$ (see Lemma 15). We can also obtain stronger accuracy guarantees by relaxing the requirement for pure DP and tuning α such that $6\alpha \leq \beta \leq \frac{\varepsilon}{2\ln(2/\delta)}$ so that we can sample our noise from the Laplace distribution which has strong concentration guarantees.

When $\mathcal{A}_f$ is randomized. The remaining challenge is to handle randomized approximation algorithms $\mathcal{A}_f$ which are only guaranteed to output a good approximation with high probability i.e., with non-zero probability $\delta > 0$ the algorithm is allowed to output an arbitrarily bad approximation. In particular, let us consider an (α, κ, δ)-approximation algorithm $\mathcal{A}_f$. For any possible input D we are always guaranteed that with probability $\geq 1 - \delta$ the algorithm $\mathcal{A}_f(D)$ outputs a good approximation $(1 - \alpha)f(D) \leq \mathcal{A}_f(D) \leq (1 + \alpha)f(D)$. Unfortunately, the function $S_f(D) = 4\alpha\mathcal{A}_f(D) + \Delta_f + 4\kappa$ is no longer guaranteed to be a β-smooth upper bound on the local sensitivity of $\mathcal{A}_f$ since $\mathcal{A}_f$ may sometimes output a value outside the specified approximation bounds.

In order to address this challenge, we define a function $g_f(D)$ that matches $\mathcal{A}_f(D)$ with probability at least $1-\delta/2$ and is *always* guaranteed to output a good approximation. We emphasize that $g_f(D)$ may not be efficiently computable, but it is well-defined and only used for the purpose of analysis. More specifically, we set $g_f(D) = \mathcal{A}_f(D)$ as long as $(1-\alpha)f(D) - \kappa \leq \mathcal{A}_f(D) \leq (1+\alpha)f(D) + \kappa$. If $\mathcal{A}_f(D) > (1+\alpha)f(D)$, then we define $g_f(D) := (1+\alpha)f(D)$, similarly, if $\mathcal{A}_f(D) < (1-\alpha)f(D)$, then we define $g_f(D) := (1-\alpha)f(D) + \kappa$. Observe that we are always guaranteed that $(1-\alpha)f(D) - \kappa \leq g_f(D) \leq (1+\alpha)f(D) + \kappa$. Thus, $S_f(D) = 4\alpha g_f(D) + \Delta_f + 4\tau$ is a $\beta = 6\alpha$-smooth upper bound on the local sensitivity of g_f (see Lemma 8). As long as $6\alpha \leq \beta \leq \frac{\varepsilon}{2\ln(2/\delta)}$ we could preserve $\left(\varepsilon, \delta\left(1+\exp\left(\frac{\varepsilon}{2}\right)\right)\right)$-differential privacy by outputting $g_f(D)$ plus Laplace Noise scaled by $\frac{8\alpha g_f(D)+\Delta_f+8\tau}{\varepsilon}$ i.e., scaled according to our β-smooth upper bound on the local sensitivity of g_f. Unfortunately, the function g_f may not be efficiently computable. Thus, we substitute g_f for $\mathcal{A}_f$ and instead output $\mathcal{A}_f(D)$ plus Laplace noise scaled according to $\frac{8\alpha\mathcal{A}_f(D)+\Delta_f+8\tau}{\varepsilon}$. While $4\alpha\mathcal{A}_f(D) + \Delta_f + 4\tau$ is not necessarily a β-smooth upper bound on the local sensitivity of $\mathcal{A}$, the key point is that the latter (efficiently computable) procedure is equivalent to the former (differentially private) procedure as long as $g_f(D) = A_f(D)$ which happens as long as $\mathcal{A}_f$ outputs a good approximation i.e., except with probability $\delta/2$. Thus, we can apply a hybrid argument to argue that the final efficiently computable algorithm is $\left(\varepsilon, \frac{\delta}{2} + \delta\left(1+\exp\left(\frac{\varepsilon}{2}\right)\right)\right)$-differential privacy (see Lemma 10). In order to ensure accuracy, we use the same strategy as before, i.e., we run $\mathcal{A}_f(D, \rho, \tau, \delta/2)$, where $\rho \leq \frac{\varepsilon\alpha}{\log(1/\delta)}$. Sampling noise proportional to $S_f(D)$ (where $S_f(D)$ is now defined in terms of ρ), and absorbing the failure probability of algorithm $\mathcal{A}_f$ into the DP failure probability term δ, results in an approximate differentially private algorithm. Finally applying the postprocessing step results in a pure differentially private algorithm. We refer to the full proofs (Section 2) for additional details.

Applications. We give some intuition on how we apply Theorem 3 to various applications by choosing appropriate parameters. Recall that with probability $1-\delta-\exp(-\gamma)$, $\mathcal{A}'_f$ outputs $(1-\alpha')f(D) - \kappa' - \frac{2\Delta_f}{\varepsilon}\cdot\gamma \leq \mathcal{A}'_f(D) \leq (1+\alpha')f(D) + \kappa' + \frac{2\Delta_f}{\varepsilon}\cdot\gamma$, where $\alpha' = \frac{\alpha(\varepsilon+16\gamma)}{12\log(4/\delta)}$, and $\kappa' = \kappa\left(\frac{2\gamma\alpha}{3\log(4/\delta)} + \frac{8\gamma}{\varepsilon} + 1\right)$ with a time/space/query complexity blow-up incurred by running the original algorithm $\mathcal{A}_f$ with multiplicative accuracy parameter $\rho = \frac{\varepsilon\alpha}{\log(4/\delta)}$. First, observe that if the original algorithm $\mathcal{A}_f$ has time/space/query complexity with a dependence on $\mathrm{poly}(\frac{1}{\alpha})$, then the resulting time/space/query complexities for $\mathcal{A}'_f$ will still have a polynomial dependence, i.e., $\mathrm{poly}(\frac{\log(4/\delta)}{\alpha})$ – this naturally leads to FPRAS or FPTAS applications, as well as other classes of approximation algorithms like sublinear time or space. On the otherhand, if the time/space/query complexity of $\mathcal{A}_f$ has a non-polynomial dependence on $1/\alpha$, e.g., $\exp(\frac{1}{\alpha})$, then since δ is typically $\mathsf{negl}(n)$ or $\frac{1}{n^c}$ for $c > 0$, the resulting DP algorithm $\mathcal{A}'_f$ could have much worse time/space/query-guarantees with respect to n, e.g., in an extreme case if we set $\delta = 2^{-\mathrm{poly}(n)}$ then $\rho = \Omega(\mathrm{poly}(n)/\alpha)$ and we could incur a $\exp(\frac{\mathrm{poly}(n)}{\alpha})$ multiplicative overhead in the running time. It is worth noting that one could optionally reduce the additive error term κ' for $\mathcal{A}'_f$ by reducing the error term κ for $\mathcal{A}$.

We further emphasize this trade-off between obtaining small failure probability bounds and the accuracy or resource guarantees. Consider the following two examples – (Example 1) if we set the probability of failure, i.e., $\exp(-\gamma) = \delta = \frac{1}{n^c}$ for any $c > 0$, then the resulting approximation parameters are roughly $\alpha' = \alpha(1+o(1))$, and $\kappa' = \kappa(\alpha + \frac{\log(n)}{\varepsilon} + 1)$[5], and

[5] In the applications we consider the original (non-private) approximation algorithm typically has only multiplicative or only additive error and not both. In particular, we typically either have $\alpha > 0$ and $\kappa = 0$ or $\kappa > 0$ and $\alpha = 0$, but not the case where $\alpha > 0$ and $\kappa > 0$. Considering the case when $\kappa \neq 0$, and $\alpha = 0$, we (roughly) have $\kappa' = \kappa(\frac{\log(n)}{\varepsilon} + 1)$.

the additional error term depending on global sensitivity is roughly $\frac{\Delta_f \log(n)}{\varepsilon}$. We incur a time/space overhead by running $\mathcal{A}_f$ with multiplicative accuracy parameter $\rho = \Omega(\varepsilon\alpha/\log n)$ instead of α. (Example 2) if we set the probability of failure, i.e., $\exp(-\gamma) = \delta = \frac{1}{n^{\log\log(n)}} - \mathsf{negl}(n)$, then α' remains the same as before, and now $\kappa' = \kappa(\alpha + \frac{\log(n)\log\log(n)}{\varepsilon} + 1)$, but the additional error term depending on global sensitivity becomes $\frac{\Delta_f \log(n)\log\log(n)}{\varepsilon}$. In this latter case we incur time or space overhead by running $\mathcal{A}_f$ with multiplicative accuracy parameter $\rho = \Omega\left(\frac{\varepsilon\alpha}{\log n \log\log n}\right)$ – to reduce the $\kappa' \log n \log\log n$ error term it could be useful to run $\mathcal{A}_f$ with additive error parameter $\frac{\kappa}{\log n \log\log n}$ which may incur additional time or space overhead. Thus these two examples illustrate how, as we decrease the failure probability, the accuracy and resource (time/space in this case) guarantees become worse. See full version [11] for applications to sublinear time and streaming algorithms.

2 General Transformation for Approximation Algorithms

In this section, we formally define our black-box differentially private transformation for (randomized) approximation algorithms. Given a tunable approximation (see Definition 2) algorithm of f, call it $\mathcal{A}_f$, that outputs an (α, κ, δ)-approximation, our framework for randomized algorithms involves two steps – (1) Apply Algorithm 1 to $\mathcal{A}_f$ to obtain an (ε, δ)-DP algorithm $\mathcal{A}'_f$ with accuracy guarantees outlined in Theorem 3 (2) Apply postprocessing step to the output of $\mathcal{A}'_f$ to obtain an ε-DP algorithm (see Theorem 5).

We first prove Theorem 3 that provides theoretical guarantees for algorithm $\mathcal{A}'_f$ (Algorithm 1). This is our main contribution as the postprocessing step to obtain pure DP applies a folkore result.

Observe that even for the case when the original algorithm $\mathcal{A}_f$ gives an $(\alpha, 0, \delta)$-approximation of f (i.e., $\kappa = 0$), the resulting DP algorithm $\mathcal{A}'_f$ will still have an additive error, this additive error is inherent due to the requirement of adding Laplace noise to preserve DP. We emphasize that the Laplace noise added to the output of algorithm $\mathcal{A}_f$ depends on the global sensitivity of the function f, therefore, we can only get meaningful DP approximation algorithms using this transformation for functions with low global sensitivity.

Proof of Theorem 3. $\mathcal{A}'_f$ is defined in Algorithm 1 – it first runs $\mathcal{A}_f(D, \rho, \kappa, \delta/2)$ where $\rho := \frac{\varepsilon\alpha}{12\log(4/\delta)}$ and then adds Laplace Noise. Thus, the resource used by $\mathcal{A}'_f$ is $R(n, \rho, \kappa, \delta)$. The privacy guarantee follows from Lemma 10, and the accuracy guarantee follows from Lemma 12. ◀

▶ Remark 7. When $\mathcal{A}_f$ is a PRAS, by definition, the output of $\mathcal{A}_f$ is an $(\alpha, 0, \delta)$-approximation of f running in time $T(n, \alpha, 0, \delta) = \text{poly}(n, 1/\alpha, \log(1/\delta))$. Applying Theorem 3 with negligible $\delta = n^{-\log n}$ and $\gamma = \log^2 n$ for any $\alpha' > 0$ we obtain a private $\left(\alpha', O\left(\frac{\Delta_f}{\varepsilon \log^2 n}\right), 2n^{-\log n}\right)$-approximation with polynomial running time $\text{poly}(n, 1/\varepsilon, 1/\alpha')$.

▶ **Lemma 8.** *Let $0 < \rho < 1/2$. Suppose that $\mathcal{A}_f$ outputs a (ρ, τ, δ)-approximation of a function $f : \mathcal{D} \to \mathbb{R}^+$ with global sensitivity Δ_f. Let $\mathcal{A}_{f,R}$ denote a deterministic run of $\mathcal{A}$ using a fixed set of random coins R. Define function $g_{f,R}$ by*

$$g_{f,R}(D) = \begin{cases} \mathcal{A}_{f,R}(D) & \textit{if } (1-\rho)f(D) - \tau \leq \mathcal{A}_{f,R}(D) \leq (1+\rho)f(D) + \tau \\ (1-\rho)f(D) - \tau & \textit{if } \mathcal{A}_{f,R}(D) < (1-\rho)f(D) - \tau \\ (1+\rho)f(D) + \tau & \textit{if } \mathcal{A}_{f,R}(D) > (1+\rho)f(D) + \tau \end{cases}$$

Then the function $S_f(D) = 4\rho g_{f,R}(D) + 4\tau + \Delta_f$ is a β-smooth upper bound for $g_{f,R}$ where $\beta \geq 6\rho$.

Proof. Fix an arbitrary set of random coin tosses R. We frequently use the fact that $(1-\rho)f(D) - \tau \le g_{f,R}(D) \le (1+\rho)f(D) + \tau$. We also note that since $0 \le \rho < 1/2$, we have that $\frac{1}{2}f(D) - \tau \le g_{f,R}(D) \le 2f(D) + \tau$.

First, we show that Condition 1 of Definition 24 holds. Without loss of generality, we assume $f(D) \ge f(D')$, where D' is any neighboring input. Then:

$$
\begin{aligned}
LS_{g_{f,R}}(D) &= \max_{D':D\sim D'} \|g_{f,R}(D) - g_{f,R}(D')\| \\
&\le \|(1+\rho)f(D) + \tau - (1-\rho)f(D') + \tau\| \\
&\le \rho\|f(D) + f(D')\| + 2\tau + \Delta_f \\
&\le 2\rho f(D) + 2\tau + \Delta_f \\
&\le 4\rho g_{f,R}(D) + 2\rho\tau + 2\tau + \Delta_f && \text{as } \frac{1}{2}f(D) - \tau \le g_{f,R}(D) \\
&\le 4\rho g_{f,R}(D) + 4\tau + \Delta_f \\
&= S_f(D)
\end{aligned}
$$

Next, we show that Condition 2 of Definition 24 holds below. We have:

$$
\begin{aligned}
S_f(D) &= 4\rho g_{f,R}(D) + 4\tau + \Delta_f \\
&\le 4\rho\,(1+\rho)\,f(D) + 4\rho\tau + 4\tau + \Delta_f && \text{by def of } g_{f,R} \\
&\le 4\rho\,(1+\rho)\,(\Delta_f + f(D')) + 4\rho\tau + 4\tau + \Delta_f && \text{since } D \sim D' \\
&\le 4\rho(1+\rho)f(D') + (4\rho(1+\rho)+1)\Delta_f + 4\rho\tau + 4\tau \\
&\le 4\rho\frac{(1+\rho)}{1-\rho}(g_{f,R}(D') + \tau) + (1+6\rho)\Delta_f + 4\rho\tau + 4\tau && \text{by def of } g_{f,R} \\
&\le 4\rho(1+\rho)(1+2\rho)(g_{f,R}(D') + \tau) + (1+6\rho)\Delta_f + 4\rho\tau + 4\tau \\
&\le 4\rho(1+\rho)(1+2\rho)g_{f,R}(D') + 12\rho\tau + (1+6\rho)\Delta_f + 4\rho\tau + 4\tau \\
&\le 4\rho(1+\rho)(1+2\rho)g_{f,R}(D') + (1+6\rho)(\Delta_f + 4\tau) \\
&\le 4\rho(1+4\rho)g_{f,R}(D') + (1+6\rho)(\Delta_f + 4\tau) \\
&\le (1+6\rho)(4\rho g_{f,R}(D') + \Delta_f + 4\tau) \\
&\le e^{6\rho}\cdot(4\rho g_{f,R}(D') + \Delta_f + 4\tau) = e^{\beta}S_f(D'),
\end{aligned}
$$

where $\beta \ge 6\rho$. ◀

▶ Remark 9. As a special case if we have a $(0,\kappa,0)$-approximation algorithm $\mathcal{A}_f$ (i.e., no multiplicative error, zero failure probability) then applying Lemma 8 yields the smooth upper bound $S_f(D) = 4\tau + \Delta_f$. We observe that this smooth upper bound is independent of D and, therefore, S_f is just an upper bound on the global sensitivity of g_f. Furthermore, in this special case we are guaranteed that $\mathcal{A}_f(D) = g_f(D)$ with probability 1. Thus, in this special case, we can achieve pure ε-DP by computing $\mathcal{A}_f(D)$ and adding Laplace noise proportional to S_f.
If we have a $(0,\kappa,\delta)$-approximation algorithm for $\delta \ne 0$ then we still have $S_f(D) = 4\tau + \Delta_f$ which means that $S_f(D)$ is an upper bound on the global sensitivity of g_f. However, computing $\mathcal{A}_f(D)$ and adding Laplace noise proportional to S_f does not necessarily yield a pure DP algorithm since we may have $\Delta_f(D) \ne \mathcal{A}_f(D)$ with non-zero probability δ. If we have $(\alpha,\kappa,0)$-approximation algorithm $\mathcal{A}_f$, and $\alpha \ne 0$, since $S_f(D) = 4\rho\mathcal{A}_f(D) + 4\tau + \Delta_f$ still depends on the input D. However, we can still achieve DP using Theorem 6.

Applying Lemma 8 to the theorem calibrating noise to smooth bounds on the smooth sensitivity [41] we show that the Algorithm 1 preserves privacy below.

▶ **Lemma 10** (Privacy). *Algorithm 1 is (ε, δ')-differentially private where $\delta' = \delta(1+\exp(\varepsilon/2))$.*

Proof. Consider a modification of Algorithm 1, call it Algorithm 1' where instead of computing $\mathcal{A}_f(D, \rho, \tau, \delta/2)$ we instead sample the random coins R that $\mathcal{A}_f$ would have used and replace the value $\mathcal{A}_f(D, \rho, \tau, \delta/2; R)$ (which we denote as $\mathcal{A}_{f,R}(D)$ in the sequel) with $g_{f,R}(D)$.The function $g_{f,R}$ may not be efficiently computable, but we only use Algorithm 1' for the purpose of analysis. We first observe that by Lemma 8, for any $\beta \geq 6\rho$ the function $S_f(D) = 4\rho g_{f,R}(D) + 4\tau + \Delta_f$ is a β-smooth upper bound on the sensitivity of $g_{f,R}$. Thus, by Theorem 25, it is sufficient to set $\rho := \frac{\varepsilon\alpha}{12\log(4/\delta)}$ for $6\rho \leq \beta \leq \frac{\varepsilon}{2\ln(\frac{4}{\delta})}$ and add noise proportional to $\mathsf{Lap}\left(\frac{2S_f(D)}{\varepsilon}\right) = \mathsf{Lap}\left(\frac{2(4\rho g_{f,R}(D)+4\tau+\Delta_f)}{\varepsilon}\right)$ to preserve $(\varepsilon, \delta/2)$-privacy of Algorithm 1'.

Since $g_{f,R}(D)$ is $\mathcal{A}_{f,R}(D)$ except with probability $\delta/2$, Algorithm 1 is identical to Algorithm 1' except with probability $\delta/2$. Thus, this shows that Algorithm 1 is (ε, δ)-private. ◀

▶ **Fact 11.** *If $Y \sim \mathsf{Lap}(b)$, then $\Pr[|Y| \geq \ell \cdot b] = \exp(-\ell)$.*

▶ **Lemma 12** (Accuracy). *For all $\gamma > 0$, with probability $1 - \exp(-\gamma) - \delta$,*

$$\big(1 - \rho(1 + \frac{16\gamma}{\varepsilon})\big)f(D) - \tau\big(\frac{8\gamma\rho}{\varepsilon} + \frac{8\gamma}{\varepsilon} + 1\big) - \frac{2\Delta_f\gamma}{\varepsilon} \leq \mathcal{A}'(D) \leq \big(1 + \rho(1 + \frac{16\gamma}{\varepsilon})\big)f(D)$$
$$+ \tau\big(\frac{8\gamma\rho}{\varepsilon} + \frac{8\gamma}{\varepsilon} + 1\big) + \frac{2\Delta_f\gamma}{\varepsilon}.$$

Proof. First, using Fact 11, for any $\gamma > 0$, we have that,

$$\mathbf{Pr}\left[|X| \geq \frac{2(4\rho\mathcal{A}(D) + 4\tau + \Delta_f)}{\varepsilon} \cdot \gamma\right] = \exp(-\gamma)$$

$\mathcal{A}_f$ is a $(\rho, \tau, \delta/2)$-approximation of f so for any $D \in \mathcal{D}$, we have that $\mathcal{A}_f(D) \leq (1+\rho)f(D)+\tau$ with probability $1-\delta/2$. Since $0 < \rho < 1/2$ we have $(1+\rho)f(D)+\tau \leq 2f(D)+\tau$. Therefore, by a union bound,

$$\Pr\big[\big(\mathcal{A}_f(D) > (1+\rho)f(D) + \tau\big) \vee \big(\mathcal{A}_f(D) < (1-\rho)f(D) - \tau\big) \vee \big(|X|$$
$$\geq \frac{2(4\rho\mathcal{A}_f(D) + 4\tau + \Delta_f)}{\varepsilon} \cdot \gamma\big)\big] \leq \delta/2 + \exp(-\gamma)$$

Thus, with probability $1 - \exp(-\gamma) - \delta/2$, we have that

$$(1-\rho)f(D) - \tau \leq \mathcal{A}_f(D) \leq (1+\rho)f(D) + \tau \leq 2f(D) + \tau \tag{1}$$

and

$$|X| < \frac{2(4\rho\mathcal{A}_f(D) + 4\tau + \Delta_f)}{\varepsilon} \cdot \gamma \tag{2}$$

By plugging in Eq. 1 into Eq. 2, we have that with probability $1 - \exp(-\gamma) - \delta/2$,

$$|X| < \frac{2(4\rho(2f(D) + \tau) + 4\tau + \Delta_f)}{\varepsilon} \cdot \gamma$$

Overall, this means that with probability $1 - \exp(-\gamma) - \delta/2$,

$$(1-\rho)f(D) - \tau - \frac{2(4\rho(2f(D) + \tau) + 4\tau + \Delta_f)}{\varepsilon} \cdot \gamma \leq \mathcal{A}'_f(D)$$
$$\leq (1+\rho)f(D) + \tau + \frac{2(4\rho(2f(D) + \tau) + 4\tau + \Delta_f)}{\varepsilon} \cdot \gamma$$

Grouping the like terms together gives the theorem statement. ◀

▶ **Theorem 5.** *Let $M = \max_D f(D)$ and let parameter $K > 0$. If $\mathcal{A}'_f(D)$ is (ε,δ)-DP algorithm with accuracy guarantee $(1-\alpha)f(D)-\kappa \le \mathcal{A}_f(D) \le (1+\alpha)f(D)+\kappa$ holding with probability $1-\eta$ then there exists an algorithm $\mathcal{A}''_f(D)$ which is ε-DP with accuracy guarantee $(1-\alpha)f(D)-\kappa-\frac{1}{KM} \le \mathcal{A}_f(D) \le (1+\alpha)f(D)+\kappa+\frac{1}{KM}$ with probability at least $1-\eta-p$ where $p = \frac{\delta K(M+1)}{e^{\varepsilon}-1+\delta K(M+1)}$.*

The proof of Theorem 5 is in Appendix D. At a high level our idea is to define $\mathcal{A}'_f(D) = \frac{\lceil K\mathcal{A}_f(D)\rceil}{KM}$. The rounding step introduces a small additive error term $\le \frac{1}{KM}$ and ensures that $\mathcal{A}'_f(D)$ now has bounded range $|\mathcal{R}| \le (M+1)K$. Since $\mathcal{A}'_f$ has bounded range we can apply a folklore result (see Theorem 19) to transform this (ε,δ)-DP algorithm into an ε-DP algorithm.

2.1 Achieving Pure DP for Approximation Algorithms with Zero Failure Probability

In this section we show how one can achieve pure differential privacy ($\delta = 0$) when we have a tunable $(\alpha,\kappa,0)$-approximation algorithm. The basic framework is the same except that we use the Cauchy distribution instead of Laplace when applying the Smooth Sensitivity framework – see Theorem 25. Since we assume $\delta = 0$ in this section we will sometimes simplify notation and write $T(n,\alpha,\kappa)$ (resp. $S(n,\alpha,\kappa)$) instead of $T(n,\alpha,\kappa,0)$ (resp. $S(n,\alpha,\kappa,0)$). We move the proofs in this section to Appendix C.

Algorithm 2 ε-differentially private framework for tunable deterministic approximation algorithms.

```
Input: Input set $D$, accuracy parameter $\alpha \in (0,1)$, differential privacy parameter $\varepsilon$, approx.
    algorithm $\mathcal{A}_f$.
Let $x_A := \mathcal{A}_f(D,\rho,\tau,0)$ where $\rho := \frac{\varepsilon\alpha}{36}$ and $\tau := \kappa$.
return $x_A + X$ where $X \sim \mathsf{C}\left(\frac{6(4\rho x_A+\Delta_f)}{\varepsilon}\right)$
```

▶ Remark 13. When $\mathcal{A}_f$ is a PTAS, by definition, the output of $\mathcal{A}_f$ is an $(\alpha,0,0)$-approximation of f running in time $T(n,\alpha,0) = \mathrm{poly}(n,1/\alpha)$. Applying Theorem 6 with for any $\alpha > 0$ we obtain a private $\left(\alpha, O\left(\frac{\Delta_f}{\varepsilon}\right), 9/10\right)$-approximation with polynomial running time $\mathrm{poly}(n,1/\varepsilon,1/\alpha)$.

▶ **Lemma 14.** *Suppose that $\mathcal{A}_f$ outputs a $(\rho,\tau,0)$-approximation where $0 < \rho < 1/2$ of a function $f : \mathcal{D} \to \mathbb{R}^+$ with global sensitivity Δ_f. Then the function $S_f(D) = 4\rho\mathcal{A}_f(D) + 4\tau + \Delta_f$ is a β-smooth upper bound for $\mathcal{A}_f$ where $\beta \ge 6\rho$.*

The proof remains the same as in Lemma 8. Applying Lemma 14 to the theorem calibrating noise to smooth bounds on the smooth sensitivity [41] we show that the Algorithm 2 preserves privacy below.

▶ **Lemma 15.** *Algorithm 2 is ε-differentially private.*

▶ **Lemma 16.** *For all $\gamma > 6.5$, with probability at least $9/10$,*

$$\left(1-\rho\left(1+\frac{48\gamma}{\varepsilon}\right)\right)f(D) - \frac{24(\rho+1)\gamma\tau}{\varepsilon} - \frac{6\Delta_f}{\varepsilon}\cdot\gamma \le \mathcal{A}'_f(D)$$
$$\le \left(1+\rho\left(1+\frac{48\gamma}{\varepsilon}\right)\right)f(D) + \frac{24(\rho+1)\gamma\tau}{\varepsilon} + \frac{6\Delta_f}{\varepsilon}\cdot\gamma$$

▶ Remark 17. For simplicity, we have chosen to sample from the standard cauchy distribution $\lambda = 2$, more generally, if we sample noise with density $h(z) \propto \frac{1}{1+|z|^\lambda}$, where $\lambda = c$, then with probability $1 - \delta$, $\gamma = \frac{1}{\delta^{1/c}}$ in Lemma 16.

Application to the Knapsack Problem. As a fun example we consider the knapsack problem. The knapsack problem is well known to be NP-Hard, but also admits an FPTAS. To define an instance of the knapsack problem we have a maximum weight capacity W for the knapsack and n items each with a value $v_{max} \geq v_i \geq 0$ and a weight $w_i \geq 0$. The goal is to find a subset $S \subseteq [n]$ of items to put in the knapsack maximizing the total value $v(S) = \sum_{i \in S} v_i$ subject to the constraint that the total weight $w(S) = \sum_{i \in S} w_i$ does not exceed our capacity i.e., $w(S) \leq W$.

For the purpose of this illustration let's fix the capacity W and weights $w_1, \ldots, w_n$ and let $f(v_1, \ldots, v_n)$ denote the value of the optimal knapsack solution given values $v_1, \ldots, v_n$. Let's say that two knapsack instances $(W, v_1, \ldots, v_n, w_1, \ldots, w_n)$ and $(W, v'_1, \ldots, v'_n, w_1, \ldots, w_n)$ are neighbors if $\sum_i \| v_i - v'_i| \leq 1$. Thus, we are viewing the exact value of each item as sensitive and the goal of differential privacy is to prevent an attacker from inferring these sensitive values exactly. Observe that the global sensitivity of f is upper bounded by $\Delta_f \leq \max_{v \sim v'} \max_{S \subseteq [n]} |v(S) - v'(S)| \leq 1$[6].

Since there is an FPTAS algorithm for Knapsack we can find a non-private approximation algorithm $\mathcal{A}_f(\vec{v}, \alpha, \kappa = 0)$ running in time $T(n, \alpha) = \text{poly}(n, 1/\alpha)$. If we apply Theorem 6 then for any target α' our ε-DP algorithm $\mathcal{A}'_f$ runs in time $\text{poly}(n, 1/\varepsilon, 1/\alpha)$ and solves Knapsack with additive error $\mathcal{O}(1/\varepsilon)$ and multiplicative error α' with probability at least 9/10. If we don't require pure DP then we can also apply Theorem 3 then for any target α' our algorithm $\mathcal{A}'_f$ runs in time $\text{poly}(n, 1/\varepsilon, 1/\alpha, \log(1/\delta))$ and solves Knapsack with probability at least $1 - \delta - \exp(-\gamma)$ with additive error at most $\mathcal{O}(\gamma/\varepsilon)$ and multiplicative error α'.

3 Conclusion and Open Questions

In this work, we introduce a general framework for transforming a non-private approximation algorithm into a differentially private approximation algorithm. We show specific applications of our framework for sublinear time and sublinear space algorithms. Although our framework applies to a large variety of problems and settings, it does incur a small penalty in both runtime and space for achieving differential privacy. A natural question is whether these losses are necessary for a general black-box framework and what are sufficient conditions for achieving a black-box reduction.

It also seems possible that our framework could provide a method for achieving differentially private algorithms when the important resource is not runtime, number of queries, or space. For example, in distributed algorithms, it is often desired to achieve sublinear communication while in learning/testing, it is often desired to achieve sublinear query complexity. We believe that exploring the limits and capabilities of our framework in those settings would be a natural future direction of work.

[6] We could also define neighboring knapsack instances such that we can completely replace the value of any item i.e., v and v' are neighbors if there exists some index $i \in [n]$ such that $v_i \neq v'_i$ and $v_j = v'_j$ for all $j \neq i$. However, in this case we can we would have large global sensitivity $\Delta_f = v_{max}$. Thus, we won't be able to design an accurate differentially private approximation even if we are willing to solve the NP-Hard knapsack problem exactly.

References

1 Jayadev Acharya, Ziteng Sun, and Huanyu Zhang. Differentially private testing of identity and closeness of discrete distributions. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6879–6891, 2018. URL: https://proceedings.neurips.cc/paper/2018/hash/7de32147a4f1055bed9e4faf3485a84d-Abstract.html.

2 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 459–467, 2012.

3 Noga Alon, Wenceslas Fernandez de la Vega, Ravi Kannan, and Marek Karpinski. Random sampling and approximation of max-csps. *J. Comput. Syst. Sci.*, 67(2):212–243, 2003. doi:10.1016/S0022-0000(03)00008-4.

4 Noga Alon, Eldar Fischer, Ilan Newman, and Asaf Shapira. A combinatorial characterization of the testable graph properties: It's all about regularity. *SIAM J. Comput.*, 39(1):143–167, 2009.

5 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

6 Gunnar Andersson and Lars Engebretsen. Property testers for dense constraint satisfaction programs on finite domains. *Random Struct. Algorithms*, 21(1):14–32, 2002.

7 Petra Berenbrink, Bruce Krayenhoff, and Frederik Mallmann-Trenn. Estimating the number of connected components in sublinear time. *Inf. Process. Lett.*, 114(11):639–642, 2014. doi:10.1016/j.ipl.2014.05.008.

8 Jaroslaw Blasiok. Optimal streaming and tracking distinct elements with high probability. *ACM Trans. Algorithms*, 16(1):3:1–3:28, 2020.

9 Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. The johnson-lindenstrauss transform itself preserves differential privacy. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 410–419, 2012.

10 Jeremiah Blocki, Elena Grigorescu, and Tamalika Mukherjee. Privately estimating graph parameters in sublinear time. In *49th International Colloquium on Automata, Languages, and Programming, ICALP*, pages 26:1–26:19, 2022.

11 Jeremiah Blocki, Elena Grigorescu, Tamalika Mukherjee, and Samson Zhou. How to make your approximation algorithm private: A black-box differentially-private transformation for tunable approximation algorithms of functions with low sensitivity, 2022. arXiv:2210.03831.

12 Vladimir Braverman, Petros Drineas, Cameron Musco, Christopher Musco, Jalaj Upadhyay, David P. Woodruff, and Samson Zhou. Near optimal linear algebra in the online and sliding window models. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 517–528, 2020.

13 Vladimir Braverman, Joel Manning, Zhiwei Steven Wu, and Samson Zhou. Private data stream analysis for universal symmetric norm estimation. In *3rd Annual Symposium on Foundations of Responsible Computing FORC*, 2022.

14 Vladimir Braverman and Rafail Ostrovsky. Effective computations on sliding windows. *SIAM J. Comput.*, 39(6):2113–2131, 2010.

15 Vladimir Braverman, Rafail Ostrovsky, and Carlo Zaniolo. Optimal sampling from sliding windows. *J. Comput. Syst. Sci.*, 78(1):260–272, 2012.

16 Vladimir Braverman, Viska Wei, and Samson Zhou. Symmetric norm estimation and regression on sliding windows. In *Computing and Combinatorics - 27th International Conference, COCOON, Proceedings*, pages 528–539, 2021.

17 Zhiqi Bu, Sivakanth Gopi, Janardhan Kulkarni, Yin Tat Lee, Judy Hanwen Shen, and Uthaipon Tantipongpipat. Fast and memory efficient differentially private-sgd via JL projections. *CoRR*, abs/2102.03013, 2021. arXiv:2102.03013.

18 Mark Bun, Jonathan R. Ullman, and Salil P. Vadhan. Fingerprinting codes and the price of approximate differential privacy. *SIAM J. Comput.*, 47(5):1888–1938, 2018. doi:10.1137/15M1033587.

19 Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM J. Comput.*, 34(6):1370–1379, 2005. doi:10.1137/S0097539702403244.

20 Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP, Proceedings, Part II*, pages 1–12, 2006.

21 Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC*, pages 371–380. ACM, 2009.

22 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC, Proceedings*, pages 265–284, 2006.

23 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. *J. Priv. Confidentiality*, 7(3):17–51, 2016.

24 Talya Eden, Amit Levi, Dana Ron, and C. Seshadhri. Approximately counting triangles in sublinear time. *SIAM J. Comput.*, 46(5):1603–1646, 2017. doi:10.1137/15M1054389.

25 Alessandro Epasto, Mohammad Mahdian, Vahab S. Mirrokni, and Peilin Zhong. Improved sliding window algorithms for clustering and coverage via bucketing-based sketches. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 3005–3042, 2022.

26 Alessandro Epasto, Jieming Mao, Andres Muñoz Medina, Vahab Mirrokni, Sergei Vassilvitskii, and Peilin Zhong. Differentially private continual releases of streaming frequency moment estimations. In *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 48:1–48:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

27 Nimrod Fiat and Dana Ron. On efficient distance approximation for graph properties. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1618–1637. SIAM, 2021.

28 Eldar Fischer and Ilan Newman. Testing versus estimation of graph properties. *SIAM J. Comput.*, 37(2):482–501, 2007.

29 Sumit Ganguly and David P. Woodruff. High probability frequency moment sketches. In *45th International Colloquium on Automata, Languages, and Programming, ICALP*, pages 58:1–58:15, 2018.

30 Badih Ghazi, Ravi Kumar, Pasin Manurangsi, and Thao Nguyen. Robust and private learning of halfspaces. In Arindam Banerjee and Kenji Fukumizu, editors, *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 1603–1611. PMLR, 2021. URL: http://proceedings.mlr.press/v130/ghazi21a.html.

31 Arijit Ghosh, Gopinath Mishra, Rahul Raychaudhury, and Sayantan Sen. Tolerant bipartiteness testing in dense graphs. In *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 69:1–69:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

32 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.

33 Oded Goldreich and Dana Ron. On estimating the average degree of a graph. *Electron. Colloquium Comput. Complex.*, 2004.

34 Maoguo Gong, Yu Xie, Ke Pan, Kaiyuan Feng, and Alex Kai Qin. A survey on differentially private machine learning [review article]. *IEEE Comput. Intell. Mag.*, 15(2):49–64, 2020. doi:10.1109/MCI.2020.2976185.

35 Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. Differentially private combinatorial optimization. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1106–1125, 2010.

36 Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. 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 705–714. ACM, 2010. doi:10.1145/1806689.1806786.

37 Rajesh Jayaram, David P. Woodruff, and Samson Zhou. Truly perfect samplers for data streams and sliding windows. In *PODS: International Conference on Management of Data*, pages 29–40, 2022.

38 Daniel M. Kane, Jelani Nelson, Ely Porat, and David P. Woodruff. Fast moment estimation in data streams in optimal space. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC*, pages 745–754, 2011.

39 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*, pages 41–52, 2010.

40 Darakhshan J. Mir, S. Muthukrishnan, Aleksandar Nikolov, and Rebecca N. Wright. Pan-private algorithms via statistics on sketches. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*, pages 37–48, 2011.

41 Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 75–84, 2007.

42 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *J. Comput. Syst. Sci.*, 72(6):1012–1042, 2006.

43 Adam D. Smith, Shuang Song, and Abhradeep Thakurta. The flajolet-martin sketch itself preserves differential privacy: Private counting with minimal space. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020.

44 Xiaoming Sun and David P. Woodruff. The communication and streaming complexity of computing the longest common and increasing subsequences. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 336–345, 2007.

45 Jakub Tetek. Additive noise mechanisms for making randomized approximation algorithms differentially private. *CoRR*, abs/2211.03695, 2022. doi:10.48550/arXiv.2211.03695.

46 Lun Wang, Iosif Pinelis, and Dawn Song. Differentially private fractional frequency moments estimation with polylogarithmic space. In *The Tenth International Conference on Learning Representations, ICLR*, 2022.

47 David P. Woodruff and Samson Zhou. Tight bounds for adversarially robust streams and sliding windows via difference estimators. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 1183–1196. IEEE, 2021.

48 Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. Improved constant-time approximation algorithms for maximum matchings and other optimization problems. *SIAM J. Comput.*, 41(4):1074–1093, 2012. doi:10.1137/110828691.

A Related Work

One of the first differentially private frameworks for computing general functions was introduced by [23] which released functions with additive noise, where the noise is calibrated according to the *global sensitivity* of the function f. This framework was generalized by [41], to handle functions which might have a high global sensitivity but are usually less sensitive in practice. The framework allows the release of functions with instance-specific noise, where the noise that is added is not just determined by f but by the input dataset as well. The noise magnitude calibrated is according to the *smooth sensitivity* of f on the input dataset which is a smooth upper bound on the *local sensitivity* of f on an input dataset. The smooth sensitivity of a function may be hard to compute, therefore in the same work, [41] give a generic method called the *sample and aggregate* method that bypasses the explicit computation of the smooth sensitivity of the function and works even when the function is given as a black-box. [21] suggested a framework called *Propose-Test-Release* to release statistical estimators with additive noise where the noise is calibrated according to the *local sensitivity* of the estimator. Note that adding noise proportional to the local sensitivity of a function with respect to an input set usually does not preserve privacy, but their approach first proposes a bound on the local sensitivity and privately tests whether this bound holds for the specific input set, and then releases the noisy response to the query.

In the context of developing differentially private frameworks for approximation algorithms, [10] formally introduced the notion of *coupled global sensitivity* of a randomized algorithm, which gives an analogous framework as that of the global sensitivity framework [23], but for randomized approximation algorithms instead of deterministic functions. In this framework, one can run a non-private randomized approximation algorithm $\mathcal{A}_f(D)$ on the dataset, and privacy is obtained by adding noise proportional to the coupled global sensitivity of $\mathcal{A}_f$. More formally, the coupled global sensitivity measures the worst-case L_1-sensitivity of the outputs of a randomized algorithm $\mathcal{A}_f$ on neighboring inputs over a minimum coupling of the internal coin tosses of $\mathcal{A}_f$.

In independent work, Tetek [45] also explores the problem of transforming randomized approximation algorithms into (pure) differentially private approximation algorithms. In contrast to our results Tetek's transformation [45] assumes that the error of the original approximation algorithm either has small subexponential diameter or bounded mean error – assumptions that would not apply generically to every (tunable) approximation algorithm. Assuming subexponential error their work shows that it is possible to achieve ε-DP by adding Laplace Noise yielding accuracy guarantees that hold with high probability. However, the assumption of the error being subexponential is quite strong and does not often hold for many randomized approximation algorithms. While assuming bounded mean error is a weaker assumption on the error of the non-private randomized algorithm, however the DP noise is sampled from the Pareto distribution, which has polynomial tail bounds. This leads to accuracy guarantees which only hold with constant probability. Note that applying the median trick commonly used to amplify success probability in the non-private literature adversely affects the privacy budget and is thus not desirable. In contrast, our transformation applies generically to any (tunably) accurate approximation algorithm and we achieve accuracy guarantees that hold with high probability for the same problems studied in their paper. Finally, we correct an outdated claim[7] from the comparison to our work detailed in [45] that says that we only achieve approximate privacy. We can achieve *pure* DP algorithms by applying a postprocessing step to the output of our transformation as outlined in Theorem 5.

B Preliminaries

We use the notation $\tilde{\mathcal{O}}\left(f(n)\right)$ to mean $f(n) \cdot \text{polylog}(f(n))$. We define datasets D and D' as *neighboring*, denoted as $D \sim D'$, if removing or adding one point in D results in D'; alternatively, if changing one data point in D results in D'.

[7] A prior version of the paper achieved pure DP, but that transformation (Theorem 1.5) only applied to deterministic tunable approximation algorithms

▶ **Definition 18** (Differential privacy). *[22] An algorithm $\mathcal{A}$ is (ε, δ)-DP if for every pair of neighboring datasets $D \sim D'$, and for all sets $\mathcal{S}$ of possible outputs, we have that $\Pr[\mathcal{A}(D) \in \mathcal{S}] \le e^{\varepsilon} \Pr[\mathcal{A}(D') \in \mathcal{S}] + \delta$. When $\delta = 0$ we simply say that the algorithm is ε-DP.*

Given an (ε, δ)-DP algorithm, one can obtain an ε-DP algorithm under certain conditions outlined below. We include the proof for completeness in Appendix D.

▶ **Theorem 19** (Approximate DP to Pure DP). *Let $\mathcal{A} : \mathcal{D} \to \mathcal{R}$. If $\mathcal{A}$ is an (ε, δ)-DP algorithm such that $\delta \le \frac{(e^{\varepsilon}-1)p}{|\mathcal{R}|(1-p)}$ then there is an algorithm $\mathcal{A}'$ such that $\mathcal{A}'$ is ε-DP defined in the following manner.*

$$\mathcal{A}'(D) = \begin{cases} \mathcal{A}(D) & \textit{with probability } 1-p \\ \mathsf{random}(\mathcal{R}) & \textit{with probability } p \end{cases}$$

where $\mathcal{R}$ is the range of $\mathcal{A}_f$.

We define the distributions we will use to sample additive noise from below.

▶ **Definition 20** (Laplace distribution). *We say a random variable X is drawn from a Laplace distribution with mean μ and scale $b > 0$ if the probability density function of X at x is $\frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right)$. We use the notation $X \sim \mathsf{Lap}(b)$ to denote that X is drawn from the Laplace distribution with scale b and mean $\mu = 0$.*

▶ **Definition 21** (Cauchy distribution). *We say a random variable X is drawn from a Cauchy distribution with location parameter x_0 and scale $b > 0$ if the probability density function of X at x is $\frac{1}{\pi b}\left(\frac{b^2}{(x-x_0)^2+b^2}\right)$. We use the notation $X \sim \mathsf{C}(b)$ to denote that X is drawn from the Cauchy distribution with scale b and location parameter $x_0 = 0$.*

We formally define the concept of global sensitivity which is a worst-case notion of sensitivity for deterministic functions below.

▶ **Definition 22** (Global sensitivity). *The global sensitivity of a function $f : \mathcal{D} \to \mathbb{R}^d$ is defined by*

$$\Delta_f = \max_{D,D' \in \mathcal{D}, D \sim D'} \|f(D) - f(D')\|_1.$$

We define the notion of local sensitivity for a fixed input, which can be much smaller than the global sensitivity, but in general, adding noise calibrated according to the local sensitivity does not preserve DP.

▶ **Definition 23** (Local sensitivity). *For $f : \mathcal{D} \to \mathbb{R}$ and $D \in \mathcal{D}$, the local sensitivity of f at D is defined as*

$$LS_f(D) = \max_{D' : D \sim D'} \|f(D) - f(D')\|_1.$$

Note: if $f : \mathcal{D} \times \mathcal{R} \to \mathbb{R}$ is a randomized function which, in addition to a dataset $D \in \mathcal{D}$ takes random coins $r \in \mathcal{R}$ as input we simply define $LS_f(D) = \max_{r \in \mathcal{R}} LS_{f_r}$ where $f_r(D) \doteq f(D; r)$.

In order to add instance-specific noise, we define the notions of β-smooth upper bound which is a smooth upper bound on the local sensitivity.

▶ **Definition 24** (Smooth upper bound on local sensitivity). *For $\beta > 0$, a function $S : \mathcal{D} \to \mathbb{R}$ is a β-smooth upper bound on the local sensitivity of $f : \mathcal{D} \to \mathbb{R}$ if*

(1) *For all $D \in \mathcal{D}$, we have $S(D) \geq LS_f(D)$.*

(2) *For all $D, D' \in \mathcal{D}$ with $\|D - D'\|_1 = 1$, we have $S(D) \leq e^\beta \cdot S(D')$.*

Finally, although one cannot add noise calibrated with local sensitivity, one can add noise proportional to a β-smooth upper bound on the local sensitivity as follows.

▶ **Theorem 25** (Corollary 2.4 in [41]). *Let $f : \mathcal{D} \to \mathbb{R}$ and $S : \mathcal{D} \to \mathbb{R}$ be a β-smooth upper bound on the local sensitivity of f.*

(1) *If $\beta \leq \frac{\varepsilon}{2(\lambda+1)}$ and $\lambda > 1$, the algorithm $D \to f(D) + \frac{2(\lambda+1)S(D)}{\varepsilon} \cdot \eta$, where η is sampled from the distribution with density $h(z) \propto \frac{1}{1+|z|^\lambda}$, is ε-differentially private.*

(2) *If $\beta \leq \frac{\varepsilon}{2\ln(2/\delta)}$ and $\delta \in (0,1)$, then the algorithm $D \to f(D) + \frac{2S(D)}{\varepsilon} \cdot \eta$ where $\eta \sim \mathsf{Lap}(1)$ is (ε, δ')-differentially private for $\delta' = \delta\left(1 + \exp\left(\frac{\varepsilon}{2}\right)\right)$*[8].

C Proofs of Section 2.1

Proof of Theorem 6.

Proof. $\mathcal{A}'_f$ is defined in Algorithm 2 – we first run $\mathcal{A}_f(D, \rho, \kappa)$ where $\rho := \frac{\varepsilon\alpha}{36}$ and then we add noise proportional to the standard Cauchy distribution. Thus, the resource used will be $R(n, \rho, \kappa)$.

The privacy guarantee follows from Lemma 15, and the accuracy guarantee follows from Lemma 16. ◀

Proof of Lemma 15

Proof. We first observe that by Lemma 14, $S_f(D) = 4\mathcal{A}_f(D) + 4\tau + \Delta_f$ is a β-smooth upper bound for $\mathcal{A}_f$. Recall that $\rho := \frac{\varepsilon\alpha}{36}$, thus we can apply Theorem 25 (with $\lambda = 2$) where $6\rho \leq \beta \leq \frac{\varepsilon}{6}$ and conclude that it is sufficient to add noise proportional to $\mathsf{C}\left(\frac{2(2+1)S_f(x)}{\varepsilon}\right) = \mathsf{C}\left(\frac{6(4\rho\mathcal{A}_f(D)+4\tau+\Delta_f)}{\varepsilon}\right)$ to preserve ε-privacy.
◀

▶ **Fact 26.** *If $Y \sim \mathsf{C}(x; 0, b)$, then $\Pr[|Y| \geq \ell b] = 1 - \frac{2\tan^{-1}(\ell)}{\pi}$.*

Proof of Lemma 16.

Proof. First, we invoke Fact 26 below,

$$\Pr\left[|X| \geq \frac{6(4\rho\mathcal{A}_f(D) + 4\tau + \Delta_f)}{\varepsilon} \cdot \gamma\right] = 1 - \frac{2\tan^{-1}(\gamma)}{\pi} \leq \frac{1}{10}$$

where the final inequality comes from using the fact that $\gamma > 6.5$. In other words, with probability $\geq 9/10$,

$$|X| \leq \frac{6(4\rho\mathcal{A}_f(D) + 4\tau + \Delta_f)}{\varepsilon} \cdot \gamma \tag{3}$$

$\mathcal{A}_f$ is a $(\rho, \tau, 0)$-approximation of f so for any $D \in \mathcal{D}$, we have that $\mathcal{A}_f(D) \leq (1+\rho)f(D) + \tau$. Since $0 < \rho < 1/2$ we have $(1+\rho)f(D) + \tau \leq 2f(D) + \tau$.

[8] These bounds differ slightly from those listed in the original paper (Corollary 2.4 in [41]). We confirmed with the authors in private communication that δ should be multiplied by $(1 + \exp(\varepsilon/2))$.

By plugging in the relation $\mathcal{A}_f(D) \leq 2f(D) + \tau$ into Eq. 3, we have that with probability at least $9/10$,

$$|X| \leq \frac{6(8\rho f(D) + 4\rho\tau + 4\tau + \Delta_f)}{\varepsilon} \cdot \gamma$$

Thus with probability at least $9/10$,

$$(1-\rho)f(D) - \tau - \frac{6(8\rho f(D) + 4\rho\tau + 4\tau + \Delta_f)}{\varepsilon} \cdot \gamma \leq \mathcal{A}'_f(D)$$
$$\leq (1+\rho)f(D) + \tau + \frac{6(8\rho f(D) + 4\rho\tau + 4\tau + \Delta_f)}{\varepsilon} \cdot \gamma$$

Rearranging the like terms together in the above expression completes the proof. ◀

D Proof of Approximate DP to Pure DP transformation

▶ **Theorem 5.** *Let $M = \max_D f(D)$ and let parameter $K > 0$. If $\mathcal{A}'_f(D)$ is (ε, δ)-DP algorithm with accuracy guarantee $(1-\alpha)f(D) - \kappa \leq \mathcal{A}_f(D) \leq (1+\alpha)f(D) + \kappa$ holding with probability $1-\eta$ then there exists an algorithm $\mathcal{A}''_f(D)$ which is ε-DP with accuracy guarantee $(1-\alpha)f(D) - \kappa - \frac{1}{KM} \leq \mathcal{A}_f(D) \leq (1+\alpha)f(D) + \kappa + \frac{1}{KM}$ with probability at least $1-\eta-p$ where $p = \frac{\delta K(M+1)}{e^\varepsilon - 1 + \delta K(M+1)}$.*

Proof. Note that WLOG we can assume that $\mathcal{A}_f(D)$ outputs a value between 0 and M since we can always truncate the output to this range – this operation preserves privacy by postprocessing and does not adversely affect accuracy. For some $K > 0$, define algorithm $\mathcal{A}''_f(D)$ as outputting $\frac{\lceil K\mathcal{A}_f(D)\rceil}{KM}$. Observe that $\mathcal{A}''_f$ is (ε, δ)-DP by postprocessing and the accuracy guarantee of $\mathcal{A}''_f$ is almost identical to that of $\mathcal{A}_f$ since by definition $|\mathcal{A}''_f(D) - \mathcal{A}_f(D)| < \frac{1}{KM}$. By post-processing we can ensure that the range $\mathcal{R}$ of $\mathcal{A}''_f(D)$ is small $|\mathcal{R}| = (M+1)K$ since $\mathcal{R} = \{\frac{i}{KM} : 0 \leq i \leq KM\}$. Thus, we can pick p such that $\delta \leq \frac{(e^\varepsilon - 1)p}{|\mathcal{R}|(1-p)}$ and apply a folklore theorem (see Theorem 19) to transform our (ε, δ)-DP algorithm $\mathcal{A}''_f(D)$ to an ε-DP algorithm $\mathcal{A}'_f(D)$ in the following manner:

$$\mathcal{A}'_f(D) = \begin{cases} \mathcal{A}''_f(D) & \text{with probability } 1-p \\ \mathsf{random}(\mathcal{R}) & \text{with probability } p \end{cases}$$

By combining the accuracy guarantees of $\mathcal{A}_f$ and $\mathcal{A}''_f$ we see that with probability $1-\eta-p$, we have that $(1-\alpha)f(D) - \kappa - \frac{1}{KM} \leq \mathcal{A}_f(D) \leq (1+\alpha)f(D) + \kappa + \frac{1}{KM}$ where $p = \frac{\delta K(M+1)}{e^\varepsilon - 1 + \delta K(M+1)}$ as claimed. ◀

▶ **Theorem 19** (Approximate DP to Pure DP). *Let $\mathcal{A} : \mathcal{D} \to \mathcal{R}$. If $\mathcal{A}$ is an (ε, δ)-DP algorithm such that $\delta \leq \frac{(e^\varepsilon - 1)p}{|\mathcal{R}|(1-p)}$ then there is an algorithm $\mathcal{A}'$ such that $\mathcal{A}'$ is ε-DP defined in the following manner.*

$$\mathcal{A}'(D) = \begin{cases} \mathcal{A}(D) & \textit{with probability } 1-p \\ \mathsf{random}(\mathcal{R}) & \textit{with probability } p \end{cases}$$

where $\mathcal{R}$ is the range of $\mathcal{A}_f$.

Proof. Recall that we define $\mathcal{A}'$ as follows:

$$\mathcal{A}'(D) = \begin{cases} \mathcal{A}(D) & \text{with probability } 1-p \\ \mathsf{random}(\mathcal{R}) & \text{with probability } p \end{cases}$$

Let $D, D' \in \mathcal{D}$ be neighboring databases and fix output $y \in \mathcal{R}$. We first give a general claim regarding the probability of $\mathcal{A}'(D) = y$ in terms of the $\Pr[\mathcal{A}(D) = y]$.

▷ Claim 27. For $D \in \mathcal{D}$,

$$\Pr[\mathcal{A}'(D) = y] = \Pr[\mathcal{A}(D) = y]\,(1-p) + \frac{p}{|\mathcal{R}|}$$

Now we need to show that $\Pr[\mathcal{A}'(D) = y] \le e^{\varepsilon} \Pr[\mathcal{A}'(D') = y]$.

$$\begin{aligned}
&\Pr[\mathcal{A}'(D) = y] \\
&= \Pr[\mathcal{A}(D) = y]\,(1-p) + \frac{p}{|\mathcal{R}|} \\
&\le (1-p)\,(e^{\varepsilon} \Pr[\mathcal{A}(D') = y] + \delta) + \frac{p}{|\mathcal{R}|} \\
&\le e^{\varepsilon} \Pr[\mathcal{A}(D') = y]\,(1-p) + \delta\,(1-p) + \frac{p}{|R|} && (4) \\
&= e^{\varepsilon}(\Pr[\mathcal{A}'(D') = y] - \frac{p}{|\mathcal{R}|}) + \delta\,(1-p) + \frac{p}{|\mathcal{R}|} && (5) \\
&\le e^{\varepsilon} \Pr[\mathcal{A}'(D') = y] + \delta\,(1-p) + \frac{p}{|\mathcal{R}|}(1-e^{\varepsilon}) \\
&\le e^{\varepsilon} \Pr[\mathcal{A}'(D') = y] && (6)
\end{aligned}$$

The transition 4 to 5 follows from the observation that $\Pr[\mathcal{A}'(D') = y] = (1-p) \Pr[\mathcal{A}(D') = y] + \frac{p}{|\mathcal{R}|}$ and therefore, $(1-p) \Pr[\mathcal{A}(D') = y] = \Pr[\mathcal{A}'(D') = y] - \frac{p}{|\mathcal{R}|}$. The last equation 6 follows because $\delta \le \frac{(e^{\varepsilon}-1)p}{|\mathcal{R}|(1-p)}$ and thus

$$\delta\,(1-p) + \frac{p}{|\mathcal{R}|}(1-e^{\varepsilon}) \le 0 \;.$$ ◀

Fast Decoding of Explicit Almost Optimal ε-Balanced q-Ary Codes And Fast Approximation of Expanding k-CSPs

Fernando Granha Jeronimo ✉
Institute for Advanced Study, Princeton, NJ, USA

Abstract

Good codes over an alphabet of constant size q can approach but not surpass distance $1 - 1/q$. This makes the use of q-ary codes a necessity in some applications, and much work has been devoted to the case of constant alphabet q. In the large distance regime, namely, distance $1 - 1/q - \varepsilon$ for small $\varepsilon > 0$, the Gilbert–Varshamov (GV) bound asserts that rate $\Omega_q(\varepsilon^2)$ is achievable whereas the q-ary MRRW bound gives a rate upper bound of $O_q(\varepsilon^2 \log(1/\varepsilon))$. In this sense, the GV bound is almost optimal in this regime. Prior to this work there was no known explicit and efficiently decodable q-ary codes near the GV bound, in this large distance regime, for any constant $q \geq 3$.

We design an $\widetilde{O}_{\varepsilon,q}(N)$ time decoder for explicit (expander based) families of linear codes $\mathcal{C}_{N,q,\varepsilon} \subseteq \mathbb{F}_q^N$ of distance $(1-1/q)(1-\varepsilon)$ and rate $\Omega_q(\varepsilon^{2+o(1)})$, for any desired $\varepsilon > 0$ and any constant prime q, namely, almost optimal in this regime. These codes are ε-balanced,i.e., for every non-zero codeword, the frequency of each symbol lies in the interval $[1/q - \varepsilon, 1/q + \varepsilon]$. A key ingredient of the q-ary decoder is a new near-linear time approximation algorithm for linear equations (k-LIN) over $\mathbb{Z}_q$ on expanding hypergraphs, in particular, those naturally arising in the decoding of these codes.

We also investigate k-CSPs on expanding hypergraphs in more generality. We show that special trade-offs available for k-LIN over $\mathbb{Z}_q$ hold for linear equations over a finite group. To handle general finite groups, we design a new matrix version of weak regularity for expanding hypergraphs. We also obtain a near-linear time approximation algorithm for general expanding k-CSPs over q-ary alphabet. This later algorithm runs in time $\widetilde{O}_{k,q}(m + n)$, where m is the number of constraints and n is the number of variables. This improves the previous best running time of $O(n^{\Theta_{k,q}(1)})$ by a Sum-of-Squares based algorithm of [AJT, 2019] (in the expanding regular case).

We obtain our results by generalizing the framework of [JST, 2021] based on weak regularity decomposition for expanding hypergraphs. This framework was originally designed for binary k-XOR with the goal of providing near-linear time decoder for explicit binary codes, near the GV bound, from the breakthrough work of Ta-Shma [STOC, 2017]. The explicit families of codes over prime $\mathbb{F}_q$ are based on suitable instatiations of the Jalan–Moshkovitz (Abelian) generalization of Ta-Shma's distance amplification procedure.

2012 ACM Subject Classification Theory of computation → Error-correcting codes; Theory of computation → Expander graphs and randomness extractors

Keywords and phrases Decoding, Approximation, GV bound, CSPs, HDXs, Regularity

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.60

Category RANDOM

Funding This material is based upon work supported by the NSF grant CCF-1900460. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.
Fernando Granha Jeronimo: NSF grant CCF-1900460

Acknowledgements We thank Vedat Alev and Shravas Rao for stimulating discussions during the initial phase of this project. We thank Shashank Srivastava and Madhur Tulsiani for stimulating discussions leading to [24].

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 60; pp. 60:1–60:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Codes over small alphabet sizes have attracted a lot of effort in coding theory [17]. There is now a vast theory about them, but important mysteries remain. One very natural alphabet is the binary alphabet, which has a myriad of uses and applications. However, it also comes with an important limitation, namely, a family of good binary codes cannot[1] surpass distance 1/2. By using a q-ary alphabet, a family of good codes can approach distance $1 - 1/q$ but not surpass it. This makes the use of q-ary codes a necessity whenever larger distances are needed. Working towards explicit and efficiently decodable codes with optimal trade-offs between rate and distance has been a challenging but fruitful guiding goal in coding theory.

In the large distance case, namely, distances are of the form $1 - 1/q - \varepsilon$ for small values of $\varepsilon > 0$, the Gilbert–Varshamov (GV) bound [13, 36] asserts that rate $\Omega_q(\varepsilon^2)$ is achievable whereas the q-ary version of McEliece, Rodemich, Rumsey and Welch (MRRW) [26] gives an impossibility upper of $O_q(\varepsilon^2 \log(1/\varepsilon))$. This means that the GV bound is nearly optimal in this regime of constant alphabet size q and large distance. To the best of our knowledge, in this regime, (prior to this work) no explicit and efficiently decodable families of q-ary codes near the GV bound were known for any $q \geq 3$.

Two widely used approaches in the construction of q-ary codes for small q are based on code concatenation [11] and on algebraic geometry (AG) constructions [30, 34]. Using code concatenation, it is possible to obtain explicit constructions achieving the suboptimal Zyablov bound trade-off between rate and distance, which gives a rate of $\Omega_q(\varepsilon^3)$. Some explicit families of AG codes are celebrated for beating the GV bound in some specific parameter regimes, e.g., the seminal work of Tsfasman, Vlădut and Zink[2] [35] or the (non-linear) construction of Elkies [9]. This surprising phenomenon of explicit AG codes beating random codes cannot happen in a major way in the large distance and constant alphabet regime since the GV bound is nearly optimal. Furthermore, known explicit constructions of linear AG codes are far from the GV bound for large distances and constant q. Another drawback of several explicit families of good AG codes is that known decoders can take much longer than linear time in the blocklength [27].

On a more combinatorial side, in a breakthrough work using expander graphs, Ta-Shma [31] gave the first explicit construction of binary codes of distance $1/2 - \varepsilon$ and rate $\Omega(\varepsilon^{2+o(1)})$, namely, near the Gilbert–Varshamov bound. A polynomial time decoder for these binary codes was first given in [23] followed by a near-linear time decoder in [24]. Subsequently, Jalan and Moshkovitz [22] extended Ta-Shma's analysis [31] to handle (in particular) codes over larger alphabets[3]. Suitable instantiations of [22] imply explicit codes over prime $\mathbb{F}_q$ of distance $1 - 1/q - \varepsilon$ with rate $\Omega_q(\varepsilon^{2+o_q(1)})$, namely, again near the (q-ary) GV bound for constant q.

Motivated by the above situation, we design a near-linear time decoder for explicit families of q-ary codes of distance $(1 - 1/q)(1 - \varepsilon)$ and rate $\Omega(\varepsilon^{2+o_q(1)})$ for any constant prime q, namely, near the GV bound in the large distance regime. More precisely, our main result is as follows (answering a question from [22]).

[1] This is a consequence of the Plotkin bound.

[2] More precisely, the TVZ bound [35] establishes a rate of $r \geq 1 - \delta - 1/(\sqrt{q} - 1)$ with respect to the relative distance δ.

[3] More precisely, [22] analyzed the (scalar) Abelian case of Ta-Shma's amplification.

▶ **Theorem 1** (Main I – Near-linear Time Unique Decoding over $\mathbb{F}_q$)**.** *Let q be a prime. For every $\varepsilon > 0$ sufficiently small, there are explicit linear Ta-Shma codes $\mathcal{C}_{N,q,\varepsilon} \subseteq \mathbb{F}_q^N$ for infinitely many values $N \in \mathbb{N}$ with*

(i) *distance at least $(1 - 1/q)(1 - \varepsilon)$ (actually ε-balanced),*

(ii) *rate $\Omega_q(\varepsilon^{2+\alpha})$ where $\alpha = O(1/(\log_2(1/\varepsilon))^{1/6})$, and*

(iii) *an $r(q/\varepsilon) \cdot \tilde{O}(N)$ time randomized unique decoding algorithm that decodes within radius $((1 - 1/q)(1 - \varepsilon))/2$,*

where $r(x) = \exp(\exp(\text{poly}(x)))$.

In fact, we actually prove the following stronger *list* decoding result.

▶ **Theorem 2** (Near-linear time List Decoding over $\mathbb{F}_q$)**.** *Let q be a prime. For every $\varepsilon > 0$ sufficiently small, there are explicit binary linear Ta-Shma codes $\mathcal{C}_{N,q,\varepsilon} \subseteq \mathbb{F}_q^N$ for infinitely many values $N \in \mathbb{N}$ with*

(i) *distance at least $(1 - 1/q)(1 - \varepsilon)$ (actually ε-balanced),*

(ii) *rate $\Omega_q(\varepsilon^{2+\alpha})$ where $\alpha = O(1/(\log_2(1/\varepsilon))^{1/6})$, and*

(iii) *an $r(q/\varepsilon) \cdot \tilde{O}(N)$ time randomized list decoding algorithm that decodes within radius $1 - 1/q - 2^{-\Theta_q((\log_2(1/\varepsilon))^{1/6})}$ and works with high probability,*

where $r(x) = \exp(\exp(\text{poly}(x)))$.

We obtain our results by building on and extending the *binary* decoding framework in [24]. This framework is based on a generalization of the weak regularity decomposition to (sparse) *expanding* hypergraphs that generalizes the seminal work of Frieze and Kannan [12]. The weak regularity decomposition of [24] was then used to approximate *expanding* k-XOR instances naturally arising in the decoding of binary Ta-Shma's codes [31]. Similarly, constraint satisfaction problems (CSPs) will play a key role in our decoder. Here, we also take the opportunity to investigate *expanding* CSPs more broadly.

An instance of a k-CSP is given by a k-uniform (ordered) constraint hypergraph $W \subseteq [n]^k$, where each vertex is associated with a variable taking values in an alphabet of size q and each edge is associated with a constraint involving the variables of its vertices. While even approximating a CSP is NP-hard in general, suitable notions of expansion of the constraint hypergraph allow for efficient approximation algorithms as in [24]. One such notion is *splittability* [2]. Roughly speaking, a τ-splittable collection of tuples for some $\tau \in (0, 1]$ is the higher-order analogue of the second largest singular value of the normalized adjacency matrix of a graph (the smaller the τ the more expanding is the collection). Approximating expanding k-CSPs is at the core of some decoding algorithms for expander based constructions of codes [8, 1, 23, 24, 6].

As mentioned above, approximating expanding k-CSPs will be again at the core of our extension of [24] to more general constraints over larger alphabets. Our new q-ary decoder will need to handle instances of linear equations over the alphabet $\mathbb{Z}_q$, where each equation involves a sum of k variables. This kind of k-CSP is commonly denoted k-LIN over alphabet $\mathbb{Z}_q$. We will see that the special algebraic structure of these linear constraints will allow to obtain some improved parameter trade-offs, which will be explored in the decoding application. More precisely, the expansion (splittability) parameter τ will have no dependence on alphabet size q and only a polynomial dependence on the arity[4] k, and this allows us to obtain better approximation guarantees. Our second result follows.

[4] In the binary case of [24], it was also possible to have a polynomial dependence on the arity k.

▶ **Theorem 3** (Main II)**.** *Let $\mathfrak{I}$ be an instance of MAX k-LIN$_q$ on n variables with alphabet $\mathbb{Z}_q$ and constraints supported on a regular*[5] *collection of tuples $W \subseteq [n]^k$. If W is τ-splittable with $\tau \le \tau_0(k, \delta) := \mathrm{poly}(\delta/k)$, then we can compute an assignment satisfying $\mathsf{OPT} - \delta$ in time $r(q/\tau_0) \cdot \widetilde{O}(|W| + n)$, where $r(x) = \exp(\exp(\mathrm{poly}(x)))$.*

We show that this phenomenon of no dependence of the expansion on the alphabet size q and only polynomial dependence on arity k also occurs for linear equations over a general finite groups $\mathfrak{G}$. Similarly, this leads to better approximation guarantees. To actually implement and obtain this advantage, we will design a new matrix version of the weak regularity decomposition for expanding hypergraphs. Our third result follows.

▶ **Theorem 4** (Main III)**.** *Let $\mathfrak{I}$ be an instance of MAX k-LIN$_{\mathfrak{G}}$ on n variables with alphabet a finite group $\mathfrak{G}$ and constraints supported on a regular collection of tuples $W \subseteq [n]^k$. If W is τ-splittable with $\tau \le \tau_0(k, \delta) := \mathrm{poly}(\delta/k)$, then we can compute an assignment satisfying $\mathsf{OPT} - \delta$ in time $O_{|\mathfrak{G}|,k,\delta}(1) \cdot \mathrm{poly}(|W| + n)$.*

▶ Remark 5. In Theorem 4, we did not attempt to make the running time near-linear in the number of constraints and variables, but it is plausible that it can be done.

We find intriguing this interplay between the type of constraint used in the CSP and the expansion requirement for a given approximation. A natural question is to investigate this interplay for more general constraint types.

In this work, we also investigate how fast we can approximate expanding k-CSPs over q-ary alphabet without making any assumptions on the constraints. We show that k-CSPs can be approximated in near-linear time in the number of constraints and variables, assuming k and q are constants, and provided the constraint hypergraph is sufficiently expanding (splittable). An important caveat of this general case is that the expansion requirements will now depend on both the alphabet size q and arity k in an exponential way (of the form $q^{-O(k)}$).

▶ **Theorem 6.** *Let $\mathfrak{I}$ be an instance of MAX k-CSP on n variables with alphabet $[q]$ and constraints supported on a regular collection of tuples $W \subseteq [n]^k$. If W is τ-splittable with $\tau \le \tau_0(k, q, \delta) := \mathrm{poly}(\delta/(kq^k))$, then we can compute an assignment satisfying $\mathsf{OPT} - \delta$ in time $r(kq/\delta) \cdot \widetilde{O}(|W| + n)$, where $r(x) = \exp(\exp(\exp(\mathrm{poly}(x))))$.*

We obtain the above result via a reduction to the "binary" weak regularity in [24] in a somewhat similar fashion to [10]. Even though it is not hard to make this connection, we think it is worth stating it since this result may be more broadly applicable. Moreover, for fixed arity k and alphabet size q, this improves the running time in the expanding regime of the Sum-of-Squares based algorithm in [2] and also the expanding regime[6] of earlier results 2-CSPs [5, 18, 19, 28].

For comparison, we recall the expanding regime[7] of [2] below.

▶ **Theorem 7** (Sum-of-Squares [2])**.** *Let $\mathfrak{I}$ be an instance of MAX k-CSP on n variables with alphabet $[q]$ and constraints supported $W \subseteq [n]^k$. If W is τ-splittable with $\tau \le \tau_0(k, q, \delta) := \mathrm{poly}(\delta/k) \cdot q^{-k}$, then we can compute an assignment satisfying $\mathsf{OPT} - \delta$ in time $n^{\mathrm{poly}(1/\tau_0)}$.*

▶ Remark 8. In the new theorem above, we do not attempt to optimize the function $r(x)$.

[5] This is an analog to tuples of a graph being d-vertex regular.

[6] We point out these approaches also consider when the expansion is defective (low threshold rank case). Since we are interested in near-linear running time, we need to focus on the expanding case.

[7] Using the improved analysis of swap walks by Dikstein and Dinur [7].

Related Work. As we mentioned above, our work is an extension of the *binary* framework of [24]. This framework was designed for approximating expanding k-XOR and to give a near-liner time decoding algorithm for the explicit binary codes of Ta-Shma [31], near the GV bound. The first polynomial time decoder for these codes was given in [23] using the Sum-of-Squares semi-definite programming hierarchy and its running time, albeit polynomial, is very far from near-linear in the blocklength.

AG codes are widely used in the study of explicit constructions over constant q-ary alphabets. Some of these constructions achieve very competitive parameter trade-offs (e.g., rate versus distance) if not the best known in several cases. However, explicit and efficiently decodable codes near GV bound for large distances,i.e., $1 - 1/q - \varepsilon$, and constant alphabet size were not known prior to this work. In fact, the first explicit construction only appeared in the breakthrough work of [31] for binary codes using more combinatorial expander based techniques. This absence of explicit construction near the GV bound in this regime means that much is yet to be discovered about this case. We view our near-linear time decoder of prime q-ary codes in this regime as not only reaching previously unattained parameter regimes with an explicit construction, but also offering a more combinatorial perspective among a wealthy of algebraic techniques.

For *non-explicit* families of codes approaching the GV bound, much more is known. Random linear codes achieve this bound, but their decoding is believed to be computationally hard. It is possible to construct more structured ensembles of random codes that allow for efficient decoding in this regime. We have the non-explicit classical Goppa codes. Another important technique is based on Thommesen's [32] technique of concatenation with random inner codes. These Thommesen based ensembles can sometimes approach the GV bound and also allow for efficient decoding [16, 14, 21, 25] and even near-linear time decoding [21, 25].

More recently, Blanc and Doron [6] used the framework in [24] to decode explicit binary codes near the GV bound with improved parameters, where they obtain a polynomial improvement on the $o(1)$ error term of the rate $\Omega(\varepsilon^{2+o(1)})$ (the α in Theorem 1) and also put forward some interesting conjectures towards further improving the rate. It is plausible that their improvement also applies here for q-ary alphabets.

In the constant alphabet case, a different parameter regime that has received much attention is the near-capacity regime [15, 20, 21, 25] of list decoding from radius $1 - r - \varepsilon$ with rate r for small values of $\varepsilon > 0$. This regime can only occur when the alphabet size q is a function of ε. Note that our near GV bound regime is the opposite, we have a fixed constant q and we can take ε arbitrarily small (smaller than some function of q).

Due to space constraints, most of our proofs only appear in the full version of this paper.

2 Proof Strategy

We will now describe our contributions in more detail. Our algorithmic results will be based on extensions of the *binary* weak regularity framework of [24]. Roughly speaking, this framework being a *"low level"* framework gives fine control over its components leading to a near-linear time decoder for Ta-Shma's codes [31] over $\mathbb{F}_2$. This same low level structure means that extensions may require suitable generalizations in several of these components as well technical work to implement them. The extensions to handle codes over prime q-ary alphabet and a matrix version of weak regularity will be no exception.

First, we will recall the weak regularity decomposition of Frieze and Kannan [10] in a more analytic form [33]. We will also first consider its *existential* form and later discuss its algorithmic form. Our setup will be as follows. Let $W \subseteq [n]^k$ be a collection of tuples endowed with the uniform probability measure μ_k. Suppose that we have a function $g\colon W \to \mathbb{C}$ that we want to approximate using a simpler approximating function, which will be made precise below. Further suppose that the quality of approximation will be measured with respect to correlations with a class of test functions $\mathcal{F}$. Given some desired approximation error $\delta > 0$, the goal will be to find a *"simple"* approximator $h \approx g$ such that

$$\max_{f \in \mathcal{F}} \left| \langle g - h, f \rangle_{\mu_k} \right| \ \le \ \delta \,.$$

As an *existential* result, it is well-known that an h of the form $h = \sum_{\ell=1}^{p} c_\ell \cdot f_\ell$ always exists, where c_ℓ's are scalars and the f_ℓ's are functions belonging to $\mathcal{F}$. Furthermore, the number of test functions p is small being at most[8] $O(1/\delta^2)$. This means that h is indeed *"simple"* since it is the sum of a small number of test functions, so h is almost as complex as the test functions it needs to fool.

To motivate the generalizations in the weak regularity framework, we will start the discussion of the important case of linear equations over $\mathbb{Z}_q$ as a motivating example. As mentioned above, approximating k-LIN over $\mathbb{Z}_q$ will be crucial in the near-linear time decoding algorithm for prime q-ary alphabets. For us, an instance $\mathfrak{I}$ of k-LIN is given by a system of linear equations[9]

$$x_{i_1} + \cdots + x_{i_k} \equiv r_w \pmod q \qquad\qquad \forall\ w = (i_1, \ldots, i_k) \in W, \tag{1}$$

where $(r_w)_{w \in W} \in \mathbb{Z}_q^W$ are given RHS coefficients. We will need to model this problem in a way that is amenable to the weak regularity approach. We will also take advantage of the algebraic structure of the constraints to avoid any dependence of the alphabet size q and to have only a mild dependence on the arity k in the expansion the framework will require from W.

"Global" Approximation of Dirac Delta Functions. An elementary property of Fourier analysis over $\mathbb{Z}_q$ is that the Dirac delta function $x \mapsto \mathbf{1}_{[x=y]}$ admits a simple but extremely handy Fourier decomposition which we now recall. Let $\omega = \exp(2\pi\sqrt{-1}/q)$. Using orthogonality of characters, we have

$$\mathbf{1}_{[x=y]} = \mathop{\mathbb{E}}_{a \in \mathbb{Z}_q} \left[\omega^{a(x-y)} \right] .$$

Suppose we have an assignment $b \in \mathbb{Z}_q^n$ to the variables of our system of linear equations $\mathfrak{I}$. Then, the fraction of satisfied constraints, which we denote by $\mathsf{val}(\mathfrak{I}, b)$ and refer as the value of this assignment, can be expressed as

$$\mathsf{val}(\mathfrak{I}, b) := \mathop{\mathbb{E}}_{w=(i_1,\ldots,i_k)\sim\mu_k} \left[\mathbf{1}_{[b_{i_1} + \cdots + b_{i_k} \equiv r_w]} \right] = \mathop{\mathbb{E}}_{w=(i_1,\ldots,i_k)\sim\mu_k} \left[\mathop{\mathbb{E}}_{a \in \mathbb{Z}_q} \left[\omega^{a(b_{i_1} + \cdots + b_{i_k} - r_w)} \right] \right] .$$

This suggests defining q functions one for each $a \in \mathbb{Z}_q$ of the form $g_a\colon W \to \mathbb{C}$ as $g_a(w) := \omega^{a \cdot b_w}$, the *"harmonic"* components. We also endow the space $\mathbb{C}^W$ with the inner product defined by the measure μ_k on W. We will need some additional notation. For $b \in \mathbb{Z}_q^n$, we define the function $\chi_{b,a}$ on $[n]$ as $\chi_{b,a}(i) = \omega^{a \cdot b_i}$. We can now rexpress $\mathsf{val}(\mathfrak{I}, b)$ in terms of its harmonic components as

[8] The ℓ_1-norm of the coefficients is "small", i.e., $\sum_{\ell=1}^{p} |c_\ell|$.

[9] The coefficients of the variables are always taken to be 1 here.

$$\begin{aligned}\mathsf{val}(\mathfrak{I}, b) &= \mathop{\mathbb{E}}_{w=(i_1,\ldots,i_k)\sim\mu_k}\left[\mathop{\mathbb{E}}_{a\in\mathbb{Z}_q}\left[\omega^{a(b_{i_1}+\cdots+b_{i_k}-r_w)}\right]\right] \\ &= \mathop{\mathbb{E}}_{a\in\mathbb{Z}_q}\left[\mathop{\mathbb{E}}_{w=(i_1,\ldots,i_k)\sim\mu_k}\left[\omega^{-a\cdot r_w}\cdot\omega^{a(b_{i_1}+\cdots+b_{i_k})}\right]\right] \\ &= \mathop{\mathbb{E}}_{a\in\mathbb{Z}_q}\left[\left\langle g_a, \underbrace{\chi_{b,a}\otimes\cdots\otimes\chi_{b,a}}_{k}\right\rangle_{\mu_k}\right] \\ &= \mathop{\mathbb{E}}_{a\in\mathbb{Z}_q}\left[\left\langle g_a, (\chi_{b,a})^{\otimes k}\right\rangle_{\mu_k}\right].\end{aligned}$$

We can now try to further approximate each g_a using a simpler function h_a that behaves similarly to g_a with respect to functions of the form $f_{b,a} = \chi_{b,a} \otimes \cdots \otimes \chi_{b,a}$ as in the inner product above. We can view functions of form $f_{b,a}$ as tests with respect to which g_a and its simpler approximator have similar correlations. This means that we can model the problem in way amenable to the existential weak regularity framework. For each a, we will consider a (slightly) more general class of test functions $\mathrm{CUT}^{\otimes k}_{\omega,q,a}$ defined as follows

$$\mathrm{CUT}^{\otimes k}_{\omega,q,a} \coloneqq \{\chi_{b^{(1)},a}\otimes\cdots\otimes\chi_{b^{(k)},a} \mid b^{(1)},\ldots,b^{(k)} \subseteq \mathbb{Z}_q^n\}.$$

A simple yet useful remark is that if we can find a decomposition fooling a larger class of test functions, this would suffice since, in particular, it fools the initial class of test.

Suppose that for some $\delta \in (0,1)$ we can find a δ-approximation $h_a = \sum_{\ell=1}^{p_a} c_{a,\ell}\cdot\chi_{b^{(a,\ell,1)},a}\otimes\cdots\otimes\chi_{b^{(a,\ell,k)},a}$ to g_a with respect to a class of test functions, i.e.,

$$\max_{f\in\mathrm{CUT}^{\otimes k}_{\omega,q,a}} \left|\langle g_a - h_a, f\rangle_{\mu_k}\right| \le \delta.$$

By replacing g_a with h_a in the computation of $\mathsf{val}(\mathfrak{I}, b)$ above, we obtain[10]

$$\mathsf{val}(\mathfrak{I}, b) = \mathop{\mathbb{E}}_{a\in\mathbb{Z}_q}\left[\left\langle g_a, (\chi_{b,a})^{\otimes k}\right\rangle\right] = \mathop{\mathbb{E}}_{a\in\mathbb{Z}_q}\left[\left\langle h_a, (\chi_{b,a})^{\otimes k}\right\rangle\right] \pm \delta.$$

We will explain how to algorithmically find h_a in near-linear time later. Now, we will argue why having access to weak regularity decomposition greatly simplifies our task of approximating $\mathsf{val}(\mathfrak{I}, b)$ and also later while decoding q-ary codes.

We can simplify the above equation for $\mathsf{val}(\mathfrak{I}, b)$ even further using the assumed expansion (splittability) of W. A suitable version of the expander mixing lemma allows us to pass from the measure μ_k to the product measure $\mu_1^{\otimes k}$, where μ_1 is the uniform measure on $[n]$. More precisely, we can show that if W is sufficiently expanding (depending on δ), then

$$\begin{aligned}\mathsf{val}(\mathfrak{I}, b) = \mathop{\mathbb{E}}_{a\in\mathbb{Z}_q}\left[\left\langle h_a, (\chi_{b,a})^{\otimes k}\right\rangle_{\mu_k}\right] \pm \delta &= \mathop{\mathbb{E}}_{a\in\mathbb{Z}_q}\left[\left\langle h_a, (\chi_{b,a})^{\otimes k}\right\rangle_{\mu_1^{\otimes k}}\right] \pm 2\delta \\ &= \mathop{\mathbb{E}}_{a\in\mathbb{Z}_q}\left[\sum_{\ell=1}^{p_a} c_{a,\ell}\cdot\prod_{j=1}^{k}\left\langle\chi_{b^{(a,\ell,1)},a}, \chi_{b,a}\right\rangle_{\mu_1}\right] \pm 2\delta.\end{aligned}$$

The *low complexity* of the approximator h_a will allow us to simplify the search for an approximately optimal assignment $b \in \mathbb{Z}_q^n$. The expression above reveals that we only need to know the values of

[10] For scalars x, y (real or complex) and real $\delta \in \mathbb{R}^+$, we use the notation $x = y \pm \delta$ if $|x - y| \le \delta$.

$$\left\{ \left\langle \chi_{b^{(a,\ell,j)},a}, \chi_{b,a} \right\rangle_{\mu_1} \right\}_{a \in \mathbb{Z}_q, \ell \in [p_a], j \in [k]} .$$

Luckily, algorithmically, there will be only $O(qk^3/\delta^2)$ such numbers (no dependence on n and only slightly more than the $O(qk/\delta^2)$ from the existential result). Using brute-force search, it is possible to find sufficiently fine and (close to valid) approximations for these numbers.

To make the entire process efficient and near-linear time we still need to say how to find the functions h_a's in near-linear time. As in [24], we will reduce the problem of finding a weak regularity decomposition with respect to a class of k-tensors, in this case the class $\mathrm{CUT}^{\otimes k}_{\omega,q,a}$, to multiple applications of the 2-tensor case (in a sparse regime). To execute this process in near-linear, we will again use the expansion of W to conveniently move to easier to handle product measures (as above). This involves finding a constant factor approximation for the following expression

$$\max_{x,y \in \mathbb{Z}_q^n} \left| \sum_{i,j=1}^{n} \mathsf{A}_{i,j} \cdot \omega^{a \cdot x_i} \cdot \omega^{a \cdot y_j} \right| , \tag{2}$$

This kind of optimization is known as the *Grothendieck problem* and, in this case, it is for roots of unity going beyond the ±1 case of Alon and Naor [3]. In [29], So, Zhang and Ye considered a more restricted version of this problem (with positive semi-definite matrices) known as the *little* Grothendieck problem. We will extend their analysis to the Grothendieck problem building on some ingredients present in their proof. In our application, the matrices A will be sparse with $m \approx n$ non-zero entries and to achieve a near-linear time we will need to find an (additive) approximation to the Grothendieck problem in time $\widetilde{O}(m)$ of Equation (2). This can be done using the fast SDP solver of Arora and Kale [4].

We now explain how the above weak regularity decomposition can be used in decoding of the expander based construction of Ta-Shma's codes [31]. We will see that the decoding problem can be naturally phrased as a k-LIN instance over $\mathbb{Z}_q$, which is a natural q-ary extension of the k-XOR over $\mathbb{Z}_2$ from [1, 23, 24]. First, we briefly describe Ta-Shma's code construction over alphabet $\mathbb{F}_q$, with q prime, as analyzed[11] in [22]. The idea is to start with a good base code $\mathcal{C}_0 \subseteq \mathbb{F}_q^n$ and to use a carefully constructed collection of tuples $W \subseteq [n]^k$ to amplify its distance via the direct-sum encoding. For any $z \in \mathbb{F}_q^n$, recall that its direct-sum encoding is a new word denoted $y = \mathrm{dsum}_W(z)$ in $\mathbb{F}_q^W$ and defined as

$$y_{(i_1,\ldots,i_k)} \;=\; z_{i_1} + \cdots + z_{i_k} \pmod q \qquad\qquad \forall\ (i_1,\ldots,i_k) \in W\,.$$

The direct-sum code $\mathcal{C} = \mathrm{dsum}_W(\mathcal{C}_0)$ is defined as $\mathcal{C} = \{\mathrm{dsum}_W(z) \mid z \in \mathcal{C}_0\}$. Note the similarity of the above equation and the system of linear equations from Equation (1). In the decoding task, we are given a (possibly) corrupted version of $\widetilde{y}$ of some codeword $y = \mathrm{dsum}_W(z) \in \mathcal{C}$, with $z \in \mathcal{C}_0$. We can view $\widetilde{y}$ as defining the RHS coefficients of an instance of k-LIN, namely, $r_W = \widetilde{y}_w$.

Having an instance of k-LIN over $\mathbb{Z}_q$, we can now use weak regularity as described above. For each $a \in \mathbb{Z}_q$, let g_a be the *harmonic* component associated with RHS vector $\widetilde{y}$ (as above). Similarly, we find a weak regularity approximation h_a for each function g_a.

[11] In [22], they considered the more general (scalar) Abelian case.

If the distance $\Delta(\widetilde{y}, \mathrm{dsum}_W(z)) \le (1 - 1/q)(1 - \beta)$ is not too large, we will be able to deduce that some harmonic function h_a "captures" the structure of the codeword z in the following sense. Set $\mathcal{R} = \{\omega^{a \cdot a'} \mid a' \in \mathbb{Z}_q\}$ and let $f_1, \dots, f_r \colon [n] \to \mathcal{R}$ be the functions appearing in the decomposition of h_a. For each tuple $(y_1, \dots, y_r) \in \mathcal{R}^r$, we can consider the set

$$\{x \in [n] \mid f_1(x) = y_1, \dots, f_r(x) = y_r\}.$$

These sets partition[12] the space $[n]$, and we can show that z is approximately constant in most of these parts. In this sense, the low complexity structure of h_a captures the structure of the codeword z. In this last argument, we use that assumption that q is prime[13].

The case of k-LIN over a finite group will also allow for a weak regularity decomposition in a similar spirit as above, where scalar Fourier characters are replaced by larger dimensional representations and "global" approximation of Dirac delta functions are performed. Extending the weak regularity framework to this case will require considering matrix valued functions. The way we model this case is done in the full version and it uses very elementary properties of representation theory. This case again exhibits an interesting interplay between the type of constraints and the requirement on expansion. (The reader who is only interested in decoding can safely ignore this extension and focus on the $\mathbb{Z}_q$ case.)

3 Constraint Types and Alphabets

We explore the role of different types of constraints and corresponding alphabets going beyond the binary k-XOR considered in [24]. For the special case of linear equations over $\mathbb{Z}_q$ or over an arbitrary finite group $\mathfrak{G}$, we will explore the special structure of the constraints and obtain results with improved parameters.

3.1 General CSPs via the Binary Regularity

We will prove our first result for approximating a general expanding k-CSPs over a q-ary alphabet in near-linear time. We obtain this result using the *binary* near-linear time weak regularity decomposition from [24] in a similar way that Frieze and Kannan modeled k-CSPs [10] using regularity. We formalize this (relatively simple) connection since we believe this result may be of independent interest and may find applications elsewhere. Moreover, it also improves the running time of [2] to near-linear time, for fixed k and q, while offering a different approach to approximating general expanding k-CSPs which could be simpler than their Sum-of-Squares based algorithm. We now restate and proceed to prove this result.

▶ **Theorem 6.** *Let $\mathfrak{I}$ be an instance of MAX k-CSP on n variables with alphabet $[q]$ and constraints supported on a regular collection of tuples $W \subseteq [n]^k$. If W is τ-splittable with $\tau \le \tau_0(k, q, \delta) \coloneqq \mathrm{poly}(\delta/(kq^k))$, then we can compute an assignment satisfying $\mathsf{OPT} - \delta$ in time $r(kq/\delta) \cdot \widetilde{O}(|W| + n)$, where $r(x) = \exp(\exp(\exp(\mathrm{poly}(x))))$.*

We will find a weak regularity decomposition with respect to $0/1$ valued test functions $\mathcal{F} = \mathrm{CUT}^{\otimes k}$ where

$$\mathrm{CUT}^{\otimes k} \coloneqq \{\pm \mathbf{1}_{S_1} \otimes \cdots \otimes \mathbf{1}_{S_k} \mid S_1, \dots, S_k \subseteq [n]\}.$$

The near-linear weak regularity decomposition of [24], which we recall below, can handle this class of functions.

[12] Possibly with empty parts.

[13] So that all non-trivial roots of unity are primitive roots. It is plausible that this restriction is not necessary.

▶ **Theorem 9** (Efficient Weak Regularity from [24]). *Let $W \subseteq [n]^k$ be a τ-splittable collection of tuples. Suppose $\mathcal{F}$ is one of $\mathrm{CUT}^{\otimes k}$, $\mathrm{CUT}_{\pm}^{\otimes k}$. Let $\mathcal{R}$ be the domain of the functions in $\mathcal{F}$, when $k = 1$. Let $g \in \mathcal{R}^{W[1]^k}$ be supported on W with $\|g\|_{\mu_k} \leq 1$. For every $\delta > 0$, if $\tau \leq \delta^2/(k^3 \cdot 2^{20})$, then we can find $h = \sum_{\ell=1}^{p} c_\ell \cdot f_\ell$ with $p = O(k^2/\delta^2)$, $c_1, \dots, c_p \in \mathbb{R}$ and functions $f_1, \dots, f_p \in \mathcal{F}$, such that $\|h\|_{\mu_1^{\otimes k}} \leq 2$, $\sum_{\ell=1}^{p} |c_\ell| = O(k/\delta)$ and h is a good approximator to g in the following sense*

$$\max_{f \in \mathcal{F}} \quad \left| \left\langle g - \left(\frac{d}{n}\right)^{k-1} h, f \right\rangle \right| \;\leq\; \delta \cdot |W| \,,$$

where the inner product is over the counting measure on $W[1]^k$. Furthermore, h can be found in $\widetilde{O}(2^{2^{\widetilde{O}(k^2/\delta^2)}} \cdot |W|)$ time.

Having access to a weak regularity decomposition as above makes the task of approximating the value of a CSP instance relatively simple, as we now describe. This is a common feature of weak regularity based arguments,e.g., [10, 28]. Here, we consider both arbitrary arity k and arbitrary alphabet size q.

We will first need some notation. Let $\alpha \in [q]^k$ and define $W_\alpha = \{w \in W \;\mid\; P_w(\alpha) = 1\}$ to be the set of tuples whose predicates P_w are satisfied by on the input α. Let $\mathcal{A}(\mathfrak{I}) = \{\alpha \in [q]^k \;\mid\; W_\alpha \neq \emptyset\}$ be the set of satisfying inputs of at least one predicate of $\mathfrak{I}$.

We will use the following claim which relates the value of an assignment to the structure of the weak regularity decomposition.

▷ Claim 10. Suppose that for every $\alpha \in [q]^k$, we have a weak regularity decomposition h_α, from Theorem 9, of the indicator $\mathbf{1}_{W(\alpha)}$ with error parameter $\delta > 0$ and with respect to the test class $\mathrm{CUT}^{\otimes k}$. Let $b \in [q]^n$ (viewed as an assignment), which induces a partition $T_1 \sqcup \cdots \sqcup T_q$ of $[n]$. Then,

$$\mathsf{val}(\mathfrak{I}, b) \;=\; \sum_{\alpha \in \mathcal{A}} \sum_{\ell=1}^{p_\alpha} c_{\alpha,\ell} \frac{\left|S_1^{\alpha,\ell} \cap T_{\alpha_1}\right|}{n} \cdots \frac{\left|S_k^{\alpha,\ell} \cap T_{\alpha_k}\right|}{n} \;\;\pm\;\; \delta \cdot |\mathcal{A}(\mathfrak{I})| \;.$$

Proof. Let $\mathcal{A} = \mathcal{A}(\mathfrak{I})$. The value of this assignment is

$$\begin{aligned}
\mathsf{val}(\mathfrak{I}, b) \;&=\; \sum_{\alpha \in \mathcal{A}} \left\langle \mathbf{1}_{W_\alpha}, \mathbf{1}_{T_{\alpha_1}} \otimes \cdots \otimes \mathbf{1}_{T_{\alpha_k}} \right\rangle_{\mu_k} \\
&=\; \frac{1}{|W|} \sum_{\alpha \in \mathcal{A}} \left\langle \left(\frac{d}{n}\right)^{k-1} h_\alpha, \mathbf{1}_{T_{\alpha_1}} \otimes \cdots \otimes \mathbf{1}_{T_{\alpha_k}} \right\rangle \;\pm\; \delta \cdot |\mathcal{A}| \\
&=\; \frac{1}{|W|} \sum_{\alpha \in \mathcal{A}} \left\langle \left(\frac{d}{n}\right)^{k-1} \sum_{\ell=1}^{p_\alpha} c_{\alpha,\ell} \cdot \mathbf{1}_{S_1^{\alpha,\ell}} \otimes \cdots \otimes \mathbf{1}_{S_k^{\alpha,\ell}}, \mathbf{1}_{T_{\alpha_1}} \otimes \cdots \otimes \mathbf{1}_{T_{\alpha_k}} \right\rangle \;\pm\; \delta \cdot |\mathcal{A}| \\
&=\; \frac{1}{n^k} \sum_{\alpha \in \mathcal{A}} \sum_{\ell=1}^{p_\alpha} c_{\alpha,\ell} \cdot \left\langle \mathbf{1}_{S_1^{\alpha,\ell}} \otimes \cdots \otimes \mathbf{1}_{S_k^{\alpha,\ell}}, \mathbf{1}_{T_{\alpha_1}} \otimes \cdots \otimes \mathbf{1}_{T_{\alpha_k}} \right\rangle \;\pm\; \delta \cdot |\mathcal{A}| \\
&=\; \sum_{\alpha \in \mathcal{A}} \sum_{\ell=1}^{p_\alpha} c_{\alpha,\ell} \cdot \left\langle \mathbf{1}_{S_1^{\alpha,\ell}}, \mathbf{1}_{T_{\alpha_1}} \right\rangle_{\mu_1} \cdots \left\langle \mathbf{1}_{S_k^{\alpha,\ell}}, \mathbf{1}_{T_{\alpha_k}} \right\rangle_{\mu_1} \;\pm\; \delta \cdot |\mathcal{A}| \\
&=\; \sum_{\alpha \in \mathcal{A}} \sum_{\ell=1}^{p_\alpha} c_{\alpha,\ell} \frac{\left|S_1^{\alpha,\ell} \cap T_{\alpha_1}\right|}{n} \cdots \frac{\left|S_k^{\alpha,\ell} \cap T_{\alpha_k}\right|}{n} \;\pm\; \delta \cdot |\mathcal{A}| \,,
\end{aligned}$$

concluding the proof. ◁

Proof of Theorem 6. Let $\mathfrak{I}$ be an instance of a k-CSP over alphabet $[q]$ supported on a collection of tuples $W \subseteq [n]^k$ and with predicates $(P_w \colon [q]^k \to \{0,1\})_{w \in W}$.

For each $\alpha \in \mathcal{A}(\mathfrak{I})$, we apply the weak regularity decomposition of Theorem 9 to the function $\mathbf{1}_{W_\alpha}$ with error parameter $\delta > 0$ and test class $\mathcal{F} = \mathrm{CUT}^{\otimes k}$. This gives an approximation $h_\alpha = \sum_{\ell=1}^{p_\alpha} c_{\alpha,\ell} \cdot \mathbf{1}_{S_1^{\alpha,\ell}} \otimes \cdots \otimes \mathbf{1}_{S_k^{\alpha,\ell}}$.

A crucial property is that instead of having to know an assignment $b \in [q]^n$, represented as a partition $T_1 \sqcup \cdots \sqcup T_q = [n]$, it is enough to know the values of the following inner products

$$\left\{ \left\langle \mathbf{1}_{S_j^{\alpha,\ell}}, \mathbf{1}_{T_{\alpha_j}} \right\rangle_{\mu_1} \right\}_{\alpha \in \mathcal{A}(\mathfrak{I}), \ell \in [p_\alpha], j \in [k]}$$

The decomposition is *low complexity*, in the sense that there are only a few of these values. However, we cannot take arbitrary values for these inner products since they may be far from *realizable*, i.e., no true assignment $b \in [q]^n$ can give rise to these values even approximately. From the inner products above, we can extract the following class of functions

$$\mathcal{F}' = \left\{ \mathbf{1}_{S_j^{\alpha,\ell}} \right\}_{\alpha \in \mathcal{A}(\mathfrak{I}), \ell \in [p_\alpha], j \in [k]},$$

whose size $r = |\mathcal{F}'| = O(|\mathcal{A}(\mathfrak{I})|\, k^3/\delta^2)$ is independent from n.

Using Claim 10, to be able to approximate $\mathsf{val}(\mathfrak{I}, b)$ within error $\delta' > 0$ we need to choose the error of the weak regularity decomposition[14] to be $\delta = \delta'/(2\,|\mathcal{A}(\mathfrak{I})|)$ In this case, we have $r = O(|\mathcal{A}(\mathfrak{I})|^2\, k^3/(\delta')^2) = O(q^{2k} k^3/(\delta')^2)$ and the τ-splittability parameter of W needs to satisfy $\tau \leq \mathrm{poly}(\delta'/(kq^k))$.

For convenience, label the functions of $\mathcal{F}'$ as $f_1, \dots, f_r$. Their range is the (simple) binary set $\mathcal{R} = \{0,1\}$. We will consider the factor $\mathcal{B}$ defined by the collection $\mathcal{F}'$, which, roughly speaking, is a partition of $[n]$ according to the values of these functions. More precisely, for every tuple $(y_1, \dots, y_r) \in \mathcal{R}^r$ we have a (possibly empty) part (or atom) of the form

$$\{x \in [n] \;\mid\; f_1(x) = y_1, \dots, f_r(x) = y_r\}.$$

In this case, we have at most $\mathcal{R}^r = 2^r$ atoms in the factor. By definition the functions $\mathcal{F}'$ are constant in each of them. An assignment b gives rise to a distribution on $[q]$ in each atom of the factor. Conversely, any approximate distribution on $[q]$ in each atom approximately corresponds to a realizable assignment b.

Let $L = \sum_{\alpha \in \mathcal{A}(\mathfrak{I}), \ell \in [p_\alpha]} |c_{\alpha,\ell}| \leq |\mathcal{A}(\mathfrak{I})|\, O(k/\delta)$. Set $\eta = \delta/(k \cdot L \cdot q)$. We can η-approximate these distributions in ℓ_1-norm on each atom[15]. The number of approximate distributions can be (crudely) bounded as

$$(1/(\eta q))^{\mathcal{R}^r} \leq \exp(\exp(\exp(\mathrm{poly}(qk/\delta')))).$$

With this fine enough discretization of the distributions on each atom, when computing the expression

$$\mathsf{val}(\mathfrak{I}, b) = \sum_{\alpha \in \mathcal{A}} \sum_{\ell=1}^{p_\alpha} c_{\alpha,\ell} \frac{\left|S_1^{\alpha,\ell} \cap T_{\alpha_1}\right|}{n} \cdots \frac{\left|S_k^{\alpha,\ell} \cap T_{\alpha_k}\right|}{n} \;\pm\; \delta \cdot |\mathcal{A}|$$

we incur an additional error of $\delta'/2$. By our choice of δ, the total approximation error is at most δ'. ◀

[14] We can assume without loss of generality that $\mathcal{A}(\mathfrak{I}) \neq \emptyset$ since otherwise the value of the CSP is always zero.

[15] If the atom is too smaller than $1/(\eta q)$, then we can consider all the possible exact distribution.

3.2 Stating the Extended Weak Regularity Framework

We now show how to obtain our main results for linear equations k-LIN over $\mathbb{Z}_q$ in Theorem 3 and over a finite group $\mathfrak{G}$ in Theorem 4.

In the full version, we see that to approximate k-LIN over $\mathbb{Z}_q$ it suffices to find a good weak regularity decomposition with respect to the test functions $\mathcal{F} = \mathrm{CUT}^{\otimes k}_{\omega,q,a}$ defined as follows

$$\mathrm{CUT}^{\otimes k}_{\omega,q,a} \coloneqq \{\chi_{b_1,a} \otimes \cdots \otimes \chi_{b_k,a} \mid b_1, \ldots, b_k \subseteq \mathbb{Z}_q^n\}.$$

In the full version, we will see that to approximate k-LIN over a finite group, it suffices to find a good weak regularity decomposition with respect to the *matrix* valued test functions $\mathcal{F}$ defined as follows

$$\mathrm{CUT}^{\otimes k}_{\rho} \coloneqq \{\rho_{b_1} \otimes \cdots \otimes \rho_{b_k} \mid b_1, \ldots, b_k \in \mathfrak{G}^n\}.$$

It will be more convenient to enlarge the test class $\mathcal{F}$ to unitary valued functions as follows

$$\mathrm{CUT}^{\otimes k}_{\mathbb{U}_{s,k,\delta}} \coloneqq \{f_1 \otimes \cdots \otimes f_k \mid f_1, \ldots, f_k \colon [n] \to \mathbb{U}_{s,k,\delta}\},$$

where $\mathbb{U}_{s,k,\delta}$ will be a fine enough discretization of the matrices[16] $M_s(\mathbb{C})$ of operator norm at most 1.

We will extend the framework to additionally handle the classes of functions $\mathrm{CUT}^{\otimes k}_{\omega,q,a}$ and $\mathrm{CUT}^{\otimes k}_{\mathbb{U}_{s,k,\delta}}$. This is proven in the full version. Let $\mathbb{K}$ be the underlying field which is either $\mathbb{R}$ or $\mathbb{C}$. Our extended framework gives the following efficient algorithmic result.

▶ **Theorem 11** (Efficient Weak Regularity (Extension of [24])). *Let $W \subseteq [n]^k$ be a τ-splittable collection of tuples. Suppose $\mathcal{F}$ is one of $\mathrm{CUT}^{\otimes k}$, $\mathrm{CUT}^{\otimes k}_{\pm}$, $\mathrm{CUT}^{\otimes k}_{\omega,q,a}$, for $q \geq 3$, or $\mathrm{CUT}^{\otimes k}_{\mathbb{U}_{s,k,\delta}}$. Let $\mathcal{R}$ be the domain of the functions in $\mathcal{F}$, when $k = 1$. Let $g \in \mathcal{R}^{W[1]^k}$ be supported on W with $\|g\|_{\mu_k} \leq 1$. For every $\delta > 0$, if $\tau \leq \delta^2/(k^3 \cdot 2^{20})$, then we can find $h = \sum_{\ell=1}^{p} c_\ell \cdot f_\ell$ with $p = O(k^2/\delta^2)$, scalars $c_1, \ldots, c_p \in \mathbb{K}$ and functions $f_1, \ldots, f_p \in \mathcal{F}$, such that $\|h\|_{\mu_1^{\otimes k}} \leq 2$, $\sum_{\ell=1}^{p} |c_\ell| = O(k/\delta)$ and h is a good approximator to g in the following sense*

$$\max_{f \in \mathcal{F}} \quad \left| \left\langle g - \left(\frac{d}{n}\right)^{k-1} h, f \right\rangle \right| \;\leq\; \delta \cdot |W|,$$

where the inner product is over the counting measure on $W[1]^k$. Furthermore, h can be found in $\widetilde{O}(2^{|\mathcal{R}|^{\widetilde{O}(k^2/\delta^2)}} \cdot |W|)$ time in the scalar valued case and in time $\widetilde{O}_{s,k,\delta}(\mathrm{poly}(|W|))$, otherwise.

3.3 Improved Case: k-LIN over $\mathbb{Z}_q$

The goal of this section is to prove Theorem 3 (restated below) assuming the new extended efficient regularity algorithm from Theorem 11.

▶ **Theorem 3** (Main II). *Let $\mathfrak{I}$ be an instance of MAX k-LIN$_q$ on n variables with alphabet $\mathbb{Z}_q$ and constraints supported on a regular*[17] *collection of tuples $W \subseteq [n]^k$. If W is τ-splittable with $\tau \leq \tau_0(k, \delta) \coloneqq \mathrm{poly}(\delta/k)$, then we can compute an assignment satisfying $\mathsf{OPT} - \delta$ in time $r(q/\tau_0) \cdot \widetilde{O}(|W| + n)$, where $r(x) = \exp(\exp(\mathrm{poly}(x)))$.*

[16] We use $M_s(\mathbb{C})$ for the set of $s \times s$ matrices over $\mathbb{C}$.

[17] This is an analog to tuples of a graph being d-vertex regular.

For k-LIN over alphabet $\mathbb{Z}_q$, we are given a collection of equations (each variable appearing with coefficient one) specified as collection of tuples $W \subseteq [n]^k$ and we are given a collection of corresponding RHS $(r_w)_{w\in W} \in \mathbb{Z}_q^W$. The system of linear equations can be written as follows

$$x_{i_1} + \cdots + x_{i_k} \;=\; r_w \pmod q \qquad\qquad \forall\ w = (i_1, \dots, i_k) \in W.$$

Orthogonality of Fourier characters will be crucially used here.

▶ **Fact 12** (Character Orthogonality). *Let ω be a non-trivial q-th root of unit. Then,*

$$\mathop{\mathbb{E}}_{a\in\mathbb{Z}_q}[\omega^a] = 0\,.$$

Orthogonality allows for a convenient way of implementing the Dirac delta function on (the alphabet) $\mathbb{Z}_q$.

▶ **Fact 13.** *Fix $y \in \mathbb{Z}_q$. The indicator function $x \mapsto \mathbf{1}_{[x=y]}$ on $\mathbb{Z}_q$ can be expressed as*

$$\mathop{\mathbb{E}}_{a\in\mathbb{Z}_q}\left[\omega^{a(x-y)}\right].$$

We now make precise the argument sketched in the proof strategy of Section 2.

Proof of Theorem 3. Let $\mathfrak{I}$ be an instance of k-LIN over $\mathbb{Z}_q$ with constraints supported on $W \subseteq [n]^k$ and RHS values $\{r_w\}_{w\in W}$. For every $a \in \mathbb{Z}_q$, we define $g_a \colon W \to \mathbb{C}$ as the map $w \in W \mapsto \omega^{a\cdot r_w}$, where $\omega = \exp(2\pi\sqrt{-1}/q)$.

Apply the efficient weak regularity decomposition of Theorem 11 to each g_a using error parameter $\delta > 0$ and test functions $\mathcal{F} = \mathrm{CUT}^{\otimes k}_{\omega,q,a}$. Note that this requires the splittability (expansion) parameter τ of W to satisfy $\tau \le O(\delta^2/k^3)$. We obtain a function $h_a = \sum_{\ell=1}^{p_a} c_{a,\ell}\cdot \chi_{b^{(a,\ell,1)},a} \otimes \cdots \otimes \chi_{b^{(a,\ell,k)},a}$, where $b^{(a,\ell,1)},\dots,b^{(a,\ell,k)} \in \mathbb{Z}_q^n$, for every $a \in \mathbb{Z}_q$ and every $\ell \in [p_a]$. Let $b \in \mathbb{Z}_q^n$ be an assignment to the variables of the system of linear equations. The value of this CSP on input b can be computed as

$$\begin{aligned}
\mathsf{val}(\mathfrak{I}, b) = \mathop{\mathbb{E}}_{a\in\mathbb{Z}_q}\left[\langle g_a, \chi_{b,a}\otimes\cdots\otimes\chi_{b,a}\rangle_{\mu_k}\right] &= \mathop{\mathbb{E}}_{a\in\mathbb{Z}_q}\left[\mathop{\mathbb{E}}_{w=(i_1,\dots,i_k)\sim\mu_k}\left[\omega^{-a\cdot r_w}\omega^{a(b_{i_1}+\cdots+b_{i_k})}\right]\right]\\
&= \mathop{\mathbb{E}}_{w=(i_1,\dots,i_k)\sim\mu_k}\left[\mathop{\mathbb{E}}_{a\in\mathbb{Z}_q}\left[\omega^{a(b_{i_1}+\cdots+b_{i_k}-r_w)}\right]\right]\\
&= \mathop{\mathbb{E}}_{w=(i_1,\dots,i_k)\sim\mu_k}\left[\mathbf{1}_{[b_{i_1}+\cdots+b_{i_k}=r_w]}\right].
\end{aligned}$$

Using the weak regularity decomposition h_a of each g_a, we obtain

$$\begin{aligned}
\mathsf{val}(\mathfrak{I}, b) &= \mathop{\mathbb{E}}_{a\in\mathbb{Z}_q}\left[\langle g_a, \chi_{b,a}\otimes\cdots\otimes\chi_{b,a}\rangle_{\mu_k}\right]\\
&= \frac{1}{|W|}\mathop{\mathbb{E}}_{a\in\mathbb{Z}_q}\left[\left\langle\left(\frac{d}{n}\right)^{k-1} h_a, \chi_{b,a}\otimes\cdots\otimes\chi_{b,a}\right\rangle\right] \pm \delta\\
&= \frac{1}{n^k}\mathop{\mathbb{E}}_{a\in\mathbb{Z}_q}\left[\sum_{\ell=1}^{p_a} c_{a,\ell}\cdot\left\langle \chi_{b^{(a,\ell,1)},a}\otimes\cdots\otimes\chi_{b^{(a,\ell,k)},a}, \chi_{b,a}\otimes\cdots\otimes\chi_{b,a}\right\rangle\right] \pm \delta\\
&= \mathop{\mathbb{E}}_{a\in\mathbb{Z}_q}\left[\sum_{\ell=1}^{p_a} c_{a,\ell}\cdot\left\langle \chi_{b^{(a,\ell,1)},a}, \chi_{b,a}\right\rangle_{\mu_1}\cdots\left\langle \chi_{b^{(a,\ell,k)},a}, \chi_{b,a}\right\rangle_{\mu_1}\right] \pm \delta\,,
\end{aligned}$$

concluding the proof.

Now it suffices to approximate the following values

$$\left\{ \left\langle \chi_{b^{(a,\ell,j)},a}, \chi_{b,a} \right\rangle_{\mu_1} \right\}_{a \in \mathbb{Z}_q, \ell \in [p_a], j \in [k]} ,$$

so that there is always a true assignment $b \in [q]^n$ which gives these values.

To this end, we first define the following collection $\mathcal{F}'$ of functions

$$\mathcal{F}' = \left\{ \chi_{b^{(a,\ell,j)},a} \right\}_{a \in \mathbb{Z}_q, \ell \in [p_a], j \in [k]} .$$

Note that $r = |\mathcal{F}'| = O(qk^3/\delta^2)$. The functions above have range $\mathcal{R} = \{\omega^{a'} \mid a' \in \mathbb{Z}_q\}$. They form a factor $\mathcal{B}$ with at most $|\mathcal{R}|^r$ atoms. By the definition of a factor, the functions $\mathcal{F}'$ are constant in each one of them, so to compute $\left\langle \chi_{b^{(a,\ell,j)},a}, \chi_{b,a} \right\rangle_{\mu_1}$ it suffices to know the distribution of symbols of b in each atom.

Let $L = \sum_{a \in \mathbb{Z}_q, \ell \in [p_a]} |c_{a,\ell}| = O(qk/\delta)$ and set $\eta = \delta/(k \cdot L \cdot q)$. The total number of η-approximate distributions in ℓ_1-norm on each atom can be (crudely) bounded as

$$(1/\eta q)^{|\mathcal{R}|^r} \leq \exp(\exp(\mathrm{poly}(qk/\delta))).$$

Using these distributions, we can approximate

$$\mathsf{val}(\mathfrak{I}, b) \;=\; \mathop{\mathbb{E}}_{a \in \mathbb{Z}_q} \left[\sum_{\ell=1}^{p_a} c_{a,\ell} \cdot \left\langle \chi_{b^{(a,\ell,1)},a}, \chi_{b,a} \right\rangle_{\mu_1} \cdots \left\langle \chi_{b^{(a,\ell,k)},a}, \chi_{b,a} \right\rangle_{\mu_1} \right] \;\pm\; \delta ,$$

incurring an additional error of δ. ◀

References

1 Vedat Levi Alev, Fernando Granha Jeronimo, Dylan Quintana, Shashank Srivastava, and Madhur Tulsiani. List decoding of direct sum codes. In *Proceedings of the 31st ACM-SIAM Symposium on Discrete Algorithms*, pages 1412–1425. SIAM, 2020.

2 Vedat Levi Alev, Fernando Granha Jeronimo, and Madhur Tulsiani. Approximating constraint satisfaction problems on high-dimensional expanders. In *Proceedings of the 60th IEEE Symposium on Foundations of Computer Science*, pages 180–201, 2019.

3 Noga Alon and Assaf Naor. Approximating the cut-norm via grothendieck's inequality. In *Proceedings of the 36th ACM Symposium on Theory of Computing*, pages 72–80, 2004.

4 Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. In *Proceedings of the 39th ACM Symposium on Theory of Computing*, STOC '07, pages 227–236, 2007.

5 Boaz Barak, Prasad Raghavendra, and David Steurer. Rounding semidefinite programming hierarchies via global correlation. In *Proceedings of the 52nd IEEE Symposium on Foundations of Computer Science*, pages 472–481, 2011. doi:10.1109/FOCS.2011.95.

6 Guy Blanc and Dean Doron. New near-linear time decodable codes closer to the GV bound. Technical Report TR22-027, Electronic Colloquium on Computational Complexity, 2022.

7 Yotam Dikstein and Irit Dinur. Agreement testing theorems on layered set systems. In *Proceedings of the 60th IEEE Symposium on Foundations of Computer Science*, 2019.

8 Irit Dinur, Prahladh Harsha, Tali Kaufman, Inbal Livni Navon, and Amnon Ta-Shma. List decoding with double samplers. In *Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms*, pages 2134–2153, 2019.

9 Noam D. Elkies. Excellent codes from modular curves. In *Proceedings of the 33rd ACM Symposium on Theory of Computing*, 2001.

10 Uriel Feige and Joe Kilian. Zero knowledge and the chromatic number. *Journal of Computer and System Sciences*, 57(2):187–199, 1998.

11 David Forney. *Concatenated Codes*. PhD thesis, MIT, 1966.

12 A. Frieze and R. Kannan. The regularity lemma and approximation schemes for dense problems. In *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science*, 1996.

13 E.N. Gilbert. A comparison of signalling alphabets. *Bell System Technical Journal*, 31:504–522, 1952.

14 Sivakanth Gopi, Swastik Kopparty, Rafael Oliveira, Noga Ron-Zewi, and Shubhangi Saraf. Locally testable and locally correctable codes approaching the Gilbert-Varshamov bound. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 2073–2091, 2017.

15 Zeyu Guo and Noga Ron-Zewi. Efficient list-decoding with constant alphabet and list sizes. *IEEE Transactions on Information Theory*, 68(3):1663–1682, 2022.

16 Venkatesan Guruswami and Piotr Indyk. Efficiently decodable codes meeting Gilbert-Varshamov bound for low rates. In *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 756–757, 2004.

17 Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. Essential coding theory. Available at `https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/index.html`, 2019.

18 Venkatesan Guruswami and Ali Kemal Sinop. Lasserre hierarchy, higher eigenvalues, and approximation schemes for graph partitioning and quadratic integer programming with psd objectives. In *FOCS*, pages 482–491, 2011.

19 Venkatesan Guruswami and Ali Kemal Sinop. Faster SDP hierarchy solvers for local rounding algorithms. In *Proceedings of the 53rd IEEE Symposium on Foundations of Computer Science*, pages 197–206. IEEE, 2012.

20 Venkatesan Guruswami and Chaoping Xing. List decoding reed-solomon, algebraic-geometric, and gabidulin subcodes up to the singleton bound. In *Proceedings of the 45th ACM Symposium on Theory of Computing*, 2013.

21 B. Hemenway, N. Ron-Zewi, and M. Wootters. Local list recovery of high-rate tensor codes applications. In *Proceedings of the 58th IEEE Symposium on Foundations of Computer Science*, pages 204–215, October 2017.

22 Akhil Jalan and Dana Moshkovitz. Near-optimal Cayley expanders for Abelian groups, 2021. `arXiv:2105.01149`.

23 Fernando Granha Jeronimo, Dylan Quintana, Shashank Srivastava, and Madhur Tulsiani. Unique decoding of explicit ε-balanced codes near the Gilbert–Varshamov bound. In *Proceedings of the 61st IEEE Symposium on Foundations of Computer Science*, 2020.

24 Fernando Granha Jeronimo, Shashank Srivastava, and Madhur Tulsiani. Near-linear time decoding of Ta-Shma's codes via splittable regularity. In *Proceedings of the 52nd ACM Symposium on Theory of Computing*, 2021.

25 Swastik Kopparty, Nicolas Resch, Noga Ron-Zewi, Shubhangi Saraf, and Shashwat Silas. On list recovery of high-rate tensor codes. *IEEE Transactions on Information Theory*, 67(1):296–316, 2021. `doi:10.1109/TIT.2020.3023962`.

26 R. McEliece, E. Rodemich, H. Rumsey, and L. Welch. New upper bounds on the rate of a code via the Delsarte-MacWilliams inequalities. *IEEE Transactions on Information Theory*, 23(2):157–166, 1977.

27 Anand Kumar Narayanan and Matthew Weidner. Subquadratic time encodable codes beating the gilbert–varshamov bound. *IEEE Transactions on Information Theory*, 65(10), 2019.

28 Shayan Oveis Gharan and Luca Trevisan. A new regularity lemma and faster approximation algorithms for low threshold rank graphs. *Theory of Computing*, 11(9):241–256, 2015. `doi:10.4086/toc.2015.v011a009`.

29 Anthony Man-Cho So, Jiawei Zhang, and Yinyu Ye. On approximating complex quadratic optimization problems via semidefinite programming relaxations. *Math. Program.*, 110(1):93–110, June 2007.

30 Henning Stichtenoth. *Algebraic Function Fields and Codes.* Springer Publishing Company, Incorporated, 2nd edition, 2008.

31 Amnon Ta-Shma. Explicit, almost optimal, epsilon-balanced codes. In *Proceedings of the 49th ACM Symposium on Theory of Computing*, STOC 2017, pages 238–251, New York, NY, USA, 2017. ACM.

32 C. Thommesen. The existence of binary linear concatenated codes with Reed- Solomon outer codes which asymptotically meet the Gilbert-Varshamov bound. *IEEE Transactions on Information Theory*, 29(6):850–853, November 1983.

33 L. Trevisan, M. Tulsiani, and S. Vadhan. Boosting, regularity and efficiently simulating every high-entropy distribution. In *Proceedings of the 24th IEEE Conference on Computational Complexity*, 2009.

34 Michael Tsfasman, Serge Vladut, and Dmitry Nogin. *Algebraic Geometric Codes: Basic Notions.* American Mathematical Society, 2007.

35 Michael A. Tsfasman, S. G. Vlădut, and Thomas Zink. Modular curves, shimura curves, and goppa codes, better than varshamov-gilbert bound. *Mathematische Nachrichten*, 109:21–28, 1982.

36 R.R. Varshamov. Estimate of the number of signals in error correcting codes. *Doklady Akademii Nauk SSSR*, 117:739–741, 1957.

Directed Poincaré Inequalities and L^1 Monotonicity Testing of Lipschitz Functions

Renato Ferreira Pinto Jr.
University of Waterloo, Canada

Abstract

We study the connection between directed isoperimetric inequalities and monotonicity testing. In recent years, this connection has unlocked breakthroughs for testing monotonicity of functions defined on discrete domains. Inspired the rich history of isoperimetric inequalities in continuous settings, we propose that studying the relationship between directed isoperimetry and monotonicity in such settings is essential for understanding the full scope of this connection.

Hence, we ask whether directed isoperimetric inequalities hold for functions $f : [0,1]^n \to \mathbb{R}$, and whether this question has implications for monotonicity testing. We answer both questions affirmatively. For Lipschitz functions $f : [0,1]^n \to \mathbb{R}$, we show the inequality $d_1^{\mathsf{mono}}(f) \lesssim \mathbb{E}\left[\|\nabla^- f\|_1\right]$, which upper bounds the L^1 distance to monotonicity of f by a measure of its "directed gradient". A key ingredient in our proof is the *monotone rearrangement* of f, which generalizes the classical "sorting operator" to continuous settings. We use this inequality to give an L^1 monotonicity tester for Lipschitz functions $f : [0,1]^n \to \mathbb{R}$, and this framework also implies similar results for testing real-valued functions on the hypergrid.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Lower bounds and information complexity; Mathematics of computing → Mathematical analysis

Keywords and phrases Monotonicity testing, property testing, isoperimetric inequalities, Poincaré inequalities

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.61

Category RANDOM

Related Version *Full Version*: `https://arxiv.org/abs/2307.02193` [24]

Funding *Renato Ferreira Pinto Jr.*: Partly funded by an NSERC Canada Graduate Scholarship Doctoral Award.

Acknowledgements We thank Eric Blais for helpful discussions throughout the course of this project, and for comments and suggestions on preliminary versions of this paper.

1 Introduction

In property testing, algorithms must make a decision about whether a function $f : \Omega \to R$ has some property $\mathcal{P}$, or is *far* (under some distance metric) from having that property, using a small number of queries to f. One of the most well-studied problems in property testing is *monotonicity testing*, the hallmark case being that of testing monotonicity of Boolean functions on the Boolean cube, $f : \{0,1\}^n \to \{0,1\}$. We call f monotone if $f(x) \le f(y)$ whenever $x \preceq y$, i.e. $x_i \le y_i$ for every $i \in [n]$.

A striking trend emerging from this topic of research has been the connection between monotonicity testing and *isoperimetric inequalities*, in particular directed analogues of classical results such as Poincaré and Talagrand inequalities. We preview that the focus of this work is to further explore this connection by establishing directed isoperimetric inequalities for functions $f : [0,1]^n \to \mathbb{R}$ with continuous domain and range, and as an application obtain monotonicity testers in such settings. Before explaining our results, let us briefly summarize the connection between monotonicity testing and directed isoperimetry.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 61; pp. 61:1–61:18
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

For a function $f : \{0,1\}^n \to \mathbb{R}$, let $d_1^{\mathsf{const}}(f)$ denote its L^1 distance to any constant function $g : \{0,1\}^n \to \mathbb{R}$, and for any point x, define its discrete gradient $\nabla f(x) \in \mathbb{R}^n$ by $(\nabla f(x))_i := f(x^{i\to 1}) - f(x^{i\to 0})$ for each $i \in [n]$, where $x^{i\to b}$ denotes the point x with its i-th coordinate set to b. Then the following inequality[1] is usually called the Poincaré inequality on the Boolean cube (see e.g. [31]): for every $f : \{0,1\}^n \to \{0,1\}$,

$$d_1^{\mathsf{const}}(f) \lesssim \mathbb{E}\left[\|\nabla f\|_1\right] . \tag{1}$$

(Here and going forward, we write $f \lesssim g$ to denote that $f \le cg$ for some universal constant c, and similarly for $f \gtrsim g$. We write $f \approx g$ to denote that $f \lesssim g$ and $g \lesssim f$.)

Now, let $d_1^{\mathsf{mono}}(f)$ denote the L^1 distance from f to any monotone function $g : \{0,1\}^n \to \mathbb{R}$, and for each point x let $\nabla^- f(x)$, which we call the *directed gradient* of f, be given by $\nabla^- f(x) := \min\{\nabla f(x), 0\}$. Then [17] were the first to notice that the main ingredient of the work of [27], who gave a monotonicity tester for Boolean functions on the Boolean cube with query complexity $O(n/\epsilon)$, was the following "directed analogue" of (1)[2]: for every $f : \{0,1\}^n \to \{0,1\}$,

$$d_1^{\mathsf{mono}}(f) \lesssim \mathbb{E}\left[\|\nabla^- f\|_1\right] . \tag{2}$$

The tester of [27] is the "edge tester", which samples edges of the Boolean cube uniformly at random and rejects if any sampled edge violates monotonicity. Inequality (2) shows that, if f is far from monotone, then many edges are violating, so the tester stands good chance of finding one.

In their breakthrough work, [17] gave the first monotonicity tester with $o(n)$ query complexity by showing a directed analogue of Margulis's inequality. This was improved by [20], and eventually the seminal paper of [30] resolved the problem of (nonadaptive) monotonicity testing of Boolean functions on the Boolean cube, up to polylogarithmic factors, by giving a tester with query complexity $\widetilde{O}(\sqrt{n}/\epsilon^2)$. The key ingredient was to show a directed analogue of *Talagrand's inequality.* Talagrand's inequality gives that, for every $f : \{0,1\}^n \to \{0,1\}$,

$$d_1^{\mathsf{const}}(f) \lesssim \mathbb{E}\left[\|\nabla f\|_2\right] .$$

Compared to (1), this replaces the ℓ^1-norm of the gradient with its ℓ^2-norm. [30] showed the natural directed analogue[3] up to polylogarithmic factors, which were later removed by [32]: for every $f : \{0,1\}^n \to \{0,1\}$,

$$d_1^{\mathsf{mono}}(f) \lesssim \mathbb{E}\left[\|\nabla^- f\|_2\right] .$$

Since then, directed isoperimetric inequalities have also unlocked results in monotonicity testing of Boolean functions on the hypergrid [7, 5, 16, 6] (see also [8, 28]) and real-valued functions on the Boolean cube [9].

Our discussion so far has focused on isoperimetric (*Poincaré-type*) inequalities on *discrete* domains. On the other hand, a rich history in geometry and functional analysis, originated in continuous settings, has established an array of isoperimetric inequalities for functions

[1] The left-hand side is usually written $\mathrm{Var}[f]$ instead; for Boolean functions, the two quantities are equivalent up to a constant factor, and writing $d_1^{\mathsf{const}}(f)$ is more consistent with the rest of our presentation.

[2] Typically the left-hand side would be the distance to a *Boolean* monotone function, rather than any real-valued monotone function, but the two quantities are equal; this may be seen via a maximum matching of violating pairs of f, see [26].

[3] In fact, they require a *robust* version of this inequality, but we omit that discussion for simplicity.

defined on continuous domains, as well as an impressive range of connections to topics such as partial differential equations [33], Markov diffusion processes [1], probability theory and concentration of measure [12], optimal transport [15], polynomial approximation [35], among others.

As a motivating starting point, we note that for suitably smooth (Lipschitz) functions $f : [0,1]^n \to \mathbb{R}$, an L^1 Poincaré-type inequality holds [13]:

$$d_1^{\mathsf{const}}(f) \lesssim \mathbb{E}\left[\|\nabla f\|_2\right] . \tag{3}$$

Thus, understanding the full scope of the connection between classical isoperimetric inequalities, their directed counterparts, and monotonicity seems to suggest the study of the continuous setting. In this work, we ask: do *directed* Poincaré-type inequalities hold for functions f with continuous domain and range? And if so, do such inequalities have any implications for monotonicity testing? We answer both questions affirmatively: Lipschitz functions $f : [0,1]^n \to \mathbb{R}$ admit a directed L^1 Poincaré-type inequality (Theorem 1.2), and this inequality implies an upper bound on the query complexity of testing monotonicity of such functions with respect to the L^1 distance (Theorem 1.4). (We view L^1 as the natural distance metric for the continuous setting; see Section 1.3 for a discussion.) This framework also yields results for L^1 testing monotonicity of real-valued functions on the hypergrid $f : [m]^n \to \mathbb{R}$. Our testers are *partial derivative testers*, which naturally generalize the classical *edge testers* [27, 18] to continuous domains.

We now introduce our model, and then summarize our results.

1.1 L^p-testing

Let (Ω, Σ, μ) be a probability space (typically for us, the unit cube or hypergrid with associated uniform probability distribution). Let $R \subseteq \mathbb{R}$ be a range, and $\mathcal{P}$ a property of functions $g : \Omega \to R$. Given a function $f : \Omega \to \mathbb{R}$, we denote the L^p distance of f to property $\mathcal{P}$ by $d_p(f, \mathcal{P}) := \inf_{g \in \mathcal{P}} d_p(f, g)$, where $d_p(f, g) := \mathop{\mathbb{E}}_{x \sim \mu}\left[|f(x) - g(x)|^p\right]^{1/p}$. For fixed domain Ω, we write $d_p^{\mathsf{const}}(f)$ for the L^p distance of f to the property of constant functions, and $d_p^{\mathsf{mono}}(f)$ for the L^p distance of f to the property of monotone functions. (See Definition 2.2 for a formal definition contemplating e.g. the required measurability and integrability assumptions.)

▶ **Definition 1.1** (L^p-testers). *Let $p \geq 1$. For probability space (Ω, Σ, μ), range $R \subseteq \mathbb{R}$, property $\mathcal{P} \subseteq L^p(\Omega, \mu)$ of functions $g : \Omega \to R$, and proximity parameter $\epsilon > 0$, we say that randomized algorithm A is an L^p*-tester for $\mathcal{P}$ *with query complexity q if, given* oracle access *to an unknown input function $f : \Omega \to R \in L^p(\Omega, \mu)$, A makes at most q oracle queries and 1) accepts with probability at least $2/3$ if $f \in \mathcal{P}$; 2) rejects with probability at least $2/3$ if $d_p(f, \mathcal{P}) > \epsilon$.*

We say that A has *one-sided error* if it accepts functions $f \in \mathcal{P}$ with probability 1, otherwise we say it has *two-sided error*. It is *nonadaptive* if it decides all of its queries in advance (i.e. before seeing output from the oracle), and otherwise it is *adaptive*. We consider two types of oracle:

Value oracle: Given point $x \in \Omega$, this oracle outputs the value $f(x)$.

Directional derivative oracle: Given point $x \in \Omega$ and vector $v \in \mathbb{R}^n$, this oracle outputs the derivative of f along v at point x, given by $\frac{\partial f}{\partial v}(x) = v \cdot \nabla f(x)$, as long as f is differentiable at x. Otherwise, it outputs a special symbol $\perp$.

A directional derivative oracle is weaker than a full first-order oracle, which would return the entire gradient [14], and it seems to us like a reasonable model for the high-dimensional setting; for example, obtaining the full gradient costs n queries, rather than a single query. This type of oracle has also been studied in optimization research, e.g. see [21]. For our applications, only the *sign* of the result will matter, in which case we remark that, for sufficiently smooth functions (say, functions with bounded second derivatives) each directional derivative query may be simulated using two value queries on sufficiently close together points.

Our definition (with value oracle) coincides with that of [4] when the range is $R = [0, 1]$. On the other hand, for general R, we keep the distance metric unmodified, whereas [4] normalize it by the magnitude of R. Intuitively, we seek testers that are efficient even when f may take large values as the dimension n grows; see Section 1.3.3 for more details.

1.2 Results and main ideas

1.2.1 Directed Poincaré-type inequalities

Our first result is a directed Poincaré inequality for Lipschitz functions $f : [0, 1]^n \to \mathbb{R}$, which may be seen as the continuous analogue of inequality (2) of [27].

▶ **Theorem 1.2.** *Let $f : [0, 1]^n \to \mathbb{R}$ be a Lipschitz function with monotone rearrangement f^*. Then*

$$d_1^{\mathsf{mono}}(f) \approx \mathbb{E}\left[|f - f^*|\right] \lesssim \mathbb{E}\left[\|\nabla^- f\|_1\right] . \tag{4}$$

As hinted in the statement, a crucial tool for this result is the *monotone rearrangement* f^* of f. We construct f^* by a sequence of axis-aligned rearrangements $R_1, \dots, R_n$; each R_i is the *non-symmetric monotone rearrangement* operator along dimension i, which naturally generalizes the *sorting* operator of [27] to the continuous case. For each coordinate $i \in [n]$, the operator R_i takes f into an equimeasurable function $R_i f$ that is monotone in the i-th coordinate, at a "cost" $\mathbb{E}\left[|f - R_i f|\right]$ that is upper bounded by $\mathbb{E}\left[|\partial_i^- f|\right]$, where $\partial_i^- f := (\nabla^- f)_i$ is the directed partial derivative along the i-th coordinate. We show that each application R_i can only decrease the "cost" associated with further applications R_j, so that the total cost of obtaining f^* (i.e. the LHS of (4)) may be upper bounded, via the triangle inequality, by the sum of all directed partial derivatives, i.e. the RHS of (4).

A technically simpler version of this argument also yields a directed Poincaré inequality for real-valued functions on the hypergrid. We also note that Theorems 1.2 and 1.3 are both tight up to constant factors.

▶ **Theorem 1.3.** *Let $f : [m]^n \to \mathbb{R}$ and let f^* be its monotone rearrangement. Then*

$$d_1^{\mathsf{mono}}(f) \approx \mathbb{E}\left[|f - f^*|\right] \lesssim m\mathbb{E}\left[\|\nabla^- f\|_1\right] .$$

Table 1 places our results in the context of existing classical and directed inequalities. In that table and going forward, for any $p, q \geq 1$ we call the inequalities

$$d_p^{\mathsf{const}}(f)^p \lesssim \mathbb{E}\left[\|\nabla f\|_q^p\right] \qquad \text{and} \qquad d_p^{\mathsf{mono}}(f)^p \lesssim \mathbb{E}\left[\|\nabla^- f\|_q^p\right]$$

a *classical* and *directed* (L^p, ℓ^q)*-Poincaré inequality*, respectively. Note that the L^p notation refers to the space in which we take norms, while ℓ^q refers to the geometry in which we measure gradients. In this paper, we focus on the L^1 inequalities.

Table 1 Classical and directed Poincaré-type inequalities on discrete and continuous domains. Cells marked with * indicate inequalities that follow from another entry in the table.

Setting Inequality		Discrete		Continuous
		$\{0,1\}^n \to \{0,1\}$	$\{0,1\}^n \to \mathbb{R}$	$[0,1]^n \to \mathbb{R}$
(L^1, ℓ^1)-Poincaré	$d_1^{\mathsf{const}}(f) \lesssim \mathbb{E}\left[\|\nabla f\|_1\right]$	* [34]	* [34]	* [13]
	$d_1^{\mathsf{mono}}(f) \lesssim \mathbb{E}\left[\|\nabla^- f\|_1\right]$	[27]	Theorem 1.3	Theorem 1.2
(L^1, ℓ^2)-Poincaré	$d_1^{\mathsf{const}}(f) \lesssim \mathbb{E}\left[\|\nabla f\|_2\right]$	* [34]	[34]	[13]
	$d_1^{\mathsf{mono}}(f) \lesssim \mathbb{E}\left[\|\nabla^- f\|_2\right]$	[30]	?	Conjecture 1.8

We also note that we have ignored in our discussion the issues of *robust* inequalities, which seem essential for some of the testing applications (see [30]), and the distinction between *inner* and *outer boundary*, whereby some inequalities on Boolean f may be made stronger by setting $\nabla f(x) = 0$ when $f(x) = 0$ (see e.g. [34]). We refer the reader to the original works for the strongest version of each inequality and a detailed treatment of these issues.

1.2.2 Testing monotonicity on the unit cube and hypergrid

Equipped with the results above, we give a monotonicity tester for Lipschitz functions $f : [0,1]^n \to \mathbb{R}$, and the same technique yields a tester for functions on the hypergrid as well. The testers are parameterized by an upper bound L on the best Lipschitz constant of f in ℓ^1 geometry, which we denote $\mathsf{Lip}_1(f)$ (see Definition 2.1 for a formal definition).

Both of our testers are *partial derivative testers.* These are algorithms which only have access to a directional derivative oracle and, moreover, their queries are promised to be axis-aligned vectors. In the discrete case, these are usually called *edge testers* [27, 18].

▶ **Theorem 1.4.** *There is a nonadaptive partial derivative L^1 monotonicity tester for Lipschitz functions $f : [0,1]^n \to \mathbb{R}$ satisfying $\mathsf{Lip}_1(f) \leq L$ with query complexity $O\left(\frac{nL}{\epsilon}\right)$ and one-sided error.*

Similarly, there is a nonadaptive partial derivative L^1 monotonicity tester for functions $f : [m]^n$ satisfying $\mathsf{Lip}_1(f) \leq L$ with query complexity $O\left(\frac{nmL}{\epsilon}\right)$ and one-sided error.

The testers work by sampling points x and coordinates $i \in [n]$ uniformly at random, and using directional derivative queries to reject if $\partial_i^- f(x) < 0$. Their correctness is shown using Theorems 1.2 and 1.3, which imply that, when f is ϵ-far from monotone in L^1-distance, the total magnitude of its negative partial derivatives must be large – and since each partial derivative is at most L by assumption, the values $\partial_i^- f(x)$ must be strictly negative in a set of large measure, which the tester stands good chance of hitting with the given query complexity.

1.2.3 Testing monotonicity on the line

The results above, linking a Poincaré-type inequality with a monotonicity tester that uses partial derivative queries and has linear dependence on n, seem to suggest a close parallel with the case of the edge tester on the Boolean cube [27, 18]. On the other hand, we also show a strong separation between Hamming and L^1 testing. Focusing on the simpler problem of monotonicity testing *on the line*, we show that the tight query complexity of L^1 monotonicity testing Lipschitz functions grows with the square root of the size of the (continuous or discrete) domain:

▶ **Theorem 1.5.** *There exist nonadaptive L^1 monotonicity testers for Lipschitz functions $f : [0, m] \to \mathbb{R}$ and $f : [m] \to \mathbb{R}$ satisfying $\mathsf{Lip}_1(f) \le L$ with query complexity $\widetilde{O}\left(\sqrt{mL/\epsilon}\right)$. The testers use value queries and have one-sided error.*

This result (along with the near-tight lower bounds in Section 1.2.4) is in contrast with the case of Hamming testing functions $f : [m] \to \mathbb{R}$, which has sample complexity $\Theta(\log m)$ [23, 25, 11, 2]. Intuitively, this difference arises because a Lipschitz function may violate monotonicity with rate of change L, so the area under the curve may grow quadratically on violating regions. The proof is in fact a reduction to the Hamming case, using the Lipschitz assumption to establish a connection between the L^1 and Hamming distances to monotonicity.

1.2.4 Lower bounds

We give two types of lower bounds: under no assumptions about the tester and for constant n, we show that the dependence of Theorem 1.4 on L/ϵ is close to optimal[4]. We give stronger bounds for the special case of partial derivative testers (such as the ones from Theorem 1.4), essentially showing that our analysis of the partial derivative tester is tight.

▶ **Theorem 1.6.** *Let n be a constant. Any L^1 monotonicity tester (with two-sided error, and adaptive value and directional derivative queries) for Lipschitz functions $f : [0,1]^n \to \mathbb{R}$ satisfying $\mathsf{Lip}_1(f) \le L$ requires at least $\Omega\left((L/\epsilon)^{\frac{n}{n+1}}\right)$ queries.*

Similarly, any L^1 monotonicity tester (with two-sided error and adaptive queries) for functions $f : [m]^n \to \mathbb{R}$ satisfying $\mathsf{Lip}_1(f) \le L$ requires at least $\Omega\left(\min\left\{(mL/\epsilon)^{\frac{n}{n+1}}, m^n\right\}\right)$ queries.

Notice that the bounds above cannot be improved beyond logarithmic factors, due to the upper bounds for the line in Theorem 1.5. It also follows that adaptivity (essentially) does not help with L^1 monotonicity testing on the line, matching the situation for Hamming testing [25, 19, 2].

Theorem 1.6 is obtained via a “hole” construction, which hides a non-monotone region of f inside an ℓ^1-ball B of radius r. We choose r such the violations of monotonicity inside B are large enough to make f ϵ-far from monotone, but at the same time, the ball B is hard to find using few queries. However, this construction has poor dependence on n.

To lower bound the query complexity of partial derivative testers with better dependence on n, we employ a simpler “step” construction, which essentially chooses a coordinate i and hides a small negative-slope region on every line along coordinate i. These functions are far from monotone, but a partial derivative tester must correctly guess both i and the negative-slope region to detect them. We conclude that Theorem 1.4 is optimal for partial derivative testers on the unit cube, and optimal for edge testers on the hypergrid for constant ϵ and L:

▶ **Theorem 1.7.** *Any partial derivative L^1 monotonicity tester for Lipschitz functions $f : [0,1]^n \to \mathbb{R}$ satisfying $\mathsf{Lip}_1(f) \le L$ (with two-sided error and adaptive queries) requires at least $\Omega(nL/\epsilon)$ queries.*

[4] Note that one may always multiply the input values by $1/L$ to reduce the problem to the case with Lipschitz constant 1 and proximity parameter ϵ/L, so this is the right ratio to look at.

Table 2 Query complexity bounds for testing monotonicity on the unit cube and hypergrid. Upper bounds are for nonadaptive (n.a.) algorithms with one-sided error (1-s.), and lower bounds are for adaptive algorithms with two-sided error, unless stated otherwise. For L^1-testing, the upper bounds derived from prior works (*) are specialized to the Lipschitz case by us; see the full version of the paper for details. Our lower bounds hold either for constant (const.) n, or for partial derivative testers (p.d.t.).

Domain	**Hamming testing** $f : \Omega \to \mathbb{R}$	L^1**-testing** (prior works) $f : \Omega \to \mathbb{R}$, $\mathsf{Lip}_1(f) \leq L$	L^1**-testing** (this work) $f : \Omega \to \mathbb{R}$, $\mathsf{Lip}_1(f) \leq L$
$\Omega = [0,1]^n$	Infeasible	$\widetilde{O}\left(\frac{n^2 L}{\epsilon}\right)$ (*) [4]	$O\left(\frac{nL}{\epsilon}\right)$ p.d.t.
		–	$\Omega\left(\left(\frac{L}{\epsilon}\right)^{\frac{n}{n+1}}\right)$ const. n $\Omega\left(\frac{nL}{\epsilon}\right)$ p.d.t.
$\Omega = [m]^n$	$O\left(\frac{n \log m}{\epsilon}\right)$ [18]	$\widetilde{O}\left(\frac{n^2 mL}{\epsilon}\right)$ (*) [4]	$O\left(\frac{nmL}{\epsilon}\right)$ p.d.t.
	$\Omega\left(\frac{n \log(m) - \log(1/\epsilon)}{\epsilon}\right)$ [19]	$\widetilde{\Omega}\left(\frac{L}{\epsilon}\right)$ n.a. 1-s. [4] $\Omega(n \log m)$ n.a. [11]	$\Omega\left(\left(\frac{mL}{\epsilon}\right)^{\frac{n}{n+1}}\right)$ const. n $\Omega(nm)$ p.d.t.

For sufficiently small constant ϵ and constant L, any partial derivative L^1 monotonicity tester for functions $f : [m]^n \to \mathbb{R}$ satisfying $\mathsf{Lip}_1(f) \leq L$ *(with two-sided error and adaptive queries) requires at least $\Omega(nm)$ queries.*

Table 2 summarizes our upper and lower bounds for testing monotonicity on the unit cube and hypergrid, along with the analogous Hamming testing results for intuition and bounds for L^1 testing from prior works. See Section 1.3.3 and the full version of the paper for a discussion and details of how prior works imply the results in that table, since to our knowledge the problem of L^1 monotonicity testing parameterized by the Lipschitz constant has not been explicitly studied before.

1.3 Discussion and open questions

1.3.1 Stronger directed Poincaré inequalities?

Classical Poincaré inequalities are usually of the ℓ^2 form, which seems natural e.g. due to basis independence. On the other hand, in the directed setting, the weaker ℓ^1 inequalities (as in [27] and Theorems 1.2 and 1.3) have more straightforward proofs than ℓ^2 counterparts such as [30]. A perhaps related observation is that monotonicity is *not* a basis-independent concept, since it is defined in terms of the standard basis. It is not obvious whether directed ℓ^2 inequalities ought to hold in every (real-valued, continuous) setting. Nevertheless, in light of the parallels and context established thus far, we are hopeful that such an equality does hold. Otherwise, we believe that the reason should be illuminating. For now, we conjecture:

▶ **Conjecture 1.8.** *For every Lipschitz function $f : [0,1]^n \to \mathbb{R}$, it holds that*

$$d_1^{\mathsf{mono}}(f) \lesssim \mathbb{E}\left[\|\nabla^- f\|_2\right] .$$

Accordingly, we also ask whether an L^1 tester with $O(\sqrt{n})$ complexity exists, presumably with a dependence on the $\mathsf{Lip}_2(f)$ constant rather than $\mathsf{Lip}_1(f)$ since ℓ^2 is the relevant geometry above.

1.3.2 Query complexity bounds

Our lower bounds either have weak dependence on n, or only apply to a specific family of algorithms (partial derivative testers). Previous works have established tester-independent lower bounds with strong dependence on n by using reductions from communication complexity [10, 11], whose translation to the continuous setting is not obvious[5], by reduction to comparison-based testers [19], whose connection to L^1 testing setting seems less immediate, or directly via a careful construction [2]. We believe that finding strong tester-independent lower bounds for L^1 testing Lipschitz functions on the unit cube is an interesting direction for further study.

We also remark that even a tight lower bound matching Theorem 1.4 may not rule out testers with better dependence on n if, for example, such a tester were parameterized by $\mathsf{Lip}_2(f)$, which can be a factor of $\sqrt{n}$ larger than $\mathsf{Lip}_1(f)$. We view the possibility of better testers on the unit cube, or otherwise a conceptual separation with [30], as an exciting direction for future work.

1.3.3 Relation to prior work on L^p-testing

[4] initiated the systematic study of L^p-testing and, most relevant to the present work, established the first (and, to our knowledge, only) results on L^p testing of the monotonicity property, on the hypergrid and on the discrete line. While our models are broadly compatible, a subtle but crucial distinction must be explained.

[4] focused their exposition on the case of functions $f : \Omega \to [0, 1]$, and in this regime, L^1 testing can only be easier than Hamming testing, which they show via a reduction based on Boolean threshold functions. On the other hand, for functions with other ranges, say $f : \Omega \to [a, b]$, their definition normalizes the notion of distance by a factor of $\frac{1}{b-a}$. In our terminology, letting $r := b - a$ and $g := f/r$, it follows that $d_1(g) = d_1(f)/r$, so testing f with proximity parameter ϵ reduces to testing g with proximity parameter ϵ/r. For Hamming testers with query complexity that depends linearly on $1/\epsilon$, this amounts to paying a factor of r in the reduction to the Boolean case[6]. This loss is indeed necessary, because by the same reasoning, testing g with proximity parameter ϵ reduces to testing f with proximity parameter $r\epsilon$. Therefore the problems of testing f with proximity parameter ϵ and testing f/r with proximity parameter ϵ/r have the same query complexity.

In this work, we do not normalize the distance metric by r; we would like to handle functions f that may take large values as the dimension n grows, as long as f satisfies a Lipschitz assumption, and our goal is to beat the query complexity afforded by the reduction to the Boolean case. We derive these benchmarks by assuming that the input f is Lipschitz, and inferring an upper bound on r based on the Lipschitz constant and the size of the domain. Combined with the hypergrid tester of [4] and a discretization argument for the unit cube inspired by [8, 28], we establish benchmarks for our testing problem. See the full version of the paper for further details.

With the discussion above in mind, it is instructive to return to Table 2. We note that our upper bounds have polynomially smaller dependence on n than the benchmarks, suggesting that our use of the Lipschitz assumption – via the directed Poincaré inequalities

[5] Note that there is no obvious reduction from testing on the hypergrid to testing on the unit cube – one idea is to simulate the unit cube tester on a multilinear interpolation of the function defined on the hypergrid, but the challenge is that simulating each query to the unit cube naively requires an exponential number of queries to the hypergrid.

[6] This factor can also be tracked explicitly in the characterization of the L^1 distance to monotonicity of [4]: it arises in Lemmas 2.1 and 2.2, where an integral from 0 to 1 must be changed to an integral from a to b, so the best threshold function is only guaranteed to be ϵ/r-far from monotone.

in Theorems 1.2 and 1.3 – exploits useful structure underlying the monotonicity testing problem (whereas the benchmark testers must work for every function with bounded range, not only the Lipschitz ones). Our lower bounds introduce an almost-linear dependence on the hypergrid length m; intuitively, this dependence is not implied by the previous bounds in [4, 11] because those construct the violations of monotonicity via Boolean functions, whereas our constructions exploit the fact that a Lipschitz function can "keep growing" along a given direction, which exacerbates the L^1 distance to monotonicity in the region where that happens. Our lower bounds for partial derivative testers show that the analysis of our algorithms is essentially tight, so new (upper or lower bound) ideas are required to establish the optimal query complexity for arbitrary testers.

On the choice of L^1 distance and Lipschitz assumption

We briefly motivate our choice of distance metric and Lipschitz assumption. For continuous range and domain, well-known counterexamples rule out testing with respect to Hamming distance: given any tester with finite query complexity, a monotone function may be made far from monotone by arbitrarily small, hard to detect perturbations. Testing against L^1 distance is then a natural choice, since this metric takes into account the magnitude of the change required to make a function monotone ([4] also discuss connections with learning and approximation theory). However, an arbitrarily small region of the input may still have disproportionate effect on the L^1 distance if the function is arbitrary, so again testing is infeasible. Lipschitz continuity seems like a natural enough assumption which, combined with the choice of L^1 distance, makes the problem tractable. Another benefit is that Lipschitz functions are differentiable almost everywhere by Rademacher's theorem, so the gradient is well-defined almost everywhere, which enables the connection with Poincaré-type inequalities.

Organization

Section 2 introduces definitions and notation. In Section 3 we prove our directed Poincaré inequality on the unit cube, and in Section 4 we give our L^1 monotonicity tester for this domain. The analogous versions for the discrete case of the hypergrid, as well as the proofs of our results for testing on the line (Section 1.2.3) and lower bounds (Section 1.2.4) may be found in the full version of the paper.

2 Preliminaries

For integer $m \geq 1$, we write $[m]$ to denote the set $\{i \in \mathbb{Z} : 1 \leq i \leq m\}$. For any $c \in \mathbb{R}$, we write c^+ for $\max\{0, c\}$ and c^- for $-\min\{0, c\}$. We denote the closure of an open set $B \subset \mathbb{R}^n$ by $\overline{B}$.

For a measure space (Ω, Σ, ν) and measurable function $f : \Omega \to \mathbb{R}$, we write $\int_\Omega f \, \mathrm{d}\nu$ for the Lebesgue integral of f over this space. Then for $p \geq 1$, the space $L_p(\Omega, \nu)$ is the set of measurable functions f such that $|f|^p$ is Lebesgue integrable, i.e. $\int_\Omega |f|^p \, \mathrm{d}\nu < \infty$, and we write the L^p norm of such functions as $\|f\|_{L^p} = \|f\|_{L^p(\nu)} = \left(\int_\Omega |f|^p \, \mathrm{d}\nu\right)^{1/p}$. We will write ν to denote the Lebesgue measure on $\mathbb{R}^n$ (the dimension n being clear from context), and simply write $L^p(\Omega)$ for $L^p(\Omega, \nu)$; we will reserve μ for the special case of probability measures.

2.1 Lipschitz functions and L^p distance

We first define Lipschitz functions with respect to a choice of ℓ^p geometry.

▶ **Definition 2.1.** *Let $p \geq 1$. We say that $f : \Omega \to \mathbb{R}$ is* (ℓ^p, L)-Lipschitz *if, for every $x, y \in \Omega$, $|f(x) - f(y)| \leq L\|x - y\|_p$. We say that f is Lipschitz if it is (ℓ^p, L)-Lipschitz for any L (in which case this also holds for any other choice of ℓ^q), and in this case we denote by $\mathsf{Lip}_p(f)$ the best possible Lipschitz constant:*

$$\mathsf{Lip}_p(f) := \inf_L \{f \text{ is } (\ell^p, L)\text{-Lipschitz}\} .$$

It follows that $\mathsf{Lip}_p(f) \leq \mathsf{Lip}_q(f)$ for $p \leq q$.

We now formally define L^p distances, completing the definition of L^p-testers from Section 1.1.

▶ **Definition 2.2** (L^p-distance)**.** *Let $p \geq 1$, let $R \subseteq \mathbb{R}$, and let (Ω, Σ, μ) be a probability space. For a property $\mathcal{P} \subseteq L^p(\Omega, \mu)$ of functions $g : \Omega \to R$ and function $f : \Omega \to R \in L^p(\Omega, \mu)$, we define the distance from f to $\mathcal{P}$ as $d_p(f, \mathcal{P}) := \inf_{g \in \mathcal{P}} d_p(f, g)$, where*

$$d_p(f, g) := \|f - g\|_{L^p(\mu)} = \mathop{\mathbb{E}}_{x \sim \mu} [|f(x) - g(x)|^p]^{1/p} .$$

For $p = 0$, we slightly abuse notation and, taking $0^0 = 0$, write $d_0(f, g)$ for the Hamming distance between f and g weighted by μ (and $\mathcal{P}$ may be any set of measurable functions on (Ω, Σ, μ)).

In our applications, we will always take μ to be the uniform distribution over Ω[7]. As a shorthand, when (Ω, Σ, μ) is understood from the context and $R = \mathbb{R}$, we will write

1. $d_p^{\mathsf{const}}(f) := d_p(f, \mathcal{P}^{\mathsf{const}})$ where $\mathcal{P}^{\mathsf{const}} := \{f : \Omega \to \mathbb{R} \in L^p(\Omega, \mu) : f = c, c \in \mathbb{R}\}$; and
2. $d_p^{\mathsf{mono}}(f) := d_p(f, \mathcal{P}^{\mathsf{mono}})$ where $\mathcal{P}^{\mathsf{mono}} := \{f : \Omega \to \mathbb{R} \in L^p(\Omega, \mu) : f \text{ is monotone}\}$.

Going forward, we will also use the shorthand $d_p(f) := d_p^{\mathsf{mono}}(f)$.

2.2 Directed partial derivatives and gradients

Let B be an open subset of $\mathbb{R}^n$, and let $f : B \to \mathbb{R}$ be Lipschitz. Then by Rademacher's theorem f is differentiable almost everywhere in B. For each $x \in B$ where f is differentiable, let $\nabla f(x) = (\partial_1 f(x), \ldots, \partial_n f(x))$ denote its gradient, where $\partial_i f(x)$ is the partial derivative of f along the i-th coordinate at x. Then, let $\partial_i^- := \min\{0, \partial_i\}$, i.e. for every x where f is differentiable we have $\partial_i^- f(x) = -(\partial_i f(x))^-$. We call ∂_i^- the *directed partial derivative* operator in direction i. Then we define the *directed gradient* operator by $\nabla^- := (\partial_1^-, \ldots, \partial_n^-)$, again defined on every point x where f is differentiable.

3 Directed Poincaré inequalities for Lipschitz functions

In this section, we establish Theorem 1.2. We start with the one-dimensional case, i.e. functions on the line, and then generalize to higher dimensions. See the full version of the paper for the discrete case of the hypergrid.

3.1 One-dimensional case

Let $m > 0$, let $I := (0, m)$, and let $f : \overline{I} \to \mathbb{R}$ be a measurable function. We wish to show that $\|f - f^*\|_{L^1} \lesssim m \|\partial^- f\|_{L^1}$, where f^* is the monotone rearrangement of f. We first introduce the monotone rearrangement, and then show this inequality using an elementary calculus argument.

[7] More precisely: when $\Omega = [0, 1]^n$, μ will be the Lebesgue measure on Ω (with associated σ-algebra Σ).

3.1.1 Monotone rearrangement

Here, we introduce the (non-symmetric, non-decreasing) monotone rearrangement of a one-dimensional function. We follow the definition of [29], with the slight modification that we are interested in the *non-decreasing* rearrangement, whereas most of the literature usually favours the non-increasing rearrangement. The difference is purely syntactic, and our choice more conveniently matches the convention in the monotonicity testing literature. Up to this choice, our definition also agrees with that of [3, Chapter 2], and we refer the reader to these two texts for a comprehensive treatment.

We define the (lower) *level sets* of $f : \overline{I} \to \mathbb{R}$ as the sets

$$\overline{I}_c := \left\{x \in \overline{I} : f(x) \leq c\right\}$$

for all $c \in \mathbb{R}$. For nonempty measurable $S \subset \mathbb{R}$ of finite measure, the *rearrangement* of S is the set

$$S^* := [0, \nu(S)]$$

(recall that ν stands for the Lebesgue measure here), and we define $\emptyset^* := \emptyset$. For a level set $\overline{I}_c$, we write $\overline{I}_c^*$ to mean $\left(\overline{I}_c\right)^*$.

▶ **Definition 3.1.** *The* monotone rearrangement *of f is the function $f^* : \overline{I} \to \mathbb{R}$ given by*

$$f^*(x) := \inf \left\{c \in \mathbb{R} : x \in \overline{I}_c^*\right\} . \tag{5}$$

Note that f^* is always a non-decreasing function.

We note two well-known properties of the monotone rearrangement: equimeasurability and order preservation. Two functions f, g are called *equimeasurable* if $\nu\{f \geq c\} = \nu\{g \geq c\}$ for every $c \in \mathbb{R}$. A mapping $u \mapsto u^*$ is called *order preserving* if $f(x) \leq g(x)$ for all $x \in \overline{I}$ implies $f^*(x) \leq g^*(x)$ for all $x \in \overline{I}$. See [3, Chapter 2, Proposition 1.7] for a proof of the following:

▶ **Fact 3.2.** *Let $f : \overline{I} \to \mathbb{R}$ be a measurable function. Then f and f^* are equimeasurable.*

▶ **Fact 3.3.** *The mapping $f \mapsto f^*$ is order preserving.*

3.1.2 Absolutely continuous functions and the one-dimensional Poincaré inequality

Let $f : \overline{I} \to \mathbb{R}$ be absolutely continuous. It follows that f has a derivative ∂f almost everywhere (i.e. outside a set of measure zero), $\partial f \in L^1(I)$ (i.e. its derivative is Lebesgue integrable), and

$$f(x) = f(0) + \int_0^x \partial f(t)\,\mathrm{d}t$$

for all $x \in \overline{I}$. It also follows that $\partial^- f \in L^1(I)$.

We may now show our one-dimensional inequality:

▶ **Lemma 3.4.** *Let $f : \overline{I} \to \mathbb{R}$ be absolutely continuous. Then $\|f - f^*\|_{L^1} \leq 2m \|\partial^- f\|_{L^1}$.*

Proof. Let $S := \{x \in \overline{I} : f^*(x) > f(x)\}$, and note that S is a measurable set because f, f^* are measurable functions (the latter by Fact 3.2). Moreover, since f and f^* are equimeasurable (by the same result), we have $\int f \,\mathrm{d}\nu = \int f^* \,\mathrm{d}\nu$ and therefore

$$\begin{aligned} \|f - f^*\|_{L^1} &= \int_I |f - f^*| \,\mathrm{d}\nu = \int_S (f^* - f)\,\mathrm{d}\nu + \int_{I\setminus S} (f - f^*)\,\mathrm{d}\nu \\ &= \int_S (f^* - f)\,\mathrm{d}\nu + \left(\int_I (f - f^*)\,\mathrm{d}\nu - \int_S (f - f^*)\,\mathrm{d}\nu\right) = 2\int_S (f^* - f)\,\mathrm{d}\nu\,. \end{aligned}$$

Hence our goal is to show that

$$\int_S (f^* - f)\,\mathrm{d}\nu \le m \left\|\partial^- f\right\|_{L^1}\,.$$

Let $x \in \overline{I}$. We claim that there exists $x' \in [0,x]$ such that $f(x') \ge f^*(x)$. Suppose this is not the case. Then since f is continuous on $[0,x]$, by the extreme value theorem it attains its maximum and therefore there exists $c < f^*(x)$ such that $f(y) \le c$ for all $y \in [0,x]$. Thus $[0,x] \subseteq \overline{I}_c$, so $\nu\left(\overline{I}_c\right) \ge x$ and hence $x \in \overline{I}_c^*$. Then, by Definition 3.1, $f^*(x) \le c < f^*(x)$, a contradiction. Thus the claim is proved.

Now, let $x \in S$ and fix some $x' \in [0,x]$ such that $f(x') \ge f^*(x)$. Since f is absolutely continuous, we have

$$f^*(x) - f(x) \le f(x') - f(x) = -\int_{x'}^{x} \partial f(t)\,\mathrm{d}t \le -\int_0^m \partial^- f(t)\,\mathrm{d}t = \left\|\partial^- f\right\|_{L^1}\,.$$

The result follows by applying this estimate to all x:

$$\int_S (f^* - f)\,\mathrm{d}\nu \le \int_S \left\|\partial^- f\right\|_{L^1} \mathrm{d}\nu = \nu(S)\left\|\partial^- f\right\|_{L^1} \le m\left\|\partial^- f\right\|_{L^1}\,. \qquad ◀$$

3.2 Multidimensional case

Although we ultimately only require an inequality on the unit cube $[0,1]^n$, we will first work in slightly more generality and consider functions defined on a *box* in $\mathbb{R}^n$, defined below. This approach makes some of the steps more transparent, and also gives intuition for the discrete case of the hypergrid.

▶ **Definition 3.5.** *Let $a \in \mathbb{R}^n_{>0}$. The* box of size a *is the closure $\overline{B} \subset \mathbb{R}^n$ of $B = (0,a_1) \times \cdots \times (0,a_n)$.*

Going forward, $\overline{B} \subset \mathbb{R}^n$ will always denote such a box.

Notation

For $x \in \mathbb{R}^n$, $y \in \mathbb{R}$ and $i \in [n]$, we will use the notation x^{-i} to denote the vector in $\mathbb{R}^{[n]\setminus\{i\}}$ obtained by removing the i-th coordinate from x (note that the indexing is not changed), and we will write (x^{-i}, y) as a shorthand for the vector $(x_1, \dots, x_{i-1}, y, x_{i+1}, \dots, x_n) \in \mathbb{R}^n$. We will also write x^{-i} directly to denote any vector in $\mathbb{R}^{[n]\setminus\{i\}}$. For function $f : \overline{B} \to \mathbb{R}$ and $x^{-i} \in \mathbb{R}^{[n]\setminus\{i\}}$, we will write $f_{x^{-i}}$ for the function given by $f_{x^{-i}}(y) = f(x^{-i}, y)$ for all $(x^{-i}, y) \in \overline{B}$. For any set $D \in \mathbb{R}^n$, we will denote by D^{-i} the projection $\{x^{-i} : x \in D\}$, and extend this notation in the natural way to more indices, e.g. D^{-i-j}.

▶ **Definition 3.6** (Rearrangement in direction i)**.** *Let $f : \overline{B} \to \mathbb{R}$ be a measurable function and let $i \in [n]$. The* rearrangement of f in direction i *is the function $R_i f : \overline{B} \to \mathbb{R}$ given by*

$$(R_i f)_{x^{-i}} := (f_{x^{-i}})^* \tag{6}$$

for all $x^{-i} \in \left(\overline{B}\right)^{-i}$. We call each R_i the rearrangement operator in direction i.

We may put (6) in words as follows: on each line in direction i determined by point x^{-i}, the restriction of $R_i f$ to that line is the monotone rearrangement of the restriction of f to that line.

▶ **Proposition 3.7.** *Let $\overline{B}$ be the box of size $a \in \mathbb{R}^n$, and let $f : \overline{B} \to \mathbb{R}$ be Lipschitz continuous. Then for each $i \in [n]$,*

$$\left\| f - R_i f \right\|_{L^1} \le 2a_i \left\| \partial_i^- f \right\|_{L^1} .$$

Proof. Since f is Lipschitz continuous, each $f_{x^{-i}} : [0, a_i] \to \mathbb{R}$ is Lipschitz continuous and *a fortiori* absolutely continuous. The result follows from Lemma 3.4, using Tonelli's theorem to choose the order of integration. ◀

A key ingredient in our multidimensional argument is that the rearrangement operator preserves Lipschitz continuity:

▶ **Lemma 3.8** ([29, Lemma 2.12])**.** *If $f : \overline{B} \to \mathbb{R}$ is Lipschitz continuous (with Lipschitz constant L), then $R_i f$ is Lipschitz continuous (with Lipschitz constant $2L$).*

We are now ready to define the (multidimensional) monotone rearrangement f^*:

▶ **Definition 3.9.** *Let $f : \overline{B} \to \mathbb{R}$ be a measurable function. The* monotone rearrangement *of f is the function*

$$f^* := R_n R_{n-1} \cdots R_1 f .$$

We first show that f^* is indeed a monotone function:

▶ **Proposition 3.10.** *Let $f : \overline{B} \to \mathbb{R}$ be Lipschitz continuous. Then f^* is monotone.*

Proof. Say that $g : \overline{B} \to \mathbb{R}$ is *monotone in direction i* if $g_{x^{-i}}$ is non-decreasing for all $x^{-i} \in \left(\overline{B}\right)^{-i}$. Then g is monotone if and only if it is monotone in direction i for every $i \in [n]$. Note that $R_i f$ is monotone in direction i by definition of monotone rearrangement. Therefore, it suffices to prove that if f is monotone in direction j, then $R_i f$ is also monotone in direction j.

Suppose f is monotone in direction j, and suppose $i < j$ without loss of generality. Let $a \in \mathbb{R}^n$ be the size of B. Let $x^{-j} \in \left(\overline{B}\right)^{-j}$ and $0 \le y_1 < y_2 \le a_j$, so that $(x^{-j}, y_1), (x^{-j}, y_2) \in \overline{B}$. We need to show that $(R_i f)(x^{-j}, y_1) \le (R_i f)(x^{-j}, y_2)$. Let $\overline{I}_i := [0, a_i]$. For each $k \in \{1, 2\}$, let $g_k : \overline{I}_i \to \mathbb{R}$ be given by

$$g_k(z) := f(x_1, \dots, x_{i-1}, z, x_{i+1}, \dots, x_{j-1}, y_k, x_{j+1}, \dots, x_n) .$$

Note that

$$g_k^*(z) = (R_i f)(x_1, \dots, x_{i-1}, z, x_{i+1}, \dots, x_{j-1}, y_k, x_{j+1}, \dots, x_n)$$

for every $z \in \overline{I}_i$, and therefore our goal is to show that $g_1^*(x_i) \le g_2^*(x_i)$. But f being monotone in direction j means that $g_1(z) \le g_2(z)$ for all $z \in \overline{I}_i$, so by the order preserving property (Fact 3.3) of the monotone rearrangement we get that $g_1^*(x_i) \le g_2^*(x_i)$, concluding the proof. ◀

It is well-known that the monotone rearrangement is a non-expansive operator. Actually a stronger fact holds, as we note below.

▶ **Proposition 3.11** ([22]). *Let $m > 0$ and let $f, g \in L^1[0,m]$. Then f^*, g^* satisfy*

$$\int_{[0,m]} (f^* - g^*)^- \,\mathrm{d}\nu \leq \int_{[0,m]} (f - g)^- \,\mathrm{d}\nu$$

and

$$\int_{[0,m]} |f^* - g^*| \,\mathrm{d}\nu \leq \int_{[0,m]} |f - g| \,\mathrm{d}\nu \,.$$

The result above is stated for functions on the interval. Taking the integral over the box B and repeating for each operator R_i yields the non-expansiveness of our monotone rearrangement operator, as also noted by [29]:

▶ **Corollary 3.12.** *Let $f, g \in L^1(\overline{B})$. Then $\|f^* - g^*\|_{L^1} \leq \|f - g\|_{L^1}$.*

We show that the rearrangement operator can only make the norm of the directed partial derivatives smaller, i.e. decrease the violations of monotonicity, which is the key step in this proof.

▶ **Proposition 3.13.** *Let $f : \overline{B} \to \mathbb{R}$ be Lipschitz continuous and let $i, j \in [n]$. Then $\left\|\partial_j^-(R_i f)\right\|_{L^1} \leq \left\|\partial_j^- f\right\|_{L^1}$.*

Proof. We may assume that $i \neq j$, since otherwise the LHS is zero. We will use the following convention for variables names: $w \in \mathbb{R}^n$ will denote points in B; $z \in \mathbb{R}^{[n]\setminus\{i,j\}}$ will denote points in B^{-i-j}; $x \in \mathbb{R}$ will denote points in $(0, a_i)$ (indexing the i-th dimension); and $y \in \mathbb{R}$ will denote points in $(0, a_j)$ (indexing the j-th dimension). For each $i \in [n]$, let e_i denote the i-th basis vector.

Since f is Lipschitz, so is $R_i f$ by Lemma 3.8. By Rademacher's theorem, these functions are differentiable almost everywhere. Therefore, let $D \subseteq B$ be a measurable set such that f and $R_i f$ are differentiable in D and $\nu(D) = \nu(B)$. We have

$$\begin{aligned}
\left\|\partial_j^-(R_i f)\right\|_{L^1} &= \int_D \left|\partial_j^-(R_i f)\right| \mathrm{d}\nu = \int_D \left[\lim_{h\to 0}\left(\frac{(R_i f)(w + he_j) - (R_i f)(w)}{h}\right)^-\right] \mathrm{d}\nu(w) \\
&\overset{(BC1)}{=} \lim_{h\to 0}\int_D \left(\frac{(R_i f)(w + he_j) - (R_i f)(w)}{h}\right)^- \mathrm{d}\nu(w) \\
&\overset{(D1)}{=} \lim_{h\to 0}\int_B \left(\frac{(R_i f)(w + he_j) - (R_i f)(w)}{h}\right)^- \mathrm{d}\nu(w) \\
&\overset{(T1)}{=} \lim_{h\to 0}\int_{B^{-i-j}}\int_{(0,a_j)}\int_{(0,a_i)} \left(\frac{(R_i f)(z, y+h, x) - (R_i f)(z, y, x)}{h}\right)^- \mathrm{d}\nu(x)\,\mathrm{d}\nu(y)\,\mathrm{d}\nu(z) \\
&\leq \lim_{h\to 0}\int_{B^{-i-j}}\int_{(0,a_j)}\int_{(0,a_i)} \left(\frac{f(z, y+h, x) - f(z, y, x)}{h}\right)^- \mathrm{d}\nu(x)\,\mathrm{d}\nu(y)\,\mathrm{d}\nu(z) \\
&\overset{(T2)}{=} \lim_{h\to 0}\int_B \left(\frac{f(w + he_j) - f(w)}{h}\right)^- \mathrm{d}\nu(w) \\
&\overset{(D2)}{=} \lim_{h\to 0}\int_D \left(\frac{f(w + he_j) - f(w)}{h}\right)^- \mathrm{d}\nu(w) \\
&\overset{(BC2)}{=} \int_D \left[\lim_{h\to 0}\left(\frac{f(w + he_j) - f(w)}{h}\right)^-\right] \mathrm{d}\nu(w) = \int_D \left|\partial_j^- f\right| \mathrm{d}\nu = \left\|\partial_j^- f\right\|_{L^1} \,.
\end{aligned}$$

Equalities (BC1) and (BC2) hold by the bounded convergence theorem, which applies because the difference quotients are uniformly bounded by the Lipschitz constants of $R_i f$ and f (respectively), and because $R_i f$ and f are differentiable in D (which gives pointwise convergence of the limits). Equalities (D1) and (D2) hold again by the uniform boundedness of the difference quotients, along with the fact that $\nu(B \setminus D) = 0$. Equalities (T1) and (T2) hold by Tonelli's theorem. Finally, the inequality holds by Proposition 3.11, since $(R_i f)(z, y+h, \cdot)$ is the monotone rearrangement of $f(z, y+h, \cdot)$ and $(R_i f)(z, y, \cdot)$ is the monotone rearrangement of $f(z, y, \cdot)$. ◀

We are now ready to prove our directed (L^1, ℓ^1)-Poincaré inequality.

▶ **Theorem 3.14.** *Let B be the box of size $a \in \mathbb{R}^n$ and let $f : \overline{B} \to \mathbb{R}$ be Lipschitz continuous. Then*

$$\|f - f^*\|_{L^1} \le 2 \sum_{i=1}^{n} a_i \left\|\partial_i^- f\right\|_{L^1} .$$

Proof. We have

$$\begin{aligned}
\|f - f^*\|_{L^1} &\le \sum_{i=1}^{n} \|R_{i-1} \cdots R_1 f - R_i \cdots R_1 f\|_{L^1} && \text{(Triangle inequality)} \\
&\le 2 \sum_{i=1}^{n} a_i \left\|\partial_i^- (R_{i-1} \cdots R_1 f)\right\|_{L^1} && \text{(Lemma 3.8 and Proposition 3.7)} \\
&\le 2 \sum_{i=1}^{n} a_i \left\|\partial_i^- f\right\|_{L^1} && \text{(Lemma 3.8 and Proposition 3.13)} .
\end{aligned}$$

◀

Setting $B = (0,1)^n$ yields the inequality portion of Theorem 1.2:

▶ **Corollary 3.15.** *Let $B = (0,1)^n$ and let $f : \overline{B} \to \mathbb{R}$ be Lipschitz continuous. Then*

$$\mathbb{E}\left[|f - f^*|\right] = \|f - f^*\|_{L^1} \le 2 \int_B \|\nabla^- f\|_1 \, \mathrm{d}\nu = 2\mathbb{E}\left[\|\nabla^- f\|_1\right] .$$

To complete the proof of Theorem 1.2, we need to show that $d_1(f) \approx \mathbb{E}\left[|f - f^*|\right]$, i.e. that the monotone rearrangement is "essentially optimal" as a target monotone function for f. The inequality $d_1(f) \le \mathbb{E}\left[|f - f^*|\right]$ is clear from the fact that f^* is monotone. The inequality in the other direction follows from the non-expansiveness of the rearrangement operator, with essentially the same proof as that of [30] for the Boolean cube:

▶ **Proposition 3.16.** *Let $f : [0,1]^n \to \mathbb{R}$ be Lipschitz continuous. Then $\mathbb{E}\left[|f - f^*|\right] \le 2d_1(f)$.*

Proof. Let $g \in L^1([0,1]^n)$ be any monotone function. It follows that $g^* = g$. By Corollary 3.12, we have that $\|f^* - g^*\|_{L^1} \le \|f - g\|_{L^1}$. Using the triangle inequality, we obtain

$$\|f - f^*\|_{L^1} \le \|f - g\|_{L^1} + \|g - f^*\|_{L^1} = \|f - g\|_{L^1} + \|f^* - g^*\|_{L^1} \le 2\|f - g\|_{L^1} .$$

The claim follows by taking the infimum over the choice of g. ◀

To check that Corollary 3.15 is tight up to constant factors, it suffices to take the linear function $f : [0,1]^n \to \mathbb{R}$ given by $f(x) = 1 - x_1$ for all $x \in [0,1]^n$. Then f^* is given by $f^*(x) = x_1$, so $\mathbb{E}\left[f - f^*\right] = 1/2$ while $\mathbb{E}\left[\|\nabla^- f\|_1\right] = 1$, as needed.

Algorithm 1 L^1 monotonicity tester for Lipschitz functions using partial derivative queries.

```
    Input: Partial derivative oracle access to Lipschitz function $f : [0,1]^n \to \mathbb{R}$.
    Output: Accept if $f$ is monotone, reject if $d_1(f) > \epsilon$.
    Requirement: $\mathsf{Lip}_1(f) \le L$.
  procedure PartialDerivativeTester($f, L, \epsilon$)
      repeat $\Theta(nL/\epsilon)$ times
          Sample $x \in [0,1]^n$ uniformly at random.
          Sample $i \in [n]$ uniformly at random.
          Reject if $\partial_i f(x) < 0$.
      end repeat
      Accept.
```

4 Application to monotonicity testing

In this section, we use the directed Poincaré inequality on the unit cube to show that the natural partial derivative tester attains the upper bound from Theorem 1.4. As noted in the introduction, we refer to the full version of the paper for the case of the hypergrid.

The tester is given in Algorithm 1. It is clear that this algorithm is a nonadaptive partial derivative tester, and that it always accepts monotone functions. It suffices to show that it rejects with good probability when $d_1(f) > \epsilon$.

▶ **Lemma 4.1.** *Let $f : [0,1]^n \to \mathbb{R}$ be a Lipschitz function satisfying $\mathsf{Lip}_1(f) \le L$. Suppose $d_1(f) > \epsilon$. Then Algorithm 1 rejects with probability at least $2/3$.*

Proof. Let $D \subseteq [0,1]^n$ be a measurable set such that f is differentiable on D and $\mu(D) = 1$, which exists by Rademacher's theorem. For each $i \in [n]$, let $S_i := \{x \in D : \partial_i f(x) < 0\}$. A standard argument gives that each $S_i \subset \mathbb{R}^n$ is a measurable set. We claim that

$$\sum_{i=1}^{n} \mu(S_i) > \frac{\epsilon}{2L} .$$

Suppose this is not the case. By the Lipschitz continuity of f, we have that $|\partial_i f(x)| \le L$ for every $x \in D$ and $i \in [n]$, and therefore

$$2 \sum_{i=1}^{n} \mathbb{E}\left[\left|\partial_i^- f\right|\right] \le 2L \sum_{i=1}^{n} \mu(S_i) \le \epsilon .$$

On the other hand, the assumption that $d_1(f) > \epsilon$ and Corollary 3.15 yield

$$\epsilon < \mathbb{E}\left[|f - f^*|\right] \le 2\mathbb{E}\left[\|\nabla^- f\|_1\right] = 2 \sum_{i=1}^{n} \mathbb{E}\left[\left|\partial_i^- f\right|\right] ,$$

a contradiction. Therefore the claim holds.

Now, the probability that one iteration of the tester rejects is the probability that $x \in S_i$ when x and i are sampled uniformly at random. This probability is

$$\mathbb{P}\left[\text{Iteration rejects}\right] = \sum_{j=1}^{n} \mathbb{P}_i\left[i = j\right] \mathbb{P}_x\left[x \in S_j\right] = \sum_{j=1}^{n} \frac{1}{n} \cdot \mu(S_j) > \frac{\epsilon}{2nL} .$$

Thus $\Theta\left(\frac{nL}{\epsilon}\right)$ iterations suffice to reject with high constant probability. ◀

References

1 Dominique Bakry, Ivan Gentil, and Michel Ledoux. *Analysis and geometry of Markov diffusion operators*, volume 103. Springer, 2014.

2 Aleksandrs Belovs. Adaptive lower bound for testing monotonicity on the line. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

3 Colin Bennett and Robert C Sharpley. *Interpolation of operators*. Academic press, 1988.

4 Piotr Berman, Sofya Raskhodnikova, and Grigory Yaroslavtsev. L^p-testing. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 164–173, 2014.

5 Hadley Black, Deeparnab Chakrabarty, and C Seshadhri. Directed isoperimetric theorems for boolean functions on the hypergrid and an $\widetilde{O}(n\sqrt{d})$ monotonicity tester. *arXiv preprint*, 2022. arXiv:2211.05281.

6 Hadley Black, Deeparnab Chakrabarty, and C Seshadhri. A $d^{1/2+o(1)}$ monotonicity tester for boolean functions on d-dimensional hypergrids. *arXiv preprint*, 2023. arXiv:2304.01416.

7 Hadley Black, Deeparnab Chakrabarty, and Comandur Seshadhri. A $o(d) \cdot \operatorname{polylog} n$ monotonicity tester for boolean functions over the hypergrid $[n]^d$. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2133–2151. SIAM, 2018.

8 Hadley Black, Deeparnab Chakrabarty, and Comandur Seshadhri. Domain reduction for monotonicity testing: A $o(d)$ tester for boolean functions in d-dimensions. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1975–1994. SIAM, 2020.

9 Hadley Black, Iden Kalemaj, and Sofya Raskhodnikova. Isoperimetric inequalities for real-valued functions with applications to monotonicity testing. *arXiv preprint*, 2020. arXiv:2011.09441.

10 Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. *computational complexity*, 21:311–358, 2012.

11 Eric Blais, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Lower bounds for testing properties of functions over hypergrid domains. In *2014 IEEE 29th Conference on Computational Complexity (CCC)*, pages 309–320. IEEE, 2014.

12 Sergey Bobkov and Michel Ledoux. Poincaré's inequalities and talagrand's concentration phenomenon for the exponential distribution. *Probability Theory and Related Fields*, 107:383–400, 1997.

13 Sergey G Bobkov and Christian Houdré. Isoperimetric constants for product probability measures. *The Annals of Probability*, pages 184–205, 1997.

14 Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

15 Lorenzo Brasco and Filippo Santambrogio. A note on some poincaré inequalities on convex sets by optimal transport methods. In *Geometric Properties for Parabolic and Elliptic PDE's: GPPEPDEs, Palinuro, Italy, May 2015 4*, pages 49–63. Springer, 2016.

16 Mark Braverman, Subhash Khot, Guy Kindler, and Dor Minzer. Improved monotonicity testers via hypercube embeddings. *arXiv preprint*, 2022. arXiv:2211.09229.

17 D. Chakrabarty and C. Seshadhri. An $o(n)$ monotonicity tester for boolean functions over the hypercube. *SIAM Journal on Computing*, 45(2):461–472, January 2016. doi:10.1137/13092770X.

18 Deeparnab Chakrabarty and C. Seshadhri. Optimal bounds for monotonicity and lipschitz testing over hypercubes and hypergrids. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 419–428, 2013.

19 Deeparnab Chakrabarty and C. Seshadhri. An optimal lower bound for monotonicity testing over hypergrids. *Theory Of Computing*, 10(17):453–464, 2014.

20 Xi Chen, Rocco A. Servedio, and Li-Yang Tan. New algorithms and lower bounds for monotonicity testing. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 286–295, October 2014. doi:10.1109/FOCS.2014.38.

21 Shuyu Cheng, Guoqiang Wu, and Jun Zhu. On the convergence of prior-guided zeroth-order optimization algorithms. *Advances in Neural Information Processing Systems*, 34:14620–14631, 2021.

22 Michael G Crandall and Luc Tartar. Some relations between nonexpansive and order preserving mappings. *Proceedings of the American Mathematical Society*, 78(3):385–390, 1980.

23 Funda Ergün, Sampath Kannan, S Ravi Kumar, Ronitt Rubinfeld, and Mahesh Viswanathan. Spot-checkers. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 259–268, 1998.

24 Renato Ferreira Pinto Jr. Directed poincaré inequalities and L^1 monotonicity testing of lipschitz functions. *arXiv preprint*, 2023. `arXiv:2307.02193`.

25 Eldar Fischer. On the strength of comparisons in property testing. *Information and Computation*, 189(1):107–116, 2004.

26 Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 474–483, 2002.

27 Oded Goldreich, Shafi Goldwasser, Eric Lehman, Dana Ron, and Alex Samorodnitsky. Testing monotonicity. *Combinatorica*, 3(20):301–337, 2000.

28 Nathaniel Harms and Yuichi Yoshida. Downsampling for testing and learning in product distributions. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

29 Bernhard Kawohl. *Rearrangements and convexity of level sets in PDE*, volume 1150 of *Lecture Notes in Mathematics*. Springer-Verlag Berlin, Heidelberg, 1985.

30 Subhash Khot, Dor Minzer, and Muli Safra. On monotonicity testing and boolean isoperimetric-type theorems. *SIAM Journal on Computing*, 47(6):2238–2276, 2018.

31 Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.

32 Ramesh Krishnan S Pallavoor, Sofya Raskhodnikova, and Erik Waingarten. Approximating the distance to monotonicity of boolean functions. *Random Structures & Algorithms*, 60(2):233–260, 2022.

33 Henri Poincaré. Sur les équations aux dérivées partielles de la physique mathématique. *American Journal of Mathematics*, pages 211–294, 1890.

34 Michel Talagrand. Isoperimetry, logarithmic sobolev inequalities on the discrete cube, and margulis' graph connectivity theorem. *Geometric & Functional Analysis GAFA*, 3(3):295–314, 1993.

35 Rüdiger Verfürth. A note on polynomial approximation in sobolev spaces. *ESAIM: Mathematical Modelling and Numerical Analysis*, 33(4):715–719, 1999.

Bias Reduction for Sum Estimation

Talya Eden
Bar-Ilan University, Ramat Gan, Israel

Jakob Bæk Tejs Houen
BARC, University of Copenhagen, Denmark

Shyam Narayanan
MIT, Cambridge, MA, USA

Will Rosenbaum
Amherst College, MA, USA

Jakub Tětek
BARC, University of Copenhagen, Denmark

Abstract

In classical statistics and distribution testing, it is often assumed that elements can be sampled exactly from some distribution $\mathcal{P}$, and that when an element x is sampled, the probability $\mathcal{P}(x)$ of sampling x is also known. In this setting, recent work in distribution testing has shown that many algorithms are robust in the sense that they still produce correct output if the elements are drawn from any distribution $\mathcal{Q}$ that is sufficiently close to $\mathcal{P}$. This phenomenon raises interesting questions: under what conditions is a "noisy" distribution $\mathcal{Q}$ sufficient, and what is the algorithmic cost of coping with this noise?

In this paper, we investigate these questions for the problem of estimating the sum of a multiset of N real values $x_1, \ldots, x_N$. This problem is well-studied in the statistical literature in the case $\mathcal{P} = \mathcal{Q}$, where the Hansen-Hurwitz estimator [Annals of Mathematical Statistics, 1943] is frequently used. We assume that for some (known) distribution $\mathcal{P}$, values are sampled from a distribution $\mathcal{Q}$ that is pointwise close to $\mathcal{P}$. That is, there is a parameter $\gamma < 1$ such that for all x_i, $(1-\gamma)\mathcal{P}(i) \leq \mathcal{Q}(i) \leq (1+\gamma)\mathcal{P}(i)$. For every positive integer k we define an estimator ζ_k for $\mu = \sum_i x_i$ whose bias is proportional to γ^k (where our ζ_1 reduces to the classical Hansen-Hurwitz estimator). As a special case, we show that if $\mathcal{Q}$ is pointwise γ-close to uniform and all $x_i \in \{0, 1\}$, for any $\varepsilon > 0$, we can estimate μ to within additive error εN using $m = \Theta(N^{1-\frac{1}{k}} / \varepsilon^{2/k})$ samples, where $k = \lceil (\lg \varepsilon)/(\lg \gamma) \rceil$. We then show that this sample complexity is essentially optimal. Interestingly, our upper and lower bounds show that the sample complexity need not vary uniformly with the desired error parameter ε: for some values of ε, perturbations in its value have no asymptotic effect on the sample complexity, while for other values, any decrease in its value results in an asymptotically larger sample complexity.

2012 ACM Subject Classification Mathematics of computing → Probabilistic algorithms; Theory of computation → Sample complexity and generalization bounds; Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Lower bounds and information complexity

Keywords and phrases bias reduction, sum estimation, sublinear time algorithms, sample complexity

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.62

Category RANDOM

Funding *Jakub Tětek*: Supported by the VILLUM Foundation grant 16582.

Acknowledgements We thank the anonymous peer reviewers, whose feedback helped improve our manuscript.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 62; pp. 62:1–62:21
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Consider the following simple problem. Let us have values $x_i \in \{0,1\}$ for $i \in [N]$ and assume we may sample i from a distribution $\mathcal{Q}$ that is pointwise γ-close to uniform (see Definition 2). It is easy to obtain an additive $\pm\gamma N$ approximation of the number of $1's$. But is it possible to get a better approximation using a number of samples that is sub-linear in N? We answer this question positively. Specifically, we solve a more general sum estimation problem, with the above problem being the simplest application. Additionally, we derive lower bounds showing that for a wide range of parameters, the sample complexity of our algorithm is asymptotically tight.

In full generality, estimating the sum of a (multi)set of numbers is a fundamental problem in statistics, and the problem plays an important role in the design of efficient algorithms for large datasets. The basic problem can be formulated as follows: given a multiset of N elements, $S = \{x_1, x_2, \ldots, x_N\}$, compute an estimate of the sum $\mu = \sum_{i\in[N]} x_i$.

Assume that the values x_i can be sampled according to some probability distribution $\mathcal{Q}$ over S (equivalently, over $[N]$). The classical work of Hansen and Hurwitz [12] examines the setting in which, when an element $x \in S$ is sampled, the probability $\mathcal{Q}(x)$ can be determined. They introduce the Hansen-Hurwitz estimator defined by

$$\mu_{\text{HH}} = \frac{1}{m}\sum_{j=1}^{m} \frac{X_j}{\mathcal{Q}(X_j)} \tag{1}$$

where $X_1, X_2, \ldots, X_m$ are samples taken from the distribution $\mathcal{Q}$. This estimator has been used extensively (though often implicitly) in sublinear algorithms [5, 20, 1, 14]. Hansen and Hurwitz prove that (1) is an accurate estimator of μ via the following theorem:

▶ **Theorem 1** (Hansen & Hurwitz, 1943 [12]). *The value μ_{HH} is an unbiased estimator of the sum μ (i.e., $\mathrm{E}(\mu_{HH}) = \mu$) and its variance is*

$$\mathrm{Var}[\mu_{HH}] = \frac{1}{m}\sum_{i=1}^{N} \mathcal{Q}(i)\left(\frac{x_i}{\mathcal{Q}(i)} - \mu\right)^2. \tag{2}$$

Theorem 1 can be applied to obtain probabilistic guarantees for estimating μ via sampling. For example, if one wishes to compute a $1\pm\varepsilon$ multiplicative estimate of μ with probability $1-\delta$, by Chebyshev's inequality, it suffices to take m sufficiently large that $\mathrm{Var}[\mu_{\text{HH}}]/(\varepsilon^2\mu^2) < \delta$.

In practice, it may however be unreasonable to assume that the probability distribution from which elements are sampled is known precisely. For example, the underlying process generating the samples may be noisy or may induce some underlying bias. We model this situation by assuming that the true sampling distribution $\mathcal{Q}$ is close to some known distribution $\mathcal{P}$. When an element x is sampled, the probability $\mathcal{P}(x)$ can be determined, but not the true probability $\mathcal{Q}(x)$. We assume that $\mathcal{Q}$ is *pointwise close* to $\mathcal{P}$ in the following sense. The assumption of pointwise closeness turns out to be necessary (as compared to weaker notions of closeness); see the discussion on page 4.

▶ **Definition 2.** *Let $\mathcal{P}$ and $\mathcal{Q}$ be probability distributions over a (multi)set S. Then for any $\gamma < 1$, we say that $\mathcal{Q}$ is* pointwise γ-close *to $\mathcal{P}$ if for every $x \in S$, we have*

$$(1-\gamma)\mathcal{P}(x) \le \mathcal{Q}(x) \le (1+\gamma)\mathcal{P}(x). \tag{3}$$

Given the situation above, one can apply the Hansen-Hurwitz estimator (1) with the known distribution $\mathcal{P}$ in place of the true sample distribution $\mathcal{Q}$. We define the *positive sum*, μ_+ to be

$$\mu_+ = \sum_{x \in S} |x| \,. \tag{4}$$

It is straightforward to show that the resulting estimator has bias at most $\gamma\mu_+$, and its variance increases by a factor of $1 + O(\gamma)$. However, the parameter γ may be too large to guarantee the desired error in the estimate of μ. For the above problem of estimating the sum of 0-1 values, this would lead to error of γN, while we want error of εN for $\varepsilon < \gamma$.[1]

Our setting is closely related to recent work in distribution testing. For example, it has been noted that many algorithms that rely on a probability oracle are “robust” in the sense that we may do distribution testing to within ε if the oracle’s answers have relative error of, say, $1 \pm \varepsilon/3$ [18, 3]. Our work goes further in the sense that our estimators work also in the setting when the error in the oracle’s answers is greater than the desired error parameter ε. Specifically, our goal is to characterize the (sample) complexity of a task as a function of the oracle error parameter γ and a desired approximation parameter ε. This can also be seen as a generalization of the learning-augmented distribution testing setting where γ is assumed to be constant [6].

1.1 Our Contributions

In this paper, our goal is to estimate the sum $\mu = \sum_{i=1}^{N} x_i$ with an error that is strictly less than the bias $\gamma\mu_+$ (Equation (4)) guaranteed by μ_{HH}. Specifically, given a desired error parameter ε with $0 < \varepsilon < \gamma$, we wish to estimate μ with bias close to $\varepsilon\mu_+$. In our setting, for each sample we are given a random index $i \in [N]$ drawn from the unknown distribution $\mathcal{Q}$, along with the value x_i and our estimate $\mathcal{P}(i)$ of the true probability $\mathcal{Q}(i)$. We introduce a family of estimators $\zeta_1, \zeta_2, \ldots$, where each ζ_k has bias at most $\gamma^k\mu_+$. To motivate the construction of ζ_k, we first re-write the Hansen-Hurwitz estimator in terms of the frequency vector of samples from S. Specifically, if $X_1, X_2, \ldots, X_m$ are the sampled elements, define the frequency vector $Y = (Y_1, Y_2, \ldots, Y_N)$ by

$$Y_i = |\{j : X_j = i\}| \,.$$

We define the estimator

$$\xi_1 = \frac{1}{m} \sum_{i=1}^{N} Y_i \cdot \frac{x_i}{\mathcal{P}(i)} .$$

Note that this estimator can be efficiently implemented, as the items that have not been sampled contribute 0 to the sum. We may thus implement this in time linear in the sample complexity, and do not need to take $O(N)$ time.

In the case where $\mathcal{P} = \mathcal{Q}$, ξ_1 is equivalent to the Hansen-Hurwitz estimator μ_{HH}. More generally, $\mathcal{Q}$ is pointwise γ-close to $\mathcal{P}$, and ξ_1 has bias at most $\gamma\mu_+$.

The estimator ξ_1 can be generalized as follows. Rather than sampling individual elements, we can examine h-wise collisions between samples, where an *h-wise collision* consists of h samples resulting in the same outcome.

[1] It is possible to get tighter bounds if parameterizing also by the sum, but for simplicity, we choose to parameterize the error only by N, ε, and γ.

▶ **Definition 3.** *For any positive integer h, we define the h-*wise collision estimator *ξ_h of $\mu = \sum_i x_i$ to be*

$$\xi_h = \frac{1}{\binom{m}{h}} \sum_{i=1}^{N} \binom{Y_i}{h} \frac{x_i}{(\mathcal{P}(i))^h}. \tag{5}$$

We note that $\binom{Y_i}{h}$ gives the number of h-wise collisions involving the value $X_j = i$. It is straightforward to show that when $\mathcal{Q} = \mathcal{P}$, all ξ_h are unbiased estimators for μ, and that ξ_h has bias $O(h\gamma\mu_+)$ when $\mathcal{Q}$ is pointwise γ-close to $\mathcal{P}$.

Individually, the estimators $\xi_1, \xi_2, \ldots$ are no better than $\xi_1 = \mu_{\mathrm{HH}}$ in terms of bias and variance. As we will show, however, for any positive integer k, a suitable linear combination of the ξ_i can be chosen such that the coefficients of γ^j in the bias cancel out for $j < k$. The resulting estimator then has bias $\leq \gamma^k \mu_+$.

▶ **Definition 4.** *For each positive integer k, we define the* bias reducing estimator of order *k, ζ_k, to be*

$$\zeta_k = \sum_{h=1}^{k} (-1)^{h+1} \binom{k}{h} \xi_h = \sum_{h=1}^{k} (-1)^{h+1} \frac{\binom{k}{h}}{\binom{m}{h}} \sum_{i \in [N]} \binom{Y_i}{h} \frac{x_i}{(\mathcal{P}(i))^h}. \tag{6}$$

▶ **Example 5.** In order to give some intuition about the expression (6), consider the case where $\mathcal{P}$ is the uniform distribution and $k = 2$. Define α_i to be such that $\mathcal{Q}(i) = (1 + \alpha_i)\mathcal{P}(i)$. Note that $|\alpha_i| \leq \gamma$ because $\mathcal{Q}$ is assumed to be pointwise γ-close to uniform. By a simple calculation, we have that $\mathrm{E}[\xi_1] = \sum_{i=1}^{N} (1 + \alpha_i) x_i$ and $\mathrm{E}[\xi_2] = \sum_{i=1}^{N} (1 + \alpha_i)^2 x_i = (1 + 2\alpha_i + \alpha_i^2) x_i$. Therefore, it holds that[2]

$$\mathrm{E}[\zeta_2] = \mathrm{E}[2\xi_1 - \xi_2] = \sum_{i=1}^{N} 2(1+\alpha_i)x_i - (1+2\alpha_i+\alpha_i^2)x_i = \sum_{i=1}^{n} (1+\alpha_i^2)x_i \subseteq \mu \pm \sum_{i=1}^{n} \alpha_i^2 |x_i| \subseteq \mu \pm \gamma^2 \mu_+.$$

This proves that the bias of the estimator is at most $\gamma^2 \mu_+$. Similarly, one can show that the bias of the estimator above is $\leq \gamma^k \mu_+$. We prove this in Section 2.

▶ **Theorem 6** (Bias portion of Theorem 10)**.** *Supoose $\mathcal{P}$ and $\mathcal{Q}$ are probability distributions over $[N]$ with $\mathcal{Q}$ pointwise γ-close to $\mathcal{P}$. Let ζ_k be defined as in (6). Then*

$$|\mathrm{E}[\zeta_k - \mu]| \leq \gamma^k \mu_+.$$

In particular, if $x_i \geq 0$ for all i, then ζ_k has bias at most $\gamma^k \mu$.

This theorem shows that ζ_k reduces the bias to γ^k compared to the bias γ for the Hansen-Hurwitz estimator (equivalent to ζ_1). Theorem 10 additionally bounds the variance of ζ_k, which is required for our applications.

We apply Theorem 6 (or more specifically, Theorem 10) to obtain our main algorithmic results. Our goal is as follows: given sample access to some $\mathcal{Q}$ that is pointwise γ-close to $\mathcal{P}$ and a desired error parameter ε, estimate μ with error ε using as few samples as possible. To this end, we employ a two-stage estimation technique (see Algorithm 2). In the first stage, we use the 1-wise collision estimator ξ_1 (i.e., the Hansen-Horwitz estimator) to obtain a coarse estimate of μ. Then, the second stage refines this estimate by applying the bias reducing estimator ζ_k with an appropriately chosen k. Specifically, we show the following:

[2] The first of the two inclusions is not tight in that the absolute value in $|x_i|$ is not necessary, but it becomes necessary for odd values of k.

▶ **Theorem 7** (Special case of Theorem 10). *Define $n = \max_i 1/\mathcal{P}(i)$.*[3] *Suppose $\mathcal{Q}$ is pointwise γ-close to $\mathcal{P}$, and let $\mathrm{Var}_{\mathrm{HH}}$ be defined as*

$$\mathrm{Var}_{\mathrm{HH}} = \frac{1}{N^2}\sum_{i=1}^{N} \mathcal{Q}(i)\left(\frac{x_i}{\mathcal{Q}(i)} - \mu\right)^2. \tag{7}$$

That is, $\mathrm{Var}_{\mathrm{HH}}$ is the variance of the Hansen-Hurwitz estimator for the mean *μ_{HH}/N with sample size $m = 1$ (cf. (Equation 2)). For $\varepsilon_1, \varepsilon_2 > 0$, define $k = \lceil (\log \varepsilon_1)/\log \gamma) \rceil$. Then using*

$$m = O\left(\sqrt[k]{n^{k-1}\varepsilon_2^{-2}\,\mathrm{Var}_{\mathrm{HH}}}\right)$$

independent samples from $\mathcal{Q}$, with probability at least $2/3$, Algorithm 2 produces an estimate $\hat{\mu}$ of $\mu = \sum_i x_i$ with absolute error

$$|\mu - \hat{\mu}| \le \varepsilon_1 \mu_+ + \varepsilon_2.$$

To understand the complexity of this algorithm, we note that when $\mathcal{P} = \mathcal{Q}$, in order to get an error ε_2 the complexity of the Hansen-Hurwitz estimator is $\varepsilon_2^{-2}\,\mathrm{Var}_{\mathrm{HH}}$. The complexity of our algorithm can thus be seen as a weighted geometric average between the complexity of the Hansen-Hurwitz estimator, and n.

As a corollary of Theorem 7, we obtain a solution to the aforementioned problem of estimating a sum of 0-1 values.

▶ **Corollary 8.** *Suppose $\mathcal{Q}$ is pointwise γ-close to the uniform distribution over $[N]$ and $x_i \in [0,1]$ for every $i \in [N]$. For any $\varepsilon > 0$ define $k = \lceil (\log \varepsilon)/\log \gamma \rceil$. Then $m = O(N^{1-1/k}\varepsilon^{-2/k})$ samples are sufficient to obtain an estimate of $\mu = \sum_i x_i$ with additive error εN with probability $2/3$.*

We note that the asymptotic sample complexities in Theorem 7 and Corollary 8 are non-uniform in γ and ε. In the case of Corollary 8, for any fixed postive integer k and constant $\gamma > 0$, if $\varepsilon = \gamma^k$, then $O(N^{1-1/k})$ samples are sufficient to obtain an εN additive estimate. On the other hand, if $\gamma^{k+1} \le \varepsilon < \gamma^k$, then our algorithm uses $O(N^{1-1/(k+1)})$ samples. Our next main result shows that this sample complexity is essentially optimal, and perhaps surprisingly, that the non-uniformity of the sample complexity is unavoidable. Specifically, we show the following lower bound.

▶ **Theorem 9.** *Suppose $x_1, x_2, \ldots, x_N \in \{0,1\}$ and N is a parameter. Then for every positive integer k, there exists a positive constant c_k such that for $\varepsilon \le c_k\gamma^k$, there is a sequence of distributions on $[N]$ that are γ-close to uniform such that the number of samples required to estimate μ within error εN is $\Omega(N^{1-1/(k+1)})$.*

This lower bound matches the upper bound of Corollary 8 for a large range of parameters. When $\gamma \le c_k$ (where c_k is as in the conclusion of the theorem), our algorithm has sample complexity $O(N^{1-1/(k+1)})$ for all $\varepsilon \in [\gamma^{k+1}, \gamma^k)$, while the lower bound shows $\Omega(N^{1-1/(k+1)})$ samples are necessary for all $\varepsilon \in [\gamma^{k+1}, c_k\gamma^k]$. Interestingly, these matching upper and lower bounds show that the asymptotic sample complexity is non-uniform as a let of ε for any fixed (sufficiently small) γ: for every $\varepsilon \in [\gamma^{k+1}, c_k\gamma^k]$, exactly $\Theta(N^{1-1/(k+1)})$ samples are

[3] Note that $N \le n$, where N is the size of the multiset being sampled. In the case where $\mathcal{P}(i) = \Omega(1/N)$ for all i, we have $n = \Theta(N)$. The convention of defining n in this way was previously used in [6].

necessary and sufficient, while for $\varepsilon = \gamma^k$, $\Theta(N^{1-1/k})$ samples are necessary and sufficient. Thus, as a function ε, the sample complexity contains "islands of stability" – intervals in which some perturbations of ε have no effect on the asymptotic sample complexity – while between these intervals, an arbitrarily small (constant) decrease in ε results in a polynomial (in N) increase in the sample complexity.

Discussion and Related Work

Throughout the paper, we assume that the probability distribution $\mathcal{Q}$ from which samples are generated is pointwise close to $\mathcal{P}$ in the sense of Definition 2. Pointwise closeness is a strictly stronger (and less commonly used) distance measure than, for example, total variation distance (i.e., L_1 distance) or other L_p distances. Nonetheless, this relatively strong assumption about the relationship between $\mathcal{P}$ and $\mathcal{Q}$ is necessary in order to obtain any non-trivial guarantee for estimating μ.[4] Algorithms for generating samples with pointwise approximation guarantees have been studied in the context of sublinear time algorithms [10, 18, 8, 7, 6, 20, 9] as well as Markov chains [16, 13]. In the latter case, *uniform mixing time* gives a bound on complexity of obtaining samples with pointwise guarantees. Interestingly, Hermon [13] shows a result that can be viewed as an analogue of our lower bound for Markov chains (specifically random walks on bounded degree graphs). Namely, small perturbations in the transition probabilities of edges can result in an asymptotic increases in the uniform mixing time.

The problem of estimating the sum is well-studied in statistics. Classical estimators for non-uniform sampling probabilities are described by Hansen and Hurwitz [12] and Horvitz and Thompson [15]. Sum estimation in the related setting where we do not know the sampling probabilities but know that they are proportional to the items' values has been studied in [17, 2].

Open problem: Sample correctors for uniform sampling

Finally, we state one interesting open problem. The concept of sample correctors from [4] assumes that we may sample from a distribution that is close to some property, and we want to be able to use it to sample from a distribution even closer to satisfying the property. It is natural to ask if one can use $o(n)$ samples from a distribution pointwise γ-close to uniform and simulate a sample with bias $o(\gamma)$.

1.2 Technical Overview

Upper Bound

The goal is to reduce the bias from γ to γ^k. Now the main observation that guides our construction is the following identity:

$$1 + (-1)^{k+1}\gamma^k = \sum_{h=1}^{k} (-1)^{h+1} \binom{k}{h} (1+\gamma)^h \ . \tag{8}$$

This should be compared to our estimator, $\zeta_k = \sum_{h=1}^{k}(-1)^{h+1}\binom{k}{h}\xi_h$, which clearly mirrors the identity. The reason for this is that the probability of an h-wise collision on position $i \in [N]$ is $\mathcal{Q}(i)^h \approx (1+\gamma)^h \mathcal{P}(i)^h$ in the worst case when $\frac{\mathcal{Q}(i)}{\mathcal{P}(i)} = 1+\gamma$. Hence the expectation of the h-wise collision estimator, ξ_h, is approximately bounded by $(1+\gamma)^h\mu$. This implies that

[4] As an extreme example, consider the case where only a γ-fraction of values x_i are nonzero. If $\mathcal{P}$ is uniform, then $\mathcal{Q}$ can assign zero mass to the nonzero elements, and still be γ-close to $\mathcal{P}$ with respect to total variation distance. Thus, *any* estimator will return a value that is independent of the actual sum.

when we take the expectation of our estimator then the expression reduces to Equation (8) which shows that the bias is reduced to γ^k. Here, we cheated slightly by assuming that the bias is the same for all the positions $i \in [N]$. This is of course not true, but actual calculation reduces to N instances of Equation (8).

The more delicate part of the analysis of our estimator is the bound on the variance. The main difficulty lies in rewriting $\zeta_k - \mathrm{E}[\zeta_k]$ into something manageable. We note that we can write our estimator ζ_k as

$$\zeta_k = \sum_{i\in[N]} x_i \sum_{h=1}^{k} (-1)^{h+1} \frac{\binom{k}{h}}{\binom{m}{h}} \frac{\binom{Y_i}{h}}{(\mathcal{P}(i))^h} \ .$$

We can express the frequency vector Y by random variables as follows: $Y_i = \sum_{j\in[m]}[X_j = i]$. (Here, we use $[X_j = i]$ to denote the indicator random variable of the event $X_j = i$.) This allows us to see that $\binom{Y_i}{h} = \sum_{\substack{I\subseteq[m]\\|I|=h}} \prod_{j\in I}[X_j = i]$. If we now define the polynomials P_i by

$$P_i(\beta_1, \dots, \beta_m) = \sum_{h=1}^{k} (-1)^{h+1} \frac{\binom{k}{h}}{\binom{m}{h}} \frac{1}{(\mathcal{P}(i))^h} \sum_{\substack{I\subseteq[m]\\|I|=h}} \prod_{j\in I} \beta_j \ ,$$

then we can write our estimator as $\zeta_k = \sum_{i\in[N]} x_i P_i([X_1 = i], \dots, [X_m = i])$. We observe that P has degree 1 in each variable so $\mathrm{E}[P_i([X_1 = i], \dots, [X_m = i])] = P_i(\mathcal{Q}(i), \dots, \mathcal{Q}(i))$. Furthermore, it seems reasonable that there should exist polynomials Q_i which have degree 1 in each variable and satisfy $P_i([X_1 = i], \dots, [X_m = i]) - P_i(\mathcal{Q}(i), \dots, \mathcal{Q}(i)) = Q_i([X_1 = i] - \mathcal{Q}(i), \dots, [X_m = i] - \mathcal{Q}(i))$. This will help us in understanding the variance of our final estimator ζ_k by decomposing into variances of the simpler events $[X_1 = i]$. We show that Q_i exist and are defined as follows

$$Q_i(\beta_1, \dots, \beta_m) = (-1)^{k+1} \sum_{h=1}^{k} \frac{\binom{k}{h}}{\binom{m}{h}} \frac{\gamma_i^{k-h}}{(\mathcal{P}(i))^h} \sum_{\substack{I\subseteq[m]\\|I|=h}} \prod_{j\in I} \beta_j$$

So we get that $\zeta_k - \mathrm{E}[\zeta_k] = \sum_{i\in[N]} x_i Q_i([X_1 = i] - \mathcal{Q}(i), \dots, [X_m = i] - \mathcal{Q}(i))$. Since $[X_1 = i] - \mathcal{Q}(i)$ is a zero mean variable and Q_i has degree 1 in each variable, it becomes easy to calculate the variance.

A technical detail that we have not touched upon yet is that if $k \geq 2$ then $\mathrm{Var}[\zeta_k]$ becomes very large if $\mu^2 \gg \mathrm{Var}[\mu_{\mathrm{HH}}]$. The reason is that $\mathrm{Var}[\zeta_k]$ depends on $\sum_{i\in[N]} \mathcal{P}(i) \left(\frac{x_i}{\mathcal{P}(i)}\right)^2 = \mathrm{Var}[\mu_{\mathrm{HH}}] + \mu^2$. Thus, if $\mu^2 \gg \mathrm{Var}[\mu_{\mathrm{HH}}]$ then our variance becomes much larger which is a problem. Now imagine that we already have an estimate, W, of μ. We then set $\bar{x}_i = x_i - \mathcal{P}(i)W$ and make a new estimator $\bar{\zeta}_k$ that uses $\bar{x}_i$ instead of x_i. Then $\bar{\zeta}_k$ will estimate $\mu - W$ so $\bar{\zeta}_k + W$ will be an estimator of μ. The trick is that $\mathrm{Var}[\bar{\zeta}_k]$ depends on $\mathrm{Var}[\mu_{\mathrm{HH}}] + (\mu - W)^2$ so if $W = \mu + O(\sqrt{\mathrm{Var}[\mu_{\mathrm{HH}}]})$ then we will have control over the variance. We can get such estimate, W, by using our estimator for $k = 1$ where there are no issues with variance (i.e., the standard deviation of $W - \mu$ only depends on $\sqrt{\mathrm{Var}[\mu_{\mathrm{HH}}]}$ rather than on μ).

Lower Bound

At a high level, our strategy is to define a reduction from the problem of distinguishing two distributions D_1 and D_2 to the problem of estimating $\mu = \sum_i x_i$. More concretely, for each fixed positive integer k, we define distributions D_1 and D_2 with support sizes n_1 and n_2, respectively, such that:

1. $n_1 = (1 + (\Theta(\gamma))^k)n_2$,
2. D_1 and D_2 are both pointwise γ-close to uniform, and
3. D_1 and D_2 have identical p^{th} frequency moments for $p = 1, 2, \ldots, k$.[5]

We describe a reduction showing that for $N = n_1 + n_2$, $\mathcal{P}$ uniform over $[N]$, and $x_i \in \{0, 1\}$ for all $i \in [N]$, any algorithm that distinguishes $\mu = n_1$ from $\mu = n_2$ can also distinguish D_1 from D_2. We then apply a framework of Raskhodnikova et al. [19], which implies that distinguishing any two distributions whose first k frequency moments are equal requires $\Omega(n^{1-1/(k+1)})$ samples. One difference between our setting and that of [19] is that in our setting, for each sample i we are also given x_i, whereas [19] focuses on support size estimation. However, since the x_i's are 0 or 1-valued, estimating the sum requires us to estimate either the number of x_i's which are 0 or the number of x_i's which are 1. We can apply this observation to reduce the problem to finding two nearly uniform distributions (i.e., both are pointwise γ-close to uniform) that differ in support size by a multiplicative $1 \pm \Theta(\gamma)^k$ factor, yet match in the first k moments.

To do this, we use a combinatorial construction that is inspired by a related lower bound in [6]. For simplicity, we assume that the probability of sampling any fixed item lies in $\{\frac{1}{n}, \frac{1+\gamma}{n}, \ldots, \frac{1+k\gamma}{n}\}$ (while these distributions would only be $k \cdot \gamma$-close to uniform, we can replace γ with γ/k). For the distribution D_1, we assume the number of elements with probability $\frac{1+i\cdot\gamma}{n}$ is a_i, and for the distribution D_2 the number of elements with probability $\frac{1+i\cdot\gamma}{n}$ is b_i. Then, our goal is for the support sizes of D_1 and D_2 (which are $\sum a_i, \sum b_i$, respectively) to differ significantly but for the first k moments to match. This means $\sum a_i$ and $\sum b_i$ differ significantly, but $\sum a_i(1 + i \cdot \gamma)^\ell = \sum b_i(1 + i \cdot \gamma)^\ell$. If we define $c_i := a_i - b_i$ and think of $(1 + i \cdot \gamma)^\ell$ as a degree ℓ polynomial $P(i)$, we want $\sum c_i P(i) = 0$ but $\sum c_i$ to be large (roughly $\gamma^k \cdot n$). Finally, we need to make sure that $\sum a_i, \sum b_i$ are both $\Theta(n)$.

To determine the values of each c_i, we utilize the observation that for any polynomial $P(x)$ of degree less than k, the successive differences, i.e., $P(x) - P(x - 1)$ is a polynomial of degree less than $k - 1$. We can repeatedly take successive differences k times to get a linear combination of $P(0), P(1), \ldots, P(k)$ that equals 0 for any polynomial P of degree less than k. We can therefore set $c_i := a_i - b_i$ to be these linear coefficients. Unfortunately, we will have $\sum c_i = 0$ as well. Instead, we replace c_i with $c'_i := c_i/(1 + i\gamma)$, so that $\sum c'_i \cdot (1 + i \cdot \gamma)^\ell = \sum c_i \cdot (1 + i \cdot \gamma)^{\ell-1} = 0$ for all $1 \le \ell \le k$. However, $\sum c'_i = \sum c_i/(1 + i \cdot \gamma)$ is not expressible as $\sum c_i P(i)$ for a polynomial P of low degree, and will in fact be nonzero, which is exactly what we want. We scale the c'_i terms so that $\sum c'_i = \gamma^k \cdot n$. If we write $a_i = \max(c'_i, 0)$ and $b_i = \max(-c'_i, 0)$, then every a_i and b_i is nonnegative but $a_i - b_i = c'_i$. One can show via some careful combinatorics that after this scaling, $\sum a_i$ and $\sum b_i$ are both $\Theta(n)$, as desired.

2 Sum Estimation

We now give the algorithm for the sum estimation problem. We then state our main result (Theorem 10) as well as the special case for estimating the sum of 0-1 values (Corollary 8) that we discussed in the introduction. We then state and prove Lemma 12 which we then use to prove Theorem 10.

We now state our main theorem. Note that this implies, by a simple substitution, the simpler version mentioned in the introduction as Theorem 7.

[5] Recall that the p^{th} frequency moment of a distribution D is $\sum_x (D(x))^p$.

Algorithm 1 *EstimateSum*(m, k, W).

```
$(X_j)_{j\in[m]} \leftarrow$ take $m$ samples from the distribution $\mathcal{Q}$
For every $i \in [N]$, let $Y_i$ denote the number of times value $i$ was sampled
  $(Y_i = \sum_{j\in[m]}[X_j = i]$ )
For every $h \in [k]$, let $\xi_h$ be the $h$-wise collision estimator
  $(\xi_h = \frac{1}{\binom{m}{h}} \sum_{i=1}^{N} \binom{Y_i}{h} \frac{x_i - \mathcal{P}(i)W}{(\mathcal{P}(i))^h}$ )
return $W + \sum_{h=1}^{k} (-1)^{h+1} \binom{k}{h} \xi_h$
```

Algorithm 2 *ImprovedEstimateSum*(m, t, k).

```
$W \leftarrow EstimateSum(t, 1, 0)$
return $EstimateSum(m, k, W)$
```

▶ **Theorem 10.** *Define $n = \max_i 1/\mathcal{P}(i)$. Suppose $\mathcal{Q}$ is pointwise γ-close to $\mathcal{P}$, and let $\mathrm{Var}_{\mathrm{HH}}$ be the variance of μ_{HH}/N defined in Equation (7). For $\varepsilon_1, \varepsilon_2 > 0$, define $k = \lceil (\lg \varepsilon_1)/\lg \gamma)\rceil$. Then using*

$$m = O\left(\sqrt[k]{n^{k-1}\varepsilon_2^{-2}\,\mathrm{Var}_{\mathrm{HH}}}\right)$$

independent samples from $\mathcal{Q}$, with probability at least $2/3$, Algorithm 2 produces an estimate $\hat{\mu}$ of $\mu = \sum_i x_i$ with absolute error

$$|\mu - \hat{\mu}| \le \varepsilon_1(1+\gamma) \mathop{\mathrm{E}}_{X\sim\mathcal{P}}[|\mathcal{P}(X)^{-1}x_X - \mu|] + \varepsilon_2$$

This theorem implies in a straightforward way a solution to the problem of estimating the sum of 0-1 values that we discussed in the introduction.

▶ **Corollary 11.** *Suppose $\mathcal{Q}$ is pointwise γ-close to the uniform distribution over $[N]$ and $x_i \in \{0,1\}$ for every $i \in [N]$. For any $\varepsilon > 0$ define $k = \lceil (\log \varepsilon)/\log \gamma\rceil$. Then $m = O(n^{1-1/k}\varepsilon^{-2/k})$ samples are sufficient to obtain an estimate of $\mu = \sum_i x_i$ with additive error $\varepsilon(\mu + \sqrt{\mu N})$ with probability $2/3$.*

Before we can prove our main result, we need the following lemma.

▶ **Lemma 12** (Analysis of *EstimateSum*$(m, k, 0)$)**.** *Define $n = \max_i 1/\mathcal{P}(i)$. Let $(x_i)_{i\in[N]}$ be a sequence of numbers and define $\mu = \sum_{i\in[N]} x_i$. Let $\mathcal{P}$ be a probability distribution over $[N]$, and let $\mathcal{Q}$ be another probability distribution over $[N]$ that is pointwise γ-close to $\mathcal{P}$.*

Consider a sequence $(X_j)_{j\in[m]}$ of independent random variables both with distribution $\mathcal{Q}$. Define $Y_i = \sum_{j\in[m]}[X_j = i]$ for every $i \in [N]$. Define

$$\zeta_k = \sum_{h=1}^{k} (-1)^{h+1} \frac{\binom{k}{h}}{\binom{m}{h}} \sum_{i\in[N]} \binom{Y_i}{h} \mathcal{P}(i)^{-h} x_i \; .$$

Then for $k \ge 2$, we get that

$$|\operatorname{E}[\zeta_k] - \mu| \le \gamma^k \sum_{i\in[N]} |x_i| \tag{9}$$

$$\mathrm{Var}[\zeta_k] \le \max\left\{2(1+\gamma)\gamma^{2k-2}k^2 \frac{\sum_{i\in[N]} \mathcal{P}(i)^{-1}x_i^2}{m}, 2^k(1+\gamma)^k k^{3k} \frac{n^{k-1}\sum_{i\in[N]} \mathcal{P}(i)^{-1}x_i^2}{m^k}\right\} , \tag{10}$$

and for $k = 1$, we get that

$$|\operatorname{E}[\zeta_1] - \mu| \leq \gamma \sum_{i \in [N]} |x_i - \mathcal{P}(i)\mu| \tag{11}$$

$$\operatorname{Var}[\zeta_1] \leq (1+\gamma)\frac{\sum_{i \in [N]} \mathcal{P}(i)^{-1}(x_i - \mathcal{P}(i)\mu)^2}{m} \tag{12}$$

We are now ready to prove the main theorem.

Proof of Theorem 10. Let W be an estimate of μ using $\mathit{EstimateSum}(t, 1, 0)$ where $t = O(1 + \gamma^{2k}\varepsilon_2^{-2}\operatorname{Var}_{\mathrm{HH}})$. Using Lemma 12 we get an estimate of the bias and the variance. Now by Chebyshev's inequality, we easily get that $|W - \mu| \leq \gamma \sum_{i \in [N]} \mathcal{P}(i)|\mathcal{P}(i)^{-1}x_i - \mu| + \max\left\{\varepsilon_2/(2\gamma^k), \sqrt{\sum_{i \in [N]} \mathcal{P}(i)(\mathcal{P}(i)^{-1}x_i - \mu)^2}\right\}$ with probability 5/6. We note that $\sum_{i \in [N]} \mathcal{P}(i)|\mathcal{P}(i)^{-1}x_i - \mu| \leq \sqrt{\sum_{i \in [N]} \mathcal{P}(i)(\mathcal{P}(i)^{-1}x_i - \mu)^2}$ since p-norms are increasing. So we also get that $|W - \mu| \leq O(\sqrt{\sum_{i \in [N]} \mathcal{P}(i)(\mathcal{P}(i)^{-1}x_i - \mu)^2})$

Now we calculate ζ using $\mathit{EstimateSum}(m, k, W)$ with $m = O\left(\sqrt[k]{n^{k-1}\varepsilon_2^{-2}\operatorname{Var}_{\mathrm{HH}}}\right)$ so ζ corresponds to $\mathit{ImprovedEstimateSum}(m, t, k)$. We now note that by Lemma 12,

$$\begin{aligned}|(\operatorname{E}[\zeta] - W) - (\mu - W)| &\leq \gamma^k \sum_{i \in [N]} \mathcal{P}(i)|\mathcal{P}(i)^{-1}x_i - W| \\ &\leq \gamma^k|W - \mu| + \gamma^k \sum_{i \in [N]} \mathcal{P}(i)|\mathcal{P}(i)^{-1}x_i - \mu|,\end{aligned}$$

and

$$\begin{aligned}\operatorname{Var}[\zeta] \leq \max\Bigg\{ & 2(1+\gamma)\gamma^{2k-2}k^2\frac{\sum_{i \in [N]} \mathcal{P}(i)(\mathcal{P}(i)^{-1}x_i - W)^2}{m}, \\ & 2^k(1+\gamma)^k k^{3k}\frac{n^{k-1}\sum_{i \in [N]} \mathcal{P}(i)(\mathcal{P}(i)^{-1}x_i - W)^2}{m^k}\Bigg\} \\ = \max\Bigg\{ & 2(1+\gamma)\gamma^{2k-2}k^2\frac{\sum_{i \in [N]} \mathcal{P}(i)(\mathcal{P}(i)^{-1}x_i - \mu)^2 + (W-\mu)^2}{m}, \\ & 2^k(1+\gamma)^k k^{3k}\frac{n^{k-1}\left(\sum_{i \in [N]} \mathcal{P}(i)(\mathcal{P}(i)^{-1}x_i - W)^2 + (W-\mu)^2\right)}{m^k}\Bigg\}\end{aligned}$$

Assuming that

$$|W - \mu| = \gamma \sum_{i \in [N]} \mathcal{P}(i)|\mathcal{P}(i)^{-1}x_i - \mu| + \max\left\{\varepsilon_1/(2\gamma^k), \sqrt{\sum_{i \in [N]} \mathcal{P}(i)(\mathcal{P}(i)^{-1}x_i - \mu)^2}\right\},$$

we get that $|(\operatorname{E}[\zeta] - W) - (\mu - W)| \leq \gamma^k(1+\gamma)\sum_{i \in [N]} \mathcal{P}(i)|\mathcal{P}(i)^{-1}x_i - \mu| + \varepsilon_2/2$. Using that $|W - \mu| \leq O(\sqrt{\sum_{i \in [N]} \mathcal{P}(i)(\mathcal{P}(i)^{-1}x_i - \mu)^2})$ we can then use Chebyshev's inequality to conclude that with probability 5/6

$$|\zeta - \operatorname{E}[\zeta]| \leq \varepsilon_2/2$$

So all in all, with probability at least 2/3 we that $|\zeta - \mu| \leq \varepsilon_1(1+\gamma)\sum_{i \in [N]} \mathcal{P}(i)|\mathcal{P}(i)^{-1}x_i - \mu| + \varepsilon_2$. ◀

3 Lower Bound

In this section, we prove that for a range of parameters – specifically in the setting of Corollary 11 – the sample complexity of our sum estimation algorithm (Algorithm 2) is asymptotically tight.

▶ **Theorem 13.** *Let k be a positive integer and let $\varepsilon < \gamma < 1$ be positive numbers. Suppose A is an algorithm such that for any $v \in \{0,1\}^N$ and any distribution $\mathcal{P}$ over $[N]$ that is pointwise γ-close to uniform, A uses samples from $\mathcal{P}$ and returns an estimate of $\|v\|_1 = \sum_i v_i$ within additive error εN with probability $2/3$. Then there exists c_k with $0 < c_k < 1$ such that if $\varepsilon \leq c_k \gamma^k$ then A requires $\Omega(N^{1-1/(k+1)})$ samples.*

While at a first glance this result might seem contradictory to our upper bound (specifically, Corollary 8), it actually reveals the following interesting phenomena. Notice that in our upper bound, the complexity depends on $k = \lceil (\log \varepsilon)/\log \gamma \rceil$, so that, e.g., for $\varepsilon = \gamma^k$, the complexity is $O(N^{1-\frac{1}{k}})$. Once ε becomes slightly smaller, i.e., $\varepsilon = c\gamma^k$ for c satisfying $\gamma \leq c < 1$, the complexity of our algorithm abruptly jumps to $O(n^{1-\frac{1}{k+1}})$. The lower bound implies that this increase in complexity is unavoidable for sufficiently small c. That is, Theorem 13 states that there exists a (sufficiently small) constant c_k, such that indeed once $\varepsilon = c_k \cdot \gamma^k$, the required number of samples is $\Omega(n^{1-\frac{1}{k+1}})$. Interestingly, for all ε satisfying $\gamma^{k+1} \leq \varepsilon \leq c_k\gamma^k$, the asymptotic complexity of sum estimation is the same; the complexity only varies for ε satisfying $c_k\gamma^k \leq \varepsilon \leq \gamma^k$. Our matching upper and lower bounds demonstrate that the sample complexity's non-uniform dependence on ε is not an artifact, but captures the true complexity of the problem (up to the dependency on $\gamma^{k/2}$ in the numerator of the upper bound). Note that if the conclusion of Theorem 13 held *every* $c < 1$, then this would capture the right dependency on n for *all* possible ranges of γ. Since we only prove the theorem for a sufficiently small c_k, it might be the case that for values ε that are not too much smaller than γ^k, the optimal dependency on N is lower than our stated upper bound. Nonetheless, our upper and lower bounds match (up to constant factors) for all ε satisfying $\gamma^{k+1} \leq \varepsilon \leq c_k\gamma^k$.

As described in Section 1.2, the main technical ingredient in the proof of the theorem is in describing two distributions D_1 and D_2 over ranges $[n_1], [n_2]$, respectively, such that D_1 and D_2 are pointwise γ-close to uniform, $n_1 = (1 + \Theta(\gamma)^k)n_2$, and D_1 and D_2 have matching frequency moments 1 through k. Given these distributions we rely on the framework for proving lower bounds by Raskhodnikova et al. [19], which states that any uniform algorithm that distinguishes two random variables with matching frequency moments 1 through k must perform $\Omega(n^{1-1/(k+1)})$ many samples.

In order to simplify our construction and its analysis, we prove the lower bound for *uniform algorithms*. Here, a uniform algorithm is an algorithm whose output depends only on the "collision statistics" of the samples – i.e., the number of collisions involving each sample, and not the identities of the samples themselves.

▶ **Definition 14** (Uniform algorithm. Definition 3.2 in [19])**.** *An algorithm is* uniform *if it samples indices $i_1, \cdots, i_m$ independently with replacement and its output is uniquely determined by (i) the value of the items x_{i_j} and (ii) the set of collisions, where two indices j and j'* collide *if $i_j = i_{j'}$.*

In particular, a uniform algorithm's output does *not* depend on the sampled indices themselves. The following lemma asserts that our restriction to uniform algorithms is without loss of generality.

▶ **Lemma 15** (cf. Theorem 11.12 in [11]). *Suppose there exists an algorithm A such that for any $v \in \{0,1\}^N$ A uses samples from $\mathcal{P}$ and returns an estimate of $\|v\|_1 = \sum_i v_i$ within additive error εN with probability $2/3$ using s samples in expectation. Then there exists a uniform algorithm A' that achieves the same approximation guarantee using s samples in expectation.*

The proof of Lemma 15 is essentially the same as that of Goldreich's Theorem 11.12 in [11]. The key idea is that $\sum_i v_i$ is a "label invariant" in the sense that it is unaffected by any permutation of the indices of v. Thus, given an algorithm A as in the hypothesis of Lemma 15, we can obtain uniform algorithm A' with the same approximation guarantee by simply choosing a uniformly random permutation of the indices of v, then using the permuted indices of v as inputs for A. See Theorem 11.12 in [11] for details.

Finally, our lower bound argument requires the following result that is a direct consequence of the work of Raskhodnikova et al. [19].

▶ **Theorem 16** (Consequence of Lemma 5.3 and Corollary 5.7 from [19]). *Let D_1 and D_2 be distributions over positive integers $b_1 < \ldots < b_\ell$, that have matching frequency moments 1 through k. Then for any uniform algorithm A with sample complexity s that distinguishes D_1 and D_2 with high constant probability, $s = \Omega(n^{1-1/(k+1)})$.*

Our main argument for the lower bound applies Theorem 16 in conjunction with a reduction from distinguishing D_1 and D_2 to sum estimation. The main technical ingredient is the following lemma, which asserts the existence of suitable distributions D_1 and D_2.

▶ **Lemma 17.** *For every positive integer k and sufficiently large integer n, there exist two distributions D_1, D_2 over $[n_1]$ and $[n_2]$ (respectively) satisfying $n_1 = (1+\Theta(\gamma)^k)n$, and $n_2 = n$ such that $p_{D_j}(i) \in (1 \pm \gamma)\frac{1}{n}$ for $j \in \{1,2\}$ and the following holds. For all $\ell \in \{1, 2, \ldots, k\}$, it holds that*

$$\sum_{i=1}^{n_1} (p_{D_1}(i))^\ell = \sum_{i=1}^{n_2} (p_{D_2}(i))^\ell.$$

In particular, there exists an absolute constant c_k such that for sufficiently large n, $n_2 \geq (1 + c_k\gamma^k)n_1$ and the above conclusion holds.

Before proving the lemma, we show how the lemma implies Theorem 13.

Proof of Theorem 13. The theorem follows from Theorem 16 together with Lemma 17. Let $N = n_1 + n_2$, where $n_1 = (1 + \Theta(\gamma)^k)n_2$ as in the conclusion of Lemma 17, and consider distinguishing between two possible outcomes $\mathcal{O}_1$ and $\mathcal{O}_2$.

In the first outcome $\mathcal{O}_1$, let $S = \{1, 2, \ldots, n_1\} \subseteq [N]$ and $T = \{t_1, \ldots, t_{n_2}\} := [N] \backslash S$. The distribution $\mathcal{Q}$ will be as follows. With exactly $1/2$ probability, we choose S: if so, we then choose a sample $i \sim D_1$, which will be in $[n_1]$, and output i. Otherwise, we choose T: we then choose a sample $i \sim D_2$, which will be in $[n_2]$, and then output t_i. Here, D_1, D_2 are the distributions from Lemma 17. Finally, we let $v_i = 1$ if $i \in S$ and $v_i = 0$ if $i \in T$.

The second outcome $\mathcal{O}_2$ is similar but "flipped". Now, we let $S = \{1, 2, \ldots, n_2\} \subseteq [N]$, and $T = \{t_1, \ldots, t_{n_1}\} := [N] \backslash S$. With exactly $1/2$ probability, we choose S: if so, we then choose a sample $i \sim D_2$, and output i. Otherwise, we choose T: we then choose a sample $i \sim D_1$, and then output t_i. Finally, we let $v_i = 1$ if $i \in S$ and $v_i = 0$ if $i \in T$.

Under $\mathcal{O}_1$, we have that $\sum v_i$ always equals n_1, whereas under $\mathcal{O}_2$, we have that $\sum v_i$ is always n_2. In addition, since both n_1 and n_2 are $\frac{N}{2} \cdot (1 \pm O(\gamma))$, and since D_1 and D_2 are γ-pointwise close to uniform, the distribution $\mathcal{Q}$ that we sample from in either case is $O(\gamma)$-pointwise close to uniform. So, we may assume that $\mathcal{P}$ is uniform over $[N]$ in either case.

Now, assume that there exists a uniform algorithm[6] A that draws samples (i, v_i) either from outcome $\mathcal{O}_1$ or outcome $\mathcal{O}_2$ and with probability at least 2/3, computes an estimate of $\sum_i v_i$ up to additive error $c_k \gamma^k N/5$, where c_k is as in the second conclusion of Lemma 17. Observe that when the error bound on A is satisfied (which occurs with probability at least 2/3), A's output distinguishes scenarios $\mathcal{O}_1$ and $\mathcal{O}_2$.

Finally, we observe that distinguishing $\mathcal{O}_1$ from $\mathcal{O}_2$ is sufficient to distinguish the distributions D_1 and D_2. Indeed, under scenario $\mathcal{O}_1$, the 1-values and sampled from D_1, while the 0-values are sampled from D_2, while the roles are reversed in $\mathcal{O}_2$. Thus, the output of A suffices to distinguish D_1 and D_2. Since A uses s samples in expectation, Theorem 16 and Lemma 17 imply that $s = \Omega(n^{1-1/(k+1)})$, as desired. ◀

We now conclude by proving our main technical lemma.

Proof of Lemma 17. First, we note that

$$\sum_{i=0}^{k}(-1)^i\binom{k}{i}\binom{i}{r} = \sum_{i=r}^{k}(-1)^i\binom{k}{i}\binom{i}{r} = \sum_{i=r}^{k}(-1)^i \cdot \frac{k!}{(k-i)!(i-r)!r!}$$
$$= \sum_{i=r}^{k}(-1)^i \cdot \binom{k}{r}\binom{k-r}{i-r} = (-1)^r\binom{k}{r} \cdot \sum_{j=0}^{k-r}(-1)^j\binom{k-r}{j}.$$

The last line follows by setting $j = i - r$. Now, note that the summation in the last line equals $(1-1)^{k-r} = 0$ if $k > r$, and equals 1 if $k = r$. So, this means that $\sum_{i=0}^{k}(-1)^i\binom{k}{i}\binom{i}{r} = 0$ for all $0 \le r < k$.

Next, note that $\binom{i}{r} = \frac{i(i-1)\cdots(i-r+1)}{r!}$ for all integers $i \ge 0$. This is a degree-r polynomial in i. From this observation, it is well-known that every degree at most $k-1$ polynomial in i can be written as a linear combination of $\binom{i}{0}, \dots, \binom{i}{k-1}$. Therefore, for any polynomial P of degree at most $k-1$, $\sum_{i=0}^{k}(-1)^i\binom{k}{i} \cdot P(i) = 0$.

Now, we let the distribution D_1 have exactly a $\frac{\binom{k}{i}}{2^{k-1}}$ fraction of its mass consisting of items each with probability $\left(1 + \frac{\gamma \cdot i}{k}\right) \cdot \frac{1}{n_0}$, for each *even* integer $0 \le i \le k$. Here, n_0 will be an integer chosen later. Note this means it must have $n_0 \cdot \frac{\binom{k}{i}}{2^{k-1} \cdot (1 + \frac{\gamma \cdot i}{k})}$ points with mass $\left(1 + \frac{\gamma \cdot i}{k}\right) \cdot \frac{1}{n_0}$ for each *even* integer $0 \le i \le k$. Likewise, we let the distribution D_2 have exactly a $\frac{\binom{k}{i}}{2^{k-1}}$ fraction of its mass consisting of items each with probability $\left(1 + \frac{\gamma \cdot i}{k}\right) \cdot \frac{1}{n_0}$, for each *odd* integer $0 \le i \le k$. Note that the total fraction of mass for both D_1 and D_2 is clearly 1.

First, we note that for any $1 \le \ell \le k$,

$$\sum_{i=1}^{n_1}(p_{D_1}(i))^\ell - \sum_{i=1}^{n_1}(p_{D_2}(i))^\ell \tag{13}$$
$$= \sum_{\substack{i=0\\ i \text{ even}}}^{k} n_0 \cdot \frac{\binom{k}{i}}{2^{k-1} \cdot (1 + \frac{\gamma \cdot i}{k})} \cdot \left(\left(1 + \frac{\gamma \cdot i}{k}\right) \cdot \frac{1}{n_0}\right)^\ell \tag{14}$$
$$- \sum_{\substack{i=0\\ i \text{ odd}}}^{k} n_0 \cdot \frac{\binom{k}{i}}{2^{k-1} \cdot (1 + \frac{\gamma \cdot i}{k})} \cdot \left(\left(1 + \frac{\gamma \cdot i}{k}\right) \cdot \frac{1}{n_0}\right)^\ell \tag{15}$$

[6] Again, the assumption that A is uniform is without loss of generality by Lemma 15. For our construction, however, the two scenarios $\mathcal{O}_1$ and $\mathcal{O}_2$ can be distinguished by a non-uniform algorithm using $O(\gamma^k)$ samples. Indeed, the two scenarios are distinguished by seeing any value v_i with $n_2 < i \le n_1$. Following the proof of Lemma 15 (cf. Theorem 11.12 in [11]), the scenarios are indistinguishable to even a non-uniform algorithm we we replace S and T with randomly chosen complementary subsets of $[N]$.

$$= \frac{1}{n_0^{\ell-1} \cdot 2^{k-1}} \cdot \sum_{i=0}^{k} (-1)^i \binom{k}{i} \cdot \left(1 + \frac{\gamma \cdot i}{k}\right)^{\ell-1}. \tag{16}$$

By letting $P(i)$ be the polynomial $\left(1+\frac{\gamma \cdot i}{k}\right)^{\ell-1}$, we have that $P(i)$ has degree at most $k-1$, so this equals 0, as desired.

Finally, we look at the difference $n_1 - n_2$, i.e., the difference in support size between D_1 and D_2. This simply equals

$$\sum_{\substack{i=0 \\ i \text{ even}}}^{k} n_0 \cdot \frac{\binom{k}{i}}{2^{k-1} \cdot (1 + \frac{\gamma \cdot i}{k})} - \sum_{\substack{i=0 \\ i \text{ odd}}}^{k} n_0 \cdot \frac{\binom{k}{i}}{2^{k-1} \cdot (1 + \frac{\gamma \cdot i}{k})} = \frac{n_0}{2^{k-1}} \cdot \sum_{i=0}^{k} (-1)^i \cdot \frac{\binom{k}{i}}{1 + \frac{\gamma}{k} \cdot i}. \tag{17}$$

We now inductively prove (by inducting on $k \ge 1$) that $\sum_{i=0}^{k} (-1)^i \cdot \frac{\binom{k}{i}}{a+\gamma \cdot i} = \frac{k! \cdot \gamma^k}{a(a+\gamma)\cdots(a+k\gamma)}$ for any real numbers a, γ. For $k = 1$, we have that $\sum_{i=0}^{k} (-1)^i \cdot \frac{\binom{k}{i}}{a+\gamma \cdot i} = \frac{1}{a} - \frac{1}{a+\gamma} = \frac{\gamma}{a(a+\gamma)}$. For general k, we can write

$$\sum_{i=0}^{k} (-1)^i \cdot \frac{\binom{k}{i}}{a + \gamma \cdot i} = \sum_{i=0}^{k} (-1)^i \cdot \frac{\binom{k-1}{i-1} + \binom{k-1}{i}}{a + \gamma \cdot i} \tag{18}$$

$$= \sum_{i=0}^{k-1} (-1)^i \cdot \frac{\binom{k-1}{i}}{a + \gamma \cdot i} + \sum_{i=1}^{k} (-1)^i \cdot \frac{\binom{k-1}{i-1}}{a + \gamma \cdot i} \tag{19}$$

$$= \sum_{i=0}^{k-1} (-1)^i \cdot \frac{\binom{k-1}{i}}{a + \gamma \cdot i} - \sum_{j=0}^{k-1} (-1)^j \cdot \frac{\binom{k-1}{j}}{(a + \gamma) + \gamma \cdot j}, \tag{20}$$

where we have set $j = i - 1$. We can now use the inductive hypothesis on $k - 1$ to obtain that this equals

$$\frac{(k-1)! \cdot \gamma^{k-1}}{a(a+\gamma)\cdots(a+(k-1)\gamma)} - \frac{(k-1)! \cdot \gamma^{k-1}}{(a+\gamma)\cdots(a+(k-1)\gamma)(a+k\gamma)} \tag{21}$$

$$= (k-1)! \cdot \gamma^{k-1} \cdot \frac{(a+k\gamma) - a}{a(a+\gamma)\cdots(a+(k-1)\gamma)(a+k\gamma)} \tag{22}$$

$$= k! \cdot \gamma^k \cdot \frac{1}{a(a+\gamma)\cdots(a+(k-1)\gamma)(a+k\gamma)}. \tag{23}$$

Therefore, by setting $a = 1$ and replacing γ with $\gamma' = \gamma/k$, we have that the difference in support size between D_1 and D_2 is

$$\frac{n_0}{2^{k-1}} \cdot \frac{k!}{k^k} \cdot \gamma^k \cdot \frac{1}{(1+\gamma/k)(1+2\gamma/k) \cdot (1+\gamma)}.$$

Assuming that $\gamma \le 1/2$, we can apply Stirling's approximation to obtain that this difference is $n_0 \cdot (\gamma/\Theta(1))^k$.

To finish, we will set n_0 appropriately. Note that we wish for D_2 to have support size exactly n. However, all of the points in D_2 has mass between $\frac{1}{n_0}$ and $\frac{1+\gamma}{n_0}$, which means that the support size n_2 must be between $\frac{n_0}{1+\gamma}$ and n_0. So, we can first set n_0, and then choose $n = n_2$ to be $n_0 \cdot \sum_{i=0, i \text{ odd}}^{k} \frac{\binom{k}{i}}{2^{k-1} \cdot (1+\frac{\gamma \cdot i}{k})}$, which is in the range $\left[\frac{n_0}{1+\gamma}, n_0\right]$, and n_1 to be $n_0 \cdot \sum_{i=0, i \text{ even}}^{k} \frac{\binom{k}{i}}{2^{k-1} \cdot (1+\frac{\gamma \cdot i}{k})}$. Both n_1 and $n = n_2$ are in the range $\left[\frac{n_0}{1+\gamma}, n_0\right]$. Indeed, we will have that $\sum_{i=1}^{n_1} (p_{D_1}(i))^\ell = \sum_{i=1}^{n_2} (p_{D_2}(i))^\ell$, and $n_1 - n_2 = \Theta(\gamma)^k \cdot n_0 = \Theta(\gamma)^k \cdot n$. In addition, because all of the values $p_{D_j}(i)$ are in the range $\left[\frac{1}{n_0}, \frac{1+\gamma}{n_0}\right]$ for both $j = 1$ and $j = 2$, this means that they are also in the range $\left[\frac{1-\gamma}{n}, \frac{1+\gamma}{n}\right]$, as desired. This completes the proof. ◀

References

1 Petra Berenbrink, Bruce Krayenhoff, and Frederik Mallmann-Trenn. Estimating the number of connected components in sublinear time. *Information Processing Letters*, 114(11):639–642, 2014.

2 Lorenzo Beretta and Jakub Tětek. Better sum estimation via weighted sampling. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2303–2338. SIAM, 2022.

3 Clément Canonne and Ronitt Rubinfeld. Testing probability distributions underlying aggregated data. In *International Colloquium on Automata, Languages, and Programming*, pages 283–295. Springer, 2014.

4 Clément L. Canonne, Themis Gouleakis, and Ronitt Rubinfeld. Sampling correctors. *SIAM J. Comput.*, 47(4):1373–1423, 2018. doi:10.1137/16M1076666.

5 Edith Cohen, Nick Duffield, Haim Kaplan, Carstent Lund, and Mikkel Thorup. Algorithms and estimators for summarization of unaggregated data streams. *Journal of Computer and System Sciences*, 80(7):1214–1244, 2014.

6 Talya Eden, Piotr Indyk, Shyam Narayanan, Ronitt Rubinfeld, Sandeep Silwal, and Tal Wagner. Learning-based support estimation in sublinear time. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=tilovEHA3YS.

7 Talya Eden, Saleet Mossel, and Ronitt Rubinfeld. Sampling multiple edges efficiently. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, volume 207 of *LIPIcs*, pages 51:1–51:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.APPROX/RANDOM.2021.51.

8 Talya Eden, Dana Ron, and Will Rosenbaum. The arboricity captures the complexity of sampling edges. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 52:1–52:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.52.

9 Talya Eden, Dana Ron, and Will Rosenbaum. Almost optimal bounds for sublinear-time sampling of k-cliques in bounded arboricity graphs. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 56:1–56:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.56.

10 Talya Eden and Will Rosenbaum. On sampling edges almost uniformly. In Raimund Seidel, editor, *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA*, volume 61 of *OASIcs*, pages 7:1–7:9. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/OASIcs.SOSA.2018.7.

11 Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.

12 Morris H Hansen and William N Hurwitz. On the theory of sampling from finite populations. *The Annals of Mathematical Statistics*, 14(4):333–362, 1943.

13 Jonathan Hermon. On sensitivity of uniform mixing times. In *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, volume 54, pages 234–248. Institut Henri Poincaré, 2018.

14 Jacob Holm and Jakub Tětek. Massively parallel computation and sublinear-time algorithms for embedded planar graphs. *arXiv preprint*, 2022. arXiv:2204.09035.

15 D. G. Horvitz and D. J. Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, 47(260):663–685, 1952. doi:10.1080/01621459.1952.10483446.

16 Ben Morris and Yuval Peres. Evolving sets and mixin. 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 279–286. ACM, 2003. doi:10.1145/780542.780585.

17 Rajeev Motwani, Rina Panigrahy, and Ying Xu. Estimating sum by weighted sampling. In *International Colloquium on Automata, Languages, and Programming*, pages 53–64. Springer, 2007.

18 Krzysztof Onak and Xiaorui Sun. Probability-revealing samples. In Amos J. Storkey and Fernando Pérez-Cruz, editors, *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, volume 84 of *Proceedings of Machine Learning Research*, pages 2018–2026. PMLR, 2018. URL: http://proceedings.mlr.press/v84/onak18a.html.

19 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.

20 Jakub Tětek and Mikkel Thorup. Edge sampling and graph parameter estimation via vertex neighborhood accesses. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20–24, 2022*, pages 1116–1129. ACM, 2022. doi:10.1145/3519935.3520059.

A Proof of Lemma 12

Here, we prove the main technical lemma for our sum estimator.

Proof. For each $i \in [N]$ we define γ_i such that $\mathcal{Q}(i) = (1+\gamma_i)\mathcal{P}(i)$. Since we know that $\mathcal{Q}$ is pointwise γ-close to $\mathcal{P}$ then $|\gamma_i| \le \gamma$, and since both $\mathcal{Q}$ and $\mathcal{P}$ are probability distributions then $\sum_{i\in[N]} \gamma_i \mathcal{P}(i) = 0$.

We start by proving the bounds on the expectation, i.e., Equation (9) and Equation (11).

$$\begin{aligned}
\mathrm{E}[\zeta_k] &= \mathrm{E}\left[\sum_{h=1}^{k}(-1)^{h+1}\frac{\binom{k}{h}}{\binom{m}{h}}\sum_{i\in[N]}\binom{Y_i}{h}\mathcal{P}(i)^{-h}x_i\right]\\
&= \sum_{h=1}^{k}(-1)^{h+1}\frac{\binom{k}{h}}{\binom{m}{h}}\sum_{i\in[N]}\mathrm{E}\left[\binom{Y_i}{h}\right]\mathcal{P}(i)^{-h}x_i\ .
\end{aligned}$$

We use the fact that $Y_i = \sum_{j\in[m]}[X_j = i]$ is a sum of 0-1 variables so $\binom{Y_i}{h} = \sum_{I\subseteq[m]:|I|=h}\prod_{j\in I}[X_j = i]$. This implies that $\mathrm{E}\left[\binom{Y_i}{h}\right] = \binom{m}{h}\mathcal{Q}(i)^h = \binom{m}{h}(1+\gamma_i)^h\mathcal{P}(i)^h$. Plugging this in, we get that

$$\begin{aligned}
\mathrm{E}[\zeta_k] &= \sum_{h=1}^{k}(-1)^{h+1}\frac{\binom{k}{h}}{\binom{m}{h}}\sum_{i\in[N]}\binom{m}{h}(1+\gamma_i)^h\mathcal{P}(i)^h\mathcal{P}(i)^{-h}x_i = \sum_{h=1}^{k}(-1)^{h+1}\binom{k}{h}\sum_{i\in[N]}(1+\gamma_i)^h x_i\\
&= \sum_{i\in[N]} x_i\sum_{h=1}^{k}(-1)^{h+1}\binom{k}{h}(1+\gamma_i)^h = \sum_{i\in[N]} x_i\left(1+\sum_{h=0}^{k}(-1)^{h+1}\binom{k}{h}(1+\gamma_i)^h\right)\\
&= \sum_{i\in[N]} x_i\left(1-(1-(1+\gamma_i))^k\right) = \sum_{i\in[N]} x_i\left(1+(-1)^{k+1}\gamma_i^k\right)
\end{aligned}$$

If $k \geq 2$ then we easily see that

$$|\operatorname{E}[\zeta_k] - \mu| = |\sum_{i\in[N]} x_i - \sum_{i\in[N]} x_i(1+(-1)^{k+1}\gamma_i^k)| = |\sum_{i\in[N]} \gamma_i^k x_i| \leq \gamma^k \sum_{i\in[N]} |x_i|$$

For $k = 1$ we will exploit that $\sum_{i\in[N]} \gamma_i \mathcal{P}(i) = 0$.

$$\begin{aligned}|\operatorname{E}[\zeta_1] - \mu| &= |\sum_{i\in[N]} \gamma_i x_i| = |\sum_{i\in[N]} \gamma_i(\mathcal{P}(i)\mu + (x_i - \mathcal{P}(i)\mu))| \\ &= |\sum_{i\in[N]} \gamma_i(x_i - \mathcal{P}(i)\mu)| \leq \gamma \sum_{i\in[N]} |x_i - \mathcal{P}(i)\mu|\end{aligned}$$

Now we will focus on bounding the variance. First we prove Equation (12).

$$\begin{aligned}\operatorname{Var}[\zeta_1] &= \operatorname{E}\left[\left(\frac{1}{m}\sum_{j\in[m]}\sum_{i\in[N]} x_i([X_j = i]\mathcal{P}(i)^{-1} - (1+\gamma_i))\right)^2\right] \\ &= \frac{1}{m^2}\sum_{j\in[m]} \operatorname{E}\left[\left(\sum_{i\in[N]} x_i([X_j = i]\mathcal{P}(i)^{-1} - (1+\gamma_i))\right)^2\right]\end{aligned}$$

We will argue that $\sum_{i\in[N]} x_i([X_j = i]\mathcal{P}(i)^{-1} - (1+\gamma_i)) = \sum_{i\in[N]}(x_i - \mathcal{P}(i)\mu)([X_j = i]\mathcal{P}(i)^{-1} - (1+\gamma_i))$.

$$\begin{aligned}&\sum_{i\in[N]} x_i([X_j = i]\mathcal{P}(i)^{-1} - (1+\gamma_i)) \\ &= \sum_{i\in[N]}(x_i - \mathcal{P}(i)\mu + \mathcal{P}(i)\mu)([X_j = i]\mathcal{P}(i)^{-1} - (1+\gamma_i)) \\ &= \sum_{i\in[N]}(x_i - \mathcal{P}(i)\mu)([X_j = i]\mathcal{P}(i)^{-1} - (1+\gamma_i)) + \mu\sum_{u\in[N]} \mathcal{P}(i)\gamma_i\end{aligned}$$

Now since $\sum_{i\in[N]} \gamma_i \mathcal{P}(i) = 0$ it follows. We can now bound the variance.

$$\begin{aligned}\operatorname{Var}[\zeta_1] &= \frac{1}{m^2}\sum_{j\in[m]} \operatorname{E}\left[\left(\sum_{i\in[N]}(x_i - \mathcal{P}(i)\mu)([X_j = i]\mathcal{P}(i)^{-1} - (1+\gamma_i))\right)^2\right] \\ &\leq \frac{1}{m^2}\sum_{j\in[m]} \operatorname{E}\left[\left(\sum_{i\in[N]} \mathcal{P}(i)^{-1}(x_i - \mathcal{P}(i)\mu)[X_j = i]\right)^2\right] \\ &= \frac{1}{m}\sum_{i\in[N]} \frac{\mathcal{Q}(i)}{\mathcal{P}(i)^2}(x_i - \mathcal{P}(i)\mu)^2 \leq \frac{1+\gamma}{m}\sum_{i\in[N]} \mathcal{P}(i)^{-1}(x_i - \mathcal{P}(i)\mu)^2\end{aligned}$$

Now we just need to focus on the case of $k \geq 2$ and prove Equation (10). For this we need the following lemma for which we defer the proof till Appendix A.

▶ **Lemma 18.** *For all sequences of numbers $(\beta_j)_{j\in[m]}$ and all α the following identity holds:*

$$\sum_{\substack{I\subseteq[m]\\0<|I|\leq k}} (-1)^{|I|+1}\frac{\binom{k}{|I|}}{\binom{m}{|I|}}\left(\prod_{j\in I}\beta_j - (1+\alpha)^h\right) = (-1)^{k+1}\sum_{\substack{I\subseteq[m]\\0<|I|\leq k}} \frac{\binom{k}{|I|}}{\binom{m}{|I|}}\alpha^{k-|I|}\prod_{j\in I}(\beta_j - (1+\alpha))$$

The idea is to use Lemma 18 to prove that

$$\zeta_k - \mathrm{E}[\zeta_k] = \sum_{\substack{I \subseteq [m] \\ 0<|I|\leq k}} \frac{\binom{m-|I|}{k-|I|}}{\binom{m}{k}} \alpha^{k-|I|} \sum_{i \in [N]} x_i \prod_{j \in I} (\mathcal{P}(i)^{-1}[X_j = i] - (1+\gamma_i)) \,. \tag{24}$$

First we use that $\mathrm{E}[\zeta_k] = \sum_{h=1}^{k} (-1)^h \frac{\binom{k}{h}}{\binom{m}{h}} \sum_{i \in [N]} \mathcal{P}(i)^{-h} x_i \, \mathrm{E}\left[\binom{Y_i}{h}\right]$ which allow us to rewrite $\zeta_k - \mathrm{E}[\zeta_k]$.

$$\zeta_k - \mathrm{E}[\zeta_k] = \sum_{h=1}^{k} (-1)^h \frac{\binom{k}{h}}{\binom{m}{h}} \sum_{i \in [N]} \mathcal{P}(i)^{-h} x_i \left(\binom{Y_i}{h} - \mathrm{E}\left[\binom{Y_i}{h}\right] \right)$$

We now again use that $Y_i = \sum_{j \in [m]} [X_j = i]$ is a sum of 0-1 variables so $\binom{Y_i}{h} = \sum_{I \subseteq [m] : |I| = h} \prod_{j \in I} [X_j = i]$ and $\mathrm{E}\left[\binom{Y_i}{h}\right] = \binom{m}{h} (1+\gamma_i)^h \mathcal{P}(i)^h$.

$$\begin{aligned}
&\sum_{h=1}^{k} (-1)^h \frac{\binom{k}{h}}{\binom{m}{h}} \sum_{i \in [N]} \mathcal{P}(i)^{-h} x_i \left(\binom{Y_i}{h} - \mathrm{E}\left[\binom{Y_i}{h}\right] \right) \\
&= \sum_{h=1}^{k} (-1)^h \frac{\binom{k}{h}}{\binom{m}{h}} \sum_{i \in [N]} \mathcal{P}(i)^{-h} x_i \sum_{\substack{I \subseteq [m] \\ |I| = h}} \left(\prod_{j \in I} [X_j = i] - (1+\gamma_i)^h \mathcal{P}(i)^h \right) \\
&= \sum_{i \in [N]} x_i \sum_{h=1}^{k} \sum_{\substack{I \subseteq [m] \\ |I| = h}} (-1)^h \frac{\binom{k}{h}}{\binom{m}{h}} \mathcal{P}(i)^{-h} \left(\prod_{j \in I} [X_j = i] - (1+\gamma_i)^h \mathcal{P}(i)^h \right) \\
&= \sum_{i \in [N]} x_i \sum_{\substack{I \subseteq [m] \\ 0<|I|\leq k}} (-1)^{|I|+1} \frac{\binom{k}{|I|}}{\binom{m}{|I|}} \mathcal{P}(i)^{-h} \left(\prod_{j \in I} [X_j = i] - (1+\gamma_i)^{|I|} \mathcal{P}(i)^{|I|} \right) \\
&= \sum_{i \in [N]} x_i \sum_{\substack{I \subseteq [m] \\ 0<|I|\leq k}} (-1)^{|I|+1} \frac{\binom{k}{|I|}}{\binom{m}{|I|}} \left(\prod_{j \in I} \mathcal{P}(i)^{-1} [X_j = i] - (1+\gamma_i)^{|I|} \right)
\end{aligned}$$

For each $i \in [N]$, the expression $\sum_{\substack{I \subseteq [m] \\ 0<|I|\leq k}} (-1)^{|I|+1} \frac{\binom{k}{|I|}}{\binom{m}{|I|}} \left(\prod_{j \in I} \mathcal{P}(i)^{-1} [X_j = i] - (1+\gamma_i)^{|I|} \right)$ is of the form of Lemma 18. So applying that N times we get that

$$\begin{aligned}
&\sum_{i \in [N]} x_i \sum_{\substack{I \subseteq [m] \\ 0<|I|\leq k}} (-1)^{|I|+1} \frac{\binom{k}{|I|}}{\binom{m}{|I|}} \left(\prod_{j \in I} \mathcal{P}(i)^{-1} [X_j = i] - (1+\gamma_i)^{|I|} \right) \\
&= \sum_{i \in [N]} x_i (-1)^{k+1} \sum_{\substack{I \subseteq [m] \\ 0<|I|\leq k}} \frac{\binom{k}{|I|}}{\binom{m}{|I|}} \gamma_i^{k-|I|} \prod_{j \in I} \left(\mathcal{P}(i)^{-1} [X_j = i] - (1+\gamma_i) \right) \\
&= (-1)^{k+1} \sum_{\substack{I \subseteq [m] \\ 0<|I|\leq k}} \frac{\binom{k}{|I|}}{\binom{m}{|I|}} \sum_{i \in [N]} \gamma_i^{k-|I|} x_i \prod_{j \in I} \left(\mathcal{P}(i)^{-1} [X_j = i] - (1+\gamma_i) \right)
\end{aligned}$$

This shows that Equation (24) is true.

Now let $I_1, I_2 \subseteq [m]$ be two different set of indices with $0 < |I_1|, |I_2| \leq k$. Without loss of generality, we can assume that there exists $h \in I_1 \setminus I_2$.

$$\begin{aligned} T = (&\sum_{i\in[N]} \gamma_i^{k-|I_1|} x_i \prod_{j\in I_1} (\mathcal{P}(i)^{-1}[X_j = i] - (1+\gamma_i))) \\ &\cdot (\sum_{i\in[N]} \gamma_i^{k-|I_2|} x_i \prod_{j\in I_2} (\mathcal{P}(i)^{-1}[X_j = i] - (1+\gamma_i))) \end{aligned}$$

Note that if we multiply this expression out then every term will contain a factor of the form $(\mathcal{P}(i)^{-1}[X_h = s] - (1+\gamma_i))$ for some $s \in [N]$ and where all the other factors are independent of X_h. Since $\mathrm{E}[(\mathcal{P}(i)^{-1}[X_h = s] - (1+\gamma_i))] = 0$ we get that $\mathrm{E}[T] = 0$. This implies that

$$\mathrm{E}[(\zeta_k - \mathrm{E}[\zeta_k])^2] = \sum_{\substack{I\subseteq[m] \\ 0<|I|\leq k}} \left(\frac{\binom{k}{|I|}}{\binom{m}{|I|}}\right)^2 \mathrm{E}[(\sum_{i\in[N]} \gamma_i^{k-|I|} x_i \prod_{j\in I} (\mathcal{P}(i)^{-1}[X_j = i] - (1+\gamma_i)))^2]$$

Let $I \subseteq [N]$ be fixed then

$$\begin{aligned} &\mathrm{E}[(\sum_{i\in[N]} \gamma_i^{k-|I|} x_i \prod_{j\in I} (\mathcal{P}(i)^{-1}[X_j = i] - (1+\gamma_i)))^2] \leq \mathrm{E}[(\sum_{i\in[N]} \gamma_i^{k-|I|} x_i \prod_{j\in I} \mathcal{P}(i)^{-1}[X_j = i])^2] \\ &= \sum_{i\in[N]} \gamma_i^{2k-2|I|} x_i^2 \frac{\mathcal{Q}(i)^{|I|}}{\mathcal{P}(i)^{2|I|}} \leq \gamma^{2k-2|I|} \sum_{i\in[N]} x_i^2 \frac{(1+\gamma_i)^{|I|}}{\mathcal{P}(i)^{|I|}} \end{aligned}$$

Collecting terms we get that

$$\begin{aligned} &\mathrm{E}[(\zeta_k - \mathrm{E}[\zeta_k])^2] \leq \sum_{\substack{I\subseteq[m] \\ 0<|I|\leq k}} \left(\frac{\binom{k}{|I|}}{\binom{m}{|I|}}\right)^2 \gamma^{2k-2|I|} \sum_{i\in[N]} x_i^2 \frac{(1+\gamma_i)^{|I|}}{\mathcal{P}(i)^{|I|}} \\ &= \sum_{h=1}^{k} \frac{\binom{k}{h}^2}{\binom{m}{h}} \gamma^{2k-2h} \sum_{i\in[N]} x_i^2 \frac{(1+\gamma_i)^h}{\mathcal{P}(i)^h} \leq \sum_{h=1}^{k} \frac{\binom{k}{h}^2}{\binom{m}{h}} \gamma^{2k-2h} (1+\gamma)^h \sum_{i\in[N]} \frac{x_i^2}{\mathcal{P}(i)^h} \\ &\leq \sum_{h=1}^{k} \frac{k^{2h}}{\binom{m}{h}} \gamma^{2k-2h} (1+\gamma)^h \sum_{i\in[N]} \frac{x_i^2}{\mathcal{P}(i)^h} \leq \max_{h=1}^{k} \frac{2^h k^{2h}}{\binom{m}{h}} \gamma^{2k-2h} (1+\gamma)^h \sum_{i\in[N]} \frac{x_i^2}{\mathcal{P}(i)^h} \end{aligned}$$

Now we note that for all $i \in [N]$, the map $h \mapsto \frac{2^h k^{2h}}{\binom{m}{h}} \gamma^{2k-2h} (1+\gamma)^h \frac{x_i^2}{\mathcal{P}(i)^h}$ is log-convex since each factor is log-convex. This implies that the map $h \mapsto \frac{2^h k^{2h}}{\binom{m}{h}} \gamma^{2k-2h} (1+\gamma)^h \sum_{i\in[N]} \frac{x_i^2}{\mathcal{P}(i)^h}$ is convex and is thus maximized at the boundary. We then get that

$$\begin{aligned} \mathrm{E}[(\zeta_k - \mathrm{E}[\zeta_k])^2] \leq \max\Big\{&2(1+\gamma)\gamma^{2k-2} k^2 \frac{\sum_{i\in[N]} \mathcal{P}(i)^{-1} x_i^2}{m}, \\ &2^k (1+\gamma)^k k^{3k} \frac{n^{k-1} \sum_{i\in[N]} \mathcal{P}(i)^{-1} x_i^2}{m^k}\Big\} \end{aligned}$$

which finishes the proof of Equation (10). ◀

Finally, we must prove Lemma 18 which will finish the proof of Lemma 12.

▶ **Lemma 19.** *For all sequences of numbers $(\beta_j)_{j\in[m]}$ and all α the following identity holds:*

$$\sum_{\substack{I\subseteq[m] \\ 0<|I|\leq k}} (-1)^{|I|+1} \frac{\binom{k}{|I|}}{\binom{m}{|I|}} \left(\prod_{j\in I} \beta_j - (1+\alpha)^h\right) = (-1)^{k+1} \sum_{\substack{I\subseteq[m] \\ 0<|I|\leq k}} \frac{\binom{k}{|I|}}{\binom{m}{|I|}} \alpha^{k-|I|} \prod_{j\in I} (\beta_j - (1+\alpha))$$

First we need a simple lemma.

▶ **Lemma 19.** *For all sequences of numbers $(\beta_j)_{j\in I}$ and all α the following identity holds:*

$$\prod_{j\in I}\beta_j-(1+\alpha)^{|I|}=\sum_{\emptyset\neq J\subseteq I}(1+\alpha)^{|I|-|J|}\prod_{j\in J}(\beta_j-(1+\alpha))$$

Proof. Let $\beta'_j:=\beta_j-(1+\alpha)$. Then,

$$\begin{aligned}\prod_{j\in I}\beta'_j-(1+\alpha)^{|I|}&=\prod_{j\in I}(\beta'_j+(1+\alpha))-(1+\alpha)^{|I|}\\&=\left[\sum_{J\subseteq I}(1+\alpha)^{|I|-|J|}\prod_{j\in J}\beta'_j\right]-(1+\alpha)^{|I|}\\&=\sum_{\emptyset\neq J\subseteq I}(1+\alpha)^{|I|-|J|}\prod_{j\in J}(\beta_j-(1+\alpha)).\end{aligned}$$ ◀

Proof of Lemma 18. We start by applying Lemma 19

$$\begin{aligned}&\sum_{\substack{I\subseteq[m]\\0<|I|\le k}}(-1)^{|I|+1}\frac{\binom{k}{|I|}}{\binom{m}{|I|}}\left(\prod_{j\in I}\beta_j-(1+\alpha)^h\right)\\&\quad=\sum_{\substack{I\subseteq[m]\\0<|I|\le k}}(-1)^{|I|+1}\frac{\binom{k}{|I|}}{\binom{m}{|I|}}\sum_{\emptyset\neq J\subseteq I}(1+\alpha)^{|I|-|J|}\prod_{j\in J}(\beta_j-(1+\alpha))\end{aligned}$$

We use that $\frac{\binom{k}{|I|}}{\binom{m}{|I|}}=\frac{\binom{m-|I|}{k-|I|}}{\binom{m}{k}}$ which follows from the fact that $\binom{m}{k}\binom{k}{|I|}=\binom{m}{|I|,k-|I|,m-k}=\binom{m}{|I|}\binom{m-|I|}{k-|I|}$.

$$\begin{aligned}&\sum_{\substack{I\subseteq[m]\\0<|I|\le k}}(-1)^{|I|+1}\frac{\binom{k}{|I|}}{\binom{m}{|I|}}\sum_{\emptyset\neq J\subseteq I}(1+\alpha)^{|I|-|J|}\prod_{j\in J}(\beta_j-(1+\alpha))\\&=\sum_{\substack{I\subseteq[m]\\0<|I|\le k}}(-1)^{|I|+1}\frac{\binom{m-|I|}{k-|I|}}{\binom{m}{k}}\sum_{\emptyset\neq J\subseteq I}(1+\alpha)^{|I|-|J|}\prod_{j\in J}(\beta_j-(1+\alpha))\\&=\frac{1}{\binom{m}{k}}\sum_{\substack{J\subseteq[m]\\0<|J|\le k}}\left(\prod_{j\in J}(\beta_j-(1+\alpha))\right)\sum_{\substack{I\subseteq[m]\\J\subseteq I,|I|\le k}}(-1)^{|I|+1}\binom{m-|I|}{k-|I|}(1+\alpha)^{|I|-|J|}\\&=\frac{1}{\binom{m}{k}}\sum_{\substack{J\subseteq[m]\\0<|J|\le k}}\left(\prod_{j\in J}(\beta_j-(1+\alpha))\right)\sum_{h=|J|}^{k}(-1)^{h+1}\binom{m-|J|}{h-|J|}\binom{m-h}{k-h}(1+\alpha)^{h-|J|}\end{aligned}$$

Now we use that $\binom{m-h}{k-h}\binom{m-|J|}{h-|J|} = \binom{m-|J|}{k-h,m-k,h-|J|} = \binom{m-|J|}{k-|J|}\binom{k-|J|}{h-|J|}$.

$$\begin{aligned}
&\frac{1}{\binom{m}{k}} \sum_{\substack{J\subseteq[m]\\0<|J|\le k}} \left(\prod_{j\in J}(\beta_j-(1+\alpha))\right) \sum_{h=|J|}^{k} (-1)^{h+1}\binom{m-|J|}{h-|J|}\binom{m-h}{k-h}(1+\alpha)^{h-|J|}\\
&= \frac{1}{\binom{m}{k}} \sum_{\substack{J\subseteq[m]\\0<|J|\le k}} \left(\prod_{j\in J}(\beta_j-(1+\alpha))\right) \sum_{h=|J|}^{k} (-1)^{h+1}\binom{m-|J|}{k-|J|}\binom{k-|J|}{h-|J|}(1+\alpha)^{h-|J|}\\
&= \frac{1}{\binom{m}{k}} \sum_{\substack{J\subseteq[m]\\0<|J|\le k}} (-1)^{|J|+1}\binom{m-|J|}{k-|J|} \left(\prod_{j\in J}(\beta_j-(1+\alpha))\right) \sum_{h=0}^{k-|J|} (-1)^{h}\binom{k-|J|}{h}(1+\alpha)^{h}\\
&= \frac{1}{\binom{m}{k}} \sum_{\substack{J\subseteq[m]\\0<|J|\le k}} (-1)^{|J|+1}\binom{m-|J|}{k-|J|} \left(\prod_{j\in J}(\beta_j-(1+\alpha))\right) (1-(1+\alpha))^{k-|J|}\\
&= \frac{(-1)^{k+1}}{\binom{m}{k}} \sum_{\substack{J\subseteq[m]\\0<|J|\le k}} \binom{m-|J|}{k-|J|}\alpha^{k-|J|} \left(\prod_{j\in J}(\beta_j-(1+\alpha))\right)
\end{aligned}$$

Finally, we again use that $\frac{\binom{k}{|I|}}{\binom{m}{|I|}} = \frac{\binom{m-|I|}{k-|I|}}{\binom{m}{k}}$ to finish the proof.

$$\begin{aligned}
&\frac{(-1)^{k+1}}{\binom{m}{k}} \sum_{\substack{J\subseteq[m]\\0<|J|\le k}} \binom{m-|J|}{k-|J|}\alpha^{k-|J|} \left(\prod_{j\in J}(\beta_j-(1+\alpha))\right)\\
&\quad = (-1)^{k+1} \sum_{\substack{J\subseteq[m]\\0<|J|\le k}} \frac{\binom{k}{h}}{\binom{m}{h}}\alpha^{k-|J|} \left(\prod_{j\in J}(\beta_j-(1+\alpha))\right)
\end{aligned}$$

◀

On the Composition of Randomized Query Complexity and Approximate Degree

Sourav Chakraborty
Indian Statistical Institute, Kolkata, India

Chandrima Kayal
Indian Statistical Institute, Kolkata, India

Rajat Mittal
Indian Institute of Technology Kanpur, India

Manaswi Paraashar
Aarhus University, Denmark

Swagato Sanyal
Indian Institute of Technology Kharagpur, India

Nitin Saurabh
Indian Institute of Technology Hyderabad, India

Abstract

For any Boolean functions f and g, the question whether $\mathrm{R}(f \circ g) = \widetilde{\Theta}(\mathrm{R}(f) \cdot \mathrm{R}(g))$, is known as the composition question for the randomized query complexity. Similarly, the composition question for the approximate degree asks whether $\widetilde{\deg}(f \circ g) = \widetilde{\Theta}(\widetilde{\deg}(f) \cdot \widetilde{\deg}(g))$. These questions are two of the most important and well-studied problems in the field of analysis of Boolean functions, and yet we are far from answering them satisfactorily.

It is known that the measures compose if one assumes various properties of the outer function f (or inner function g). This paper extends the class of outer functions for which R and $\widetilde{\deg}$ compose.

A recent landmark result (Ben-David and Blais, 2020) showed that $\mathrm{R}(f \circ g) = \Omega(\mathrm{noisyR}(f) \cdot \mathrm{R}(g))$. This implies that composition holds whenever $\mathrm{noisyR}(f) = \widetilde{\Theta}(\mathrm{R}(f))$. We show two results:

1. When $\mathrm{R}(f) = \Theta(n)$, then $\mathrm{noisyR}(f) = \Theta(\mathrm{R}(f))$. In other words, composition holds whenever the randomized query complexity of the outer function is full.
2. If R composes with respect to an outer function, then noisyR also composes with respect to the same outer function.

On the other hand, no result of the type $\widetilde{\deg}(f \circ g) = \Omega(M(f) \cdot \widetilde{\deg}(g))$ (for some non-trivial complexity measure $M(\cdot)$) was known to the best of our knowledge. We prove that

$$\widetilde{\deg}(f \circ g) = \widetilde{\Omega}(\sqrt{\mathrm{bs}(f)} \cdot \widetilde{\deg}(g)),$$

where $\mathrm{bs}(f)$ is the block sensitivity of f. This implies that $\widetilde{\deg}$ composes when $\widetilde{\deg}(f)$ is asymptotically equal to $\sqrt{\mathrm{bs}(f)}$.

It is already known that both R and $\widetilde{\deg}$ compose when the outer function is symmetric. We also extend these results to weaker notions of symmetry with respect to the outer function.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Approximate degree, Boolean functions, Composition Theorem, Partial functions, Randomized Query Complexity

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.63

Category RANDOM

Related Version *Full Version*: https://eccc.weizmann.ac.il/report/2023/099/

Funding *Swagato Sanyal*: supported by an ISIRD grant by Sponsored Research and Industrial Consultancy, IIT Kharagpur.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 63; pp. 63:1–63:23
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

For studying the complexity of Boolean functions, a number of simple complexity measures (like decision tree complexity, randomized query complexity, degree, certificate complexity and so on) have been studied over the years. (Refer to the survey [15] for an introduction to complexity measures of Boolean functions.) Understanding how these measures are related to each other [1, 2, 4, 27], and how they behave for various classes of Boolean functions has been a central area of research in complexity theory [34, 23, 42].

A crucial step towards understanding a complexity measure is: how does the complexity measure behave when two Boolean functions are combined to obtain a new function (i.e., what is the relationship between the measure of the resulting function and the measures of the two individual functions) [16, 12, 25, 44]? One particularly natural combination of functions is called *composition*, and it takes a central role in complexity theory.

For any two Boolean functions $f : \{0,1\}^n \to \{0,1\}$ and $g : \{0,1\}^m \to \{0,1\}$, the composed function $f \circ g : \{0,1\}^{nm} \to \{0,1\}$ is defined as the function

$$f \circ g(x_1, \ldots, x_n) = f(g(x_1), \ldots, g(x_n)),$$

where $x_i \in \{0,1\}^m$ for $i \in [n]$. For the function $f \circ g$, the function f is called the outer function and g is called the inner function. See Definition 13 for a natural extension to partial functions.

Let $M(\cdot)$ be a complexity measure of Boolean functions. A central question in complexity theory is the following.

▶ **Question 1** (Composition question for M). *Is the following true for all Boolean functions f and g:*

$$M(f \circ g) = \widetilde{\Theta}(M(f) \cdot M(g))?$$

The notation $\widetilde{\Theta}(\cdot)$ hides poly-logarithmic factors of the arity of the outer function f.

Composition of Boolean functions with respect to different complexity measures is a very important and useful tool in areas like communication complexity, circuit complexity and many more. To take an example, a popular application of composition is to create new functions demonstrating better separations (refer to [33, 44, 3, 25] for some other results of similar flavour).

It is known that the answer to the composition question is in the affirmative for complexity measures like deterministic decision tree complexity [37, 44, 31], degree [44] and quantum query complexity [35, 29, 28]. While it is well understood how several complexity measures behave under composition, there are two important measures (even though well studied) for which this problems remains wide open: randomized query complexity (denoted by R) and approximate degree (denoted by $\widetilde{\deg}$) [38, 33, 3, 39, 17, 40]. (See Definition 30 and Definition 31 for their respective formal definitions.)

For both R and $\widetilde{\deg}$ the upper bound inequality is known, i.e., $\mathrm{R}(f \circ g) = \widetilde{O}(\mathrm{R}(f) \cdot \mathrm{R}(g))$ (folklore) and $\widetilde{\deg}(f \circ g) = O(\widetilde{\deg}(f) \cdot \widetilde{\deg}(g))$ [41]. Thus it is enough to prove the lower bound on the complexity of composed function in terms of the individual functions. Most of the attempts to prove this direction of the question have focused on special cases when the outer function has certain special properties[1].

[1] We note that some works have also studied the composition of R and $\widetilde{\deg}$ when the inner functions have special properties, for example, [1, 13, 6, 24, 30, 10].

The initial steps taken towards answering the composition question for R were to show a lower bound by using a weaker measure of outer function than the randomized query complexity. In particular, it was shown that $\mathrm{R}(f \circ g) = \Omega(\mathrm{s}(f) \cdot \mathrm{R}(g))$ [26, 5], where $\mathrm{s}(f)$ is the sensitivity of f (Definition 32). Since $\mathrm{s}(f) = \Theta(\mathrm{R}(f))$ for any symmetric function[2] f, these results show that R composes when the outer function is a symmetric function (like OR, AND, Majority, Parity, etc.). The lower bound was later improved to obtain $\mathrm{R}(f \circ g) = \Omega(\mathrm{fbs}(f) \cdot \mathrm{R}(g))$ [7, 8], where $\mathrm{fbs}(f)$ is the fractional block sensitivity of f (Definition 33).

Unfortunately, at this stage, there could even be a cubic gap between R and fbs; the best known bound is $\mathrm{R}(f) = O(\mathrm{fbs}(f)^3)$ [2]. Given that there are already known Boolean functions with quadratic gap between $\mathrm{fbs}(f)$ and $\mathrm{R}(f)$ (e.g., BKK function [1]), it is natural to consider composition question for randomized query complexity when R is *big* but fbs is *small*. We take a step towards this problem by showing that composition for R holds when the outer function has full randomized query complexity, i.e., $\mathrm{R}(f) = \Theta(n)$, where n is the arity of the outer function f.

For composition of $\widetilde{\deg}$, Sherstov [38] already showed that $\widetilde{\deg}(f \circ g)$ composes when the approximate degree of the outer function f is $\Theta(n)$, where n is the arity of the outer function. Thus approximate degree composes for several symmetric functions (having approximate degree $\Theta(n)$, like Majority and Parity). Though, until recently it was not even clear if $\widetilde{\deg}(\mathsf{OR} \circ \mathsf{AND}) = \Omega(\widetilde{\deg}(\mathsf{OR})\,\widetilde{\deg}(\mathsf{AND}))$ (arguably the simplest of composed functions). OR has approximate degree $O(\sqrt{n})$, and thus the result of [38] does not prove $\widetilde{\deg}$ composition when the outer function is OR (similarly for AND). In a long series of work by [33, 3, 39, 17, 40], the question was finally resolved; it was later generalized to the case when the outer function is symmetric [11].

In contrast to R composition, no lower bound on the approximate degree of composed function is known with a weaker measure on the outer function. It is well known that for any Boolean function f, $\sqrt{\mathrm{s}(f)} \le \sqrt{\mathrm{bs}(f)} = O(\widetilde{\deg}(f))$ [33]. So a natural step towards proving $\widetilde{\deg}$ composition is: prove a lower bound on $\widetilde{\deg}(f \circ g)$ by $\sqrt{\mathrm{bs}(f)} \cdot \widetilde{\deg}(g)$. We show this result by generalizing the approach of [11].

While the techniques used for the composition of R and $\widetilde{\deg}$ are quite different, one can still observe similarities between the classes of outer functions for which the composition of R and $\widetilde{\deg}$ is known to hold respectively. We further expand these similarities, by extending the classes of outer functions for which the composition theorem hold.

2 Our Results and Techniques

It is well-known, by amplification, that $\mathrm{R}(f \circ g) = O(\mathrm{R}(f) \cdot \mathrm{R}(g) \cdot \log \mathrm{R}(f))$. In the case of approximate degree, Shrestov [41] showed that $\widetilde{\deg}(f \circ g) = O(\widetilde{\deg}(f) \cdot \widetilde{\deg}(g))$. So, to answer the composition question for R (or $\widetilde{\deg}$), we are only concerned about proving a lower bound on $\mathrm{R}(f \circ g)$ (or $\widetilde{\deg}(f \circ g)$) in terms of $\mathrm{R}(f)$ and $\mathrm{R}(g)$ (or $\widetilde{\deg}(f)$ and $\widetilde{\deg}(g)$) respectively.

We split our results into three parts. In the first part we prove the tight lower bound on $\mathrm{R}(f \circ g)$ when the outer function has full randomized complexity. In the second part we provide a tight lower bound on $\widetilde{\deg}(f \circ g)$ in terms of $\mathrm{bs}(f)$ and $\widetilde{\deg}(g)$. Our results on the lower bound of $\mathrm{R}(f \circ g)$ and $\widetilde{\deg}(f \circ g)$ are summarized in Table 1. Finally, we also prove composition theorems for R and $\widetilde{\deg}$ when the outer functions have a slightly relaxed notion of symmetry.

[2] Functions that depend only on the Hamming weight of their input.

Table 1 Composition of R and $\widetilde{\deg}$ depending on the complexity of the outer function in terms of block-sensitivity and arity.

	In terms of $\mathrm{bs}(f)$	In terms of arity of f
R	$\mathrm{R}(f \circ g) = \widetilde{\Omega}(\mathrm{bs}(f) \cdot \mathrm{R}(g))$ [26]	$\mathrm{R}(f \circ g) = \widetilde{\Omega}(\mathrm{R}(f) \cdot \mathrm{R}(g))$ when $\mathrm{R}(f) = \Theta(n)$ Theorem 2
$\widetilde{\deg}$	$\widetilde{\deg}(f \circ g) = \widetilde{\Omega}(\sqrt{\mathrm{bs}(f)} \cdot \widetilde{\deg}(g))$ Theorem 7	$\widetilde{\deg}(f \circ g) = \widetilde{\Omega}(\widetilde{\deg}(f) \cdot \widetilde{\deg}(g))$ when $\widetilde{\deg}(f) = \Theta(n)$ [38]

2.1 Lower bounds on $\mathrm{R}(f \circ g)$ when the outer function has full randomized query complexity

Sherstov [38] proved that $\widetilde{\deg}(f \circ g) = \Omega(\widetilde{\deg}(f) \cdot \widetilde{\deg}(g))$ when the approximate degree of the outer function f is $\Theta(n)$, where n is the arity of f. Though, a corresponding result for the case of randomized query complexity was not known. Our main result is to prove the corresponding theorem for randomized query complexity.

▶ **Theorem 2.** *Let f be a partial Boolean function on n-bits such that $\mathrm{R}(f) = \Theta(n)$. Then for all partial functions g, we have*

$$\mathrm{R}(f \circ g) = \Omega(\mathrm{R}(f) \cdot \mathrm{R}(g)).$$

The proof of this theorem is given in Section 4. Notice, since $\mathrm{R}(f \circ g) = O(\mathrm{R}(f) \cdot \mathrm{R}(g) \log \mathrm{R}(f))$ (by error reduction), Theorem 2 says that composition of R holds when the randomized query complexity of the outer function, f, is $\Theta(n)$. Next, we give main ideas behind the proof of the above theorem.

Ideas behind proof of Theorem 2

A crucial complexity measure that we use for the proof of Theorem 2 is called the *noisy randomized query complexity*, first introduced by Ben-David and Blais [9] in order to study randomized query complexity. Noisy randomized query complexity can be seen as a generalization of randomized query complexity where the algorithm can query a bit with any bias and only pays proportionally to the square of the bias in terms of cost (see Definition 16). They give the following characterization of $\mathrm{noisyR}(f)$ (the noisy randomized query complexity of f).

▶ **Theorem 3** (Ben-David and Blais [9])**.** *For all partial functions f on n-bits, we have*

$$\mathrm{noisyR}(f) = \Theta\left(\frac{\mathrm{R}(f \circ \mathrm{GapMaj}_n)}{n}\right), \tag{1}$$

where GapMaj_n is the Gap-Majority function on n bits whose input is promised to have Hamming weight either $(n/2 + 2\sqrt{n})$ (in which case the output is -1) or $(n/2 - 2\sqrt{n})$ (in which case the output is 1).

We want to point out that the arity of f and Gap-Majority is the *same* in Theorem 3. Towards a proof of Theorem 2, we first make the following crucial observation.

▶ **Observation 4.** *Let f be a partial Boolean function on n bits. If $t(n) \geq 1$ is a non-decreasing function of n and*

$$\mathrm{noisyR}(f) = \Omega\left(\frac{\mathrm{R}(f \circ \mathrm{GapMaj}_{t(n)})}{t(n)}\right),$$

then $\mathrm{R}(f \circ g) = \Omega((\mathrm{R}(f) \cdot \mathrm{R}(g))/t(n))$ for all partial functions g.

In particular choosing $t(n)$ to be $(\log n)$, if the outer function f satisfies

$$\mathrm{noisyR}(f) = \Omega\left(\frac{\mathrm{R}(f \circ \mathrm{GapMaj}_{\log n})}{\log n}\right). \tag{2}$$

then the above observation gives $\mathrm{R}(f \circ g) = \Omega((\mathrm{R}(f) \cdot \mathrm{R}(g))/(\log n))$ for all partial functions g.

The Observation 4 allows us to approach the composition question for randomized query complexity in a conceptually fresh manner. The goal of proving that randomized query complexity composes for a function or a class of functions, say upto $(\log n)$-factor, reduces to showing that Equation 2 holds for that function or class of functions for $t(n) = \log n$.

We are able to show that Equation 2 holds for all non-decreasing functions $t(n)$ with a slight overhead.

▶ **Theorem 5.** *Let f be a partial function on n bits and let $t \geq 1$, then* $\mathrm{R}(f \circ \mathrm{GapMaj}_t) = O\left(t \cdot \mathrm{noisyR}(f) + n\right)$.

Notice that this is a generalization of Ben-David and Blais' characterization of noisyR given by Theorem 3 in one direction. To give an idea of the proof, their characterization (Theorem 3) shows that any noisy oracle algorithm for f can be simulated using only two biases, 1 and $1/\sqrt{n}$ (where n is the arity of f), with only constant overhead. We generalize this by showing that the same simulation works with a slight overhead even when the bias $1/\sqrt{n}$ is replaced by a bias $1/\sqrt{t}$, for some $t \geq 1$. A detailed proof the above theorem has been included in the full version of this paper [20].

This seemingly inconsequential generalization allows us to complete the proof of Theorem 2, i.e. if for an n-bit partial function f, $\mathrm{R}(f) = \Theta(n)$, then $\mathrm{R}(f \circ g) = \widetilde{\Theta}(\mathrm{R}(f) \cdot \mathrm{R}(g))$ for all partial functions g (see Section 4 for details).

Furthermore, Theorem 5 even sheds light on *the composition question for* noisyR. A corollary of this theorem is that if R composes with respect to an outer function, then noisyR also composes with respect to the same outer function (see Section 4 for a proof).

▶ **Corollary 6.** *Let f be a partial Boolean function. If* $\mathrm{R}(f \circ g) = \widetilde{\Theta}(\mathrm{R}(f) \cdot \mathrm{R}(g))$ *for all partial functions g then* $\mathrm{noisyR}(f \circ g) = \widetilde{\Theta}(\mathrm{noisyR}(f) \cdot \mathrm{noisyR}(g))$.

2.2 Lower bound on $\widetilde{\deg}(f \circ g)$ in terms of block sensitivity of f and $\widetilde{\deg}(g)$

As discussed in the introduction, the composition question for $\widetilde{\deg}$ is only known to hold when the outer function f is symmetric [11] or has high approximate degree [38]. There are also no known lower bounds on $\widetilde{\deg}(f \circ g)$ in terms of weaker measures of f and $\widetilde{\deg}(g)$. Compare this with the situation with respect to composition of R. It was shown in [26] that $\mathrm{R}(f \circ g) = \Omega(\mathrm{s}(f)\,\mathrm{R}(g))$, where $\mathrm{s}(f)$ denotes the sensitivity of f. This was later strengthened to $\Omega(\mathrm{fbs}(f)\,\mathrm{R}(g))$ [7, 8], where $\mathrm{fbs}(f)$ is the fractional block sensitivity of f.

In this second part we show analogous lower bounds on approximate degree of composed function $f \circ g$. Our main result here is the following.

▶ **Theorem 7.** *For all non-constant (possibly partial)*[3] *Boolean functions* $f : \{0,1\}^n \to \{0,1\}$ *and* $g : \{0,1\}^m \to \{0,1\}$, *we have*

$$\widetilde{\deg}(f \circ g) = \widetilde{\Omega}(\sqrt{\mathrm{bs}(f)} \cdot \widetilde{\deg}(g)).$$

[3] For definitions of block sensitivity and approximate degree in the context of partial functions, please see Definitions 32 and 17.

We first note that the above theorem is tight in terms of block sensitivity, i.e., we cannot have $\widetilde{\deg}(f \circ g) = \widetilde{\Omega}(\mathrm{bs}(f)^c \cdot \widetilde{\deg}(g))$ for any $c > 1/2$. This is because the OR function over n bits witnesses the tight quadratic separation between $\widetilde{\deg}$ and bs, i.e., $\widetilde{\deg}(\mathsf{OR}_n) = \Theta(\sqrt{n}) = \Theta(\sqrt{\mathrm{bs}(\mathsf{OR}_n)})$ [33].

We also get the following composition theorem as a corollary. It says that the composition for $\widetilde{\deg}$ holds when the outer function has minimal approximate degree with respect to its block sensitivity. Recall, $\widetilde{\deg}(f) = \Omega(\sqrt{\mathrm{bs}(f)})$ [33].

▶ **Corollary 8.** *For all Boolean function $f : \{0,1\}^n \to \{0,1\}$ with $\widetilde{\deg}(f) = \Theta(\sqrt{\mathrm{bs}(f)})$ and for all $g : \{0,1\}^m \to \{0,1\}$, we have $\widetilde{\deg}(f \circ g) = \widetilde{\Theta}(\widetilde{\deg}(f) \cdot \widetilde{\deg}(g))$.*

This complements a result of Sherstov [38, Theorem 6.6], which shows that composition of $\widetilde{\deg}$ holds when the outer function has maximal $\widetilde{\deg}$ with respect to its arity.

We further note that Corollary 8 covers new set of composed functions $f \circ g$ for which the composition theorem for $\widetilde{\deg}$ doesn't follow from the known results [11, 38]. For example, consider the Rubinstein function RUB with arity n(see [36] for Definition) as the outer function f. It is clearly not a symmetric function. It also doesn't have high approximate degree, i.e., $\widetilde{\deg}(\mathsf{RUB}) = O(\sqrt{n}\log n)$ (see Lemma A.7 from [20]). Therefore, the composition of $\widetilde{\deg}(\mathsf{RUB} \circ g)$ doesn't follow from the existing results. However, it follows from Corollary 8, since $\mathrm{bs}(\mathsf{RUB}) = \Omega(n)$ and so $\widetilde{\deg}(\mathsf{RUB}) = \widetilde{\Theta}(\sqrt{\mathrm{bs}(\mathsf{RUB})})$.

Another example is the sink function SINK over $\binom{n}{2}$ variables ([22]), which is also not a symmetric function. Furthermore, its approximate degree is $O(\sqrt{n}\log n)$ (Lemma A.7 from [20]). Therefore, the composition of $\widetilde{\deg}(\mathsf{SINK} \circ g)$ also doesn't follow from the existing results. Again, it follows from Corollary 8, since $\mathrm{bs}(\mathsf{SINK}) = \Theta(n)$ (Observation A.4, [20]) and $\widetilde{\deg}(\mathsf{SINK}) = \widetilde{\Theta}(\sqrt{n})$.

Ideas behind proof of Theorem 7

We will first sketch the proof ideas in the case when f and g are total Boolean functions, and then explain how to extend it to partial functions too.

Our starting point is the well known Nisan-Szegedy's embedding of PrOR (see Definition 18) over $\mathrm{bs}(f)$ many bits in a Boolean function f [33]. Carrying out this transformation in $f \circ g$ embeds $\mathsf{PrOR}_{\mathrm{bs}(f)} \circ (g_1, \ldots, g_{\mathrm{bs}(f)})$ into $f \circ g$, where $g_1, \ldots, g_{\mathrm{bs}(f)}$ are different partial functions such that $\widetilde{\mathrm{bdeg}}(g_i) \geq \widetilde{\deg}(g)$ for all $i \in [\mathrm{bs}(f)]$[4]. Since the transformation is just substitutions of variables by constants, we further have

$$\widetilde{\deg}(f \circ g) \geq \widetilde{\mathrm{bdeg}}(\mathsf{PrOR}_{\mathrm{bs}(f)} \circ (g_1, \ldots, g_{\mathrm{bs}(f)})). \tag{3}$$

It now looks like that we can appeal to the composition theorem for PrOR (Theorem 21) [11] to obtain our theorem. However, there is a technical difficulty – Theorem 21 doesn't hold for *different* inner *partial* functions. It only deals with a single total inner function. We therefore generalize the proof of Theorem 21 to obtain the following general version of the composition theorem for PrOR.

▶ **Theorem 9.** *For any partial Boolean functions $g_1, g_2, \ldots, g_n$, we have*

$$\widetilde{\mathrm{bdeg}}\left(\mathsf{PrOR}_n \circ (g_1, g_2, \ldots, g_n)\right) = \Omega\left(\frac{\sqrt{n} \cdot \min_{i=1}^{n} \widetilde{\mathrm{bdeg}}(g_i)}{\log n}\right).$$

[4] $\widetilde{\mathrm{bdeg}}$ is the notion of approximate degree in the context of partial functions. For a formal definition, see Definition 17.

We can now obtain Theorem 7 from Equation (3) and Theorem 9. The proof of Theorem 9 is a generalization of a result due to [11] (see Theorem 21). For lack of space, we present the proof of this theorem in the complete version of this paper [20].

2.3 Composition results when the outer functions has some symmetry

The class of symmetric functions capture many important function like OR, AND, Parity and Majority. Recall that a function is symmetric when the function value only depends on the Hamming weight of the input; in other words, a function is symmetric iff its value on an input remains unchanged even after permuting the bits of the input. As noted earlier, both for R and $\widetilde{\deg}$, composition was known to hold when the outer function was symmetric.

A natural question is, whether one can prove composition theorems when the outer function is *weakly* symmetric (it is symmetric with respect to a weaker notion of symmetry). In this paper we consider one such notion of symmetry – junta-symmetric functions.

▶ **Definition 10** (k-junta symmetric function)**.** *A function $f : \{0,1\}^n \to \{0,1\}$ is called a k-junta symmetric function if there exists a set $\mathcal{J}$ of size k of variables such that the function value depends on assignments to the variables in $\mathcal{J}$ as well as on the Hamming weight of the whole input.*

k-junta symmetric functions can be seen as a mixture of symmetric functions and k-juntas. This class of functions has been considered previously in literature, particularly in [19, 14] where these functions plays a crucial role. [19] even presents multiple characterisations of k-junta symmetric functions for constant k. Note that by definition an arbitrary k-junta (i.e., a function that depend on k variables) is also a k-junta symmetric function, since we can consider the dependence on Hamming weight to be trivial. Thus, this notion loses out on the symmetry of the function considered. We, therefore, consider the class of *strongly* k-junta symmetric functions.

▶ **Definition 11** (Strongly k-junta symmetric function)**.** *A k-junta symmetric function is called strongly k-junta symmetric if every variable is influential. In other words, there exists a setting to the junta variables such that the function value depends on the Hamming weight of the whole input in a non-trivial way.*

We prove that if the outer function is strongly $\sqrt{n}$-junta symmetric ("strongly" indicating that the dependence on the Hamming weight is non-trivial) then $\widetilde{\deg}$ composes. On the other hand, Theorem 2 implies that R composes for any strongly k-junta symmetric functions (as long as $n-k=\Theta(n)$).

▶ **Theorem 12.** *For any strongly k-junta symmetric function $f : \{0,1\}^n \to \{0,1\}$ and any Boolean function $g : \{0,1\}^m \to \{0,1\}$, we have*

- $\widetilde{\deg}(f \circ g) = \widetilde{\Theta}(\widetilde{\deg}(f) \cdot \widetilde{\deg}(g))$ *where* $k = O(\sqrt{n})$.
- $\mathrm{R}(f \circ g) = \widetilde{\Theta}(\mathrm{R}(f) \cdot \mathrm{R}(g))$ *where* $n-k=\Theta(n)$.

For the lack of space, the proof of the above theorem is given in Appendix C. Note that if one is able to prove the above theorem for k-junta-symmetric functions (without the requirement of "strongly") for any non-constant k then we would have the full composition theorem.

Organization of the paper

We have formally defined complexity measures and Boolean functions needed for our results in Section 3 and Appendix A. Section 4 contains proofs of our results related to the composition of randomized query complexity (Theorem 2). In Section 5 we give the proof of our result for the composition of approximate degree (Theorem 7). Finally, for the sake of space, the results about the composition of functions with the weak notion of symmetry are in Appendix C.

3 Preliminaries

Notations: We will use $[n]$ to represent the set $\{1, \dots, n\}$. For any (possibly partial) Boolean function $f : \{0,1\}^n \to \{0,1,*\}$ we will denote by $\mathrm{Dom}(f)$ the set $f^{-1}(\{0,1\})$. The arity of f is the number of variables - in this case n. A Boolean function $f : \{0,1\}^n \to \{0,1,*\}$ is said to be total if $\mathrm{Dom}(f) = \{0,1\}^n$. Any function (not otherwise stated) will be a total Boolean function.

For any $x \in \{0,1\}^n$, we will use $|x|$ to denote the number of 1s in x, that is, the Hamming weight of the string x. The string x^i denotes the modified string x with the i-th bit flipped. Similarly, x^B is defined to be the string such that all the bits whose index is contained in the set $B \subseteq [n]$ are flipped in x.

Following is a formal definition of (partial) function composition.

▶ **Definition 13** (Generalized composition of functions). *For any (possibly partial) Boolean function $f : \{0,1\}^n \to \{0,1,*\}$ and n (possibly partial) Boolean functions $g_1, g_2, \dots, g_n$, define the (possibly partial) composed function*

$$f \circ (g_1, g_2, \dots, g_n)(x_1, x_2, \dots, x_n) = f(g_1(x_1), g_2(x_2), \dots, g_n(x_n)),$$

where g_i's can have different arities and, moreover, if $x_i \notin \mathrm{Dom}(g_i)$ for any $i \in [n]$ or the string $(g_1(x_1), g_2(x_2), \dots, g_n(x_n)) \notin \mathrm{Dom}(f)$, then the function $f \circ g$ outputs $*$.

In this paper we use the standard definitions of various complexity measures like randomized query complexity, sensitivity, block-sensitivity, fractional block sensitivity and approximate degree. We present the formal definitions in Appendix A.

3.1 Standard definitions and functions for the composition of R

The function Gap-Majority has played an important role in the study of composition of R.

▶ **Definition 14** (Gap-Majority). *The function $\mathrm{GapMaj}_t : \{0,1\}^t \to \{0,1,*\}$ is a partial function with arity t such that*

$$\mathrm{GapMaj}_t(x) = \begin{cases} 1 & \text{if } |x| = t/2 + 2\sqrt{t}, \\ 0 & \text{if } |x| = t/2 - 2\sqrt{t}, \\ * & \text{otherwise.} \end{cases}$$

It can be shown that $\mathrm{R}(\mathrm{GapMaj}_t) = \Theta(t)$ [9].

In regards to the composition question of R, one of the most significant complexity measures (defined by Ben-David and Blais [9]) is that of noisyR. We first define the noisy oracle model.

▶ **Definition 15** (Noisy Oracle Model and Noisy Oracle Access to a String ([9])). *For $b \in \{0,1\}$, a noisy oracle to b takes a parameter $-1 \leq \gamma \leq 1$ as input and returns a bit b' such that $\Pr[b' = b] = (1+\gamma)/2$. The cost of one such query is γ^2. Each query to noisy oracle returns independent bits.*

For $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, noisy oracle access to x is access to n independent noisy oracles, one for each bit x_i, $i \in [n]$.

Next, we define the noisy oracle model of computation.

▶ **Definition 16** (Noisy Oracle Model of Computation ([9])). *Let $f : \{0,1\}^n \to \{0,1,*\}$ be a partial Boolean function. A* noisyR *query algorithm A computes f if for all $x \in \mathrm{Dom}(f)$, $\Pr[A(x) \neq f(x)] \leq 1/3$, where A is a randomized algorithm given noisy oracle access to x, and the probability is over both noisy oracle calls and the internal randomness of the algorithm A. The cost of the algorithm A for an input x is the sum of the cost of all noisy oracle calls made by A on x, and the cost of A, cost(A), is the maximum cost over all $x \in \mathrm{Dom}(f)$. The* noisyR *randomized query complexity of f, denoted by* $\mathrm{noisyR}(f)$, *is defined as*

$$\mathrm{noisyR}(f) = \min_{A \text{ computes } f} cost(A).$$

Again, 1/3 in the above definition can be replaced by any constant $< 1/2$. If only queries with $\gamma = 1$ are allowed in the noisy query model, then we obtain the usual randomized algorithm for f, thus $\mathrm{noisyR}(f) = O(\mathrm{R}(f))$.

3.2 Standard definitions and functions for the composition of $\widetilde{\deg}$

The definition of $\widetilde{\deg}$ can naturally be extended to partial functions f by restricting the definition to hold only for inputs in $\mathrm{Dom}(f)$, but the approximating polynomial can take arbitrarily large values on points outside the domain. However, for the purpose of understanding the composition of approximate degree of Boolean functions (or even total Boolean functions) one need a measure of approximate degree of partial Boolean functions which is bounded on all the points of the Boolean cube.

▶ **Definition 17** (Bounded approximate degree ($\widetilde{\mathrm{bdeg}}$)). *For a partial Boolean function $f : \{0,1\}^n \to \{0,1,*\}$, the bounded approximate degree of f ($\widetilde{\mathrm{bdeg}}(f)$) is the minimum possible degree of a polynomial p such that*
- $|p(x) - f(x)| \leq 1/3, \quad \forall x \in \mathrm{Dom}(f)$, *and*
- $0 \leq p(x) \leq 1 \quad \forall x \in \{0,1\}^n$.

In other words, we take the minimum possible degree of a polynomial which is bounded for all possible inputs ($p(x) \in [0,1]$ for all $x \in \{0,1\}^n$), and it approximates f in the usual sense over $\mathrm{Dom}(f)$.

Over the years people have tried to study the composition of $\widetilde{\deg}$ with different outer functions. In this context the following restriction of OR is an important partial function:

▶ **Definition 18** (Promise-OR). *Promise-*OR *(denoted by* PrOR_n*) is the function* PrOR_n : $\{0,1\}^n \to \{0,1,*\}$ *such that* $\mathsf{PrOR}_n(x) = 0$ *if* $|x| = 0$, *equals to 1 if* $|x| = 1$, *and* $*$ *otherwise.*

Some useful previous results. We will also be crucially using a few results from prior works in our proofs. The following are a couple of useful results on noisyR.

▶ **Lemma 19** ([9]). *Let f be a non-constant partial Boolean function then* $\mathrm{noisyR}(f) = \Omega(1)$.

▶ **Theorem 20** ([9]). *For all partial functions f and g, $\mathrm{R}(f \circ g) = \Omega(\mathrm{noisyR}(f) \cdot \mathrm{R}(g))$.*

We will also be using the following theorem of [11] regarding the composition question of $\widetilde{\mathrm{bdeg}}$ when the outer function is PrOR_n. Informally, we will call it the Promise-OR composition theorem.

▶ **Theorem 21** ([11]). *For any Boolean function $g : \{0,1\}^m \to \{0,1\}$ we have,*

$$\widetilde{\mathrm{bdeg}}\left(\mathsf{PrOR}_n \circ g\right) = \Omega\left(\sqrt{n} \cdot \widetilde{\deg}(g)/\log n\right).$$

4 Results about composition of R

This section is devoted to the results related to the composition of randomized query complexity. Our main result states that composition of R holds if the outer function has full randomized query complexity (Theorem 2). As mentioned in the proof idea, the proof critically depends on the notion of noisy randomized query complexity and its properties (introduced by Ben-David and Blais [9]).

Recall the definition of noisy randomized query complexity of a function f from Definition 16. As mentioned in the introduction (Theorem 3), Ben-David and Blais [9] proved that

$$\mathrm{noisyR}(f) = \Theta\left(\frac{\mathrm{R}(f \circ \mathrm{GapMaj}_n)}{n}\right), \tag{4}$$

where GapMaj_n is the Gap-Majority function on n bits. Note that Ben-David and Blais proved Equation 4 when the arity of functions f and Gap-Majority is the same. We show that if Equation 4 can be generalized for Gap-Majority functions of arbitrary arity for some outer function f, then randomized query complexity composes for the function f. We restate the following observation from the introduction.

▶ **Observation 4.** *Let f be a partial Boolean function on n bits. If $t(n) \geq 1$ is a non-decreasing function of n and*

$$\mathrm{noisyR}(f) = \Omega\left(\frac{\mathrm{R}(f \circ \mathrm{GapMaj}_{t(n)})}{t(n)}\right),$$

then $\mathrm{R}(f \circ g) = \Omega((\mathrm{R}(f) \cdot \mathrm{R}(g))/t(n))$ for all partial functions g.

Proof. Suppose $\mathrm{noisyR}(f) = \Omega\left(\frac{\mathrm{R}(f \circ \mathrm{GapMaj}_{t(n)})}{t(n)}\right)$, since $\mathrm{R}(f \circ \mathrm{GapMaj}_t) \geq \mathrm{R}(f)$, we have $\mathrm{noisyR}(f) = \Omega(\mathrm{R}(f)/(t(n))$. Theorem 20 implies that a lower bound on noisyR translates to a lower bound on $\mathrm{R}(f \circ g)$. We have,

$$\begin{aligned} \mathrm{R}(f \circ g) &= \Omega(\mathrm{noisyR}(f) \cdot \mathrm{R}(g)) && \text{(Theorem 20)} \\ &= \Omega\left(\frac{\mathrm{R}(f) \cdot \mathrm{R}(g)}{t(n)}\right). \end{aligned}$$ ◀

Observation 4 follows from the above observation by choosing $t(n)$ to be a small function of n.

We restate from Section 1 our generalized characterization of noisyR (i.e., generalization of Equation 4). For a complete proof of Theorem 5 we refer to the full version of this paper [20].

▶ **Theorem 5.** *Let f be a partial function on n bits and let $t \geq 1$, then* $\mathrm{R}(f \circ \mathrm{GapMaj}_t) = O\left(t \cdot \mathrm{noisyR}(f) + n\right)$.

This allows us to show that if for an n-bit partial function f, $\mathrm{R}(f) = \Theta(n)$, then $\mathrm{R}(f \circ g) = \widetilde{\Theta}(\mathrm{R}(f) \cdot \mathrm{R}(g))$ for all partial functions g (Theorem 2).

The proof of Theorem 2 is discussed in Section 4.1. A corollary of this theorem is that if R composes with respect to an outer function, then noisyR also composes with respect to the same outer function (Corollary 6).

We give proof of Theorem 2 in the next section and prove Corollary 6 in Section 4.3. We need the following theorem for these proofs, which lower bounds $R(f \circ g)$ in terms of $R(f)$ and $R(g)$.

▶ **Theorem 22** ([24])**.** *Let f and g be partial functions then* $\mathrm{R}(f \circ g) = \Omega(\mathrm{R}(f) \cdot \sqrt{\mathrm{R}(g)})$.

4.1 Composition for functions with $\mathrm{R}(f) = \Theta(n)$

We restate the theorem below.

▶ **Theorem 2.** *Let f be a partial Boolean function on n-bits such that $\mathrm{R}(f) = \Theta(n)$. Then for all partial functions g, we have*

$$\mathrm{R}(f \circ g) = \Omega(\mathrm{R}(f) \cdot \mathrm{R}(g)).$$

Proof. From Theorem 22 we have a lower bound on the randomized query complexity of $(f \circ \mathrm{GapMaj}_t)$:

$$\mathrm{R}(f \circ \mathrm{GapMaj}_t) = \Omega(\mathrm{R}(f) \cdot \sqrt{t}). \tag{5}$$

On the other hand, Theorem 5 gives an upper bound of $O\left(t \cdot \mathrm{noisyR}(f) + n\right)$ on $\mathrm{R}(f \circ \mathrm{GapMaj}_t)$. Thus, choosing $t = \left(\frac{C \cdot n}{\mathrm{noisyR}(f)}\right)$ for a large enough constant C, we have

$$\mathrm{R}(f) \cdot \sqrt{\frac{n}{\mathrm{noisyR}(f)}} = O\left(\frac{n}{\mathrm{noisyR}(f)} \cdot \mathrm{noisyR}(f) + n\right).$$

This implies that

$$\mathrm{R}(f) = O\left(\sqrt{n \cdot \mathrm{noisyR}(f)}\right). \tag{6}$$

Thus, if $\mathrm{R}(f) = \Theta(n)$, then $\mathrm{noisyR}(f) = \Omega(\mathrm{R}(f))$, which implies composition from Theorem 20. ◀

Notice that Equation 6 is equivalent to the following observation.

▶ **Observation 23.** *Let f be a partial Boolean function on n-bits. Then,* $\mathrm{noisyR}(f) = \Omega\left(\frac{\mathrm{R}(f)^2}{n}\right)$.

When $\mathrm{R}(f) = \Theta(n)$, we have already seen that Observation 23 implies composition of randomized query complexity when the outer function is f.

Though, Observation 23 implies a more general result. When $\mathrm{R}(f)$ is close to n (arity of f), Observation 23 places a limit on the gap between $\mathrm{R}(f)$ and $\mathrm{noisyR}(f)$ (consequently on the violation of composition with outer function being f). These implications are formally discussed in Appendix 4.2.

Another implication of Theorem 2 is that composition of R for an outer function f implies the composition of noisyR for outer function being f (Corollary 6).

4.2 Additional implications of Observation 23

Without loss of generality we can assume $\mathrm{R}(f \circ g) = \Omega(\mathrm{R}(g))$ (note that this is true when f is non-constant).

Ben-David and Blais [9] gave a counterexample for composition, but the arity of the used function was very high compared to the randomized query complexity. They observed that the composition can still be true in the weaker sense:

$$\mathrm{R}(f \circ g) = \Omega\left(\frac{\mathrm{R}(f) \cdot \mathrm{R}(g)}{\log n}\right).$$

Observation 23 shows that a much weaker composition result is true.

▶ **Corollary 24.** *Let f and g be partial functions on n and m bits respectively. If $\mathrm{R}(f \circ g) = \Omega(\mathrm{R}(g))$, then*

$$\mathrm{R}(f \circ g) = \Omega\left(\frac{\mathrm{R}(f) \cdot \mathrm{R}(g)}{\sqrt{n}}\right).$$

Proof.

$$\begin{aligned} \mathrm{R}(f \circ g) &= \Omega(\mathrm{noisyR}(f) \cdot \mathrm{R}(g)) && \text{(Theorem 20)} \\ &= \Omega\left(\frac{\mathrm{R}(f)^2 \cdot \mathrm{R}(g)}{n}\right). && (7) \end{aligned}$$

Where the last equality follows from Observation 23 [5] Now there are two cases:

- **Case 1.** $\mathrm{R}(f) = O(\sqrt{n})$. In this case $\mathrm{R}(f)/\sqrt{n} = O(1)$ and since we assumed $\mathrm{R}(f \circ g) = \Omega(\mathrm{R}(g))$, the claim follows from Equation 7.
- **Case 2.** $\mathrm{R}(f) = \Theta(n^{1/2} \cdot t(n))$ where $t(n)$ is a strictly increasing function of n. Thus,

$$\frac{\mathrm{R}(f)^2 \cdot \mathrm{R}(g)}{n} = \Omega\left(t(n)^2 \cdot \mathrm{R}(g)\right) = \Omega\left(\frac{\mathrm{R}(f) \cdot \mathrm{R}(g)}{\sqrt{n}}\right).$$

Again, the claim follows from Equation 7. ◀

The weaker composition, Corollary 24, implies that if $\mathrm{R}(f)$ and $\mathrm{R}(g)$ are comparable to the arity of these functions, the randomized query complexity of $f \circ g$ is "not far" from the conjectured randomized query complexity $\mathrm{R}(f) \cdot \mathrm{R}(g)$. In other words, if there is a large polynomial separation between $\mathrm{R}(f \circ g)$ and $(\mathrm{R}(f) \cdot \mathrm{R}(g))$, then $\mathrm{R}(f)$ and $\mathrm{R}(g)$ can not be too large.

▶ **Corollary 25.** *Let f and g be partial functions such that f is a function on n-bits and g is a function on $t(n)$-bits where $t(n)$ is a strictly increasing function of n. If $\mathrm{R}(f) = \Theta(n^{\beta})$, $\mathrm{R}(g) = \Theta(n^{\gamma})$ and $\mathrm{R}(f \circ g) = O((\mathrm{R}(f) \cdot \mathrm{R}(g))^{\alpha})$, where $\alpha < 1$ is a constant, then $(1 - \alpha)(\alpha + \beta) < 1/2$.*

Proof. For some constants A and B we have

$$A \cdot \frac{\mathrm{R}(f) \cdot \mathrm{R}(g)}{\sqrt{n}} \leq \mathrm{R}(f \circ g) \leq B \cdot (\mathrm{R}(f) \cdot \mathrm{R}(g))^{\alpha},$$

[5] Sherstov [38] proved that for Boolean functions f and g, $\widetilde{\deg}(f \circ g) = \Omega((\widetilde{\deg}(f)^2 \widetilde{\deg}(g))/n)$. Thus in Equation 7 we prove the same result but in the randomized world.

where the first inequality follows from Corollary 24 and second from assumption. Assigning the values of $\mathrm{R}(f)$ and $\mathrm{R}(g)$ in terms on n we have,

$$\begin{aligned}
&A \cdot n^{\beta+\gamma-1/2} \leq B \cdot n^{\alpha(\beta+\gamma)} \\
&n^{(1-\alpha)(\beta+\gamma)-1/2} \leq \frac{B}{A}.
\end{aligned}$$

which implies, for large enough n, $(1-\alpha)(\beta+\gamma) \leq 1/2$. ◀

A special case of the above corollary is when arity and randomized query complexity of g are superpolynomial in n. In this case a polynomial gap between $\mathrm{R}(f \circ g)$ and $(\mathrm{R}(f) \cdot \mathrm{R}(g))$ is not possible.

4.3 Proof of Corollary 6

First, we need the following lemma which follows from Theorem 5, Theorem 20 and Lemma 19.

▶ **Lemma 26.** *Let f be a partial function on n bits and let $t = \Omega(n)$. Then*

$$\mathrm{noisyR}(f) = \Theta\left(\frac{\mathrm{R}(f \circ \mathrm{GapMaj}_t)}{t}\right).$$

Proof. From Theorem 5 we have for all $t \geq 1$, $\mathrm{R}(f \circ \mathrm{GapMaj}_t) = O(t \cdot \mathrm{noisyR}(f) + n)$. Since we have assumed $t = \Omega(n)$ and $\mathrm{noisyR}(f) = \Omega(1)$ (Lemma 19), we get $\mathrm{R}(f \circ \mathrm{GapMaj}_t) = O(t \cdot \mathrm{noisyR}(f))$. Thus, $\mathrm{noisyR}(f) = \Omega\left(\frac{\mathrm{R}(f \circ \mathrm{GapMaj}_t)}{t}\right)$.

The upper bound $\mathrm{noisyR}(f) = O\left(\frac{\mathrm{R}(f \circ \mathrm{GapMaj}_t)}{t}\right)$ follows from Theorem 20 and the fact that $\mathrm{R}(\mathrm{GapMaj}_t) = \Theta(t)$. ◀

Now we prove that if R composes for f then noisyR composes for that f. For convenience, we recall the statement of the corollary from the introduction.

▶ **Corollary 6.** *Let f be a partial Boolean function. If $\mathrm{R}(f \circ g) = \widetilde{\Theta}(\mathrm{R}(f) \cdot \mathrm{R}(g))$ for all partial functions g then $\mathrm{noisyR}(f \circ g) = \widetilde{\Theta}(\mathrm{noisyR}(f) \cdot \mathrm{noisyR}(g))$.*

Proof. From Theorem 3, we have

$$\mathrm{noisyR}(f \circ g) = \Theta\left(\frac{\mathrm{R}\left((f \circ g) \circ \mathrm{GapMaj}_{mn}\right)}{mn}\right).$$

Since $(f \circ g) \circ h = f \circ (g \circ h)$, the right hand side of the above expression is equal to

$$\Theta\left(\frac{\mathrm{R}\left(f \circ (g \circ \mathrm{GapMaj}_{mn})\right)}{mn}\right).$$

The proof follows from the assumption that R composes and Lemma 26.

$$\begin{aligned}
\mathrm{noisyR}(f \circ g) &= \Theta\left(\frac{\mathrm{R}\left(f\right) \cdot \mathrm{R}\left(g \circ \mathrm{GapMaj}_{mn}\right)}{mn}\right) && \text{(assuming R composes)} \\
&= \Theta\left(\mathrm{R}(f) \cdot \mathrm{noisyR}(g)\right) && \text{(from Lemma 26)} \\
&= \Theta\left(\mathrm{noisyR}(f) \cdot \mathrm{noisyR}(g)\right). && \text{(assuming R composes)}
\end{aligned}$$

◀

5 Composition of approximate degree in terms of block sensitivity

In this section we study the composition question for approximate degree. Recall that the composition question asks: whether for all Boolean functions f and g

$$\widetilde{\deg}(f \circ g) = \widetilde{\Omega}(\widetilde{\deg}(f)\,\widetilde{\deg}(g))?$$

Following our discussion from the introduction, we know that the above composition is known to hold for only two sub-classes of outer functions, namely symmetric functions [11] and functions with high approximate degree [38]. It is thus natural to seek weaker lower bounds to make progress towards the composition question. One way to weaken the expression on the right-hand side would be to replace the measure $\widetilde{\deg}(f)$ by a weaker measure (like $\sqrt{\mathrm{s}(f)}$, $\sqrt{\mathrm{bs}(f)}$ or $\sqrt{\mathrm{fbs}(f)}$). Here we will establish one such lower bound of $\sqrt{\mathrm{bs}(f)}\,\widetilde{\deg}(g)$.

We restate our theorem now.

▶ **Theorem 7.** *For all non-constant (possibly partial)*[6] *Boolean functions $f : \{0,1\}^n \to \{0,1\}$ and $g : \{0,1\}^m \to \{0,1\}$, we have*

$$\widetilde{\deg}(f \circ g) = \widetilde{\Omega}(\sqrt{\mathrm{bs}(f)} \cdot \widetilde{\deg}(g)).$$

We note that many analogous results are known in the setting of composition of R; see, for example, [26, 7, 8, 13, 6, 24, 10]. To the best of our knowledge, this is the first such result in the setting of $\widetilde{\deg}$. We present only a proof sketch here; most of the technical parts of the proof appear in Appendix B.

Further, we present the sketch of the proof in two parts. For simplicity, in the first part we sketch a proof of the lower bound $\sqrt{\mathrm{s}(f)}\,\widetilde{\deg}(g)$ for total function f, and then in the second part we modify the arguments to obtain Theorem 7.

We begin with a proof sketch for a lower bound of $\sqrt{\mathrm{s}(f)}\,\widetilde{\deg}(g)$. Let $x \in \{0,1\}^n$ be an input having the maximum sensitivity with respect to f, and $S \subseteq [n]$ be the set of sensitive bits at x ($|S| = \mathrm{s}(f)$). Consider the subfunction f' obtained from f by fixing the set of variables *not* in S according to x. By construction, f' is defined over $\mathrm{s}(f)$ many variables and is fully sensitive at the input $x|_S$ given by x restricted to the indices in S. Since f' is a subfunction of f and g is non-constant, we have $\widetilde{\deg}(f \circ g) \geq \widetilde{\deg}(f' \circ g)$.

Notice that f' at the neighbourhood of x, in the Boolean cube, is the partial function PrOR (Definition 18) or its negation. Therefore, we have $\widetilde{\deg}(f \circ g) \geq \widetilde{\deg}(f' \circ g) \geq \widetilde{\mathrm{bdeg}}(\mathsf{PrOR}_{|S|} \circ g)$ (see Definition 17 for a definition of the bounded approximate degree). We can now invoke the composition theorem for PrOR (Theorem 21) [11] to obtain our lower bound:

$$\widetilde{\deg}(f \circ g) \geq \widetilde{\deg}(f' \circ g) \geq \widetilde{\mathrm{bdeg}}(\mathsf{PrOR}_{|S|} \circ g) = \widetilde{\Omega}(\sqrt{\mathrm{s}(f)}\,\widetilde{\deg}(g)).$$

However, there is a technical issue with our argument above. When we claimed that f' looks like a PrOR function we were not quite correct. Technically, it is a Shifted-PrOR function $\mathsf{PrOR}_{|S|}^{x|_S}$, where $\mathsf{PrOR}_n^a(y_1, y_2, \ldots, y_n) := \mathsf{PrOR}_n(y_1 \oplus a_1, y_2 \oplus a_2, \ldots, y_n \oplus a_n)$ for $a \in \{0,1\}^n$. Formally, we have

$$\widetilde{\deg}(f \circ g) \geq \widetilde{\deg}(f' \circ g) \geq \widetilde{\mathrm{bdeg}}(\mathsf{PrOR}_{|S|}^{x|_S} \circ g) = \widetilde{\mathrm{bdeg}}(\mathsf{PrOR}_{|S|} \circ (g_1, \ldots, g_{|S|})), \tag{8}$$

where $g_i = g$ or $\neg g$ depending on the corresponding i-th bit in $x|_S$.

[6] For definitions of block sensitivity and approximate degree in the context of partial functions, please see Definitions 32 and 17.

We, therefore, need a composition theorem for PrOR with *different* inner functions, while Theorem 21 requires that all the inner functions be same. In fact, we would need a more general composition theorem with *different* inner *partial* functions, which we restate below. This generalization is crucially used when dealing with block sensitivity.

▶ **Theorem 9.** *For any partial Boolean functions $g_1, g_2, \ldots, g_n$, we have*

$$\widetilde{\mathrm{bdeg}}\left(\mathsf{PrOR}_n \circ (g_1, g_2, \ldots, g_n)\right) = \Omega\left(\frac{\sqrt{n}\cdot \min_{i=1}^{n} \widetilde{\mathrm{bdeg}}(g_i)}{\log n}\right).$$

The proof of Theorem 9 is a generalization of proof of Theorem 21. For the sake of completeness we have added a proof in the full version of the paper [20].

Now returning to Equation (8) and using Theorem 9, we obtain the desired lower bound:

$$\widetilde{\deg}(f\circ g) \geq \widetilde{\deg}(f'\circ g) \geq \widetilde{\mathrm{bdeg}}(\mathsf{PrOR}_{|S|}^{x|_S}\circ g) = \widetilde{\Omega}(\sqrt{\mathrm{s}(f)}\,\widetilde{\deg}(g)).$$

We are now ready to present the modifications required to improve the lower bound to $\widetilde{\Omega}(\sqrt{\mathrm{bs}(f)}\,\widetilde{\deg}(g))$.

Proof of Theorem 7. Let $b = \mathrm{bs}(f)$ and $a = (a_1, a_2, \ldots, a_n)$ be an input where f achieves the maximum block sensitivity. Further, let $B_1, B_2, \ldots, B_b$ be disjoint minimal sets of variables that achieves the block sensitivity at a, i.e., $f(a) \neq f(a^{B_i})$ for all $i \in [b]$. Recall, a^{B_i} denotes the Boolean string obtained from a by flipping the bits at all the indices given by B_i. Define a partial function $f' : \{0,1\}^n \to \{0,1,*\}$ such that,

$$f'(x) = \begin{cases} 0 & \text{if } x = a, \\ 1 & \text{if } x = a^{B_i}, \text{ for some } i \in [b], \\ * & \text{otherwise.} \end{cases}$$

Note that f contains f' or its negation as a sub function. Thus, $\widetilde{\deg}(f\circ g) \geq \widetilde{\mathrm{bdeg}}(f'\circ g)$.

Since g is non-constant, we can fix the indices *not* in $\bigcup_{i=1}^{b} B_i$ according to a to obtain $f''\circ g$. We would now like to embed PrOR_b over the remaining variables in f''. For this purpose we define the following partial functions: for every $i \in [b]$, let $I_i : \{0,1\}^{B_i} \to \{0,1,*\}$ be such that

$$I_i(x) = \begin{cases} 0 & \text{if } x = a|_{B_i}, \\ 1 & \text{if } x = a^{B_i}|_{B_i}, \\ * & \text{otherwise.} \end{cases}$$

Now observe that $f''\circ g$ can be rewritten as $\mathsf{PrOR}_b \circ (I_1\circ g, \ldots, I_b \circ g)$. We therefore have

$$\begin{aligned} \widetilde{\deg}(f\circ g) \geq \widetilde{\mathrm{bdeg}}(f'\circ g) \geq \widetilde{\mathrm{bdeg}}(f''\circ g) &= \widetilde{\mathrm{bdeg}}(\mathsf{PrOR}_b \circ (I_1\circ g, \ldots, I_b\circ g)) \\ &= \Omega\left(\frac{\sqrt{b}\cdot \min_i \widetilde{\mathrm{bdeg}}(I_i\circ g)}{\log b}\right) = \widetilde{\Omega}\left(\sqrt{b}\cdot \widetilde{\deg}(g)\right), \end{aligned}$$

where the second last equality follows from Theorem 9 and the last equality uses the fact $\widetilde{\mathrm{bdeg}}(I_i\circ g) \geq \widetilde{\deg}(g)$ for all i, which in turn follows from each I_i being non-constant. ◀

We end this section with few final remarks. As a corollary to Theorem 7 we have the following composition for $\widetilde{\deg}$ when the outer function has minimal approximate degree with respect to its block sensitivity.

▶ **Corollary 8.** *For all Boolean function* $f : \{0,1\}^n \to \{0,1\}$ *with* $\widetilde{\deg}(f) = \Theta(\sqrt{\mathrm{bs}(f)})$ *and for all* $g : \{0,1\}^m \to \{0,1\}$*, we have* $\widetilde{\deg}(f \circ g) = \widetilde{\Theta}(\widetilde{\deg}(f) \cdot \widetilde{\deg}(g))$*.*

We also note that the set of Boolean functions with $\widetilde{\deg}(f) = \Theta(\sqrt{\mathrm{bs}(f)})$ includes examples of *non-symmetric* functions f with *low* approximate degree. In other words, when such functions are outer function in a composed function then the composition of $\widetilde{\deg}$ doesn't follow from the known results [11, 38]. Such examples are described in Subsection 2.2.

As stated in the introduction, we recall that Theorem 7 is tight in terms of block-sensitivity, i.e., the lower bound can not be improved to $\widetilde{\Omega}(\mathrm{bs}(f)^c \cdot \widetilde{\deg}(g))$ for some $c > 1/2$.

6 Conclusion

While our work makes progress on the composition problem for R and $\widetilde{\deg}$, the main problems of whether $\widetilde{\deg}$ and R composes for any pair of Boolean functions still remains open. In this light, we would like to highlight some questions that can be useful stepping stones towards the main questions.

We showed that the composition question for R is equivalent to the following open question (which is a generalization of Ben-David and Blais [9] result):

▶ **Open question 27.** *Let* $f : \{0,1\}^n \to \{0,1,*\}$ *be a Boolean function. Then, is it true that for arbitrary* t*,* $\mathrm{noisyR}(f) = \Theta\left(\mathrm{R}(f \circ \mathrm{GapMaj}_t)/t\right)$*?*

In case of approximate degree composition, a natural question is whether $\sqrt{\mathrm{bs}(f)}$ can be replaced by some other complexity measures. In this regards we state the following open problems:

▶ **Open question 28.** *For all Boolean functions* f *and* g*, can we prove either of the following:*

- $\widetilde{\deg}(f \circ g) = \Omega(\sqrt{\widetilde{\deg}(f)} \cdot \widetilde{\deg}(g))$*?*
- $\widetilde{\deg}(f \circ g) = \Omega(\sqrt{\mathrm{fbs}(f)} \cdot \widetilde{\deg}(g))$*?*

Recently, in [43, 42, 23, 18, 21], the classes of transitive functions got a lot of attention as natural generalization of the classes of symmetric functions. Can the result for symmetric functions be extended to transitive functions?

▶ **Open question 29.** *Can we prove that* $\widetilde{\deg}$ *and* R *compose when the outer function is transitive?*

References

1 Scott Aaronson, Shalev Ben-David, and Robin Kothari. Separations in query complexity using cheat sheets. In *STOC*, pages 863–876, 2016. doi:10.1145/2897518.2897644.

2 Scott Aaronson, Shalev Ben-David, Robin Kothari, Shravas Rao, and Avishay Tal. Degree vs. approximate degree and quantum implications of Huang's sensitivity theorem. In *STOC*, pages 1330–1342, 2021. doi:10.1145/3406325.3451047.

3 Andris Ambainis. Polynomial degree and lower bounds in quantum complexity: Collision and element distinctness with small range. *Theory Comput.*, 1(1):37–46, 2005. doi:10.4086/toc.2005.v001a003.

4 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.

5 Andris Ambainis, Martins Kokainis, and Robin Kothari. Nearly optimal separations between communication (or query) complexity and partitions. In *CCC*, volume 50, pages 4:1–4:14, 2016. doi:10.4230/LIPIcs.CCC.2016.4.

6 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 *FSTTCS*, volume 93, pages 10:1–10:13, 2017. doi:10.4230/LIPIcs.FSTTCS.2017.10.

7 Andrew Bassilakis, Andrew Drucker, Mika Göös, Lunjia Hu, Weiyun Ma, and Li-Yang Tan. The power of many samples in query complexity. In *ICALP*, volume 168, pages 9:1–9:18, 2020. doi:10.4230/LIPIcs.ICALP.2020.9.

8 Shalev Ben-David and Eric Blais. A tight composition theorem for the randomized query complexity of partial functions. *arXiv:2002.10809*, 2020.

9 Shalev Ben-David and Eric Blais. A tight composition theorem for the randomized query complexity of partial functions: Extended abstract. In *FOCS*, pages 240–246, 2020. doi:10.1109/FOCS46700.2020.00031.

10 Shalev Ben-David, Eric Blais, Mika Göös, and Gilbert Maystre. Randomised composition and small-bias minimax. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 624–635. IEEE, 2022.

11 Shalev Ben-David, Adam Bouland, Ankit Garg, and Robin Kothari. Classical lower bounds from quantum upper bounds. In *FOCS*, pages 339–349, 2018. doi:10.1109/FOCS.2018.00040.

12 Shalev Ben-David, Mika Göös, Robin Kothari, and Thomas Watson. When Is Amplification Necessary for Composition in Randomized Query Complexity? In *APPROX/RANDOM 2020*, volume 176, pages 28:1–28:16, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.28.

13 Shalev Ben-David and Robin Kothari. Randomized query complexity of sabotaged and composed functions. *Theory Comput.*, 14(1):1–27, 2018. doi:10.4086/toc.2018.v014a005.

14 Eric Blais, Amit Weinstein, and Yuichi Yoshida. Partially symmetric functions are efficiently isomorphism testable. *SIAM J. Comput.*, 44(2):411–432, 2015. doi:10.1137/140971877.

15 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.

16 Mark Bun, Robin Kothari, and Justin Thaler. Quantum algorithms and approximating polynomials for composed functions with shared inputs. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 662–678. SIAM, 2019. doi:10.1137/1.9781611975482.42.

17 Mark Bun and Justin Thaler. Dual lower bounds for approximate degree and Markov-Bernstein inequalities. In *ICALP*, volume 7965, pages 303–314, 2013. doi:10.1007/978-3-642-39206-1_26.

18 Sourav Chakraborty. On the sensitivity of cyclically-invariant boolean functions. *Discret. Math. Theor. Comput. Sci.*, 13(4):51–60, 2011.

19 Sourav Chakraborty, Eldar Fischer, David García-Soriano, and Arie Matsliah. Junto-symmetric functions, hypergraph isomorphism and crunching. In *CCC*, pages 148–158, 2012. doi:10.1109/CCC.2012.28.

20 Sourav Chakraborty, Chandrima kayal, Rajat Mittal, Manaswi Paraashar, Swagato Sanyal, and Nitin Saurabh. On the composition of randomized query complexity and approximate degree. *Electron. Colloquium Comput. Complex.*, 1(1):37–46, 2023. doi:TR23-099.

21 Sourav Chakraborty, Chandrima Kayal, and Manaswi Paraashar. Separations between combinatorial measures for transitive functions. In *ICALP*, volume 229, pages 36:1–36:20, 2022. doi:10.4230/LIPIcs.ICALP.2022.36.

22 Arkadev Chattopadhyay, Nikhil S. Mande, and Suhail Sherif. The log-approximate-rank conjecture is false. *Journal of the ACM*, 67, 2020. URL: https://eccc.weizmann.ac.il/report/2018/176/.

23 Andrew Drucker. Block sensitivity of minterm-transitive functions. *Theor. Comput. Sci.*, 412(41):5796–5801, 2011. doi:10.1016/j.tcs.2011.06.025.

24 Dmitry Gavinsky, Troy Lee, Miklos Santha, and Swagato Sanyal. A Composition Theorem for Randomized Query Complexity via Max-Conflict Complexity. In *ICALP*, volume 132, pages 64:1–64:13, 2019. doi:10.4230/LIPIcs.ICALP.2019.64.

25 Justin Gilmer, Michael E. Saks, and Srikanth Srinivasan. Composition limits and separating examples for some Boolean function complexity measures. *Combinatorica*, 36(3):265–311, 2016. doi:10.1007/s00493-014-3189-x.

26 Mika Göös, T. S. Jayram, Toniann Pitassi, and Thomas Watson. Randomized communication versus partition number. *ACM Trans. Comput. Theory*, 10(1), 2018. doi:10.1145/3170711.

27 Hao Huang. Induced subgraphs of hypercubes and a proof of the sensitivity conjecture. *Annals of Mathematics*, 190(3):949–955, 2019. doi:10.4007/annals.2019.190.3.6.

28 Shelby Kimmel. Quantum adversary (upper) bound. *Chicago Journal of Theoretical Computer Science*, 2013(4), April 2013. doi:10.4086/cjtcs.2013.004.

29 Troy Lee, Rajat Mittal, Ben W. Reichardt, Robert Špalek, and Mario Szegedy. Quantum query complexity of state conversion. In *FOCS*, pages 344–353, 2011. doi:10.1109/FOCS.2011.75.

30 Yaqiao Li. Conflict complexity is lower bounded by block sensitivity. *Theor. Comput. Sci.*, 856:169–172, 2021. doi:10.1016/j.tcs.2020.12.038.

31 Ashley Montanaro. A composition theorem for decision tree complexity. *Chicago Journal of Theoretical Computer Science*, 2014(6), July 2014.

32 Noam Nisan. Crew prams and decision trees. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 327–335, 1989.

33 Noam Nisan and Mario Szegedy. On the degree of boolean functions as real polynomials. *Comput. Complex.*, 4:301–313, 1994. doi:10.1007/BF01263419.

34 Ramamohan Paturi. On the degree of polynomials that approximate symmetric boolean functions. In *STOC*, pages 468–474, 1992. doi:10.1145/129712.129758.

35 Ben W. Reichardt. *Reflections for quantum query algorithms*, pages 560–569. SODA. SIAM, 2011. doi:10.1137/1.9781611973082.44.

36 David Rubinstein. Sensitivity vs. block sensitivity of Boolean functions. *Combinatorica*, 15(2):297–299, 1995. doi:10.1007/BF01200762.

37 Petr Savický. On determinism versus unambiquous nondeterminism for decision trees. *Electron. Colloquium Comput. Complex.*, 9(009), 2002. URL: https://eccc.weizmann.ac.il//report/2002/009/.

38 A. A. Sherstov. Strong direct product theorems for quantum communication and query complexity. *SIAM J. Comput.*, 41(5):1122–1165, 2012. doi:10.1137/110842661.

39 Alexander A. Sherstov. Approximating the AND-OR tree. *Theory Comput.*, 9:653–663, 2013. doi:10.4086/toc.2013.v009a020.

40 Alexander A. Sherstov. The intersection of two halfspaces has high threshold degree. *SIAM J. Comput.*, 42(6):2329–2374, 2013. doi:10.1137/100785260.

41 Alexander A. Sherstov. Making polynomials robust to noise. *Theory Comput.*, 9:593–615, 2013. doi:10.4086/toc.2013.v009a018.

42 Xiaoming Sun. Block sensitivity of weakly symmetric functions. *Theor. Comput. Sci.*, 384(1):87–91, 2007. doi:10.1016/j.tcs.2007.05.020.

43 Xiaoming Sun, Andrew Chi-Chih Yao, and Shengyu Zhang. Graph properties and circular functions: How low can quantum query complexity go? In *CCC*, pages 286–293, 2004. doi:10.1109/CCC.2004.1313851.

44 Avishay Tal. Properties and applications of boolean function composition. In *ITCS*, pages 441–454, 2013. doi:10.1145/2422436.2422485.

A Standard definition of complexity measures

We look at many different complexity measures in the paper, let us start with the formal definition of R and $\widetilde{\deg}$.

▶ **Definition 30** (Randomized query complexity (R))**.** *Let $f : \{0,1\}^n \to \{0,1,*\}$ be a (possibly partial) Boolean function. A randomized query algorithm A computes f if $\forall x \in \mathrm{Dom}(f), \Pr[A(x) \neq f(x)] \leq 1/3$, where the probability is over the internal randomness of the algorithm. The cost of the algorithm A, cost(A), is the number of queries made in the worst case over any input as well as internal randomness. The randomized query complexity of f, denoted by $\mathrm{R}(f)$, is defined as*

$$\mathrm{R}(f) = \min_{A \text{ computes } f} cost(A).$$

▶ **Definition 31** (Approximate degree ($\widetilde{\deg}$))**.** *A polynomial $p : \mathbb{R}^n \to \mathbb{R}$ is said to approximate a Boolean function $f : \{0,1\}^n \to \{0,1\}$ if $|p(x) - f(x)| \leq 1/3, \quad \forall x \in \{0,1\}^n$. The approximate degree of f, $\widetilde{\deg}(f)$, is the minimum possible degree of a polynomial which approximates f.*

Note that the constant $1/3$ in the above definitions can be replaced by any constant strictly smaller than $1/2$ which changes $\widetilde{\deg}(f)$ by only a constant factor.

Other than R and $\widetilde{\deg}$, two important related measures are sensitivity ($\mathrm{s}(f)$) and block sensitivity ($\mathrm{bs}(f)$). While the sensitivity and block sensitivity of a total function is well defined, we note that for the case of partial functions there are at least two valid ways of extending the definition from total functions to partial functions. All our results in this paper will hold for partial functions with the following definitions of sensitivity and block sensitivity.

▶ **Definition 32.** *The sensitivity $\mathrm{s}(f,x)$ of a function $f : \{0,1\} \to \{0,1,*\}$ on x is the maximum number s such that there are indices $i_1, i_2, \ldots, i_s \in [n]$ with $f(x^{i_j}) = 1 - f(x)$, for all $1 \leq j \leq s$. Here x^i is obtained from x by flipping the i^{th} bit. The sensitivity of f is defined to be $\mathrm{s}(f) = \max_{x \in \mathrm{Dom}(f)} \mathrm{s}(f,x)$.*

Similarly, the block sensitivity $\mathrm{bs}(f,x)$ of a function $f : \{0,1\} \to \{0,1,*\}$ on x is the maximum number b such that there are disjoint sets $B_1, B_2, \ldots, B_b \subseteq [n]$ with $f(x^{B_j}) = 1 - f(x)$ for all $1 \leq j \leq b$. Recall x^{B_j} is obtained from x by flipping all bits inside B_j. The block sensitivity of f is defined to be $\mathrm{bs}(f) = \max_{x \in \mathrm{Dom}(f)} \mathrm{bs}(f,x)$.

In the definition of block sensitivity, the constraint that the blocks has to be disjoint can be relaxed by extending the definition to "fractional blocks". This gives the measure of fractional block sensitivity.

▶ **Definition 33.** *The fractional block sensitivity $\mathrm{fbs}(f,x)$ of a function $f : \{0,1\} \to \{0,1,*\}$ on x is the maximum value of $\sum_{j=i}^{b} p_j$ such that there are sets $B_1, B_2, \ldots, B_b \subseteq [n]$ and $p_1, \ldots, p_b \in (0,1]$ satisfying the following two conditions.*

- *For each $1 \leq j \leq b$, $f(x^{B_j}) = 1 - f(x)$, and*
- *For each $1 \leq i \leq n$, $\sum_{j:\, i \in B_j} p_j \leq 1$.*

The fractional block sensitivity of f is defined to be $\mathrm{fbs}(f) = \max_{x \in \mathrm{Dom}(f)} \mathrm{fbs}(f,x)$.

B Approximate degree of Promise-OR composed with different inner functions

In this section we show that the approximate degree composes when the outer function is PrOR and the inner functions are (possibly) different partial functions. The proof is essentially a straightforward generalization of the proof of Theorem 21 [11, Theorem 16 (arXiv version)]. However, for the sake of completeness and reader's convenience, we give an overview of the proof here. We will need some definitions and theorems from [11] which we state now. We start with the definition of a problem called “singleton combinatorial group testing”. It generalizes the combinatorial group testing problem.

▶ **Definition 34** (Singleton CGT). *Let D be the set of all $w \in \{0,1\}^{2^n}$ for which there exists an $x \in \{0,1\}^n$ such that for all $S \subseteq [n]$ satisfying $\sum_{i \in S} x_i \in \{0,1\}$, we have $\sum_{i \in S} x_i = w_S$. Note that for all $w \in D$, the string x is uniquely defined by $x_i = w_{\{i\}}$. Let us denote this string by $x(w)$. we then define the partial function $\mathsf{SCGT}_{2^n} : D \to \{0,1\}^n$ by $\mathsf{SCGT}_{2^n}(w) = x(w)$.*

▶ **Theorem 35** ([11, Theorem 19 (arXiv version)]). *The bounded-error quantum query complexity of SCGT_{2^n} is $\Theta(\sqrt{n})$.*

For a formal Definition of bounded error quantum query complexity we refer the survey by [15]. Before we state the next result that we need from [11] we are defining robustness of a polynomial to input noise.

▶ **Definition 36** (Robustness to input noise). *For any function $f : \{0,1\}^n \to \{0,1,*\}$ we say a polynomial $p : \{0,1\}^n \to \mathrm{R}$ approximately computes f with δ-robustness where $\delta \in [0, \frac{1}{2})$ if for any $x \in \mathrm{Dom}(f)$ and $\Delta \in [-\delta, \delta]^n$, we have $|f(x) - p(\Delta + x)| \leq \frac{1}{3}$.*

Now we are ready to state the next result.

▶ **Theorem 37** ([11, Theorem 17 (arXiv version)]). *For a partial Boolean function f, there exists a bounded multilinear polynomial p of degree $O(\mathsf{Q}(f))$ that approximates f with robustness $\Omega(1/\mathsf{Q}(f)^2)$ where $\mathsf{Q}(f)$ is the bounded error quantum query complexity of the function f.*

We refer [11] for more details about robustness of a polynomial induces by quantum algorithm. We also need the existence of a multilinear robust polynomial for $\mathsf{XOR}_n \circ \mathsf{SCGT}_{2^n}$, which follows from Theorems 35 and 37 above, where $\mathsf{XOR}_n \circ \mathsf{SCGT}_{2^n}$ is the parity of n output bits of SCGT_{2^n}.

▶ **Theorem 38** ([11, Theorem 20 (arXiv version)]). *There is a real polynomial p of degree $O(\sqrt{n})$ over 2^n variables $\{w_S\}_{S \subseteq [n]}$ and a constant $c \geq 10^{-5}$ such that for any input $w \in \{0,1\}^{2^n}$ with $\mathsf{XOR}_n \circ \mathsf{SCGT}_{2^n}(w) \neq *$ and any $\Delta \in [-c/n, c/n]^{2^n}$,*

$$|p(w + \Delta) - \mathsf{XOR}_n \circ \mathsf{SCGT}_{2^n}(w)| \leq 1/3.$$

Furthermore, p is multilinear and for all $w \in \{0,1\}^{2^n}$, $p(w) \in [0,1]$.

We also need the following result of Sherstov that shows composition holds for the approximate degree of the parity of n different functions.

▶ **Theorem 39** ([38, Theorem 5.9]). *For any partial Boolean functions $f_1, \dots, f_n$, we have*

$$\widetilde{\mathrm{bdeg}}(\mathsf{XOR} \circ (f_1, \dots, f_n)) = \Omega\left(\sum_{i=1}^{n} \widetilde{\mathrm{bdeg}}(f_i)\right).$$

Theorem 9 can be proved in the similar line of [11, Theorem 20 (arXiv version)], which we are restating below. For the sake of completeness a detailed proof has been added in the full version of the paper [20].

▶ **Theorem 40.** *For any partial Boolean functions $f_1, f_2, \dots, f_n$, we have*

$$\widetilde{\mathrm{bdeg}}\left(\mathsf{PrOR}_n \circ (f_1, f_2, \dots, f_n)\right) = \Omega\left(\frac{\sqrt{n} \cdot \min_{i=1}^{n} \widetilde{\mathrm{bdeg}}(f_i)}{\log n}\right).$$

Furthermore the following upper bound also holds,

$$\widetilde{\mathrm{bdeg}}\left(\mathsf{PrOR}_n \circ (f_1, f_2, \ldots, f_n)\right) = O\left(\sqrt{n} \cdot \max_{i=1}^{n} \widetilde{\mathrm{bdeg}}(f_i) \cdot \log n\right).$$

We will now use this weak bound to establish nearly optimal bound for the approximate degree of PrOR composed with n different *partial* functions. This will again be a simple generalization of OR composed with different functions [11, Theorem 37]. For the sake of completeness, we work out some of the details.

▶ **Theorem 41.** *For any partial Boolean functions $f_1, f_2, \ldots, f_n$, we have*

$$\widetilde{\mathrm{bdeg}}\left(\mathsf{PrOR}_n \circ (f_1, f_2, \ldots, f_n)\right) = \widetilde{\Theta}\left(\sqrt{\sum_{i=1}^{n} \widetilde{\mathrm{bdeg}}(f_i)^2}\right),$$

when the lcm of $\widetilde{\mathrm{bdeg}}(f_i)^2$ for $i \in [n]$ is $\Theta(\max_i\ \widetilde{\mathrm{bdeg}}(f_i)^2)$.

Proof. As mentioned before, the proof is merely working out the details of [11, Theorem 37] while keeping in mind that we are working with *partial* functions.

Let $F = \mathsf{PrOR}_n \circ (f_1, f_2, \ldots, f_n)$, $d_i = \widetilde{\mathrm{bdeg}}(f_i)^2$ for $i \in [n]$, and ℓ be the lcm of d_i's. Now consider the function $G = \mathsf{PrOR}_\ell \circ F$. From Theorem 40, we have the following bounds on $\widetilde{\mathrm{bdeg}}(G)$ up to constants

$$\frac{\sqrt{\ell} \cdot \widetilde{\mathrm{bdeg}}(F)}{\log \ell} \leq \widetilde{\mathrm{bdeg}}(G) \leq \sqrt{\ell} \cdot \widetilde{\mathrm{bdeg}}(F) \cdot \log \ell. \tag{9}$$

Now using the associativity of PrOR we can rewrite G as

$$G = \mathsf{PrOR}_{n\ell} \circ (\underbrace{f_1, \ldots, f_1}_{\ell \text{ times}}, \ldots, \underbrace{f_n, \ldots, f_n}_{\ell \text{ times}}). \tag{10}$$

Further regrouping f_i's, we can rewrite G as follows

$$G = \mathsf{PrOR}_d \circ (\underbrace{\mathsf{PrOR}_{\ell/d_1} \circ f_1, \ldots, \mathsf{PrOR}_{\ell/d_1} \circ f_1}_{d_1 \text{ times}}, \ldots, \underbrace{\mathsf{PrOR}_{\ell/d_n} \circ f_n, \ldots, \mathsf{PrOR}_{\ell/d_n} \circ f_n}_{d_n \text{ times}}), \tag{11}$$

where $d = \sum_{i=1}^{n} d_i$. Now using Theorem 40 and $\sqrt{d_i} = \widetilde{\mathrm{bdeg}}(f_i)$, we obtain following bounds for $\mathsf{PrOR}_{\ell/d_i} \circ f_i$ (up to constants)

$$\frac{\sqrt{\ell}}{\log(\ell/d_i)} \leq \widetilde{\mathrm{bdeg}}(\mathsf{PrOR}_{\ell/d_i} \circ f_i) \leq \sqrt{\ell} \cdot \log(\ell/d_i). \tag{12}$$

Now consider (11) and using Theorem 40 along with (12), we obtain

$$\frac{\sqrt{d\ell}}{\log d \cdot \log \ell} \leq \widetilde{\mathrm{bdeg}}(G) \leq \sqrt{d\ell} \cdot \log \ell \cdot \log d \tag{13}$$

Now from (13) and (9) it follows

$$\frac{\sqrt{d}}{\log d \cdot \log^2 \ell} \leq \widetilde{\mathrm{bdeg}}(F) \leq \sqrt{d} \cdot \log^2 \ell \cdot \log d.$$ ◀

C Composition theorems for strongly-k-junta symmetric outer functions

In this section we will prove the composition result of $\widetilde{\deg}$ and R when the outer function has some amount of symmetry. Of course, there are various notion of symmetry. Traditionally a function is said to have the maximum amount of symmetry when the function value is invariant under any permutation of the variables. Such functions are called symmetric. Symmetric functions are very well studied in the literature of Boolean function analysis. In the terms of composition theorems of $\widetilde{\deg}$ and R it was proved in [11] and [26] that $\widetilde{\deg}$ and R respectively composes when the outer function is symmetric.

In terms of weaker notions of symmetry there are various possible definitions. In this paper we consider the case of strongly-k-junta symmetric functions. The composition theorem for $\widetilde{\deg}$ when the outer function is strongly-k-junta symmetric (Theorem 12(Part(i)) is presented in Appendix C.1. The proof of the composition theorem for R when the outer function is strongly-k-junta symmetric (Theorem 12(Part(ii)) follows easily from Theorem 2.

▶ **Observation 42.** *For any strongly $k-$junta symmetric function $f : \{0,1\}^n \to \{0,1\}$ and any Boolean function $g : \{0,1\}^m \to \{0,1\}$, we have $\mathrm{R}(f \circ g) = \widetilde{\Omega}(\mathrm{R}(f) \cdot \mathrm{R}(g))$ where $n - k = \Theta(n)$.*

Proof. There exists an assignment of the k-bits such that the resulting function is a non-constant symmetric function on $(n-k)$ bits. Since the sensitivity of the restricted function is $\Omega(n)$, the randomized query complexity is also $\Omega(n)$ (see [32]). Hence, from Theorem 2 the result follows. ◀

C.1 Composition of approximate degree for $\sqrt{n}$-junta symmetric functions

In this section, first, we define Multiplexer Function or Addressing Function that will be useful is the analysis.

▶ **Definition 43** (Multiplexer Function or Addressing Function).
The function MUX : $\{0,1\}^{k+2^k} \to \{0,1\}$ *with input* $(x_0, \ldots, x_{k-1}, y_0, \ldots, y_{2^k-1})$ *outputs the bit* y_t, *where* $t = \sum_{i=0}^{k-1} x_i 2^i$.

A crucial result that we use in the prove of composition theorem of $\widetilde{\deg}$ is the following result from [34].

▶ **Theorem 44** ([34]). *For any non-constant symmetric function $f : \{0,1\}^n \to \{0,1\}$, let k be the closest integer to $n/2$ such that f takes different values on inputs of Hamming weight k and $k+1$. Define,*

$$\gamma(f) = \begin{cases} k & \text{if } k \leq n/2, \\ n-k & \text{otherwise.} \end{cases}$$

Then

$$\widetilde{\deg}(f) = \Theta\left(\sqrt{n(\gamma(f)+1)}\right).$$

Using the result of [34] we prove the following proposition about the approximate degree of a k-junta symmetric function. Recall the multiplexer function from Definition 43.

▶ **Proposition 45.** *For any k-junta symmetric function $f : \{0,1\}^n \to \{0,1\}$, we have $\widetilde{\deg}(f) = \Omega\left(\sqrt{(n-k)\gamma_{\max}}\right)$ and $\widetilde{\deg}(f) = O\left(\max\{k, \sqrt{(n-k)\gamma_{\max}}\}\right)$, where $\gamma_{\max} = \max_{i \in \{0,1\}^k}\{\gamma(f_i)\}$ such that f_i is the symmetric function obtained by restricting the junta variables according to i.*

Proof. Fixing the junta variables in f we obtain a symmetric function on $n-k$ variables with approximate degree $\Omega(\sqrt{(n-k)\gamma_{\max}})$ (Theorem 44), which in turn implies the same lower bound on $\widetilde{\deg}(f)$.

For the upper bound, we obtain an approximating polynomial for f by composing the (exact) polynomial for the multiplexer function $\mathsf{MUX} : \{0,1\}^{k+2^k} \to \{0,1\}$ with the approximating polynomials for different symmetric functions obtained by restricting the k junta variables. Therefore, $\widetilde{\deg}(f) = k + O(\sqrt{(n-k)\gamma_{\max}}) = O\left(\max\{k, \sqrt{(n-k)\gamma_{\max}}\}\right)$. ◀

As mentioned earlier, the composition of $\widetilde{\deg}$ when the outer function is symmetric was proved in [11]. The following is their result that we crucially use in the proof of Theorem 12.

▶ **Theorem 46** ([11])**.** *For any symmetric Boolean function $f : \{0,1\}^n \to \{0,1\}$ and any Boolean function $g : \{0,1\}^m \to \{0,1\}$ we have,*

$$\widetilde{\deg}(f \circ g) = \widetilde{\Omega}(\widetilde{\deg}(f) \cdot \widetilde{\deg}(g))$$

We now present the proof of Part (i) of Theorem 12, that the proof of composition of $\widetilde{\deg}$ when the outer function is strongly-k-junta symmetric.

Proof of Theorem 12(Part (i)). Since f is a strongly-k-junta symmetric function so there exists a setting of the k junta variables such that the resulting function is a non-constant symmetric function. Let f' be the symmetric function obtained by restricting the junta variables of f so that f' is non-constant. Then by Theorem 44 the approximate degree of f' is $\Omega(\sqrt{(n-k)\gamma_{\max}})$. Then clearly we have

$$\widetilde{\deg}(f \circ g) \geq \widetilde{\deg}(f' \circ g) = \widetilde{\Omega}(\widetilde{\deg}(f') \cdot \widetilde{\deg}(g)) = \widetilde{\Omega}(\sqrt{(n-k)\gamma_{\max}} \cdot \widetilde{\deg}(g)), \tag{14}$$

where the first equality follows from Theorem 46. Now from Proposition 45 we know that $\widetilde{\deg}(f) = O(\sqrt{(n-k)\gamma_{\max}})$ if $k = O(\sqrt{(n-k)\gamma_{\max}})$, which is satisfied when $k = O(\sqrt{n})$. Thus from (14) we obtain

$$\widetilde{\deg}(f \circ g) = \widetilde{\Omega}(\widetilde{\deg}(f) \cdot \widetilde{\deg}(g)).$$ ◀

Sampling from the Random Cluster Model on Random Regular Graphs at All Temperatures via Glauber Dynamics[*]

Andreas Galanis
Department of Computer Science, University of Oxford, UK

Leslie Ann Goldberg
Department of Computer Science, University of Oxford, UK

Paulina Smolarova
Department of Computer Science, University of Oxford, UK

Abstract

We consider the performance of Glauber dynamics for the random cluster model with real parameter $q > 1$ and temperature $\beta > 0$. Recent work by Helmuth, Jenssen and Perkins detailed the ordered/disordered transition of the model on random Δ-regular graphs for all sufficiently large q and obtained an efficient sampling algorithm for all temperatures β using cluster expansion methods. Despite this major progress, the performance of natural Markov chains, including Glauber dynamics, is not yet well understood on the random regular graph, partly because of the non-local nature of the model (especially at low temperatures) and partly because of severe bottleneck phenomena that emerge in a window around the ordered/disordered transition.

Nevertheless, it is widely conjectured that the bottleneck phenomena that impede mixing from worst-case starting configurations can be avoided by initialising the chain more judiciously. Our main result establishes this conjecture for all sufficiently large q (with respect to Δ). Specifically, we consider the mixing time of Glauber dynamics initialised from the two extreme configurations, the all-in and all-out, and obtain a pair of fast mixing bounds which cover *all* temperatures β, including in particular the bottleneck window. Our result is inspired by the recent approach of Gheissari and Sinclair for the Ising model who obtained a similar-flavoured mixing-time bound on the random regular graph for sufficiently low temperatures. To cover all temperatures in the RC model, we refine appropriately the structural results of Helmuth, Jenssen and Perkins about the ordered/disordered transition and show spatial mixing properties "within the phase", which are then related to the evolution of the chain.

2012 ACM Subject Classification Mathematics of computing → Gibbs sampling; Mathematics of computing → Random graphs; Mathematics of computing → Markov-chain Monte Carlo methods

Keywords and phrases approximate counting, Glauber dynamics, random cluster model, approximate sampling, random regular graphs

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.64

Category RANDOM

Related Version *Full Version*: https://arxiv.org/abs/2305.13239

1 Introduction

For real numbers $q, \beta > 0$ and a graph $G = (V, E)$, the *random cluster model* on G with parameters q and β is a probability distribution on the set $\Omega = \Omega_G$ of all assignments $\mathcal{F} : E \to \{0, 1\}$; we typically refer to assignments in Ω as configurations. For a configuration $\mathcal{F}$, we say that edges mapped to 1 are *in-edges*, and edges mapped to 0 are *out-edges*. We

[*] For the purpose of Open Access, the authors have applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission. All data is provided in full in the results section of this paper.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 64; pp. 64:1–64:12
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

use $\mathrm{In}(\mathcal{F})$ to denote the set of edges e with $\mathcal{F}(e) = 1$, $\mathrm{Out}(\mathcal{F})$ to denote the set of edges e with $\mathcal{F}(e) = 0$, $|\mathcal{F}|$ for the cardinality of $\mathrm{In}(\mathcal{F})$ and $c(\mathcal{F})$ for the number of connected components in the graph $(V, \mathrm{In}(\mathcal{F}))$. Then, the weight of $\mathcal{F}$ in the RC model is given by $w_G(\mathcal{F}) = q^{c(\mathcal{F})}(e^{\beta} - 1)^{|\mathcal{F}|}$.

For integer values of q, the RC model is closely connected to the (ferromagnetic) Ising/-Potts models; $q = 2$ is the Ising model and $q \geq 3$ is the Potts model whose configurations are all possible assignments of q colours to the vertices of the graph where an assignment σ has weight proportional to $\mathrm{e}^{\beta m(\sigma)}$ with $m(\sigma)$ being the number of monochromatic edges under σ. The RC model is an alternative edge representation of the models (for integer q) that has also been studied extensively in its own right due to its intricate behaviour (see, e.g., [19]).

We will be primarily interested in sampling from the so-called *Gibbs distribution* on Ω induced by these weights, denoted by $\pi_G(\cdot)$, where for a configuration $\mathcal{F}$, $\pi_G(\mathcal{F}) = w_G(\mathcal{F})/Z_G$ where the normalising factor $Z_G = \sum_{\mathcal{F}' \in \Omega_G} w_G(\mathcal{F}')$ is the aggregate sum of weight of all configurations (known as the partition function). We focus on the *Glauber dynamics* which is a classical Markov chain for sampling from Gibbs distributions which is a particularly useful tool for developing approximate sampling algorithms. We will refer to Glauber dynamics for the RC model as the RC dynamics. Roughly, the RC dynamics is a Markov chain $(X_t)_{t \geq 0}$ initialised at some configuration X_0 which evolves by iteratively updating at each step $t \geq 1$ a randomly chosen edge based on whether its endpoints belong to the same component in the graph $(V, \mathrm{In}(X_t))$. The mixing time of the chain is the number of steps to get within total variation distance $\leq 1/4$ from π_G, see Section 2 for details.

Our goal is to obtain a fast algorithm for the RC model using Glauber dynamics on the random regular graph. There are two key obstacles that arise, especially at low temperatures (large β): (i) Glauber dynamics for the RC model has a highly non-local behaviour, and (ii) there are severe bottleneck phenomena and worst case graphs which prohibit a general fast-convergence result, and more generally an efficient algorithm. The random regular graph is a particularly interesting testbed in this front since it exhibits all the relevant phase transition phenomena and has also been used as the main gadget in hardness reductions [15].

To overview the phenomena that are most relevant for us, the following picture was detailed in a remarkable development by Jenssen, Helmuth, and Perkins [22]: for $\Delta \geq 5$ and all sufficiently large q, they established the ordered/disordered transition occurring at some β_c satisfying $\beta_c = (1 + o_q(1))\frac{2 \log q}{\Delta}$ (see also [15] for integer $q \geq 3$).[1] Roughly, for $\beta < \beta_c$ a typical configuration of the model is disordered, whereas for $\beta > \beta_c$ it is ordered: disordered configurations resemble the all-out configuration (in that all components are of size $O(\log n)$) whereas ordered configurations resemble the all-in configuration (where there is a giant component with $\Omega(n)$ vertices). The two types of configurations coexist at $\beta = \beta_c$, i.e., each appears with some probability bounded away from zero. The methods in [22] are based on cluster expansion techniques which also yielded an efficient sampling algorithm at all temperatures $\beta > 0$. This is a surprising algorithmic result given that the coexistence causes multimodality in π_G and severe bottleneck phenomena for Markov chains in a window around β_c; it was shown for instance in [22] that the RC dynamics (and the related non-local Swendsen-Wang dynamics) have exponential mixing time, essentially because of the number of steps needed for the chain to move from ordered to disordered (and vice versa).

These results pose a rather bleak landscape for the RC dynamics; yet, on random regular graphs it is widely conjectured that the multimodality and the associated bottlenecks can be circumvented by initialising the chain more judiciously, in particular at either the all-out

[1] Recent results of Bencs, Borbényi, and Csikvári [1] yield the exact formula $\beta_c = \log \frac{q-2}{(q-1)^{1-2/\Delta} - 1}$ for all $q > 2$ and $\Delta \geq 3$, which was previously only known for integer q [15].

or the all-in configurations (depending on whether $\beta \leq \beta_c$). However the tools available for analysing Markov chains are typically insensitive to the initial configuration, and even more so when working at a critical range of the parameters.

Our main result establishes this conjecture for all $\Delta \geq 5$ and q sufficiently large (conditions which we inherit from [22]). For an integer n such that Δn is even, let $\mathcal{G}_{n,\Delta}$ denote the set of all Δ-regular graphs with n vertices.[2] Throughout, we use $O(1)$ to denote a constant depending on q, β, Δ but independent of n.

▶ **Theorem 1.** *Let $\Delta \geq 5$ be an integer. There exists $C = C(\Delta) > 0$ such that, for all sufficiently large q, the following holds for any $\beta > 0$, w.h.p. over $G \sim \mathcal{G}_{n,\Delta}$.*

1. *For $\beta < \beta_c$, the mixing time of the RC dynamics starting from all-out is $O(n \log n)$.*
2. *For $\beta > \beta_c$, the mixing time of the RC dynamics starting from all-in is $O(n^C)$. For integer q, the mixing time is in fact $O(n \log n)$.*

Note that Theorem 1 implies an $O(n \log n)$ sampling algorithm from the Potts model for all $\beta \neq \beta_c$ (and all sufficiently large q). Intuitively, and as we will see later in more detail, Theorem 1 asserts that the RC dynamics starting from all-in mixes quickly within the set of ordered configurations for $\beta > \beta_c$, and similarly it mixes well within the disordered set of configurations starting from all out when $\beta < \beta_c$. In fact the same is true for $\beta = \beta_c$ and hence the RC dynamics can be used to sample even at criticality, see Remark 12 for details.

Finally, let us note that the RC dynamics can be used analogously to the theorem above to produce a sample within total variation distance ε of π_G for any $\varepsilon \geq \mathrm{e}^{-\Theta(n)}$, by running it for a number of steps which is $\log(1/\varepsilon)$ times the corresponding mixing time bound.[3] The lower bound on the error comes from the total variation distance between π_G and the conditional "ordered" and "disordered" configurations, see Lemma 3.

1.1 Further related work

Our approach to proving Theorem 1 is inspired from a recent paper by Gheissari and Sinclair [16] who established similar flavoured results for the Ising model ($q = 2$) on the random regular graph for large β. To obtain our results for all β, we adapt suitably their notion of "spatial mixing within the phase", see Section 2.2 for details.

Among the results in [16], it was established that Glauber dynamics on the random regular graph, initialised appropriately, mixes in $O(n \log n)$ time when β is sufficiently large.[4] More recently, Gheissari and Sinclair [17] obtained mixing-time bounds for the RC dynamics on the lattice $\mathbb{Z}^d$ under appropriate boundary conditions. They also analyse the mixing time starting from a mixture of the all-in/all-out initialisation. Note that the phase transition on grid lattices is qualitatively different than that of the random regular graph; there, instead of a window/interval of temperatures, the three points β_u, β_u^* and β_c all coincide into a single phase transition point. See also [6, 23] for related algorithmic results on $\mathbb{Z}^d$ using cluster expansion methods.

For the random regular graph, Blanca and Gheissari [4] showed for all integer $\Delta \geq 3$ and real $q \geq 1$ that the mixing time is $O(n \log n)$ provided that $\beta < \beta_u(q, \Delta)$ where β_u is the uniqueness threshold on the tree. A sampling algorithm (not based on MCMC) for

[2] We write $G \sim \mathcal{G}_{n,\Delta}$ to denote a graph in $\mathcal{G}_{n,\Delta}$ chosen uniformly at random, and we say that a property holds w.h.p. for $G \sim \mathcal{G}_{n,\Delta}$ as a shorthand for "with probability $1 - o_n(1)$ over a graph $G \in \mathcal{G}_{n,\Delta}$ chosen uniformly at random.

[3] The standard submultiplicative argument to bootstrap the total-variation distance goes through using the monotonicity of the RC model (to account for the constraint on the initial configuration), see also [25].

[4] Note that, for $q = 2$, an $O(n^{10})$ upper bound for the RC dynamics on any graph G was previously known at all temperatures β by Guo and Jerrum [20] (see also [13]).

$\beta < \beta_c(q,\Delta)$ and $q,\Delta \geq 3$ was designed by Efthymiou [12] (see also [3]), albeit achieving weaker approximation guarantees. Coja-Oghlan et al. [10] showed that, for all integer $q,\Delta \geq 3$ and $\beta \in (\beta_u, \beta'_u)$ the mixing time is $\mathrm{e}^{\Omega(n)}$ where $\beta'_u = \log(1+\frac{q}{\Delta-2}) > \beta_u$ is (conjectured to be) another uniqueness threshold on the tree (see [21, 24]). More generally, for integer $q \geq 3$, the hardness results/techniques of [18, 15] yield that for any $\beta > \beta_c$, there are graphs G where the mixing time of the RC dynamics is $\exp(n^{\Omega(1)})$ and the problem of appoximately sampling on graphs of max-degree Δ becomes #BIS-hard; on the other hand, for $\beta \leq (1-o_{q,\Delta}(1))\beta_c$ it has been shown in [11, 7] that the cluster-expansion technique of [22] yields a sampling algorithm on any max-degree Δ graph.

As a final note, another model of interest where analogous mixing results for Glauber dynamics (initialised appropriately) should be obtainable is for sampling independent sets on random bipartite regular graphs. However, in contrast to the RC/Potts models, the phase transition there is analogous to that of the Ising model, and hence, establishing the relevant spatial mixing properties close to the criticality threshold is likely to require different techniques, see, e.g., [9] for more discussion.

1.2 Independent results of Blanca and Gheissari

In an independent and simultaneous work, Blanca and Gheissari [5] obtain related (but incomparable) results. For $\Delta \geq 3, q \geq 1$ and arbitrarily small $\tau > 0$, they show for sufficiently large β a mixing time bound of $O(n^{1+\tau})$ for the RC dynamics on the random regular graph starting from an arbitrary configuration (and obtain an analogous result for the grid and the Swendsen-Wang dynamics). Our result instead applies to all β for the random regular graph (even the critical window) by taking into consideration the initial configuration; the two papers have different approaches to obtain the main ingredients.

2 Proof of Theorem 1

We start with the formal description of the RC dynamics. Given a graph $G = (V, E)$ and an initial configuration $X_0 : E \to \{0, 1\}$, the RC dynamics on G is a Markov chain $(X_t)_{t\geq 0}$ on the set of configurations Ω_G. Let $p := 1 - \mathrm{e}^{-\beta}$ and $\hat{p} := \frac{p}{(1-p)q+p}$ (note that for $q > 1$ it holds that $\hat{p} \in (p/q, p)$). For $t \geq 0$, to obtain X_{t+1} from X_t:

1. Choose u.a.r. an edge $e \in E$. If e is a cut-edge in the graph $(V, \mathrm{In}(X_t) \cup \{e\})$, set $X_{t+1}(e) = 1$ with probability $\hat{p}$ (and $X_{t+1}(e) = 0$ otherwise). Else, set $X_{t+1}(e) = 1$ with probability p, and $X_{t+1}(e) = 0$ otherwise.
2. Set $X_{t+1}(f) = X_t(f)$ for all $f \in E\backslash\{e\}$.

It is a standard fact that the distribution of X_t converges to the RC distribution π_G. Let $T_{\mathrm{mix}}(G; X_0) = \min_{t\geq 0}\{t \mid \mathrm{dist}_{\mathrm{TV}}(X_t, \pi_G) \leq 1/4\}$ be the number of steps needed to get within total-variation distance $\leq 1/4$ from π_G starting from X_0, and $T_{\mathrm{mix}}(G) = \max_{X_0} T_{\mathrm{mix}}(G; X_0)$ be the mixing time from the worst starting state.

2.1 The ordered and disordered phases on random regular graphs

We review in more detail the ordered/disordered transition, following [22].

▶ **Definition 2.** *For $\Delta \geq 3$, let $\eta = \eta(\Delta) \in (0, 1/2)$ be a small constant (see Definition 17). For $G \in \mathcal{G}_{n,\Delta}$, the* ordered phase *is the set of configurations $\Omega^{\mathrm{ord}} := \{\mathcal{F} \in \Omega : |\mathrm{In}(\mathcal{F})| \geq (1-\eta)|E|\}$, whereas the* disordered phase *is the set $\Omega^{\mathrm{dis}} := \{\mathcal{F} \in \Omega : |\mathrm{In}(\mathcal{F})| \leq \eta|E|\}$. For $q, \beta > 0$, let $\pi^{\mathrm{ord}}_G, \pi^{\mathrm{dis}}_G$ be the conditional distributions of π_G on $\Omega^{\mathrm{ord}}, \Omega^{\mathrm{dis}}$, respectively.*

We will use the following result of Helmuth, Jenssen and Perkins [22, Lemma 9].

▶ **Lemma 3** ([22, Theorem 1]). *Let $\Delta \geq 5$ be an integer. Then, for all sufficiently large q, there exists $\beta_c > 0$ satisfying $\beta_c = (1 + o_q(1))\frac{2\log q}{\Delta}$ such that the following holds for any $\beta > 0$ w.h.p. for $G \sim \mathcal{G}_{n,\Delta}$.*

$$\text{if } \beta < \beta_c \text{, then } \left\|\pi_G - \pi_G^{\mathrm{dis}}\right\|_{\mathrm{TV}} = \mathrm{e}^{-\Omega(n)}; \qquad \text{if } \beta > \beta_c \text{, then } \left\|\pi_G - \pi_G^{\mathrm{ord}}\right\|_{\mathrm{TV}} = \mathrm{e}^{-\Omega(n)}. \tag{1}$$

Moreover, there exists $\zeta = \zeta(\Delta) > 0$ with $\zeta < \eta$ such that

$$\begin{aligned} &\text{for } \beta \leq \beta_c, \quad \pi_G^{\mathrm{dis}}\Big(|\operatorname{In}(\mathcal{F})| \geq \zeta|E|\Big) = \mathrm{e}^{-\Omega(n)}, \qquad \text{and} \\ &\text{for } \beta \geq \beta_c, \quad \pi_G^{\mathrm{ord}}\Big(|\operatorname{In}(\mathcal{F})| \leq (1-\zeta)|E|\Big) = \mathrm{e}^{-\Omega(n)}. \end{aligned} \tag{2}$$

Proof. The claims about the total variation distance are shown in [22, Theorem 1, Items (2), (3), (8)]. Equation (2) shows a bit of slack in the definitions of Ω^{dis} and Ω^{ord} that will be useful later; it follows essentially from the same theorem, we defer the details to Lemma 25 of the full version [14]. ◀

2.2 Main ingredient: Weak spatial mixing within a phase

Let $G = (V, E)$ be a graph. For $v \in V$ and $r \geq 0$, let $B_r(v)$ denote the set of all vertices in V whose distance from v is at most r. Let $\pi = \pi_G$ be the RC distribution on G and let $\pi_{\mathcal{B}_r^+(v)}$ be the conditional distribution of π where all edges in $E\backslash E(B_r(v))$ are "in". We define analogously $\pi_{\mathcal{B}_r^-(v)}$ by conditioning the edges in $E\backslash E(B_r(v))$ to be "out".

▶ **Definition 4.** *Let G be a graph with m edges. Let $q, \beta > 0$ be reals and $r \geq 1$ be an integer. We say that the graph G has* WSM within the ordered phase at radius r *if for every $v \in V(G)$ and every edge e incident to v, $\|\pi_{\mathcal{B}_r^+(v)}(e \mapsto \cdot) - \pi_G^{\mathrm{ord}}(e \mapsto \cdot)\|_{\mathrm{TV}} \leq \frac{1}{100m}$. Analogously, we say that G has* WSM within the disordered phase at radius r *if for every $v \in V(G)$ and every edge e incident to v, $\|\pi_{\mathcal{B}_r^-(v)}(e \mapsto \cdot) - \pi_G^{\mathrm{dis}}(e \mapsto \cdot)\|_{\mathrm{TV}} \leq \frac{1}{100m}$.*

The bulk of our arguments consists of showing the following two theorems.

▶ **Theorem 5.** *Let $\Delta \geq 5$ be an integer. There exists $M = M(\Delta) > 0$ such that for all q sufficiently large, the following holds for any $\beta \geq \beta_c$. W.h.p. over $G \sim \mathcal{G}_{n,\Delta}$, G has WSM within the ordered phase at a radius r which satisfies $r \leq \frac{M}{\beta}\log n$.*

The upper bound on the radius r in terms of $1/\beta$ ensures that we can remove the dependence on β of the mixing time in Theorem 1 (caused by a loose bound on the mixing time on the tree, see Lemma 9 below). For the disordered phase, we have

▶ **Theorem 6.** *For all integer $\Delta \geq 5$, for all q sufficiently large and any $\beta \leq \beta_c$, w.h.p. over $G \sim \mathcal{G}_{n,\Delta}$, G has WSM within the disordered phase at a radius r which satisfies $r \leq \frac{1}{3}\log_{\Delta-1} n$.*

2.3 Second ingredient: Local mixing on tree-like neighbourhoods

We first define a local version of RC dynamics where we perform only updates in a small ball around a vertex. Here, we need to consider the extreme boundary conditions that all vertices outside of the ball belong in distinct components ("free boundary") and where they belong to the same component ("wired boundary"); we will refer to these two chains as the free and wired RC dynamics, respectively. For the random regular graph, these "local-mixing" considerations are strongly connnected to the Δ-regular tree.

Formally, given a graph $G = (V, E)$ and a subset $U \subseteq V$, let $G[U]$ be the induced subgraph of G on U. The tree-excess of a connected graph G is given by $|E| - |V| + 1$. For a vertex v in G and integer $r \geq 0$, let $B_r(v)$ denote the set of vertices at distance at most r from v and $S_r(v)$ those at distance exactly r from v. For $K > 0$, a max-degree Δ graph G is locally K-treelike if for every $v \in V$ and $r \leq \frac{1}{3}\log_{\Delta-1}|V|$, the graph $G[B_r(v)]$ has tree excess $\leq K$.

▶ **Lemma 7** (see, e.g., [16, Lemma 5.8]). *For any integer $\Delta \geq 3$, there is $K > 0$ such that w.h.p. $G \sim G_{n,\Delta}$ is locally K-treelike.*

For a graph G, a vertex ρ in G and an integer $r \geq 1$, the *free RC dynamics* on $B_r(\rho)$ is the RC dynamics where all edges outside of $B_r(\rho)$ are conditioned to be out and only edges of G with both endpoints in $B_r(\rho)$ are updated.

▶ **Lemma 8** ([4, Lemma 6.5]). *Let $\Delta \geq 3$ be an integer, and $q, K > 1$, $\beta > 0$ be reals. There exists $C > 0$ such that the following holds for any Δ-regular graph G and integer $r \geq 1$.*

Suppose that $\rho \in V$ is such that $G[B_r(\rho)]$ is K-treelike. Then, with $n = |B_r(\rho)|$, the mixing time of the free RC dynamics on $B_r(\rho)$ is $\leq Cn\log n$.

To define the wired RC dynamics, for a graph G, a vertex ρ in G and an integer $r \geq 1$, let H be the graph obtained by removing all vertices and edges outside of $B_r(\rho)$, and adding a new vertex v_∞ connected to all vertices in $S_r(\rho)$. The *wired RC dynamics* on $B_r(\rho)$ is the RC dynamics on H where the edges adjacent to v_∞ are conditioned to be in and only edges of G with both endpoints in $B_r(\rho)$ are updated. Denote by $\hat{\pi}_{B_r(\rho)}$ the stationary distribution of the wired RC dynamics. Note that when the graph outside of $B_r(\rho)$ is connected, $\hat{\pi}_{B_r(\rho)}$ induces the same distribution as $\pi_{\mathcal{B}_r^+(\rho)}$.[5]

▶ **Lemma 9.** *Let $\Delta \geq 3$ be an integer, and $q, K > 1$, $\beta > 0$ be reals. There exists $\hat{C} > 0$ such that the following holds for every Δ-regular graph $G = (V, E)$ and any integer $r \geq 1$.*

Suppose that $\rho \in V$ is such that $G[B_r(\rho)]$ is K-treelike. Then, with $n = |B_r(\rho)|$, the mixing time of the wired RC dynamics on $B_r(\rho)$ is $\leq \hat{C}n^3(q^4\mathrm{e}^\beta)^{\Delta r}$.

▶ Remark 10. For integer $q > 1$, the mixing time bound in Lemma 9 can be improved to $O(n\log n)$ using results of [2], see Appendix A.2. in the full version [14] for details.

2.4 Proof of Theorem 1

In this section, we will prove the $\beta > \beta_c$ part of Theorem 1, given below as Theorem 11 for convenience. The proof of the $\beta < \beta_c$ part of Theorem 1 is in Appendix A of the full version [14].

▶ **Theorem 11.** *Let $\Delta \geq 5$ be an integer. Then, for all sufficiently large q, there exists $C = C(q, \Delta)$ such that the following holds w.h.p. for $G \sim \mathcal{G}_{n,\Delta}$. For $\beta > \beta_c$, the mixing time of the RC dynamics starting from all-in is $O(n^C)$. For integer q, the mixing time is in fact $O(n\log n)$.*

Proof of Theorem 11 (Theorem 1(b)). The argument resembles that of [16], a bit of care is required to combine the pieces. Consider $G = (V, E) \sim \mathcal{G}_{n,\Delta}$ with $n = |V|$ and $m = |E|$. Let q be sufficiently large so that both Lemma 3 and Theorem 5 apply; assume also that Lemma 7 applies so that G is locally K-treelike.

[5] More precisely, the weight of a configuration $\mathcal{F} : E(B_r(\rho)) \to \{0,1\}$ in $\hat{\pi}_{B_r(\rho)}$ is proportional to $q^{\hat{c}(\mathcal{F})}(\mathrm{e}^\beta - 1)^{|\mathcal{F}|}$ where $\hat{c}(\mathcal{F})$ denotes the number of components in the graph $(B_r(\rho), \mathrm{In}(F))$ that do not include any of the vertices in $S_r(\rho)$ (since all of these belong to the same component in the wired dynamics and hence contribute just a single extra factor of q).

Consider arbitrary $\beta > \beta_c$ and set $\beta_0 := \log(q^{1.9/\Delta} + 1)$. Since $\beta_c = (1 + o_q(1))\frac{2\log q}{\Delta}$, we have that $\beta \geq \beta_0$ for all sufficiently large q. By Lemma 3, a graph $G = (V, E) \sim \mathcal{G}_{n,\Delta}$ satisfies w.h.p. $\left\|\pi_G - \pi_G^{\mathrm{ord}}\right\|_{\mathrm{TV}} = \mathrm{e}^{-\Omega(n)}$ and $\pi_G^{\mathrm{ord}}\big(|\operatorname{In}(\mathcal{F})| \leq (1-\zeta)|E|\big) = \mathrm{e}^{-\Omega(n)}$. Moreover, by Theorem 5, G has WSM within the ordered phase at radius r for some $r \leq \frac{M}{\beta}\log n$, where $M = M(\Delta) > 0$ is a constant independent of β. Note that by taking q large, we can ensure that β_0 and hence β are at least M so that $r \leq \frac{1}{3}\log_{\Delta-1} n$. Theorem 11 will follow by showing that the mixing time is bounded by $T = O(n^{2+\log W})$, where $W = \Delta^{2M/\beta_0}\mathrm{e}^{M\Delta(\Delta+1)}$ is independent of q, β; for integer q we will show the stronger upper bound $\tilde{T} = O(n\log n)$.

Consider the RC dynamics $(X_t)_{t\geq 0}$ with X_0 being the all-in configuration on the edges. Consider also the "ordered" RC dynamics $\hat{X}_t$ with $\hat{X}_0 \sim \pi_G^{\mathrm{ord}}$ where we reject moves that lead to configurations outside of Ω^{ord}; note that $\hat{X}_t \sim \pi_G^{\mathrm{ord}}$ for all $t \geq 0$. For $t \geq 0$, let $\mathcal{E}_t$ be the event that $\operatorname{In}(\hat{X}_t) \geq (1-\zeta)|E|$ and let $\mathcal{E}_{<t} := \bigcap_{t'=0,\dots,t-1}\mathcal{E}_{t'}$. From Lemma 3 we have that $\pi_G^{\mathrm{ord}}(\mathcal{E}_t) \geq 1 - \mathrm{e}^{-\Omega(n)}$ and hence by a union bound $\pi_G^{\mathrm{ord}}(\mathcal{E}_{<t}) \geq 1 - t\mathrm{e}^{-\Omega(n)}$ as well.

We couple the evolution of X_t and $\hat{X}_t$ using the monotone coupling, i.e., at every step of the two chains choose the same edge e_t to update and use the same uniform number $U_t \in [0, 1]$ to decide whether to include e_t in each of $X_{t+1}, \hat{X}_{t+1}$. Using the monotonicity of the model for $q \geq 1$ (and in particular that $p > \hat{p}$), under the monotone coupling, for all $t \geq 0$ such that $\mathcal{E}_{<t}$ holds (and hence no reject move has happened in $\hat{X}_t$ so far), we have that $\hat{X}_t \leq X_t$ (i.e., $\operatorname{In}(\hat{X}_t) \subseteq \operatorname{In}(X_t)$). To complete the proof, it therefore suffices to show that

$$\Pr(X_T \neq \hat{X}_T) \leq 1/4. \tag{3}$$

Consider an arbitrary time $t \geq 0$. By a union bound, we have that

$$\Pr\left(X_t \neq \hat{X}_t\right) \leq \sum_e \Pr\left(X_t(e) \neq \hat{X}_t(e)\right) \leq m\Pr\left(\overline{\mathcal{E}_{<t}}\right) + \sum_e \Pr\left(X_t(e) \neq \hat{X}_t(e) \mid \mathcal{E}_{<t}\right). \tag{4}$$

Fix an arbitrary edge e incident to some vertex v, and let (X_t^v) be the wired RC dynamics on $G[B_r(v)]$. We couple the evolution of (X_t^v) with that of (X_t) and $(\hat{X}_t)$ using the monotone coupling analogously to above, where in X_t^v we ignore updates of edges outside the ball $G[B_r(v)]$). We have $X_t^v \geq X_t$ for all $t \geq 0$, and hence, conditioned on $\mathcal{E}_{<t}$, we have that $X_t^v \geq X_t \geq \hat{X}_t$. It follows that

$$\begin{aligned}\Pr\left(X_t(e) \neq \hat{X}_t(e) \mid \mathcal{E}_{<t}\right) &= \Pr(X_t(e) = 1 \mid \mathcal{E}_{<t}) - \Pr(\hat{X}_t(e) = 1 \mid \mathcal{E}_{<t})\\ &\leq |\Pr(X_t^v(e) = 1 \mid \mathcal{E}_{<t}) - \Pr(\hat{X}_t(e) = 1 \mid \mathcal{E}_{<t})|.\end{aligned}$$

For any two events A, B, we have $|\Pr(A) - \Pr(A \mid B)| \leq 2\Pr(\overline{B})$, so using this for $B = \mathcal{E}_{<t}$ and A the events $\{X_t^v(e) = 1\}, \{\hat{X}_t(e) = 1\}$, the triangle inequality gives

$$\Pr\left(X_t(e) \neq \hat{X}_t(e) \mid \mathcal{E}_{<t}\right) \leq 4\Pr\left(\overline{\mathcal{E}_{<t}}\right) + |\Pr(X_t^v(e) = 1) - \Pr(\hat{X}_t(e) = 1)|$$

Note that $\Pr(\hat{X}_t(e) = 1) = \pi_G^{\mathrm{ord}}(e \mapsto 1)$, so another application of triangle inequality gives

$$\begin{aligned}\Pr\left(X_t(e) \neq \hat{X}_t(e) \mid \mathcal{E}_{<t}\right) \leq 4\Pr\left(\overline{\mathcal{E}_{<t}}\right) &+ \left|\Pr(X_t^v(e) = 1) - \pi_{\mathcal{B}_r^+(v)}(e \mapsto 1)\right|\\ &+ \left|\pi_{\mathcal{B}_r^+(v)}(e \mapsto 1) - \pi_G^{\mathrm{ord}}(e \mapsto 1)\right|.\end{aligned} \tag{5}$$

Since G has WSM within the ordered phase at radius r, we have that

$$\left|\pi_{\mathcal{B}_r^+(v)}(e \mapsto 1) - \pi_G^{\mathrm{ord}}(e \mapsto 1)\right| \leq 1/(100m). \tag{6}$$

Moreover, let T_v be the mixing time of the wired RC dynamics on $G[B_r(v)]$ and let $N_v = |E(B_r(v))| \le \Delta^{r+1}$. Since $r \le \frac{1}{3}\log_{\Delta-1} n$, $G[B_r(v)]$ is K-treelike, so from Lemma 9, with $\hat{C} = O(1)$ denoting the constant there (and absorbing a couple of factors of Δ into it),

$$T_v \le \hat{C}(N_v)^3(q^4 \mathrm{e}^{\beta})^{\Delta r} \le \hat{C}N_v\Delta^{2r}q^{4\Delta r}\mathrm{e}^{\beta\Delta r} = \hat{C}N_v\Big(\Delta^{2M/\beta}q^{4\Delta M/\beta}\mathrm{e}^{M\Delta}\Big)^{\log n} \le \hat{C}N_v W^{\log n},$$

where in the last inequality we used that $\beta > \beta_0$, $\beta_0 > \frac{1}{\Delta}\log q$ and $W = \Delta^{2M/\beta_0}\mathrm{e}^{M\Delta(\Delta+1)}$. For $T = \Theta(n^{2+\log W})$, we have $T \ge 40T_v\frac{m}{N_v}\log m$, so by Chernoff bounds, with probability $1-\exp(-n^{\Omega(1)})$, we have at least $10T_v \log m$ updates inside $B_r(v)$ among $t = 1,\dots,T$. For integer $k \ge 1$ the distance from stationarity after kT_v steps is at most $(1/4)^k$, we obtain

$$\big|\Pr(X_T^v(e)=1) - \pi_{\mathcal{B}_r^+(v)}(e \mapsto 1)\big| \le \exp(-n^{\Omega(1)}) + \mathrm{e}^{-4\log m} \le 1/m^3. \tag{7}$$

Plugging (6) and (7) into (5) for $t = T$, and then back into (4), we obtain using $\Pr(\overline{\mathcal{E}_{<T}}) \le T\mathrm{e}^{-\Omega(n)}$ that $\Pr(X_T \ne \hat{X}_T) \le 5mT\mathrm{e}^{-\Omega(n)} + m/m^3 + 1/100 \le 1/4$, as needed.

For integer q, to get the improved mixing time bound $O(n\log n)$ in Theorem 11 the reasoning is similar. The main difference is that for integer q, we have that for any vertex v the mixing time T_v is bounded by $T_v = O(N_v \log N_v)$ (cf. Remark 10), and therefore the above argument yields a mixing time upper bound of $O(n(\log n)^2)$. With a bit more care, for $\tilde{T} = \Theta(n\log n)$, we show in Appendix A.3 in the full version [14] using a log-Sobolev inequality that

$$\big|\Pr(X_{\tilde{T}}^v(e)=1) - \pi_{\mathcal{B}_r^+(v)}(e \mapsto 1)\big| \le 1/m^3, \tag{8}$$

which analogously to above yields $\Pr(X_{\tilde{T}} \ne \hat{X}_{\tilde{T}}) \le 1/4$, and hence the desired mixing time bound of $O(n\log n)$ for integer q. ◀

▶ Remark 12. All the ingredients to show the coupling of the RC dynamics starting from all-in with π_G^{ord} (i.e., (3)) work even at criticality, i.e., for $\beta = \beta_c$; a similar observation applies at $\beta = \beta_c$ for π_G^{dis} when starting the RC dynamics from all-out. The difference at criticality is that π_G is a mixture of π_G^{ord} and π_G^{dis}, i.e., to obtain a sample for π_G, one should output a sample for π_G^{ord} with some probability Q and otherwise a sample from π_G^{dis}. The value of Q can be computed in time $\tilde{O}(n^2)$ by approximating the corresponding partition functions, by using, e.g., the algorithms in [22, 8] (or even the RC dynamics itself). See also [22, Theorems 2 & 3] for precise results characterising the distribution of Q; it is shown for example that Q converges to $1/(q+1)$ as q grows large.

3 Proof outline of the WSM within the ordered phase

3.1 Locally tree-like expanders

Analogously to [22], we work a bit more generally with Δ-regular expanders, which are also tree-like. The *expansion profile* of an n-vertex graph $G = (V, E)$ for $\varepsilon > 0$ is given by

$$\phi_G(\varepsilon) := \min_{S\subseteq V;\ 0<|S|\le\varepsilon n} \frac{|E(S, V\backslash S)|}{\Delta|S|}.$$

Then the classes $G_{\Delta,\delta}$ and $\mathcal{G}_{\Delta,\delta,K}$ are as follows.

▶ **Definition 13.** *Let $\Delta \ge 5$ be an integer, and $\delta \in (0, 1/2), K > 0$ be reals. $G_{\Delta,\delta}$ is the class of Δ-regular graphs such that $\phi_G(1/2) \ge 1/10$ and $\phi_G(\delta) \ge 5/9$. $\mathcal{G}_{\Delta,\delta,K}$ is the class of all locally K-treelike graphs $G \in \mathcal{G}_{\Delta,\delta}$.*

We use the following lemma.

▶ **Lemma 14** ([22, Proposition 37]). *Fix $\Delta \geq 5$. There is a constant $\delta \in (0, 1/2)$ such that w.h.p. a uniformly random Δ-regular graph belongs to $\mathcal{G}_{\Delta,\delta}$.*

Lemma 14 and Lemma 7 show that there is also a positive integer K such that, w.h.p, $G \in \mathcal{G}_{\Delta,\delta,K}$. Next we state an important property of expanders from [26].

▶ **Lemma 15** ([26, Lemma 2.3]). *Let $G = (V, E)$ be a regular graph and consider $E' \subseteq E$ with $|E'| \leq \theta|E|$ for some $\theta \in (0, \phi_G(1/2))$. Then $(V, E\backslash E')$ has a component of size at least $(1 - \frac{\theta}{2\phi_G(1/2)})|V|$.*

We use Lemma 15 to establish the existence of a giant component.

▶ **Definition 16.** *The size of a component of a graph is the number of vertices in the component. A* giant component *in an n-vertex graph is a component whose size is greater than $n/2$. Given a graph $G = (V, E)$ and a subset $F \subseteq E$, $G[F]$ denotes the graph (V, F).*

▶ **Definition 17.** *Fix $\Delta \geq 5$. Fix $\delta \in (0, 1/2)$ satisfying Lemma 14. Let $\eta = \min(\delta/5, 1/100)$.*

▶ **Corollary 18.** *Fix integers $\Delta \geq 5$ and $K \geq 0$ and a real number $\delta \in (0, 1/2)$. Let G be a graph in $\mathcal{G}_{\Delta,\delta,K}$ and let $\mathcal{F}$ be a configuration in Ω^{ord} or a partial configuration with $|\operatorname{In}(\mathcal{F})| \geq (1 - \eta)|E|$. Then there is a giant component in $G[\operatorname{In}(\mathcal{F})]$ whose size is at least $(1 - \delta)|V|$.*

Proof. Apply Lemma 15 with $E' = \operatorname{Out}(\mathcal{F})$ and $\theta = \eta = \min(\delta/5, 1/100)$. Note $|\operatorname{Out}(\mathcal{F})| \leq \eta|E|$ and $\phi_G(1/2) \geq 1/10$. Thus the lemma say that $G[\operatorname{In}(\mathcal{F})]$ has a component of size at least $\left(1 - \frac{\delta/5}{2 \cdot 1/10}\right)|V| = (1 - \delta)|V| > |V|/2$. ◀

3.2 Sketch of proof of Theorem 5

Let $\Delta \geq 5$ be an integer. Consider any sufficiently large q and any $\beta \geq \beta_c$. For sufficiently large n, choose a "radius" $r \approx \frac{1}{\beta}\log n$ and let $G = (V, E) \sim \mathcal{G}_{n,\Delta}$. Fix a vertex $v \in V$ and an edge e incident to v. We wish to show, with sufficiently high probability, that $\|\pi_{\mathcal{B}_r^+(v)}(e \mapsto \cdot) - \pi_G^{\text{ord}}(e \mapsto \cdot)\|_{\text{TV}} \leq 1/(100|E|)$.

Our goal is essentially to construct a coupling of $\mathcal{F}^+ \sim \pi_{\mathcal{B}_r^+(v)}$ and $\mathcal{F}^{\text{ord}} \sim \pi^{\text{ord}}$, such that $\Pr(\mathcal{F}^+(e) \neq \mathcal{F}^{\text{ord}}(e))$ is sufficiently small. In order to construct the coupling, we take advantage of the fact that $G[B_r(v)]$ is locally tree-like. In fact, we identify a suitable subgraph of $G[B_r(v)]$ without cycles and restrict the coupling to this subgraph.

Consider a breadth-first search from v in $G[B_r(v)]$. Let T_0 be the rooted tree consisting of all forward edges in this breadth-first search. All other edges in $B_r(v)$ are called "excess edges". W.h.p., since $G \sim \mathcal{G}_{n,\Delta}$, there are at most K excess edges in $B_r(v)$ for some absolute constant $K > 0$. In particular, since G is locally tree-like, we can identify integers r_1 and r_2 satisfying $r \geq r_1 > r_2 \geq 0$ such that $E(B_{r_1}(v)) \setminus E(B_{r_2}(v))$ contains no excess edges and $r_1 - r_2 \geq r/(2K) = \Omega(r)$. The fact that $r_1 - r_2 = \Omega(r)$ ensures that $B_{r_1}(v) \setminus B_{r_2}(v)$ is a sufficiently large subgraph of G, and the coupling focuses on this subgraph.

In order to describe the coupling process we need a small amount of notation. A *partial configuration* $\mathcal{F}$ is a map from the edges of G to the set $\{0, 1, *\}$. In-edges and out-edges (those that are mapped to 1 or to 0) are "revealed" and edges that are mapped to $*$ are "unrevealed". A refinement of a partial configuration is obtained by revealing more edges. We use $\mathcal{F} \subseteq \mathcal{F}'$ to denote the fact that $\mathcal{F}'$ refines $\mathcal{F}$.

In the coupling, we generate a sequence of edge subsets $F_0 \subseteq F_1 \subseteq \cdots \subseteq E$ such that, after iteration i, the edges in F_i are revealed. We also construct two sequences of partial configurations $\mathcal{F}_0^+ \subseteq \mathcal{F}_1^+ \subseteq \cdots \subseteq \mathcal{F}^+$ and $\mathcal{F}_0^{\mathrm{ord}} \subseteq \mathcal{F}_1^{\mathrm{ord}} \subseteq \cdots \subseteq \mathcal{F}^{\mathrm{ord}}$, maintaining the invariant that the revealed edges in $\mathcal{F}_i^{\mathrm{ord}}$ and $\mathcal{F}_i^+$ are exactly the edges in F_i. The coupling will have the crucial property that $\mathcal{F}^{\mathrm{ord}} \sim \pi^{\mathrm{ord}}$ and $\mathcal{F}^+ \sim \pi_{\mathcal{B}_{r_1}^+(v)}$

- The process starts with iteration $i = 0$. The initial set F_0 of revealed edges is all edges except those in $E(B_{r_1}(v))$. In $\mathcal{F}_0^{\mathrm{ord}}$ these revealed edges are sampled from the projection $\pi_{F_0}^{\mathrm{ord}}$ of π^{ord} onto F_0. It is likely that the configuration $\mathcal{F}_0^{\mathrm{ord}}$ has at least $(1-\eta)|E|$ in-edges. If not, then the coupling terminates (unsuccessfully), generating $\mathcal{F}^{\mathrm{ord}}$ and $\mathcal{F}^+$ from the right distributions. We will show that the probability of this unsuccessful termination is low. On the other hand, if $\mathcal{F}_0^{\mathrm{ord}}$ has a least $(1-\eta)|E|$ in-edges, then we are off to a good start. All configurations refining $\mathcal{F}_0^{\mathrm{ord}}$ are in Ω^{ord}, so the projection of π and π^{ord} onto subsequent edges that get revealed are the same (making it easier to continue the coupling). At this point $\mathcal{F}_0^+$ is taken to be the configuration with revealed edges F_0 where all revealed edges are in-edges.
- After iteration $i = 0$, iterations continue with $i = 1, 2, \ldots$ until an edge is revealed whose distance from v is at most r_2 or until the in-edges in $\mathcal{F}_i^{\mathrm{ord}}$ induce a giant component, and this giant component contains all vertices on the boundary of F_i. We will show that it is very unlikely that an edge at distance at most r_2 from v is reached. So it is likely the giant component in $\mathcal{F}_i^{\mathrm{ord}}$ contains all vertices on the boundary of F_i. This is a good situation because the conditional distribution of π, conditioned on refining $\mathcal{F}_i^{\mathrm{ord}}$ and the conditional distribution of $\mathcal{F}^+$, conditioned on refining $\mathcal{F}_i^+$ induce the same distribution on edges incident to v, which enables us to show that $\Pr(\mathcal{F}^+(e) \neq \mathcal{F}^{\mathrm{ord}}(e))$ is sufficiently small.
- The process at iteration $i+1$ is as follows. W_i is taken to be the set of all vertices on the boundary of F_i whose components (induced by the in-edges in $\mathcal{F}_i^{\mathrm{ord}}$) are all small. By "boundary" we mean that vertices in W_i are adjacent to revealed edges, and to unrevealed edges. If W_i is empty, then the coupling finishes. Otherwise, a vertex $w_i \in W_i$ is chosen to be as far from v as possible. The edges in the subtree of T_0 below the parent of w_i are revealed in F_{i+1}.

The main remaining ingredient in the proof is showing that the unsuccessful terminations of the coupling are unlikely. To do this, we use the polymer framework of [22]. (Ordered) polymers are defined using an inductive definition. For a set of edges $A \subseteq E$, let $\mathcal{B}_0(A) = A$, and inductively for $j = 0, 1, 2, \ldots$ define $\mathcal{B}_{j+1}(A)$ to be the set of all edges such that they are either in $\mathcal{B}_j(A)$ or edges that are incident to a vertex that has at least $5\Delta/9$ incident edges in $\mathcal{B}_j(A)$. Let $\mathcal{B}_\infty(A) = \bigcup_{j\in\mathbb{N}} \mathcal{B}_j(A)$. An ordered polymer of a configuration $\mathcal{F}$ is a connected component of $B_\infty(\mathrm{Out}(\mathcal{F}))$. The bulk of the work is to prove the following lemma, which is repeated in the appendix of the full version [14] (with more detail) as Lemma 44.

▶ **Lemma 19.** *Fix $\Delta \geq 5$ and $K, M > 0$. Suppose that $\beta \geq 3M$. Suppose that n is sufficiently large so that $r := \frac{M}{\beta}\log_{\Delta-1} n > K$ and $|B_r(v)| \leq 9\Delta n/200$. Define r_1 as above. Let $\mathcal{F}^{\mathrm{ord}}$ and $\mathcal{F}^+$ be generated by the process. Then at least one of the following conditions holds.*

1. *$\mathcal{F}^{\mathrm{ord}}$ and $\mathcal{F}^+$ agree on the edges that are incident to v.*
2. *$|\mathrm{In}(\mathcal{F}^{\mathrm{ord}}) \setminus E(B_{r_1}(v))| < (1-\eta)|E|$.*
3. *$\mathcal{F}^{\mathrm{ord}}$ contains a polymer of size at least $\frac{r}{400\Delta(1+K)} - 1$.*

To complete the proof of Theorem 5, we show that items 2 and 3 are unlikely. The proof that item 2 is unlikely, Lemma 45 in the full version [14], follows from the slack specified in Equation (2) of Lemma 3. The proof that item 3 is unlikely, Lemma 29 in the full version [14], follows from an analysis on the size of polymers by adapting appropriately the cluster expansion techniques of [22] (a bit of extra work is needed there to capture the $1/\beta$ dependence in the size of the polymer, see [14, Lemma 29] for details).

The proof of WSM for the disordered phase (Theorem 6) follows a similar strategy, the details are given in Appendix D of the full version [14].

References

1 Ferenc Bencs, Márton Borbényi, and Péter Csikvári. Random cluster model on regular graphs. *Communications in Mathematical Physics*, pages 1–46, 2022.

2 Antonio Blanca, Zongchen Chen, Daniel Štefankovič, and Eric Vigoda. The Swendsen-Wang dynamics on trees. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques* (APPROX/RANDOM 2021), volume 207, pages 43:1–43:15, 2021.

3 Antonio Blanca, Andreas Galanis, Leslie Ann Goldberg, Daniel Štefankovič, Eric Vigoda, and Kuan Yang. Sampling in uniqueness from the Potts and random-cluster models on random regular graphs. *SIAM Journal on Discrete Mathematics*, 34(1):742–793, 2020.

4 Antonio Blanca and Reza Gheissari. Random-cluster dynamics on random regular graphs in tree uniqueness. *Communications in Mathematical Physics*, 386(2):1243–1287, 2021.

5 Antonio Blanca and Reza Gheissari. On the tractability of sampling from the Potts model at low temperatures via Swendsen-Wang dynamics. *CoRR*, abs/2304.03182, 2023.

6 Christian Borgs, Jennifer Chayes, Tyler Helmuth, Will Perkins, and Prasad Tetali. Efficient sampling and counting algorithms for the Potts model on $\mathbb{Z}^d$ at all temperatures. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing,* (STOC '20) , pages 738–751, 2020.

7 Charlie Carlson, Ewan Davies, Nicolas Fraiman, Alexandra Kolla, Aditya Potukuchi, and Corrine Yap. Algorithms for the ferromagnetic Potts model on expanders. In *63rd Annual Symposium on Foundations of Computer Science* (FOCS 2022), pages 344–355, 2022.

8 Zongchen Chen, Andreas Galanis, Leslie A. Goldberg, Will Perkins, James Stewart, and Eric Vigoda. Fast algorithms at low temperatures via markov chains†. *Random Structures & Algorithms*, 58(2):294–321, 2021.

9 Zongchen Chen, Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Sampling colorings and independent sets of random regular bipartite graphs in the non-uniqueness region. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms,* (SODA '22), pages 2198–2207, 2022.

10 Amin Coja-Oghlan, Andreas Galanis, Leslie Ann Goldberg, Jean Bernoulli Ravelomanana, Daniel Štefankovič, and Eric Vigoda. Metastability of the Potts ferromagnet on random regular graphs. *Communications in Mathematical Physics*, pages 1–41, 2023.

11 Matthew Coulson, Ewan Davies, Alexandra Kolla, Viresh Patel, and Guus Regts. Statistical physics approaches to unique games. In *35th Computational Complexity Conference,* (CCC 2020), volume 169 of *LIPIcs*, pages 13:1–13:27, 2020.

12 Charilaos Efthymiou. On sampling symmetric Gibbs distributions on sparse random graphs and hypergraphs. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, pages 57:1–57:16, 2022.

13 Weiming Feng, Heng Guo, and Jiaheng Wang. Sampling from the ferromagnetic Ising model with external fields. *CoRR*, abs/2205.01985, 2022. arXiv:2205.01985.

14 Andreas Galanis, Leslie Ann Goldberg, and Paulina Smolarova. Sampling from the random cluster model on random regular graphs at all temperatures via Glauber dynamics, 2023. arXiv:2305.13239.

15 Andreas Galanis, Daniel Štefankovic, Eric Vigoda, and Linji Yang. Ferromagnetic Potts model: Refined #bis-hardness and related results. *SIAM Journal on Computing*, 45(6):2004–2065, 2016.

16 Reza Gheissari and Alistair Sinclair. Low-temperature Ising dynamics with random initializations. In *54th Annual ACM SIGACT Symposium on Theory of Computing,* (STOC '22), pages 1445–1458, 2022.

17 Reza Gheissari and Alistair Sinclair. Spatial mixing and the random-cluster dynamics on lattices. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms,* (SODA '23) , pages 4606–4621, 2023.

18 Leslie Ann Goldberg and Mark Jerrum. Approximating the partition function of the ferromagnetic Potts model. *Journal of the ACM*, 59(5):1–31, 2012.

19 Geoffrey Grimmett. *The random-cluster model*, volume 333. Springer, 2006.

20 Heng Guo and Mark Jerrum. Random cluster dynamics for the Ising model is rapidly mixing. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms,* (SODA 2017), pages 1818–1827, 2017.

21 Olle Häggström. The random-cluster model on a homogeneous tree. *Probability Theory and Related Fields*, 104:231–253, 1996.

22 Tyler Helmuth, Matthew Jenssen, and Will Perkins. Finite-size scaling, phase coexistence, and algorithms for the random cluster model on random graphs. *Annales de l'Institut Henri Poincare (B) Probabilites et statistiques*, 59(2):817–848, 2023.

23 Tyler Helmuth, Will Perkins, and Guus Regts. Algorithmic Pirogov-Sinai theory. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, ,* (STOC '19) , pages 1009–1020, 2019.

24 Johan Jonasson. The random cluster model on a general graph and a phase transition characterization of nonamenability. *Stochastic Processes and their Applications*, 79(2):335–354, 1999.

25 Yuval Peres and Peter Winkler. Can extra updates delay mixing? *Communications in Mathematical Physics*, 323:1007–1016, 2013.

26 Luca Trevisan. Lecture notes on graph partitioning, expanders and spectral methods, 2016. URL: `https://lucatrevisan.github.io/books/expanders-2016.pdf`.

Range Avoidance for Constant Depth Circuits: Hardness and Algorithms

Karthik Gajulapalli ✉ 🏠
Georgetown University, Washington, DC, USA

Alexander Golovnev ✉ 🏠
Georgetown University, Washington, DC, USA

Satyajeet Nagargoje ✉ 🏠
Georgetown University, Washington, DC, USA

Sidhant Saraogi ✉ 🏠
Georgetown University, Washington, DC, USA

Abstract

Range Avoidance (Avoid) is a total search problem where, given a Boolean circuit $\mathsf{C}\colon \{0,1\}^n \to \{0,1\}^m$, $m > n$, the task is to find a $y \in \{0,1\}^m$ outside the range of C. For an integer $k \geq 2$, NC^0_k-Avoid is a special case of Avoid where each output bit of C depends on at most k input bits. While there is a very natural randomized algorithm for Avoid, a deterministic algorithm for the problem would have many interesting consequences. Ren, Santhanam, and Wang (FOCS 2022) and Guruswami, Lyu, and Wang (RANDOM 2022) proved that explicit constructions of functions of high formula complexity, rigid matrices, and optimal linear codes, reduce to NC^0_4-Avoid, thus establishing conditional hardness of the NC^0_4-Avoid problem. On the other hand, NC^0_2-Avoid admits polynomial-time algorithms, leaving the question about the complexity of NC^0_3-Avoid open.

We give the first reduction of an explicit construction question to NC^0_3-Avoid. Specifically, we prove that a polynomial-time algorithm (with an NP oracle) for NC^0_3-Avoid for the case of $m = n + n^{2/3}$ would imply an explicit construction of a rigid matrix, and, thus, a super-linear lower bound on the size of log-depth circuits.

We also give deterministic polynomial-time algorithms for all NC^0_k-Avoid problems for $m \geq n^{k-1}/\log(n)$. Prior work required an NP oracle, and required larger stretch, $m \geq n^{k-1}$.

2012 ACM Subject Classification Theory of computation → Circuit complexity

Keywords and phrases Boolean function analysis, Explicit Constructions, Low-depth Circuits, Range Avoidance, Matrix Rigidity, Circuit Lower Bounds

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.65

Category RANDOM

Related Version *arXiv Version*: `https://arxiv.org/abs/2303.05044`

Acknowledgements We would like to thank Justin Thaler, Sam King, and anonymous reviewers for their helpful comments on our paper.

1 Introduction

The Range Avoidance (Avoid) problem is: given a Boolean circuit $\mathsf{C}\colon \{0,1\}^n \to \{0,1\}^m$ for some *stretch* $m > n$, find an element $y \in \{0,1\}^m$ outside the range of C. By the pigeonhole principle, such a y always exists. This problem was first introduced by Kleinberg, Korten, Mitropolsky, and Papadimitriou [15] as a complete problem for the class APEPP (Abundant Polynomial Empty Pigeonhole Principle). Informally, APEPP contains total search problems where the existence of a solution follows via the union bound (such as Shannon's classical proof that most functions require circuits of exponential size).

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 65; pp. 65:1–65:18
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Korten [16] proved that a deterministic algorithm for Avoid would imply explicit constructions of objects that are central to the field of computational complexity, and would resolve several long-standing open problems. Such objects include functions of high circuit complexity, rigid matrices, pseudorandom generators, and Ramsey graphs. The key idea is that there is a succinct way to encode all "easy" objects (such as descriptions of functions of low circuit complexity) in the input space of a small circuit that acts as a decoder. Then a solution to the Avoid problem yields a "hard" object (such as a function of high circuit complexity), implying an explicit construction. In fact, the aforementioned works [15, 16] showed that even a deterministic algorithm *with an* NP *oracle* solving Avoid in polynomial time would lead to breakthrough results in complexity theory.

The only known deterministic algorithm for Avoid is the trivial brute force algorithm running in time $2^n \cdot \mathrm{poly}(n, |\mathsf{C}|)$.[1] No better algorithms are known for Avoid even if the algorithm is allowed to use an NP oracle. On the other hand, using both an NP oracle and randomness, one can solve Avoid in polynomial time: Pick a random string $y \in \{0,1\}^m$, and simply check if $y \in \mathrm{Range}(\mathsf{C})$ using the NP oracle. This shows that $\textsc{Avoid} \in \mathsf{FZPP}^{\mathsf{NP}}$, and, using the standard trick of simulating randomness by non-uniformity, $\textsc{Avoid} \in \mathsf{FP}^{\mathsf{NP}}/\mathsf{Poly}$.[2] Interestingly, for the class of polynomial-time algorithms with an NP oracle, the Avoid problem is equally hard for all values of stretch $n + 1 \leq m \leq \mathrm{poly}(n)$ [15].

For a class of circuits $\mathcal{C}$, the $\mathcal{C}$-Avoid problem is a special case of Avoid where each output is computed by a circuit from $\mathcal{C}$. Recent works by Ren, Santhanam, and Wang [19] and Guruswami, Lyu, and Wang [9] proved that efficient algorithms for $\mathcal{C}$-Avoid even for certain simple circuit classes $\mathcal{C}$ would be sufficient for getting various explicit constructions. Later, Chen, Huang, Li, and Ren [6] re-derived the best known lower bounds against ACC^0 circuits from an efficient algorithm for a certain $\mathcal{C}$-Avoid problem. While this suggests that designing efficient algorithms for Avoid problems is a promising approach to various explicit construction questions, the work of Ilango, Li, and Williams [10] proves barriers for designing polynomial time algorithms under certain cryptographic assumptions.

Let NC^1 denote the class of Boolean fan-in-2 circuits of depth $O(\log(n))$, and NC^0_k denote the class of Boolean functions where each output depends on at most k inputs for a constant k. [19] used perfect encodings of [13, 14, 2] to reduce NC^1-Avoid to NC^0_4-Avoid in polynomial time. Consequently, [9] reduced most of the aforementioned explicit constructions in [16] (and several new ones!) to NC^1-Avoid, and, thus, to NC^0_4-Avoid. In particular, polynomial-time *deterministic* algorithms (even with an NP oracle) for NC^0_4-Avoid would now imply breakthrough results in complexity theory.

[9] gave a polynomial-time algorithm solving NC^0_2-Avoid for any stretch $m \geq n + 1$. As mentioned above, NC^0_4-Avoid might be hard to solve efficiently. This leaves the question about the complexity of NC^0_3-Avoid open.

▶ **Open Problem 1** ([9]). *Can we reduce explicit construction problems to solving* NC^0_3*-Avoid? Or can we solve* NC^0_3*-Avoid in polynomial time?*

Unlike the case of the general Avoid problem, NC^0_k-Avoid may be much easier for large

[1] This trivial algorithm is (conditionally) tight for a related problem studied in [15], where the range of C has size much smaller than 2^{n+1}, and is given by a circuit computing a function from $[N]$ to $[M]$. [15] gives a deterministic reduction from SAT on n variables to Avoid for a circuit $\mathsf{C}\colon [2^n] \to [2^n + 2^{o(n)}]$ running in subexponential time. Thus, under the Exponential Time Hypothesis [12, 11], this problem does not admit deterministic (and randomized) algorithms running in time $2^{o(n)}$.

[2] Here, the complexity classes FP, FE, FZPP are simply the functional analogs of the decision classes P, E, ZPP.

values of the stretch m. Indeed, on one hand, NC^0_k-Avoid for small stretch $m = n + o(n)$ is capable of encoding hard explicit construction problems [19, 9]. On the other hand, NC^0_k-Avoid for $m = \Omega(n^k)$ is easily solvable in polynomial time: since the number of distinct functions depending on at most k out of n inputs is $O(n^k)$, every such instance of the problem must have two outputs computing identical functions. Assigning different values to these outputs solves NC^0_k-Avoid.

[9] presented an algorithm solving NC^0_k-Avoid for stretch $m \geq \Omega(n^{k-1})$ in polynomial time with an NP oracle.[3] This improvement on the trivial algorithm suggests a natural question of whether one can solve NC^0_k-Avoid for even smaller values of stretch m.

▶ **Open Problem 2.** *Design a polynomial-time algorithm (with an* NP *oracle) solving* NC^0_k*-Avoid with* n *inputs and stretch* $m = o(n^{k-1})$ *for* $k \geq 3$.

1.1 Our Results

The classical result of Shannon [20] shows that most Boolean functions of n variables require Boolean circuits of exponential size. Despite that, the best known lower bound on the size of a circuit (or even a circuit of logarithmic depth, i.e., NC^1) for a function in P (or even E^{NP}) is $3.1n - o(n)$ proven by Li and Yang [17]. A central problem in circuit complexity is to prove a super-linear lower bound on the number of gates of NC^1 circuits computing an explicit function [22, 3, Frontier 3].

Similarly, for the class of *linear* NC^1 circuits – NC^1 circuits where each gate computes the XOR (or its negation) of its two inputs – no super-linear lower bound on the complexity of an explicit linear map $M \in \mathbb{F}_2^{n\times n}$ is known. The best lower bound against linear circuits is $3n - o(n)$ proven by Chashkin [5].

In our first result (Theorem 9), we answer Open Problem 1 by showing that a polynomial-time algorithm for NC^0_3-Avoid would imply an explicit construction of a map requiring linear NC^1 circuits of super-linear size (thus, demonstrating the hardness of NC^0_3-Avoid).

▶ **Theorem 1.** *An* FP *(resp.* $\mathsf{FP}^{\mathsf{NP}}$*) algorithm for* NC^0_3*-Avoid with stretch* $m = n + O(n^{2/3})$ *implies an explicit construction of a linear map in* FP *(resp.* $\mathsf{FP}^{\mathsf{NP}}$*) that cannot be computed by linear* NC^1 *circuits of size* $o(n \log\log(n))$.

Our proof of Theorem 1 first reduces an explicit construction of a rigid matrix to NC^0_3-Avoid (Theorem 9). A matrix $M \in \mathbb{F}_2^{n\times n}$ is called (r, s)-rigid if it cannot be written as a sum $M = L + S$ of a rank-r matrix L and a matrix S with at most s non-zeros per row. In a seminal work, Valiant [22] introduced an approach for proving super-linear lower bounds on the size of linear NC^1 circuits via matrix rigidity. Valiant proved that an $(\varepsilon n, n^\varepsilon)$-rigid matrix $M \in \mathbb{F}_2^{n\times n}$ for any constant $\varepsilon > 0$ requires linear NC^1 circuits of size $\Omega(n \log\log(n))$. Theorem 1 now follows straightforwardly as a corollary of Theorem 9.

The best known constructions of rigid matrices do not yet achieve the parameters sufficient for Valiant's circuit lower bound. [7, 18, 21] construct an $\left(r, \Omega\left(\frac{n}{r}\log(\frac{n}{r})\right)\right)$-rigid matrix in polynomial time, [8] gives an $\left(r, \Omega\left(\frac{n^2}{r^2\log(n)}\right)\right)$-rigid matrix in time $2^{O(n)}$ for $r \geq \sqrt{n}$, and [1, 4] give $(2^{\varepsilon\log(n)/\log\log(n)}, \Omega(n))$-rigid matrices in polynomial time with an NP oracle. However, even an $\mathsf{FP}^{\mathsf{NP}}$ algorithm for NC^0_3-Avoid with stretch $m = n + n^{12/17-\varepsilon}$ for any constant $\varepsilon > 0$ would already improve on these known constructions of rigid matrices.

[3] The algorithm of [9] does not use the full power of $\mathsf{FP}^{\mathsf{NP}}$: it outputs a hitting set $H \subseteq \{0,1\}^m$ such that for every NC^0_k function C, at least one point $y \in H$ is outside the range of C. Only then the algorithm looks at the input function and finds a solution $y \in H$ using the NP oracle.

In fact, we reduce the problem of constructing explicit rigid matrices to a problem that we call degree-2-Avoid, where each output computes a degree-2 polynomial of the inputs. Following Ren, Santhanam, and Wang's approach [19], this problem can be reduced to NC^0_3-Avoid using the perfect encoding scheme of Applebaum, Ishai, and Kushilevitz [2].[4]

On the algorithmic side, we make partial progress towards resolving Open Problem 2. We first give a simple deterministic polynomial-time algorithm for NC^0_3-Avoid for stretch $m \geq \binom{n}{2}/3 + 2n$ (presented in Appendix A). This algorithm already improves on the best known algorithm for NC^0_3, as it does not use an NP oracle. Then, in Theorem 4 we extend this algorithm to solve NC^0_k-Avoid for all constant k. Recall that the current best algorithms for this problem solve the case where $m \geq \Omega(n^{k-1})$ in polynomial time using an NP oracle [9]. We improve this result in two directions: our algorithm does not use an NP oracle, and it works in polynomial time for stretch $m \geq n^{k-1}/\log(n)$.

▶ **Theorem 2.** *There is a deterministic polynomial-time algorithm that solves the NC^0_k-Avoid problem with n inputs and stretch m for every $k \geq 3$ and $m \geq n^{k-1}/\log(n)$.*

1.2 Proof Overview

1.2.1 Hardness of NC^0_3-Avoid

Valiant [22] proved that linear NC^1 circuits with a linear number of gates can only compute non-rigid linear maps $M \in \mathbb{F}_2^{n\times n}$, i.e., maps M that can be written as a sum $M = Q + S$, where $\operatorname{rank}(Q) \leq \varepsilon n$ and each row of S has at most n^δ ones in it. For the rest of the section, our non-rigid matrices can be written as the sum of a matrix with rank $\leq n/10$ and a matrix with row sparsity at most $n^{0.1}$. Therefore, constructing a rigid matrix would imply a super-linear lower bound on the size of linear NC^1 circuits computing it.

To reduce an explicit construction of an $n \times n$ rigid matrix to solving an instance of NC^0_3-Avoid, we design an NC^0_3 function $f\colon \{0,1\}^{p(n)} \to \{0,1\}^{n^2}$, for some polynomial $p(n) < n^2$, such that for every non-rigid matrix $M \in \{0,1\}^{n\times n}$, there exists $x \in \{0,1\}^{p(n)}$ satisfying $f(x) = M$. Now, any solution $M' \in \{0,1\}^{n\times n}$ to the NC^0_3-Avoid problem for the function f must be a rigid matrix.

Before constructing such an NC^0_3 function f, we first design a function $g\colon \mathbb{F}_2^{n^2/2} \to \mathbb{F}_2^{n^2}$, where each output bit of g is a degree-2 polynomial of the inputs, and the range of g contains all non-rigid matrices. A solution to the degree-2-Avoid problem for the function g would give us a rigid matrix. Following [19], we can then apply a perfect encoding scheme [13, 14, 2] to g, and obtain an NC^0_3 function f, as required (see Lemma 6). Effectively, this reduces solving Avoid on g to solving Avoid on f.

Now we construct a degree-2 function $g\colon \mathbb{F}_2^{n^2/2} \to \mathbb{F}_2^{n^2}$ whose inputs encode all non-rigid matrices, i.e., for every non-rigid matrix M, there is an $x \in \{0,1\}^{n^2/2}$ such that $f(x) = M$. A non-rigid matrix M can be written as $M = LR + S$, where $L, R^T \in \mathbb{F}_2^{n\times n/10}$, and each row of S contains at most $n^{0.1}$ ones. The first $n^2/5$ inputs of the function g will correspond to the elements of L and R. Note that every entry of LR is a degree-2 function of the entries of L and R since it just computes the inner product of a row in L and a column in R. Now, for each $n^{0.1}$-sparse row of S, we show how to encode it using $n^{0.6}$ inputs and a degree 2 function. Repeating this procedure for each row of S will finish the proof.

[4] Ren, Santhanam, and Wang use the following definition of perfect encodings: A function $\widehat{f}$ is a perfect encoding of a function f if there exists a polynomial time algorithm Dec such that for all x, y: $\mathrm{Dec}(y) = f(x) \iff \exists r, y = \widehat{f}(x, r)$.

We interpret the $n^{0.1}$-sparse row with n entries as a $\sqrt{n} \times \sqrt{n}$ matrix A. Since A has at most $n^{0.1}$ non-zero entries, $\operatorname{rank}(A) \leq n^{0.1}$. It can be written as a product $A = BC$ where $B, C^T \in \mathbb{F}_2^{\sqrt{n} \times n^{0.1}}$. Therefore, there is a degree-2 function h that takes as input B, C of size $2n^{0.6}$ and outputs the sparse matrix A.

The presented encoding of s-sparse vectors in $\mathbb{F}_2^n$ is only non-trivial for $s < \sqrt{n}$ (as otherwise the number of inputs of h exceeds the number of outputs). As a result, we cannot encode the entire matrix S as an $n^{1.1}$-sparse vector in $\mathbb{F}_2^{n^2}$. However, for Valiant's approach of proving circuit lower bounds, we can assume that S is $n^{0.1}$-*row* sparse.[5] Thus, we can separately encode each row of S using only $O(n^{0.6})$ inputs to obtain an encoding of S in $O(n^{1.6})$ bits. In Lemma 8, we will demonstrate how to accommodate slightly better sparsity parameter when we are allowed higher (but still constant) degree d for the encoding function.

1.2.2 Simple algorithm for NC_3^0-Avoid

We start with a short description of a simple deterministic polynomial-time algorithm for NC_3^0-Avoid for stretch $m \geq \binom{n}{2}/3 + 2n$ (presented in Appendix A). This algorithm already improves on the best known algorithm for NC_3^0-Avoid, as our algorithm does not use an NP oracle.

If we had a #SAT oracle, then we could solve NC_3^0-Avoid even for stretch $m = n + 1$. Our algorithm would iteratively find constant assignments to each of the first n outputs to minimize the number of inputs that map to the current (partial) output assignments. Throughout our exposition, we say such inputs are *consistent with the (partial) output assignments*. Before we describe our algorithm, it is important to note that we always fix circuit outputs to constant assignments. At each iteration, we would use the #SAT oracle to find the output assignment that reduces the size of the input set by at least half. After fixing the first n outputs, we still have at least $m - n \geq 1$ unassigned outputs, and only one input point $x \in \mathbb{F}_2^n$ that is consistent with the previously assigned output bits. This allows us to find an assignment of the $(n+1)$-th output bit such that the string specified by the output bits lies outside the range of the circuit.

Unfortunately, solving #SAT (even approximately) is hard for this class of multi-output circuits. In the absence of an efficient #SAT algorithm, our algorithm maintains an affine subspace $\mathcal{S}$ that contains all inputs from $\mathbb{F}_2^n$ that are consistent with the current partial assignment ($\mathcal{S}$ may also contain inputs that are not consistent with the current partial assignment). We carefully set output values so that at each iteration, we reduce the dimension of $\mathcal{S}$ by at least one. This way, after $n + 1$ steps we will find a solution to the NC_3^0-Avoid problem. However, our algorithm can only work when the stretch is $m \geq \Omega(n^2)$.

Without loss of generality, we assume that each output reads exactly three input bits. At each iteration the number of currently unassigned outputs is $> \binom{n}{2}/3$. This allows us to find a pair of outputs y_1 and y_2 that share a pair of input variables.[6] Say, $y_1 = f_1(x_1, x_2, x_3)$ and $y_2 = f_2(x_2, x_3, x_4)$. We will find a constant assignment to y_1 and y_2 that reduces the dimension of the affine subspace $\mathcal{S}$.

Note that there are 16 assignments to (x_1, x_2, x_3, x_4) and four different values of (y_1, y_2). Therefore, there is a way to assign $y_1 = c_1, y_2 = c_2$ such that at most four points (x_1, x_2, x_3, x_4) map to these values of the outputs. Note that there always exists a hyperplane H containing

[5] Alternatively, one can argue by Markov's inequality that if a matrix M cannot be written as a sum of rank-r and $n^{0.1}$-*row* sparse matrices, then M also cannot be written as a sum of rank-$2r$ and $rn^{0.1}$-*globally* sparse matrices.

[6] Each output sees 3 pairs of input bits, giving a total of $3m > \binom{n}{2}$ pairs. By the pigeonhole principle, at least one pair of inputs appears in two outputs.

any four points in $\mathbb{F}_2^4$. Let $\mathcal{H}$ be the affine subspace obtained by extending H to all n inputs. Then, $\mathcal{S} \cap \mathcal{H}$ gives us an affine subspace containing all inputs consistent with the assignment $y_1 = c_1, y_2 = c_2$.

If $\mathcal{S} \nsubseteq \mathcal{H}$, then we have an assignment of two outputs that reduces the dimension of our affine subspace as $\dim(\mathcal{S} \cap \mathcal{H}) < \dim(\mathcal{S})$. Otherwise, if $\mathcal{S} \subseteq \mathcal{H}$, then instead of considering all 16 assignments to the inputs (x_1, x_2, x_3, x_4), we can restrict our attention to at most 8 such assignments that belong to H, which makes the problem only easier (as we show in Theorem 16).

1.2.3 Better algorithm for NC_k^0-Avoid

The main bottleneck of this simple algorithm is that it maintains an affine subspace that must contain all consistent inputs. While affine subspaces are easy to work with, they are not expressive enough to accurately describe all inputs that are consistent with an arbitrary partial assignment. In order to improve the previous algorithm, we will maintain a more expressive structure than an affine subspace – a union of affine subspaces. Below we sketch our approach to solving NC_3^0-Avoid with stretch $m \geq \Omega(n^2/\log(n))$. Theorem 4 generalizes this to solving NC_k^0-Avoid problems with stretch $m \geq \Omega(n^{k-1}/\log(n))$ for all values of k.

Again, without loss of generality, we assume that each output depends on exactly three inputs. Consider a bipartite graph, where the left vertices correspond to n inputs, the right vertices correspond to m outputs, and an input-output pair (x_i, y_j) is connected by an edge if the output y_j depends on the input x_i. First we select $t = 3n^2/m$ highest-degree inputs $I = \{x_1, \ldots, x_t\}$. Their neighborhood must contain at least $3n$ distinct outputs $O = \{y_1, \ldots, y_{3n}\}$.[7] Let C be the sub-circuit defined on outputs from O and their corresponding inputs. Now, we will find a $y \in \mathbb{F}_2^{3n}$ outside $\mathrm{Range}(\mathsf{C})$.

First, consider all 2^t assignments to the inputs in $I = \{x_1, \ldots, x_t\}$, resulting in circuits $\mathsf{C}_1, \ldots, \mathsf{C}_{2^t}$. Since every output in O is connected to at least one input from I, fixing an assignment to the inputs I reduces each C_i to an NC_2^0 circuit. In a way, we have reduced NC_3^0-Avoid to an OR of 2^t instances of NC_2^0-Avoid: we need to find a $y \in \{0,1\}^{3n}$ outside the ranges of all the C_i's. For each circuit C_i, we will maintain an affine subspace $\mathcal{S}_i$ containing all inputs consistent with the current partial assignment of the outputs.

Our algorithm works by iteratively fixing the output bits from $\{y_1, \ldots, y_{3n}\}$ such that at each step the total number of points in the (disjoint) union of the affine subspaces $\mathcal{S}_i$ is reduced by a constant factor, eventually making all the subspaces empty. We observe (in Lemma 13) that for any affine subspace $\mathcal{S}_i$, one of the assignments $y_i = 0$ or $y_i = 1$ always reduces the dimension of $\mathcal{S}_i$ by one. Therefore, by picking the "best" assignment $y_i = c$ across all the subspaces $\mathcal{S}_i$, we can reduce the size of the union of such affine subspaces by a constant factor of $4/3$. Repeating this procedure for $\log_{4/3}(2^n) + 1 < 3n$ steps finishes the proof.

1.3 Open Problems

Our work motivates several natural questions about the complexity of NC_3^0-Avoid and degree-2-Avoid. We reduce explicit constructions of rigid matrices to solving degree-2-Avoid, and then NC_3^0-Avoid, with appropriate stretch.

[7] Since the number of outputs is m, and each output has degree 3, the number of edges in the graph is $3m$, and the average degree of an input is $3m/n$. The t highest-degree inputs then have total degree at least $3mt/n$, and must be connected to at least $mt/n = 3n$ distinct outputs.

▶ **Open Problem 3.** *Can other explicit construction questions be reduced to* NC_3^0*-*Avoid *or degree-*2*-*Avoid*?*

Particularly, we suspect that the construction of linear and list-decodable codes with optimal parameters [9] might be good candidates for these reductions.

Using the encoding of [2] in the reduction from degree-2-Avoid to NC_3^0-Avoid almost always decreases the required stretch to $m = n + o(n)$ (as highlighted in Section 3). It would also be interesting to find a more efficient encoding or reduction from degree-2-Avoid to NC_3^0-Avoid. This could potentially increase the stretch for NC_3^0-Avoid required to obtain explicit constructions thereby making the problem easier.

▶ **Open Problem 4.** *Can we construct a more efficient reduction from degree-*2*-*Avoid *to* NC_0^3*-*Avoid*?*

We believe degree-2-Avoid might be of independent interest since it allows for a larger stretch. For example, for improved constructions of rigid matrices, it suffices to solve degree-2-Avoid for super-linear stretch $m \geq n^{12/11-\varepsilon}$ for a constant $\varepsilon > 0$. In fact, degree-2-Avoid is easy to solve when the stretch is $m \geq n^2$. Note that there are at most $\binom{n}{2}$ unique degree-2 monomials on n variables. If $m \geq n^2$, then we can replace each unique monomial with a new variable. As a result, we will have m linear functions in $< m$ variables. We can solve Avoid on this linear function instance by a dimension reduction strategy similar to the one outlined in the previous section.

▶ **Open Problem 5.** *Are there algorithmic techniques to solve degree-*2*-*Avoid *that do not use a reduction to* NC_3^0*-*Avoid*?*

For the NC_3^0-Avoid problem, our algorithm runs in deterministic time $2^{O(n^2/m)}$ for any stretch $m \geq n+1$. In particular, this recovers the exponential-time brute force algorithm for the hardest case of $m = n+1$. It would be interesting to obtain matching conditional lower bounds for deterministic algorithms for NC_3^0-Avoid.

▶ **Open Problem 6.** *Is there a conditional lower bound of* $2^{\Omega(n^2/m)}$ *on the complexity of deterministic algorithms without an* NP *oracle for* NC_3^0*-*Avoid*?*

Finally, it is natural to ask if algorithms with NP oracles can solve NC_3^0-Avoid more efficiently.

▶ **Open Problem 7.** *Do there exist polynomial-time algorithms with* NP *oracles that solve* NC_3^0*-*Avoid *for stretch* $m = o(n^2/\log(n))$*?*

1.4 Structure

The rest of the paper is organized as follows. In Section 2, we give all necessary background material, including a reduction from degree-d-Avoid to NC_{d+1}^0-Avoid in Section 2.3. In Section 3, we reduce the problem of constructing explicit rigid matrices to NC_3^0-Avoid. In Section 4, we give deterministic algorithms solving NC_k^0-Avoid in polynomial time for stretch $m \geq n^{k-1}/\log(n)$. Finally, Appendix A contains an alternative deterministic polynomial-time algorithm for NC_3^0 for the case where stretch $m \geq \Omega(n^2)$.

2 Preliminaries

For every $a \in \mathbb{F}_2^n$ and subspace L of $\mathbb{F}_2^n$, we can define an affine subspace $\mathcal{A} \subseteq \mathbb{F}_2^n$ where $\mathcal{A} = \{a + v \mid v \in L\}$. The dimension of the affine subspace $\dim(\mathcal{A})$ is the same as the dimension of the linear subspace L that defines it. Equivalently, the set of points that lie on a specified set of hyperplanes over $\mathbb{F}_2^n$ also characterize an affine subspace of $\mathbb{F}_2^n$. The hyperplanes can be written as a system of linear equations $Ax = b$, and the dimension of the corresponding affine subspace $\mathcal{A}$ can be calculated as $\dim(\mathcal{A}) = n - \operatorname{rank}(A)$.

The circuits and algorithms in this paper generally work over the boolean hypercube $\{0,1\}^n$. We work with multi-output circuits $\mathsf{C} : \{0,1\}^n \to \{0,1\}^m$ where $m > n$ and m is called the stretch of the circuit. A partial output assignment, $y \in \{0,1,*\}^m$, is a fixing of a subset of the output bits of the circuit to constants. For an input $x \in \{0,1\}^n$ to the circuit, we say x *is consistent with a partial output assignment* $y \in \{0,1,*\}^m$, if $\mathsf{C}(x)$ agrees with y on the fixed bits. When specified, the input (resp. output) space of a circuit might instead be viewed as the vector space $\mathbb{F}_2^n$ (resp. $\mathbb{F}_2^m$) over the finite field $\mathbb{F}_2$.

The complexity classes $\mathsf{FP}, \mathsf{FP}^{\mathsf{NP}}, \mathsf{FE}$, and $\mathsf{FE}^{\mathsf{NP}}$ are classes of search problems analogous to the classes of decision problems $\mathsf{P}, \mathsf{P}^{\mathsf{NP}}, \mathsf{E}$, and E^{NP}. For example, the class FP contains all functions that can be computed by deterministic polynomial-time Turing machines.

2.1 Circuits and Matrix Rigidity

In this paper, we work with circuit classes NC^0_k and NC^1, which we define below.

▶ **Definition 1** (NC Circuits). *The circuit class* NC^i *contains multi-output Boolean circuits on* n *inputs of depth* $O(\log^i(n))$ *where each gate has fan-in* 2*. We are particularly concerned with the following classes of circuits:*

- *For every constant* $k \geq 1$, NC^0_k *is the class of circuits where each output depends on at most* k *inputs.*
- NC^1 *is the class of circuits of depth* $O(\log(n))$ *where all gates have fan-in* 2*.*
- *Linear* NC^1 *circuits are circuits of depth* $O(\log(n))$ *where every gate has fan-in* 2 *and computes an affine function, i.e., the XOR of its two inputs or its negation.*

It is a long-standing open problem in circuit complexity to prove super-linear lower bounds on the size of (linear) NC^1 circuits computing an n-output function from FP or even $\mathsf{FE}^{\mathsf{NP}}$ [22, 3, Frontier 3]. Valiant [22] suggested an approach for proving super-linear lower bounds for linear NC^1 circuits using the notion of matrix rigidity.

▶ **Definition 2** (Matrix Rigidity). *For* $r, s \in \mathbb{Z}^+$*, a matrix* $M \in \mathbb{F}_2^{n\times n}$ *is* (r,s)-rigid *if* M *cannot be written as a sum*

$$M = L + S\,,$$

where $L, S \in \mathbb{F}_2^{n\times n}$*,* L *is low rank, i.e.,* $\operatorname{rank}(L) \leq r$*, and* S *is row sparse, i.e., every row of* S *has at most* s *non-zero entries.*

Valiant [22] proved that a linear operator given by a sufficiently rigid matrix requires linear NC^1 circuits of size at least $\Omega(n \log\log(n))$, but there are still no known constructions of such rigid matrices even in $\mathsf{FE}^{\mathsf{NP}}$.

▶ **Theorem 3** ([22]). *If a family of matrices* $(M_n)_{n\geq 1}$*,* $M_n \in \mathbb{F}^{n\times n}$*, is* $(\varepsilon n, n^{\delta})$*-rigid for constant* $\varepsilon, \delta > 0$*, then the linear map* $x \mapsto Mx$ *requires linear* NC^1 *circuits of size* $\Omega(n \log\log(n))$*.*

2.2 Range Avoidance for Circuits

In the range avoidance problem, given a circuit C with n inputs and m outputs, $m > n$, the goal is to find an m-bit string outside the range of C.

▶ **Definition 3** (Avoid). *In the* Avoid *problem, given a description of a circuit* $\mathsf{C} : \{0,1\}^n \to \{0,1\}^m$ *for* $m > n$*, the task is to find a* $y \in \{0,1\}^m$ *such that* $\forall x \in \{0,1\}^n \colon \mathsf{C}(x) \neq y$*.*

The function $m = m(n)$ is called the *stretch* of the multi-output circuit C. Note that Avoid is a total search problem, i.e., there always exists such a $y \in \{0,1\}^m$ since $m > n$. We focus on a more restricted problem where there is an additional promise that the input circuit C is from a fixed circuit class $\mathcal{C}$.

▶ **Definition 4** ($\mathcal{C}$-Avoid). *In the* $\mathcal{C}$-Avoid *problem, given a description of a circuit* $\mathsf{C} : \{0,1\}^n \to \{0,1\}^m$ *for* $m > n$*, where* $\mathsf{C} \in \mathcal{C}$*, the task is to find a* $y \in \{0,1\}^m$ *such that* $\forall x \in \{0,1\}^n \colon \mathsf{C}(x) \neq y$*.*

In particular, we are concerned with NC^1-Avoid and NC^0_k-Avoid for constant $k \geq 1$. We will also consider the class of functions where each output is a multivariate polynomial of the inputs of degree at most d over $\mathbb{F}_2$.

▶ **Definition 5** (degree-d-Avoid). *In the* degree-d-Avoid *problem, given a description of a function* $\mathsf{C} \colon \mathbb{F}_2^n \to \mathbb{F}_2^m$ *for* $m > n$*, where each output can be computed by a polynomial of degree* $\leq d$ *in the* n *inputs, the task is to find a* $y \in \mathbb{F}_2^m$ *such that* $\forall x \in \mathbb{F}_2^n \colon \mathsf{C}(x) \neq y$*.*

2.3 Low Degree and Low Locality

Perfect randomized encodings were introduced by [2] for various cryptographic applications. We are interested in the following property of perfect encodings: For a Boolean function $f : \{0,1\}^n \to \{0,1\}^m$ and its encoding $\widehat{f} \colon \{0,1\}^{n+\ell} \to \{0,1\}^{m+\ell}$, there exists a polynomial-time decoding algorithm, $\mathrm{Dec} \colon \{0,1\}^{m+\ell} \to \{0,1\}^m$, such that for all $y \in \{0,1\}^{m+\ell}$ and $x \in \{0,1\}^n$ satisfying $\mathrm{Dec}(y) = f(x)$, there exists $r \in \{0,1\}^\ell$ such that $y = \widehat{f}(x,r)$. This property can be used in Avoid reductions as follows. Given a solution to the Avoid problem for the function $\widehat{f}$, i.e., $y \notin \mathrm{Range}(\widehat{f})$, one can find a solution to the Avoid problem for the function f in polynomial time by simply computing $\mathrm{Dec}(y) \notin \mathrm{Range}(f)$.

[2] first encode NC^1 functions as degree-3 functions. Then, they encode every degree-d function as an NC^0_{d+1} function. Composing these two encodings provides an encoding of NC^1 functions in NC^0_4. Using this encoding, [19] provides a polynomial time reduction from NC^1-Avoid to NC^0_4-Avoid. We use only one part of the result from [2]: there is a polynomial time reduction from degree-d-Avoid to NC^0_{d+1}-Avoid. For completeness, we include the proof here.

▶ **Lemma 6.** *Let* $d \geq 2$ *be a constant, and* $f \colon \mathbb{F}_2^n \to \mathbb{F}_2^m$ *be a multi-output function where every output computes a sum of* k *monomials of degree* $\leq d$*. Then there exists a function* $\widehat{f} \colon \{0,1\}^{n+(2k-1)m} \to \{0,1\}^{2km}$ *computed by an* NC^0_{d+1} *circuit and a polynomial time algorithm* $\mathrm{Dec} \colon \{0,1\}^{2km} \to \{0,1\}^m$ *such that for all* x, y*, if* $\mathrm{Dec}(y) = f(x)$*, there exists an* $r \in \{0,1\}^{(2k-1)m}$ *such that* $\widehat{f}(x,r) = y$*.*

Proof. We follow the encoding constructed in [2]. First, we construct an encoding $\widehat{g}$ for each single output function g of f. Let $g(x) = T_1(x) + T_2(x) + \cdots + T_k(x)$ be a single output degree-d function where each $T_i(x)$ is a monomial of degree at most d. Consider the encoding of g, $\widehat{g} : \{0,1\}^n \times \{0,1\}^k \times \{0,1\}^{k-1} \to \{0,1\}^{2k}$ defined as follows

$$\begin{aligned}\widehat{g}(x,r,s) = (T_1(x) - r_1, \quad & T_2(x) - r_2, \quad & \ldots \quad & T_{k-1}(x) - r_{k-1}(x), \quad & T_k(x) - r_k, \\ r_1 - s_1, \quad & s_1 + r_2 - s_2, \quad & \ldots \quad & s_{k-2} + r_{k-1} - s_{k-1}, \quad & s_{k-1} + r_k)\,.\end{aligned}$$

Clearly, each output bit of $\widehat{g}$ can be computed by an NC^0_{d+1} circuit. We define a polynomial-time algorithm $\mathrm{Dec}_{\widehat{g}}\colon \{0,1\}^{2k} \to \{0,1\}$ such that if $\mathrm{Dec}_{\widehat{g}}(y) = g(x)$ then there exist r and s satisfying $\widehat{g}(x,r,s) = y$. Given $y \in \{0,1\}^{2k}$, $\mathrm{Dec}_{\widehat{g}}(y)$ sums up the bits of y modulo 2.

Suppose $\mathrm{Dec}_{\widehat{g}}(y) = g(x)$ for some $x \in \{0,1\}^n$, i.e.,

$$\mathrm{Dec}_{\widehat{g}}(y) = \sum_{j=1}^{2k} y_j = g(x) = \sum_{j=1}^{k} T_j(x)\,. \tag{1}$$

We will now show that there exist r and s such that $\widehat{g}(x,r,s) = y$. For each $j \in [k]$, we set $r_j = T_j(x) - y_j$. We also set $s_1 = r_1 - y_{k+1}$ and sequentially set $s_j = s_{j-1} + r_j - y_{k+j}$ for each $j \in \{2,\dots,k-1\}$. By definition, the first $2k-1$ bits of $\widehat{g}(x,r,s)$ equal the first $2k-1$ bits of y. For the last bit, note that:

$$\begin{aligned} s_{k-1} + r_k &= \sum_{i=1}^{k} r_i - \sum_{i=k+1}^{2k-1} y_i && \text{(by the definition of } s\text{)} \\ &= \sum_{i=1}^{k} T_i(x) - \sum_{i=1}^{2k-1} y_i && \text{(by the definition of } r\text{)} \\ &= y_{2k}\,. && \text{(by Equation (1))} \end{aligned}$$

Therefore, for the constructed r and s, $\widehat{g}(x,r,s) = y$, as required.

Suppose $f(x) = (f_1(x), f_2(x), \dots, f_m(x))$, where each $f_i(x)$ is a sum of at most k monomials of degree $\le d$. Let $\widehat{f_i}$ be the encoding of f_i as defined above. Then our encoding of f is simply a concatenation of the encodings of its individual outputs, $\widehat{f}\colon \{0,1\}^{n+(2k-1)m} \to \{0,1\}^{2km}$, where

$$\widehat{f}(x, r^{(1)}, r^{(2)}, \dots, r^{(m)}, s^{(1)}, s^{(2)}, \dots, s^{(m)}) = (\widehat{f_1}(x, r^{(1)}, s^{(1)}), \dots, \widehat{f_m}(x, r^{(m)}, s^{(m)}))\,. \tag{2}$$

On input $y = (y_1, y_2, \dots, y_m) \in \{0,1\}^{2km}$, the decoding algorithm returns

$$\mathrm{Dec}(y) = (\mathrm{Dec}_{\widehat{f_1}}(y_1), \dots, \mathrm{Dec}_{\widehat{f_m}}(y_m))\,. \tag{3}$$

Suppose $\mathrm{Dec}(y) = f(x)$ for some $x \in \{0,1\}^n$. Then $\mathrm{Dec}_{\widehat{f_i}}(y_i) = f_i(x)$ for all $i \in [m]$. By our proof above, there exists $r^{(i)}$ and $s^{(i)}$ such that $y_i = \widehat{f_i}(x, r^{(i)}, s^{(i)})$ and, thereby,

$$y = (y_1, \dots, y_m) = (\widehat{f_1}(x, r^{(1)}, s^{(1)}), \dots, \widehat{f_m}(x, r^{(m)}, s^{(m)})) = \widehat{f}(x, r^{(1)}, s^{(1)}, \dots, r^{(m)}, s^{(m)})\,.$$

Finally, Dec runs in time $O(mk)$ since it runs m iterations of $\mathrm{Dec}_{\widehat{f_i}}$ for each i, each of which simply computes a sum of $2k$ bits. Since each $\widehat{f_i}$ is in NC^0_{d+1}, so is $\widehat{f}$. ◀

Now, following [19], we conclude that there is a polynomial-time reduction from degree-d-Avoid to NC^0_{d+1}-Avoid.

▶ **Corollary 7.** *For every $d \ge 1$, if there exists an* FP *(resp.* $\mathsf{FP}^{\mathsf{NP}}$*) algorithm for* NC^0_{d+1}*-Avoid, then there exists an* FP *(resp.* $\mathsf{FP}^{\mathsf{NP}}$*) algorithm for degree-$d$-Avoid.*

Proof. Let f be an input to a degree-d-Avoid problem with m output bits. Then, each output bit of f is a sum of at most $k = O(n^d)$ monomials of degree d. Let $\widehat{f}$ be the encoding of f in NC^0_{d+1} guaranteed by Lemma 6. Note that $\widehat{f}\colon \{0,1\}^{n+(2k-1)m} \to \{0,1\}^{2km}$. By the assumption of the Corollary, there is an FP (resp. $\mathsf{FP}^{\mathsf{NP}}$) algorithm that returns a $y \notin \mathrm{Range}(\widehat{f})$. Then, by Lemma 6, $\mathrm{Dec}(y) \notin \mathrm{Range}(f)$ and Dec runs in polynomial time. Therefore, there is an FP (resp. $\mathsf{FP}^{\mathsf{NP}}$) algorithm for degree-$d$-Avoid. ◀

3 Hardness of NC^0_3-Avoid

In this section, we reduce the problem of constructing explicit rigid matrices to the algorithmic task of solving NC^0_3-Avoid. First, in Lemma 8 we give an explicit degree-2 function $f\colon \{0,1\}^k \to \{0,1\}^n$, $k \ll n$, whose range contains all sparse vectors of length n. Note that such a function f must have degree at least 2. Indeed, if f was affine and its range contained all vectors of sparsity at most 1, then its range must have dimension n, and the number of inputs of f would be $k \geq n$.

Next, in Theorem 9, we apply this lemma, together with the reduction from degree-2-Avoid to NC^0_3-Avoid from Lemma 6, to conclude that an efficient algorithm for NC^0_3-Avoid would provide an explicit construction of rigid matrices.

▶ **Lemma 8.** *For every $d \geq 1$ and every polynomial-time computable $s := s(n) < \frac{n^{1-1/d}}{d}$, there exists a polynomial-time computable function $f\colon \mathbb{F}_2^{dsn^{1/d}} \to \mathbb{F}_2^n$ whose range contains all vectors of sparsity at most s, and each output of f is a degree-d polynomial.*

Proof. Let G be an arbitrary d-uniform hypergraph on $\ell = dn^{1/d}$ vertices and n hyperedges (such a graph exists because $\binom{\ell}{d} \geq (\ell/d)^d = n$). Fix an ordering $\{1,\dots,\ell\}$ of the vertices and $\{1,\dots,n\}$ of the edges. Each vertex of G will be labeled by a vector from $\mathbb{F}_2^s$. Our function $f\colon \mathbb{F}_2^{s\ell} \to \mathbb{F}_2^n$ will take as input the labels of the vertices of G and output n elements corresponding to the n hyperedges of G: the ith output is the generalized inner product of the labels of the d vertices in the ith hyperedge. We interpret the input as a matrix $X \in \mathbb{F}_2^{s\times\ell}$, where the jth column $X_j \in \mathbb{F}_2^s$ is the label corresponding to the jth vertex. Suppose the hyperedge i contains the vertices $\{j_1,\dots,j_d\}$ then the ith output is $f_i(X) = \sum_{k=1}^{s} X_{k,j_1}\cdots X_{k,j_d}$. Clearly, f is a degree-d function, it only remains to show that its output contains all vectors of sparsity $\leq s$. For this, we show that for every vector $y \in \mathbb{F}_2^n$ of sparsity $\leq s$, there is an input, i.e., a labeling of the vertices of G, such that f outputs y. Let the s non-zero elements of y correspond to the distinct edges $i_1,\dots,i_s$ in G. For each vertex j in G we set its label $X_j \in \mathbb{F}_2^s$ to be such that $(X_j)_k = 1$ if $j \in i_k$ and $(X_j)_k = 0$ otherwise.

Consider any edge $i = \{j_1,\dots,j_d\}$ and the submatrix $X_{j_1,\dots,j_d}$ of X containing the labels of these vertices connected by i.

- If $i = i_k$ for some $k \in [s]$, then $y_i = 1$. The i_kth row of $X_{j_1,\dots,j_d}$ contains all 1 entries. Furthermore, every other row contains at least one zero. Therefore, $f_i(X) = 1$.
- If $i \neq i_k$ for all $k \in [s]$, then $y_i = 0$ and each row of $X_{j_1,\dots,j_d}$ contains at least one zero. Therefore, $f_i(X) = 0$. ◀

Equipped with Lemma 8, we are ready to show that an efficient algorithm for degree-2-Avoid or NC^0_3-Avoid would imply an explicit construction of rigid matrices.

▶ **Theorem 9.** *For every constant $1/2 \leq \delta \leq 1$, an FP (resp. $\mathsf{FP}^{\mathsf{NP}}$) algorithm for degree-2-Avoid with stretch $m = 2n^{2/(1+\delta)}$ will provide an FP (resp. $\mathsf{FP}^{\mathsf{NP}}$) algorithm for finding an $(n^\delta/10, n^{\delta-1/2}/10)$-rigid matrix.*

Furthermore, for every $1/2 \leq \delta \leq 1$, an FP (resp. $\mathsf{FP}^{\mathsf{NP}}$) algorithm for NC^0_3-Avoid with stretch $m = n + O(n^{2/(2+\delta)})$ will provide an FP (resp. $\mathsf{FP}^{\mathsf{NP}}$) algorithm for finding an $(n^\delta/10, n^{\delta-1/2}/10)$-rigid matrix.

Proof. Let $r = n^\delta/10$ and $s = n^{\delta-1/2}/10$. First, we reduce (in deterministic polynomial time) the problem of finding an (r,s)-rigid matrix to solving degree-2-Avoid for a function $g\colon \mathbb{F}_2^{4rn} \to \mathbb{F}_2^{n^2}$.

Suppose $M \in \mathbb{F}_2^{n\times n}$ is not (r,s)-rigid. Then, M can be written as a sum $M = Q + S$, where $\operatorname{rank}(Q) \leq r$ and S is s-row sparse. Furthermore, $Q = L \cdot R$ for some matrices $L, R^T \in \mathbb{F}_2^{n\times r}$.

We view the input of g as $2rn$ entries of the matrices L and R, and $2sn^{3/2}$ inputs of n copies of the degree-2 function f from Lemma 8 needed to encode the entries of n sparse rows of S. Then the function g simply outputs all n^2 entries of $M = L \cdot R + S$. Note that each output of g computes a dot-product of a row of L and a column of R, and adds a degree-2 output of f. Therefore, we constructed a degree-2 function g whose range contains all non-rigid matrices. A solution to degree-2-Avoid on input g would therefore give an (r,s)-rigid matrix. The number of inputs of g is $n' = 2rn + 2sn^{3/2} = 4rn = 2n^{1+\delta}/5$, and the stretch of the function g is at least $m'(n') \geq 2(n')^{2/(1+\delta)}$. This concludes the proof of the first part of the theorem.

For the second part, we use the polynomial-time reduction from degree-2-Avoid to NC_3^0-Avoid from Lemma 6. By the construction above, we have a degree-2 function $g\colon \mathbb{F}_2^{4rn} \to \mathbb{F}_2^{n^2}$ where each output bit is the sum of at most $t = r + s \leq 2r$ degree-2 monomials. We apply Lemma 6 to reduce Avoid for g to Avoid for an NC_3^0 function $\widehat{g}\colon \{0,1\}^{\widehat{n}} \to \{0,1\}^{\widehat{m}}$, where $\widehat{n} = n' + (2t-1)n^2$ and $\widehat{m} = 2tn^2$. This yields a stretch of $\widehat{m}(\widehat{n}) = \widehat{n} + O(\widehat{n}^{2/(2+\delta)})$ for the function $\widehat{g}$. Therefore, an algorithm for NC_3^0-Avoid for stretch $\widehat{m}(n)$ yields an (r,s)-rigid matrix. ◀

We remark that in the regime $\delta > 1/2$, Theorem 9 would give matrices that for rank n^δ have higher rigidity than all known constructions of rigid matrices in $\mathsf{FP}, \mathsf{FP}^{\mathsf{NP}}$ and $\mathsf{FE}^{\mathsf{NP}}$. Therefore, for every $\varepsilon > 0$, an $\mathsf{FP}^{\mathsf{NP}}$ algorithm for degree-2-Avoid with stretch $n^{12/11-\varepsilon}$ or an $\mathsf{FP}^{\mathsf{NP}}$ algorithm for NC_3^0-Avoid with stretch $n + n^{12/17+\varepsilon}$ would lead to new rigidity lower bounds. Since the regime of $\delta = 1$ in Theorem 9 is sufficient for Valiant's program of proving super-linear lower bounds on the size of linear NC^1 circuits (see Theorem 3), we have the following corollary.

▶ **Corollary 10.** *An* FP *(resp.* $\mathsf{FP}^{\mathsf{NP}}$*) algorithm for degree-2-Avoid with stretch $m = 2n$ or for* NC_3^0*-Avoid with stretch $m = n + O(n^{2/3})$ will provide a linear function in* FP *(resp.* $\mathsf{FP}^{\mathsf{NP}}$*) that cannot be computed by linear* NC^1 *circuits of size $o(n \log\log(n))$.*

4 Algorithms for NC_k^0-Avoid

In this section, we describe polynomial-time algorithms for solving NC_k^0-Avoid with non-trivial stretch. More specifically, we provide an algorithm that runs in time $2^{O(n^{k-1}/m)}\cdot\operatorname{poly}(n)$ when the stretch of the input circuit is at least $m \geq \Omega(n^{k-2})$. First, we describe a useful structural property of NC_k^0 circuits, which follows from the following simple graph-theoretic result.

▶ **Lemma 11.** *For any constants $c \geq 1$ and $k \geq 3$, every k-uniform hypergraph $G = (V,E)$ with n vertices and $m \geq cn^{k-2}$ hyperedges contains a subset of vertices $V' \subseteq V, |V'| \leq cn^{k-1}/m$ and a subset of hyperedges $E' \subseteq E, |E'| \geq cn$ such that each hyperedge in E' contains at least $k-2$ vertices from V'. Furthermore, there is a polynomial-time algorithm that finds such a V' and E'.*

Proof. Consider the following bipartite graph H with vertex set $A \sqcup B$ where $A = \{u_S \mid S \subseteq V, |S| = k-2\}$ is the set of vertices indexed by the $k-2$ sized subsets of V and $B = \{v_e \mid e \in E\}$ is indexed by the edges of G. Furthermore there is an edge (u_S, v_e) in H if $S \subseteq e$, i.e., if all the vertices in S are contained in the hyperedge $e \in E$. Since each hyperedge

e contains k vertices, the degree of each vertex in B is exactly $\binom{k}{k-2}$. Then, the average degree of the vertices in A is $\frac{|B|\binom{k}{k-2}}{|A|} = \frac{m\binom{k}{k-2}}{\binom{n}{k-2}}$. Let $A' \subseteq A$ be the subset of t vertices with highest degree in A and let $N(A')$ be their neighbors in B. Then total degree of vertices in A' is at least $\frac{tm\binom{k}{k-2}}{\binom{n}{k-2}}$. Since each vertex in B has degree $\binom{k}{k-2}$, $|N(A')| \geq \frac{tm\binom{k}{k-2}}{\binom{n}{k-2}\binom{k}{k-2}} = \frac{tm}{\binom{n}{k-2}}$. Therefore, setting $t = \frac{cn\binom{n}{k-2}}{m}$, V' to be the set of $t(k-2) = \frac{cn(k-2)\binom{n}{k-2}}{m} \leq cn^{k-1}/m$ vertices of G contained in the union of the vertex subsets in A', and $E' = N(A')$ completes our proof.

To find V' and E', we first construct the graph H which has polynomial size, and then find the vertices in A' by finding the t vertices in A with maximum degree. It is now straightforward to construct V' from A' and to find $E' = N(A')$. ◀

▶ **Corollary 12.** *For any constants $c \geq 1$ and $k \geq 3$, given an NC^0_k circuit C with n inputs and $m \geq cn^{k-2}$ outputs, there exists a subset of outputs O of size $|O| \geq cn$, and a subset of inputs I of size $|I| \leq cn^{k-1}/m$, such that for every output bit $\mathsf{C}_i \in O$, at least $k-2$ of the input bits feeding into C_i are from I. Furthermore, there is a polynomial-time algorithm that finds such sets I and O.*

Proof. Without loss of generality we assume that each output of C reads exactly k inputs (as if it reads $\ell < k$ inputs, we let it additionally read arbitrary $k - \ell$ inputs and ignore them). Consider the hypergraph where each vertex corresponds to one of the n inputs $\{x_1, \ldots, x_n\}$ of C. Each edge of the hypergraph corresponds to an output C_i, $e_i = \{j \mid \mathsf{C}_i \text{ reads } x_j\}$. Now, we apply Lemma 11 on this hypergraph and set $I = V'$ and $O = E'$. Note that $|I| = |V'| \leq cn^{k-1}/m$ and $|O| = |E'| \geq cn$. ◀

This corollary finds a linear number of output bits O of the circuit that mostly depend on a small number of common input bits I. Our algorithm for NC^0_k-Avoid will "branch" on all possible assignments to the inputs from I. Each such assignment will correspond to an affine subspace $\mathcal{S} \subseteq \mathbb{F}_2^n$ of the input space. Then, our algorithm works by fixing the output bits from O such that the sum of the dimensions of these affine subspaces is significantly reduced at each step, eventually making all subspaces empty. Note that by the guarantee of Corollary 12, after fixing the inputs I, each output from O depends on at most two inputs. Thus, we need an efficient way to reduce the dimension of the affine subspace containing the consistent inputs for the case where output functions depend on at most two inputs. In Lemma 13, we provide such a subroutine AffineReduce (Algorithm 1).

▶ **Lemma 13.** *Let $\mathcal{S} \subseteq \mathbb{F}_2^n$ be an affine subspace, and $f \colon \mathbb{F}_2^n \to \mathbb{F}_2$ be a function that depends on at most two inputs. The algorithm AffineReduce in deterministic polynomial time finds two affine subspaces (or empty sets) $\mathcal{S}_0, \mathcal{S}_1 \subseteq \mathcal{S}$ such that*

(1) *$\forall x \in \mathcal{S}, b \in \mathbb{F}_2$, if $f(x) = b$, then $x \in \mathcal{S}_b$;*

(2) *$|\mathcal{S}_0| + |\mathcal{S}_1| \leq 3|\mathcal{S}|/2$.*

Proof. Without loss of generality we assume that $f(x)$ depends on (a subset of) x_1 and x_2. We will consider three cases depending on the degree of f, and in each case we will find affine subspaces (or empty sets) $\mathcal{S}_0, \mathcal{S}_1 \subseteq \mathcal{S}$ such that at least one of them has dimension strictly smaller than the dimension of $\mathcal{S}$ (or at least one of them is an empty set). This will ensure that $|\mathcal{S}_0| + |\mathcal{S}_1| \leq 3|\mathcal{S}|/2$.

- If $f(x) = c$ for some $c \in \mathbb{F}_2$ is a constant function, then we set $\mathcal{S}_c = \mathcal{S}$ and $\mathcal{S}_{1-c} = \emptyset$. Clearly, $\mathcal{S}_c = \mathcal{S}$ and $\mathcal{S}_{1-c}$ contain all points $x \in \mathcal{S}$ that are consistent with $f(x) = c$ and $f(x) = 1 - c$, respectively.

- If $f(x) = a_1x_1 + a_2x_2 + c$ for some constants $a_1, a_2, c \in \mathbb{F}_2$ is an affine function, then for each $b \in \mathbb{F}_2$ let H_b be the hyperplane defined by $a_1x_1 + a_2x_2 + c = b$, and let $\mathcal{S}_b = \mathcal{S} \cap H_b$. Again, $\mathcal{S} \cap H_b$ contains all the inputs in $\mathcal{S}$ that are consistent with $f(x) = b$. Furthermore, if $\dim(\mathcal{S}_0) = \dim(\mathcal{S})$, then $\mathcal{S} \subseteq H_0$ and $\mathcal{S}_1 = \mathcal{S} \cap H_1 = \emptyset$. Therefore, either $\dim(\mathcal{S}_0) < \dim(\mathcal{S})$ or $\mathcal{S}_1 = \emptyset$.
- If $f(x) = (x_1 + a_1)(x_2 + a_2) + c$ for some constants $a_1, a_2, c \in \mathbb{F}_2$ is a quadratic function, then let $\mathcal{H}$ be the affine subspace defined by $\mathcal{H} = \{x \in \mathbb{F}_2^n \mid x_1 = 1 + a_1, x_2 = 1 + a_2\}$. Consider the affine subspace $\mathcal{S}_{1-c} = \mathcal{S} \cap \mathcal{H}$ which contains all points $x \in \mathcal{S}$ satisfying $f(x) = 1 - c$.
  - If $\dim(\mathcal{S}_{1-c}) < \dim(\mathcal{S})$, then we are done as we can take $\mathcal{S}_c = S$
  - If $\dim(\mathcal{S}_{1-c}) = \dim(\mathcal{S})$, then $\mathcal{S} \subseteq \mathcal{H}$. Then, every point in $\mathcal{S}$ satisfies $f(x) = 1 - c$, thus, setting $\mathcal{S}_{1-c} = \mathcal{S}$ and $\mathcal{S}_c = \emptyset$ completes our construction.

In each case, either $|\mathcal{S}_0| \leq \frac{|\mathcal{S}|}{2}$ or $|\mathcal{S}_1| \leq \frac{|\mathcal{S}|}{2}$. Therefore, $|\mathcal{S}_0| + |\mathcal{S}_1| \leq 3|\mathcal{S}|/2$.

The only computation made by AffineReduce is to compute the dimensions of explicitly given affine subspaces, which can be performed in polynomial time. ◀

Algorithm 1 AffineReduce($\mathcal{S}, f$).

```
Input: Affine subspace $\mathcal{S} \subseteq \mathbb{F}_2^n$, $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ that may depend only on $x_1$ and $x_2$
Output: $\mathcal{S}_0, \mathcal{S}_1 \subseteq \mathcal{S}$
  if $f(x) = c$ then
      return $\mathcal{S}_c = \mathcal{S}$ and $\mathcal{S}_{1-c} = \emptyset$
  if $f(x) = a_1x_1 + a_2x_2 + c$ then
      For $b \in \mathbb{F}_2$, let $H_b = \{x \in \mathbb{F}_2^n\colon a_1x_1 + a_2x_2 + c = b\}$
      return $\mathcal{S}_0 = \mathcal{S} \cap H_0$ and $\mathcal{S}_1 = \mathcal{S} \cap H_1$
  if $f(x) = (x_1 + a_1)(x_2 + a_2) + c$ then
      Let $\mathcal{H} = \{x \in \mathbb{F}_2^n\colon x_1 = 1 + a_1, x_2 = 1 + a_2\}$
      Let $\mathcal{S}_{1-c} = \mathcal{S} \cap \mathcal{H}$
      if $\dim(\mathcal{S}_{1-c}) < \dim(\mathcal{S})$ then
          return $\mathcal{S}_{1-c}$ and $\mathcal{S}_c = \mathcal{S}$
      else
          return $\mathcal{S}_{1-c}$ and $\mathcal{S}_c = \emptyset$
```

A simple application of AffineReduce recovers a polynomial-time algorithm for NC_2^0-Avoid from [9].

▶ **Corollary 14.** *There is a deterministic polynomial-time algorithm that, given an* NC_2^0 *circuit* $\mathsf{C}\colon \{0,1\}^n \to \{0,1\}^m$, $m \geq n+1$, *finds an element* $y \in \{0,1\}^m, y \notin \mathrm{Range}(\mathsf{C})$.

Proof. At iteration $1 \leq i \leq n+1$, our algorithm will fix the value of the ith output bit y_i. The algorithm also maintains an affine subspace $\mathcal{S} \subseteq \mathbb{F}_2^n$ that contains all inputs $x \in \mathbb{F}_2^n$ consistent with the partial output assignments of $y_1, \ldots, y_i$. By Lemma 13, there exists an assignment $y_i = b$, such that either none of the inputs in $\mathcal{S}_b$ are consistent with y or the dimension of $\mathcal{S} = \mathcal{S}_b$ reduces at least by one. In the former case, we already find our desired output y (we can just set the unassigned bits of y to arbitrary values). Otherwise, after fixing the first n outputs, we have $\dim(\mathcal{S}) = 0$, i.e., $S = \{x\}$ for some $x \in \mathbb{F}_2^n$. Let $b \in \{0,1\}$ be the value of the $(n+1)$th output bit of $\mathsf{C}(x)$. Then setting $y_{n+1} = 1 - b$ produces our desired output y. This algorithm runs in polynomial time since it makes at most n calls to AffineReduce and one call to $\mathsf{C}(x)$. ◀

Finally, equipped with Lemma 13, we are ready to present our main algorithm for NC^0_k-Avoid.

▶ **Theorem 4.** *Given an NC^0_k circuit $\mathsf{C}\colon \{0,1\}^n \to \{0,1\}^m$, where $m \geq 3n^{k-2}$, the algorithm* SubspaceUnion *finds an element $y \in \{0,1\}^m, y \notin \text{Range}(\mathsf{C})$ in deterministic time $2^{O(n^{k-1}/m)} \cdot \text{poly}(n)$.*

Proof. First we apply Corollary 12 with $c = 3$ to the circuit C, and select in polynomial time a subset of inputs $I = \{x_1, \dots, x_t\}$ and a set of outputs $O = \{y_1, y_2, ..., y_{3n}\}$ for $t \leq 3n^{k-1}/m$. This ensures that each y_i has at most two inputs outside of I. For each of the 2^t assignments of the inputs from I, we consider a circuit where the values of these t inputs are fixed. Namely, for $j \in \{0, \dots, 2^t - 1\}$, we fix the inputs in I to the bits in the binary representation of j. Then we restrict the circuit C to the outputs $y_1, \dots, y_{3n}$ and all the inputs that feed them, and obtain a circuit C_j, where each output depends on at most 2 inputs. We'll find a value $y \in \{0,1\}^{3n}$ that no C_j outputs, and this will give us a solution to the original NC^0_k-Avoid instance.

Our algorithm will maintain the following invariant. At the ith iteration of the algorithm after we fix the values of the outputs $y_1, \dots, y_i$, we maintain $\mathcal{U} = \bigcup_{j=0}^{2^t-1} \mathcal{U}_j$, a disjoint union of 2^t affine subspaces, such that all inputs $x \in \mathbb{F}_2^n$ that are consistent with $y_1, \dots, y_i$ belong to $\mathcal{U}$ (and $\mathcal{U}$ may contain points that are inconsistent with $y_1, \dots, y_i$, too).

In the beginning of the algorithm, for every $0 \leq j < 2^t$, we let $\mathcal{U}_j$ be the affine subspace where the inputs in I are fixed to the bits in the binary representation of j. Then $\mathcal{U} = \bigcup_j \mathcal{U}_j = \mathbb{F}_2^n$ is the set of all inputs consistent with our initial empty partial assignment.

At every step i, we will show how to find a constant $b \in \{0,1\}$ such that after fixing $y_i = b$, the size of our disjoint union $|\mathcal{U}|$ reduces by a factor of $4/3$. Therefore, after repeating this procedure for the $3n$ outputs from O, we will have an empty $\mathcal{U}$, and the constructed partial assignment will give us a solution to the NC^0_k-Avoid problem.

At the ith iteration of the algorithm, we have values of outputs $y_1, \dots y_{i-1}$ fixed, and are to fix the value of y_i. We have two choices: either set $y_i = 0$ or set $y_i = 1$. By Lemma 13, we have two affine subspaces (or empty sets) $\mathcal{U}_{j,0}, \mathcal{U}_{j,1} \subseteq \mathcal{U}_j$ containing all inputs $x \in \mathcal{U}_j$ mapping to $y_i = 0$ and $y_i = 1$, respectively. Moreover, Lemma 13 guarantees that $|\mathcal{U}_{j,0}| + |\mathcal{U}_{j,1}| \leq 3|\mathcal{U}_j|/2$. Summing over all $0 \leq j < 2^t$, we get

$$\sum_j |\mathcal{U}_{j,0}| + \sum_j |\mathcal{U}_{j,1}| \leq \sum_j 3|\mathcal{U}_j|/2 = 3|\mathcal{U}|/2\,.$$

Let $b \in \{0,1\}$ be the value minimizing $\sum_j |\mathcal{U}_{j,b}|$. In particular, we have that $\sum_j |\mathcal{U}_{j,b}| \leq 3|\mathcal{U}|/4$. Therefore, setting $y_i = b$ reduces the size of $\mathcal{U}$ at least by a factor of $4/3$. Repeating this procedure $\log_{4/3}(2^n) + 1 \leq 3n$ times will result in a partial assignment to the output bits O with no inputs that map to it.

The algorithm SubspaceUnion maintains 2^t affine subspaces of $\mathbb{F}_2^n$, computes their dimensions and calls the deterministic polynomial-time AffineReduce procedure polynomial number of times. Therefore, this algorithm runs in time $2^t \cdot \text{poly}(n) = 2^{O(n^{k-1}/m)} \cdot \text{poly}(n)$. ◀

We conclude this section with a corollary stating that SubspaceUnion solves NC^0_k-Avoid efficiently for certain non-trivial values of stretch m.

▶ **Corollary 15.** *For any constants $k \geq 3$ and $\varepsilon > 0$, the algorithm* SubspaceUnion *solves NC^0_k-Avoid on n inputs and m outputs in deterministic polynomial and deterministic sub-exponential $2^{O(n^{1-\varepsilon})}$ time for $m \geq n^{k-1}/\log(n)$ and $m \geq n^{k-2+\varepsilon}$, respectively.*

Algorithm 2 SubspaceUnion(C).

```
Input: NC$_k^0$ circuit $\mathsf{C} : \{0,1\}^n \to \{0,1\}^m$, where $m \geq 3n^{k-2}$
Output: $y \in \{0,1\}^m$, $y \notin \mathrm{Range}(\mathsf{C})$
  Find $x_1, \dots, x_t$ and $y_1, \dots, y_{3n}$ via Corollary 12 for $t \leq 3n^{k-1}/m$
  For $0 \leq j < 2^t$, set $\mathcal{U}_j = \{x \in \{0,1\}^n : \sum_{i=1}^t x_i 2^{i-1} = j\}$
  for i=1 to 3n do
      Find function $f$ at $y_i$
      For $0 \leq j < 2^t$, set $\mathcal{U}_{j,0}$,$\mathcal{U}_{j,1} \leftarrow$ AffineReduce($\mathcal{U}_j, f$)
      Find $b \in \{0,1\}$ minimizing $\sum_j |\mathcal{U}_{j,b}|$
      Set $y_i = b$
      For $0 \leq j < 2^t$, set $\mathcal{U}_j = \mathcal{U}_{j,b}$
  Set all remaining $y_k = 0$
  return $y$
```

References

1 Josh Alman and Lijie Chen. Efficient construction of rigid matrices using an NP oracle. In *FOCS*, 2019. 3

2 Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM Journal on Computing*, 36(4):845–888, 2006. 2, 4, 7, 9

3 Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009. 3, 8

4 Amey Bhangale, Prahladh Harsha, Orr Paradise, and Avishay Tal. Rigid matrices from rectangular PCPs. In *FOCS*, 2020. 3

5 Aleksandr V. Chashkin. On the complexity of Boolean matrices, graphs and their corresponding Boolean functions. *Discrete Mathematics and Applications*, 4(3):229–257, 1994. 3

6 Yeyuan Chen, Yizhi Huang, Jiatu Li, and Hanlin Ren. Range avoidance, remote point, and hard partial truth table via satisfying-pairs algorithms. In *STOC*, 2023. 2

7 Joel Friedman. A note on matrix rigidity. *Combinatorica*, 13(2):235–239, 1993. 3

8 Oded Goldreich and Avishay Tal. Matrix rigidity of random Toeplitz matrices. In *STOC*, 2016. 3

9 Venkatesan Guruswami, Xin Lyu, and Xiuhan Wang. Range avoidance for low-depth circuits and connections to pseudorandomness. In *RANDOM*, 2022. 2, 3, 4, 7, 14

10 Rahul Ilango, Jiatu Li, and Ryan Williams. Indistinguishability obfuscation, range avoidance, and bounded arithmetic. In *STOC*, 2023. 2

11 Russell Impagliazzo and Ramamohan Paturi. The complexity of k-SAT. In *CCC*, 1999. 2

12 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? In *FOCS*, 1998. 2

13 Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *FOCS*, 2000. 2, 4

14 Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *ICALP*, 2002. 2, 4

15 Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos Papadimitriou. Total functions in the polynomial hierarchy. In *ITCS*, 2021. 1, 2

16 Oliver Korten. The hardest explicit construction. In *FOCS*, 2021. 2

17 Jiatu Li and Tianqi Yang. $3.1n - o(n)$ circuit lower bounds for explicit functions. In *STOC*, 2022. 3

18 Pavel Pudlák and Vojtech Rödl. Some combinatorial-algebraic problems from complexity theory. *Discrete Mathematics*, 1(136):253–279, 1994. 3

19 Hanlin Ren, Rahul Santhanam, and Zhikun Wang. On the range avoidance problem for circuits. In *FOCS*, 2022. 2, 3, 4, 9, 10

20 Claude E. Shannon. The synthesis of two-terminal switching circuits. *The Bell System Technical Journal*, 28:59–98, 1949. 3

21 Mohammad A. Shokrollahi, Daniel A. Spielman, and Volker Stemann. A remark on matrix rigidity. *Information Processing Letters*, 64(6):283–285, 1997. 3

22 Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *MFCS*, 1977. 3, 4, 8

A An Alternative Algorithm for NC^0_3-Avoid

▶ **Theorem 16.** *Given an NC^0_3 circuit $\mathsf{C} : \{0,1\}^n \to \{0,1\}^m$, where $m \geq \frac{1}{3}\binom{n}{2} + 2n$, the algorithm* OneSubspace *finds an element $y \in \{0,1\}^m, y \notin \mathrm{Range}(\mathsf{C})$ in deterministic polynomial time.*

Proof. The algorithm maintains an affine subspace $\mathcal{S} \subseteq \mathbb{F}_2^n$ over the inputs, and a partial output assignment $y \in \{0,1,*\}^m$ such that $\mathcal{S}$ contains all inputs $x \in \mathbb{F}_2^n$ consistent with y. Initially, $y = (*,\ldots,*)$ and $\mathcal{S} = \mathbb{F}_2^n$. At each iteration, OneSubspace assigns at most two outputs and reduces the dimension of $\mathcal{S}$ by at least 1. After n steps, $\mathcal{S}$ must have dimension 0. Then the algorithm assigns one more output bit, and terminates with an element $y \notin \mathrm{Range}(\mathsf{C})$.

Now, we only need to argue that the algorithm can reduce the dimension of $\mathcal{S}$ in each iteration and that we can perform each step in polynomial time.

First, if there is an output y_1 that depends on at most 2 inputs x_1, x_2, let f be the function computed at that output: $y_1 = f(x_1, x_2)$. By Lemma 13, AffineReduce$(\mathcal{S}, f)$ outputs an affine subspace $\mathcal{S}_b$ of lower dimension $\dim(\mathcal{S}_b) < \dim(\mathcal{S})$, containing all inputs consistent with $y_1 = b$. Thus, in the following we assume that each output depends on exactly 3 inputs.

Since we fix at most 2 bits of the output at each iteration, the number of unassigned outputs m is always greater than $\frac{1}{3}\binom{n}{2}$. Then, there exists a pair of outputs y_1, y_2 that both depend on the same pair of inputs x_2, x_3. Since each output depends on three pairs of inputs, and the number of such pairs is $\binom{n}{2} < 3m$, there must be a pair of inputs that feeds into two outputs. Let x_1, x_4 be the remaining inputs that feed into the outputs y_1, y_2, respectively: $y_1 = f_1(x_1, x_2, x_3), y_2 = f_2(x_2, x_3, x_4)$.

There exist 4 possible values of (y_1, y_2) and at most 16 possible values of the input bits (x_1, x_2, x_3, x_4) appearing in $\mathcal{S}$. Then, there exists a pair of constants $(b_1, b_2) \in \{0,1\}^2$ such that at most 4 different assignments $A \subseteq \{0,1\}^4$ to (x_1, x_2, x_3, x_4) are consistent with the partial assignment $(y_1, y_2) = (b_1, b_2)$. Since $|A| \leq 4$, there is a 3-dimensional affine subspace in $\mathbb{F}_2^4$ that contains all points from A. Therefore, there is a hyperplane $\mathcal{H} \subseteq \mathbb{F}_2^4$ defining this 3-dimensional affine subspace. Extending $\mathcal{H}$ to all n inputs, gives us an affine subspace $\mathcal{H}' = \{x \in \mathbb{F}_2^n : (x_1, x_2, x_3, x_4) \in \mathcal{H}\} \subseteq \mathbb{F}_2^n$ that contains all inputs in $\mathbb{F}_2^n$ consistent with the partial assignment $(y_1, y_2) = (b_1, b_2)$.

If $\mathcal{S} \not\subseteq \mathcal{H}'$, then setting $(y_1, y_2) = (b_1, b_2), \mathcal{S} = \mathcal{S} \cap \mathcal{H}'$ reduces the dimension of the affine subspace $\mathcal{S}$. In the following we assume that $\mathcal{S} \subseteq \mathcal{H}'$.

- Suppose there exists $(c_1, c_2) \in \{0,1\}^2$ such that for all points (x_1, x_2, x_3, x_4) in $\mathcal{H}'$, $(f_1(x_1, x_2, x_3), f_2(x_2, x_3, x_4)) \neq (c_1, c_2)$. Then we can set $(y_1, y_2) = (c_1, c_2)$ and $\mathcal{S} = \emptyset$ as no points in $\mathcal{S} \subseteq \mathcal{H}'$ can output (c_1, c_2). In this case we found a $y \notin \mathrm{Range}(\mathsf{C})$.
- If there are no such assignments, then since $|\mathcal{H}| = 8$, there must exist an assignment $(c_1, c_2) \in \{0,1\}^2$ such that at most two points from $\mathcal{H}$ are consistent with $(y_1, y_2) = (c_1, c_2)$. These (at most) two points form a 0- or 1-dimensional affine subspace $\mathcal{U} \subseteq \mathbb{F}_2^4$, which we extend to all n inputs $\mathcal{U}' = \{x \in \mathbb{F}_2^n : (x_1, x_2, x_3, x_4) \in \mathcal{U}\} \subseteq \mathbb{F}_2^n$.

- If $\mathcal{S} \not\subseteq \mathcal{U}'$, we can set $(y_1, y_2) = (c_1, c_2)$ and $\mathcal{S} = \mathcal{S} \cap \mathcal{U}'$, reducing the dimension of $\mathcal{S}$.
- Otherwise, all inputs in $\mathcal{S} \subseteq \mathcal{U}'$ have $(y_1, y_2) = (c_1, c_2)$, and we can set $(y_1, y_2) = (1 - c_1, c_2)$ to obtain $\mathcal{S} = \emptyset$.

This algorithm performs n iterations, each of which computes dimensions of a constant number of explicitly given affine subspaces in polynomial time. ◀

Algorithm 3 OneSubspace(C).

```
Input: $\mathsf{NC}^0_3$ circuit $\mathsf{C}: \{0,1\}^n \to \{0,1\}^m$, where $m \geq \frac{1}{3}\binom{n}{2} + 2n$
Output: $y \in \{0,1\}^m$, $y \notin \mathrm{Range}(\mathsf{C})$
  Let $\mathcal{S} = \mathbb{F}_2^n$
  for i=1 to n do
      if $\mathcal{S} = \emptyset$ then
          Set all remaining $y_k = 0$
          return $y$
      if $\exists y_1, x_1, x_2$ s.t. $y_1 = f(x_1, x_2)$ then
          $\mathcal{S}_0, \mathcal{S}_1 = \textsc{AffineReduce}(\mathcal{S}, f)$
          Find $b \in \{0,1\}$ that minimizes $|\mathcal{S}_b|$, Set $y_1 = b$, $\mathcal{S} = \mathcal{S}_b$
      else
          Find $y_1, y_2, x_1, x_2, x_3, x_4$ s.t. $y_1 = f_1(x_1, x_2, x_3)$, $y_2 = f_2(x_2, x_3, x_4)$
          Find $b_1, b_2 \in \{0,1\}$, s.t.
              $A = \{(x_1, x_2, x_3, x_4) \in \mathbb{F}_2^4 \colon (f_1(x_1, x_2, x_3), f_2(x_2, x_3, x_4)) = (b_1, b_2)\}$ and $|A| \leq 4$
          Let $\mathcal{H} \subseteq \mathbb{F}_2^4$ be the hyperplane defined by points in $A$
          Let $\mathcal{H}' = \{x \in \mathbb{F}_2^n \colon (x_1, x_2, x_3, x_4) \in \mathcal{H}\}$
          if $\mathcal{S} \not\subseteq \mathcal{H}$ then
              Set $(y_1, y_2) = (b_1, b_2)$, $\mathcal{S} = \mathcal{S} \cap \mathcal{H}'$
          else
              if $\exists (c_1, c_2)$ s.t. $\forall (x_1, x_2, x_3, x_4) \in \mathcal{H}$ $(f_1(x_1, x_2, x_3), f_2(x_2, x_3, x_4)) \neq (c_1, c_2)$ then
                  Set $\mathcal{S} = \emptyset$, $(y_1, y_2) = (c_1, c_2)$
              else
                  Find $(c_1, c_2)$ s.t.
                      $\mathcal{U} = \{(x_1, x_2, x_3, x_4) \in \mathcal{H} \colon (f_1(x_1, x_2, x_3), f_2(x_1, x_2, x_3)) = (c_1, c_2)\}$,$|\mathcal{U}| \leq 2$
                  Let $\mathcal{U}' = \{x \in \mathbb{F}_2^n | (x_1, x_2, x_3, x_4) \in \mathcal{U}\}$
                  if $\mathcal{S} \not\subseteq \mathcal{U}'$ then
                      Set $\mathcal{S} = S \cap \mathcal{U}'$, $(y_1, y_2) = (c_1, c_2)$
                  else
                      $\mathcal{S} = \emptyset$, $(y_1, y_2) = (1 - c_1, c_2)$
  return $y$
```

Testing Connectedness of Images

Piotr Berman
Unaffiliated Researcher

Meiram Murzabulatov
Computer Science Department, School of Digital Sciences,
Nazarbayev University, Astana, Kazakhstan

Sofya Raskhodnikova
Boston University, MA, USA

Dragos Ristache
Boston University, MA, USA

Abstract

We investigate algorithms for testing whether an image is connected. Given a proximity parameter $\epsilon \in (0,1)$ and query access to a black-and-white image represented by an $n \times n$ matrix of Boolean pixel values, a (1-sided error) connectedness tester accepts if the image is connected and rejects with probability at least 2/3 if the image is ϵ-far from connected. We show that connectedness can be tested nonadaptively with $O(\frac{1}{\epsilon^2})$ queries and adaptively with $O(\frac{1}{\epsilon^{3/2}}\sqrt{\log\frac{1}{\epsilon}})$ queries. The best connectedness tester to date, by Berman, Raskhodnikova, and Yaroslavtsev (STOC 2014) had query complexity $O(\frac{1}{\epsilon^2}\log\frac{1}{\epsilon})$ and was adaptive. We also prove that every nonadaptive, 1-sided error tester for connectedness must make $\Omega(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$ queries.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Property testing, sublinear-algorithms, lower bounds, connectivity, graphs

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.66

Category RANDOM

Funding *Meiram Murzabulatov*: Supported by the Nazarbayev University Social Policy Grant.

1 Introduction

Connectedness is one of the most fundamental properties of images [16]. In the context of property testing, it was first studied two decades ago [18], but the query complexity of this property is still unresolved. We improve the algorithms for testing this property and also give the first lower bound on the query complexity of this task.

We focus on black-and-white images. For simplicity, we only consider square images, but everything in this paper can be easily generalized to rectangular images. We represent an image by an $n \times n$ binary matrix M of pixel values, where 0 denotes white and 1 denotes black. To define *connectedness*, we consider *the image graph* G_M of an image M. The vertices of G_M are $\{(i,j) \mid M[i,j] = 1\}$, and two vertices (i,j) and (i',j') are connected by an edge if $|i-i'| + |j-j'| = 1$. In other words, the image graph consists of black pixels connected by the grid lines. The image is *connected* if its image graph is connected.

We study connectedness in the property testing model [20, 11], first considered in the context of images in [18]. A (1-sided error) property tester for connectedness gets query access to the input matrix M. Given a proximity parameter $\epsilon \in (0,1)$, the tester has to accept if M is connected and reject with probability at least 2/3 if M is ϵ-far from connected. An image is ϵ-far from connected if at least an ϵ fraction of pixels have to be changed to make it connected. The tester is *nonadaptive* if it makes all its queries before receiving any answers; otherwise, it is *adaptive*.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).
Editors: Nicole Megow and Adam D. Smith; Article No. 66; pp. 66:1–66:15
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In [18], it was shown that connectedness can be tested adaptively with $O(\frac{1}{\epsilon^2}\log^2\frac{1}{\epsilon})$ queries. The adaptive complexity of testing connectedness was later improved to $O(\frac{1}{\epsilon^2}\log\frac{1}{\epsilon})$ in [8].

1.1 Our Results

1.1.1 Connectedness Testers

We give two new algorithms for testing connectedness of images: one adaptive, one nonadaptive. Both improve on the best connectedness tester to date in terms of query complexity. Previously, no nonadaptive testers for connectedness were proposed.

▶ **Theorem 1.1.** *Given a proximity parameter $\epsilon \in (0,1)$, connectedness of $n \times n$ images, where $n > \epsilon^{-2} \cdot 256$, can be ϵ-tested adaptively and with 1-sided error with query and time complexity $O(\frac{1}{\epsilon^{3/2}}\sqrt{\log\frac{1}{\epsilon}})$.*

It can be tested nonadaptively and with 1-sided error with query and time complexity $O(\frac{1}{\epsilon^2})$.

Previous algorithms for testing connectedness of images are modeled on the connectedness tester for bounded-degree graphs by Goldreich and Ron [12]: they pick a uniformly random pixel and adaptively try to find a small connected component by querying its neighbors. As discussed in [18], even though connectedness of an image is defined in terms of the connectedness of the corresponding (degree-4) image graph, these two properties are different because of how the distance is defined. In the bounded-degree graph model, the (absolute) distance between graphs is the number of edges that need to be changed to transform one graph into the other. In contrast, the (absolute) distance between two image graphs is the number of pixels (vertices) on which they differ; in other words, the edge structure of the image graph is fixed, and only vertices can be added or removed to transform one graph into another. However, previous connectedness testers in the image model did not take advantage of the differences.

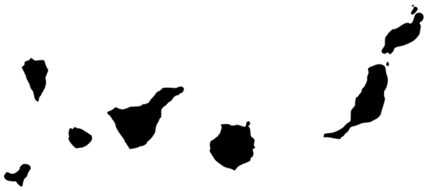


Figure 1 An image M.

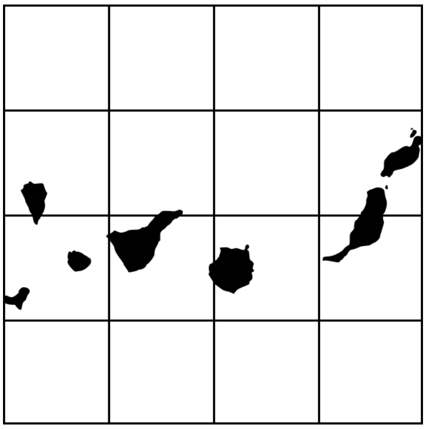


Figure 2 The same image with a grid.

As our starting point, we use an idea from [7] that gave an algorithm for approximating the (relative) distance to the nearest connected image with additive error ϵ with query $O(\frac{1}{\epsilon^4})$ and running time $\exp\left(O\left(\frac{1}{\epsilon}\right)\right)$. They observed that one can modify an image in a small number of pixels by drawing a grid on the image (as shown in Figures 1 and 2). In the resulting image, the distance to connectedness is determined by the properties of individual squares into which the grid lines partition the image.

Our algorithms consider different partitions of the grid (a logarithmic number of partitions in $\frac{1}{\epsilon}$). For each partition, they sample random squares and try to check whether the squares satisfy the following property called *border-connectedness.*

▶ **Definition 1.2** (Border connectedness)**.** *A (sub)image s is* border-connected *if for every black pixel (i, j) of s, the image graph G_s contains a path from (i, j) to a pixel on the border of s. The property* border connectedness, *denoted C', is the set of all border-connected images.*

Our nonadaptive algorithm reads all pixels in each sampled square. Our adaptive algorithm further partitions each square into diamonds, as shown in Figure 4. It queries all pixels on the diamond lattice and then adaptively tries to catch a witness (that is, a small connected component) in one of the diamonds, using the lattice structure. (We could have partitioned into squares again, but partitioning into diamonds makes the proof cleaner and saves a constant factor in the analysis.)

1.1.2 The Lower Bound

We also prove the first nontrivial lower bound for testing connectedness of images. Note that all nontrivial properties have query complexity $\Omega(1/\epsilon)$, even for adaptive testers. In particular, for connectedness, this is the number of queries needed to distinguish between the white image and an image, where we color black a random subset of size ϵn^2 of the set of pixels with both coordinates divisible by 3. By standard arguments, it implies a lower bound of $\Omega(1/\epsilon)$. Some properties of images, such is being a halfplane, can be tested nonadaptively (with 1-sided error) with $O(1/\epsilon)$ queries [5]. We show that it is impossible for connectedness.

▶ **Theorem 1.3.** *Every nonadaptive (1-sided error) ϵ-tester for connectedness of images must query $\Omega(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$ pixels (for some family of images).*

Every 1-sided error tester must catch a witness of disconnectedness in order to reject. This witness could include a connected component completely surrounded by white pixels. The difficulty for proving hardness is that, unlike in the case of finding a witness for disconnectedness of graphs, the algorithm does not have to read the whole connected component. Instead, it is sufficient to find a closed white loop with a black pixel inside it (and another black pixel outside it). As we discussed, it is sufficient for an algorithm to look for witnesses inside relatively small squares (specifically, squares with side length $O(1/\epsilon)$), since adding a grid around such squares, as shown in Figure 2, would change $O(\epsilon n^2)$ pixels. But no matter how you had a witness inside such a square, it can be easily captured with $O(1/\epsilon)$ queries if the border of the square is white.

To overcome this difficulty, we consider a checkerboard-like pattern with white squares replaced by many parallel lines, called *bridges*, with one white (disconnecting) pixel positioned randomly on each bridge. See Figure 5. To catch a white border around a connected component, a tester has to query all disconnecting pixels of at least one square. To make this difficult, we hide the checkerboard pattern inside a randomly positioned *interesting window.* The sizes of interesting windows and their positions are selected so that the tester cannot effectively reuse queries needed to succeed in catching the disconnecting pixels in each interesting window.

1.2 Other Related Work

In addition to [12], connectedness testing and approximating the number of connected components in graphs in sublinear time was explored in [9, 8, 4]. Other property testing tasks studied in the pixel model of images, the model considered in this paper, include testing whether an image is a half-plane [18], convexity [18, 6, 5], and image partitioning properties [13]. Early implementations and applications to vision were provided in [13, 14, 15, 17]. Finally, general classes of matrix properties were investigated, including matrix-poset properties [10], *earthmover resilient* properties [2], *hereditary* properties [1], and classes of matrices that are free of specified patterns [3].

Testing connectedness has also been studied by Ron and Tsur [19] with a different input representation suitable for testing sparse images.

2 Definitions and Notation

We use $[0..n)$ to denote the set of integers $\{0, 1, \ldots, n-1\}$ and $[n]$ to denote $\{1, 2, \ldots, n\}$.

2.1 Image Representation

We represent an image by an $n \times n$ binary matrix M of pixel values, where 0 denotes white and 1 denotes black. The object is a subset of $[0..n)^2$ corresponding to black pixels; namely, $\{(i,j) \mid M[i,j] = 1\}$. The *border of the image* is the set $\{(i,j) \in [0..n)^2 \mid i \in \{0, n-1\}$ or $j \in \{0, n-1\}\}$.

2.2 Property Testing Definitions

A *property* $\mathcal{P}$ is a set of images. The set of connected images is denoted C. The *absolute distance* from an image M to a property $\mathcal{P}$, denoted $Dist(M, \mathcal{P})$, is the smallest number of pixels in M that need to be modified to get an image in $\mathcal{P}$. The (relative) distance between an $n \times n$ image M and a property $\mathcal{P}$ is $dist(M, \mathcal{P}) = Dist(M, \mathcal{P})/n^2$. We say that M is ϵ-far from $\mathcal{P}$ if $dist(M, \mathcal{P}) \geq \epsilon$; otherwise, M is ϵ-close to $\mathcal{P}$.

3 Adaptive and Nonadaptive Property Testers for Connectedness

In this section, we present our testers for connectedness, proving Theorem 1.1. Both testers use the same top-level procedure, described in Algorithm 1. First, it samples random pixels to ensure that a black pixel is found. It will be used later to certify non-connectedness by producing a black pixel and an isolated black component. Then Algorithm 1 considers a logarithmic number of partitions of the image into subimages of the same size. For each partition, it samples a carefully selected number of these subimages and tests them for border connectedness (see Definition 1.2). This is where the two algorithms diverge. The nonadaptive algorithm tests for border connectedness using the subroutine *Exhaustive-Square-Tester* which queries all pixels in the sampled square and determines exactly if the square is border connected. The adaptive algorithm uses subroutine *Diagonal-Square-Tester* (Algorithm 2). If the top-level procedure finds a subimage that violates border connectedness and a black pixel outside that subimage, it rejects; otherwise, it accepts.

To simplify the analysis of the algorithm, we assume[1] that $n-1$ and $1/\epsilon$ are powers of 2. Next, we define terminology used to describe the partitions considered by Algorithm 1.

[1] This assumption can be made w.l.o.g. because if $n \in (2^{i-1}+1, 2^i+1)$ for some i , instead of the original image M we can consider a $(2^i+1) \times (2^i+1)$ image M', which is equal to M on the corresponding coordinates and has white pixels everywhere else. Let $\epsilon' = \epsilon n^2/(2^i+1)^2$. To ϵ-test M for connectedness, it suffices to ϵ'-test M' for connectedness. The resulting tester for M has the desired query complexity because $\epsilon' = \Theta(\epsilon)$. If $\epsilon \in (1/2^j, 1/2^{j-1})$ for some j, to ϵ-test a property $\mathcal{P}$, it suffices to run an ϵ''-test for $\mathcal{P}$ with $\epsilon'' = 1/2^j < \epsilon$.

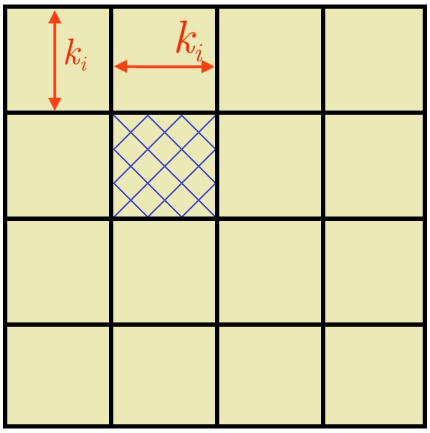


Figure 3 An illustration to Definition 3.1: black lines consist of grid pixels; the 16 yellow $k_i \times k_i$ squares represent squares of S_i. One of the squares includes diagonal lattice pixels from Definition 3.6 that are used in Algorithm 2.

▶ **Definition 3.1** (Grid pixels, squares of different levels, witnesses). *For* $i \in [0..\log \frac{1}{\epsilon})$, *let* $k_i = \frac{4}{\epsilon} \cdot 2^{-i} - 1$. *Call* (x, y) *a* grid pixel of level i *if* $(k_i + 1)|x$ *or* $(k_i + 1)|y$. *For all coordinates* u, v, *which are divisible by* $k_i + 1$, *the* $k_i \times k_i$ *subimage that consists of pixels* $[k_i]^2 + (u, v)$ *is called a* square of level i. *The set of all squares of level* i *is denoted* $\mathcal{S}_i$. Boundary pixels *of a square of level* i *are the pixels of the square which are adjacent to the grid pixels of level* i. *A square of any level that violates property* C' *(see Definition 1.2) is called a* witness.

Algorithm 1 *ϵ-tester* for connectedness.

```
input : parameter $\epsilon \in (0,1)$ ; access to a $n \times n$ binary matrix $M$.
1 Query $\frac{4}{\epsilon}$ pixels uniformly at random with replacement.
2 For $i = 0$ to $\log \frac{1}{\epsilon}$
    (a) Sample $2^{i+3}$ uniformly random squares of level $i$ (see Definition 3.1) with replacement.
    (b) For every sampled square $s$ from Step 2a, let $[k_i]^2 + (u, v)$ be the set of its pixels.
        Run the border-connectedness subroutine with inputs $i, u, v$: if the tester is nonadaptive,
        use Exhaustive-Square-Tester; otherwise use Diagonal-Square-Tester (Algorithm 2).
        If the subroutine rejects and Step 1 detected a black pixel outside $s$, reject.
3 Accept.
```

3.1 Effective Local Cost and the Structural Lemma

In this section, we prove our main structural lemma (Lemma 3.5) used in the analysis of Algorithm 1. It relates the distance to connectedness to the properties of individual squares, defined next.

▶ **Definition 3.2** (Local cost and effective local cost). *For a level* i, *consider a square* $s \in \mathcal{S}_i$. *The* local cost *of* s *is* $lc(s) = Dist(s, C')$. *The* effective local cost *of* s *is* $elc(s) = \min(2k_i, lc(s))$.

Next we state and prove two claims used in the proof of Lemma 3.5.

▷ Claim 3.3. For any square s of level $i \in [0..\log\frac{1}{\epsilon} - 1)$, let $ch(s)$ denote the set of its 4 children (i.e., squares of level $i+1$ inside it). Then $lc(s) \leq elc(s) + \sum_{q \in ch(s)} lc(q)$.

Proof. If $lc(s) \leq 2k_i$ then $elc(s) = lc(s)$. Since all costs are nonnegative, the inequality in Claim 3.3 becomes trivial.

Now assume that $lc(s) > 2k_i$. Then $elc(s) = 2k_i$. We can modify $\sum_{q \in ch(s)} lc(q)$ pixels in s so that all its children satisfy the property C'. (Note that here C' is the set of $k_i \times k_i$ (sub)images.) Then we can make black all pixels of s that partition it into its children, i.e., pixels $\{(x,y) \mid x = \frac{k_i+1}{2} \text{ or } y = \frac{k_i+1}{2}\}$. There are at most $2k_i$ such pixels, and after this modification s will satisfy C'. Hence, $lc(s) \leq elc(s) + \sum_{q \in ch(s)} lc(q)$. ◁

▷ Claim 3.4 (Distance to Border Connectedness). Let s be a $k \times k$ image. Then

$$Dist(s, C') \leq \frac{k^2}{4}.$$

Proof. If s contains at most $\frac{k^2}{4}$ black pixels, we can make all of them white, i.e., modify at most $\frac{k^2}{4}$ pixels and obtain an image that satisfies C'. Now consider an image s with more than $\frac{k^2}{4}$ black pixels, i.e., with less than $\frac{3k^2}{4}$ white pixels. Partition all pixels of s into 3 groups such that group $i \in \{0,1,2\}$ contains all pixels (x,y), where $y \equiv i \pmod 3$. Making all pixels of one group black produces an image that satisfies C'. By averaging, at least one group has less than $\frac{k^2}{4}$ white pixels. Making all these white pixels black results in an image that satisfies C'. This completes the proof. ◁

▶ **Lemma 3.5** (Structural Lemma). *Let M be an $n \times n$ image that is ϵ-far from C. Then the sum of effective local costs of all squares of all levels inside M is at least $\frac{\epsilon n^2}{2}$.*

Proof. To obtain a connected image, we can make all the $\frac{\epsilon n^2}{2}$ grid pixels of level 0 black and modify pixels inside every square of $\mathcal{S}_0$ to ensure it satisfies the property C'. Thus,

$$\sum\nolimits_{s \in \mathcal{S}_0} lc(s) \geq Dist(M, C) - \frac{\epsilon n^2}{2} \geq \frac{\epsilon n^2}{2}.$$

Consequently, it suffices to show that $\sum_{i=0}^{\log\frac{1}{\epsilon}-1} \sum_{s \in \mathcal{S}_i} elc(s) \geq \sum_{s \in \mathcal{S}_0} lc(s)$.

Let s be a square of level i. We use $desc(s,j)$ to denote the set of all squares of level $j \geq i$ inside s. (In particular, $desc(s,i)$ contains only s.) We will prove by induction that for any integer $j \in [i, \log\frac{1}{\epsilon} - 1)$,

$$lc(s) \leq \sum_{h=i}^{j} \sum\nolimits_{q \in desc(s,h)} elc(q) + \sum\nolimits_{q \in desc(s,j+1)} lc(q). \tag{1}$$

For $j = i$ (base case), the inequality in (1) holds since it is equivalent to the statement in Claim 3.3. Assume that (1) holds for $j = m$, that is,

$$lc(s) \leq \sum_{h=i}^{m} \sum\nolimits_{q \in desc(s,h)} elc(q) + \sum\nolimits_{q \in desc(s,m+1)} lc(q).$$

We will prove (1) holds for $j = m+1$. By Claim 3.3,

$$lc(q) \leq elc(q) + \sum\nolimits_{f \in ch(q)} lc(f).$$

Thus,

$$\sum\nolimits_{q\in desc(s,m+1)} lc(q) \le \sum\nolimits_{q\in desc(s,m+1)} elc(q) + \sum\nolimits_{q\in desc(s,m+2)} lc(q)$$

and

$$\begin{aligned} lc(s) &\le \sum_{h=i}^{m} \sum\nolimits_{q\in desc(s,h)} elc(q) + \sum\nolimits_{q\in desc(s,m+1)} lc(q) \\ &\le \sum_{h=i}^{m+1} \sum\nolimits_{q\in desc(s,h)} elc(q) + \sum\nolimits_{q\in desc(s,m+2)} lc(q), \end{aligned}$$

completing the inductive argument.

By (1) applied with $j = \log\frac{1}{\epsilon} - 2$, we get that for every square s of level i,

$$\begin{aligned} lc(s) &\le \sum_{h=i}^{\log\frac{1}{\epsilon}-2} \sum\nolimits_{q\in desc(s,h)} elc(q) + \sum\nolimits_{q\in desc(s,\log\frac{1}{\epsilon}-1)} lc(q) \\ &= \sum_{h=i}^{\log\frac{1}{\epsilon}-1} \sum\nolimits_{q\in desc(s,h)} elc(q), \end{aligned} \tag{2}$$

where the final equality holds because in every square of level $i = \log\frac{1}{\epsilon} - 1$, we have $k_i = 7$, and consequently, by Claim 3.4, the local cost is at most $\frac{7^2}{4} < 2\cdot 7$, i.e., it is equal to the effective local cost of that square.

Summing up (2) for all squares in $\mathcal{S}_0$, we get

$$\sum\nolimits_{s\in\mathcal{S}_0} lc(s) \le \sum\nolimits_{s\in\mathcal{S}_0} \sum\nolimits_{h=i}^{\log\frac{1}{\epsilon}-1} \sum\nolimits_{q\in desc(s,h)} elc(q) = \sum\nolimits_{h=i}^{\log\frac{1}{\epsilon}-1} \sum\nolimits_{s\in\mathcal{S}_i} elc(s),$$

where the last equality is obtained by switching the order of summations and rearranging the second summation in terms of levels. ◀

3.2 Testing Border-Connectedness

In this section, we state and analyze our adaptive border connectedness subroutine after defining the concepts used in it. We state the guarantees of both border connectedness subroutines in Lemma 3.7.

For the adaptive subroutine, we partition the square into diamonds and fences surrounding them, as described in Definition 3.6. The subroutine queries all pixels on the fences and categorizes diamonds into those whose black pixels are potentially connected to the border (set B in Algorithm 2) and those whose black pixels are definitely not (set A in Algorithm 2). Then it tries to find a black pixel in a diamond from set A or (using BFS) an isolated black component in diamonds from set B. Observe that either of the two provides evidence that the square is not border-connected.

▶ **Definition 3.6** (Diagonal lattice pixels, diamonds and fences). *For a fixed value of i, consider a square s in $\mathcal{S}_i$. Let m_i be the largest odd integer less than or equal to $\lceil\sqrt{k_i/\log k_i}\rceil$.* Diagonal lattice pixels *of the square is the set of pixels $L = \{(x,y)\in s \mid m_i|(x+y)$ or $m_i|(x-y)\}$. Let D be a $k_i\times k_i$ image whose pixels with coordinates from $[k_i]^2 - L$ are black and the remaining pixels are white. A set of pixels of the square whose corresponding pixels in D form a connected component is called a* diamond *of the square. A set of all diagonal lattice pixels that have some neighbouring pixel(s) from a particular diamond is called the* fence *of that diamond.*

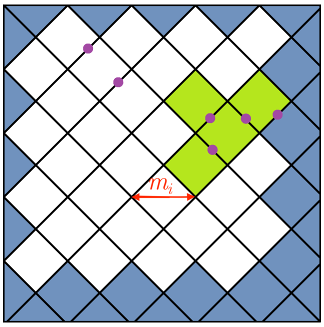


Figure 4 An example of execution of Algorithm 2. Black lines represent lattice pixels. The blue diamonds are included in B because they are adjacent to the border of the square. The purple dots represent black pixels in the lattice. The green diamonds are added to B during the BFS because their fences contain black pixels. The white diamonds remain in A.

Note that lattice pixels are not part of any diamond. Moreover, some diamonds are partial (those that have pixels from the border of the square.)

Algorithm 2 Border-connectedness subroutine *Diagonal-Square-Tester*.

```
input  : parameters $i$, $u$ and $v$; access to an $n \times n$ matrix $M$.
Let $s$ be a square of level $i$ that consists of pixels $[k_i]^2 + (u, v)$ and $m_i$ be the largest odd
  integer less than or equal to $\lceil \sqrt{k_i / \log k_i} \rceil$.
Query all the diagonal lattice pixels of $s$ (see Definition 3.6).
Initialize $B$ to be the set of all diamonds of $s$ that contain a border pixel of $s$. Initialize
  $A$ to be the set of the remaining diamonds of $s$.
While $\exists d_1 \in B$ and $\exists d_2 \in A$ such that $d_1$ and $d_2$ have a black pixel in the common
  portion of their fences, move $d_2$ from $A$ to $B$.
Query $k_i m_i$ pixels in $s$ uniformly at random with replacement.
  (a) If a black pixel from a diamond in $A$ or its fence is discovered, reject.
  (b) If a black pixel from a diamond in $B$ is discovered, pick a natural number $x \in [k_i^2]$ from
      the distribution with the probability mass function $f(j) = \frac{1}{j(j+1)}$ for all $j \in [k_i^2 - 1]$
      and $f(j) = \frac{1}{j}$ for $j = k_i^2$. (Observe that $\Pr(x \geq j) = \frac{1}{j}$ for all $j \in [k_i^2]$.) Starting from
      the black pixel, perform a BFS of its connected component. If the search halted after
      discovering at most $x$ black pixels, none of which are on the border of $s$, reject. Else (if
      $x + 1$ black pixels were found for this component or a black pixel on the border of
      square $s$ is reached) stop the search and proceed with the remaining queried pixels.
Accept.
```

Recall from Definition 3.1 that a witness is a square of one of the levels that violates border-connectedness.

▶ **Lemma 3.7.** *Fix level $i \in [0..\log \frac{1}{\epsilon})$. Let $s \in \mathcal{S}_i$ be a witness that consists of pixels $[k_i]^2 + (u, v)$. A border-connectedness subroutine of Algorithm 1 rejects s with probability at least $\frac{elc(s) \cdot \alpha}{2k_i}$, where $\alpha = 1$ for* Exhaustive-Square-Tester *and $\alpha = 1 - e^{-1}$ for* Diagonal-Square-Tester.

Proof. *Exhaustive-Square-Tester* determines that s is a witness with probability $1 \geq \frac{elc(s)}{2k_i} \cdot 1$.

Now we prove the statement for *Diagonal-Square-Tester*. Let A and B be defined as in Algorithm 2 after Step 3. Let a be the number of black pixels in all the diamonds of the set A and on the fences of those diamonds. Let b be the number of connected components of the image graph that are formed by black pixels in all the diamonds in the set B and in their fences and that contain no pixels on the border of s.

Next, we prove that

$$bm_i + a \geq lc(s) \geq elc(s). \tag{3}$$

We claim that we can connect all b connected components to each other and to the border of the square by modifying at most bm_i pixels. To see this, notice that any two black pixels in the same diamond can be connected to each other by changing less than m_i pixels (by taking any Manhattan-distance shortest path between the two pixels and making it all black). To prove the claim, we can connect the b connected components to the border of the square in the order their diamonds were added to B by the algorithm. The initial diamonds placed in B have at least one pixel on the border of the square, so their connected components can be directly connected to the border (using at most m_i pixels per connected component). Now assume that we already connected to the border all components that have pixels in the diamonds added to B so far. When the algorithm moves some diamond d_2 from A to B, it is done because there is already a diamond d_1 in B such that d_1 and d_2 have a black pixel β in the common portion of their fences. Then β must be already connected to the border. If there are any connected components in d_2 that are not connected to the border yet, we can fix that by connecting them to β (using at most m_i pixels per connected component). We proceed like this until all b connected components of the diamonds that were added to B by the algorithm are connected to the border of the square. At this point, we have changed at most bm_i pixels.

Recall that we have a black pixels in all the diamonds of the set A and on the fences of those diamonds. We make all of them white. After these at most $bm_i + a$ modifications, the square s satisfies C'. Thus, Equation (3) holds.

Observe that for $x \in [0, 1]$,

$$x \geq 1 - e^{-x} \geq x(1 - e^{-1}) > x/2. \tag{4}$$

By (4), the probability that a specific pixel from s is selected by Algorithm 2 in Item b is

$$1 - (1 - 1/k_i^2)^{k_i m_i} \geq 1 - e^{-\frac{k_i m_i}{k_i^2}} > \frac{m_i}{2k_i}.$$

Consider one of the b connected components defined above. Let p be the number of pixels in it. *Diagonal-Square-Tester* finds this component completely if in Step 4b of the subroutine, one of the p pixels from the component is selected and $x \geq p$ is chosen. This happens with probability at least $\frac{1}{p} \cdot \frac{m_i}{2k_i} \cdot p = \frac{m_i}{2k_i}$. The subroutine determines that s is a witness if it finds one of the b connected components or one of the a black pixels considered above. Thus, by (3) and (4), the probability that Algorithm 2 determines that square s is a witness is at least

$$1 - \left(1 - \frac{m_i}{2k_i}\right)^{a+b} \geq 1 - e^{-\frac{m_i(a+b)}{2k_i}} \geq 1 - e^{-\frac{bm_i+a}{2k_i}} \geq 1 - e^{-\frac{elc(s)}{2k_i}} \geq \frac{elc(s)(1 - e^{-1})}{2k_i},$$

completing the proof of Lemma 3.7. ◀

3.3 Proof of Theorem 1.1

In this section, we use Lemmas 3.5 and 3.7 to complete the analysis of our connectedness tester and the proof of Theorem 1.1.

Algorithm 1 always accepts connected images, since it sees no violation of C' in Step 2b. Consider an image M that is ϵ-far from connectedness. For every $i \in [0..\log\frac{1}{\epsilon})$, there are $\frac{n^2}{k_i^2}$ squares in $\mathcal{S}_i$. Let $\mathcal{W}_i$ be the set of all witnesses in $\mathcal{S}_i$. Let s be a square in $\mathcal{W}_i$. The probability that Algorithm 1 chooses s in Step 2a is

$$1-\left(1-\frac{k_i^2}{n^2}\right)^{2^{i+3}} \geq 1-e^{\frac{k_i^2\cdot 2^{i+3}}{n^2}} > \frac{k_i^2\cdot 2^{i+2}}{n^2},$$

where the inequalities follow from (4). By Lemma 3.7, the probability that the algorithm rejects s after choosing it is at least $\alpha\cdot elc(s)/2k_i$. Thus, the probability that the algorithm samples s in Step 2a and then rejects in Step 2b is at least

$$\frac{2^{i+2}k_i\cdot\alpha\cdot elc(s)}{2n^2} \geq \frac{6\alpha\cdot elc(s)}{\epsilon n^2}.$$

Thus, the probability that Algorithm 1 does not catch $s \in \mathcal{W}_i$ as a witness is at most $1-\frac{6\alpha\cdot elc(s)}{\epsilon n^2} \leq e^{-\frac{6\alpha\cdot elc(s)}{\epsilon n^2}}$. The probability that the algorithm does not catch any witness of $\mathcal{W}_i$ is at most

$$P_i = e^{-\frac{6\alpha}{\epsilon n^2}\cdot\sum_{s\in\mathcal{W}_i} elc(s)}.$$

Observe that $elc(s)=0$, for $s \in \mathcal{S}_i - \mathcal{W}_i$. Thus, by Lemma 3.5,

$$\sum_{s\in\mathcal{W}_i} elc(s) = \sum_{s\in\mathcal{S}_i} elc(s) \geq \frac{\epsilon n^2}{2}.$$

By a union bound over all levels, the probability that Algorithm 1 fails to catch any witness is at most

$$\sum\nolimits_{i\in\{0\}\cup[\log(1/\epsilon)]} P_i \leq e^{-\frac{6\alpha}{\epsilon n^2}\cdot\frac{\epsilon n^2}{2}} = e^{-3\alpha}.$$

Therefore, the probability that the algorithm detects at least one witness is at least $1-e^{-3\alpha}$. Since at least an ϵ fraction of pixels in M are black and every square of every level contains at most $\frac{16}{\epsilon^2} < \frac{\epsilon n^2}{2}$ pixels, the probability that Algorithm 1 detects a black pixel outside of that witness in Step 1 is at least $1-(1-\frac{\epsilon}{2})^{\frac{4}{\epsilon}} > 1-e^{-2}$. Thus, for both values of α, the probability that Algorithm 1 rejects M is at least

$$(1-e^{-2})(1-e^{-3\alpha}) \geq 2/3.$$

3.3.1 Query Complexity

We prove that Algorithm 1 has query complexity $O(\frac{1}{\epsilon^2})$ if it uses *Exhaustive-Square-Tester* as a subroutine. Algorithm 1 samples 2^{i+1} squares of level $i \in \{0\}\cup[\log\frac{1}{\epsilon}]$ and, for each sampled square, it calls *Exhaustive-Square-Tester* which makes $(\frac{4}{\epsilon}\cdot 2^{-i}-1)(\frac{4}{\epsilon}\cdot 2^{-i}-1) < \frac{16}{\epsilon^2 2^{2i}}$ queries in each sampled square of level i. Thus, the query complexity of Algorithm 1 is

$$\sum\nolimits_{i=0}^{\log\frac{1}{\epsilon}} 2^{i+1}\cdot\frac{16}{\epsilon^2 2^{2i}} < \sum\nolimits_{i=0}^{\log\frac{1}{\epsilon}} \frac{32}{\epsilon^2 2^{i}} = O\left(\frac{1}{\epsilon^2}\right).$$

When Algorithm 1 uses *Diagonal-Square-Tester*, it queries at most $\frac{2k_i^2}{m_i}$ diagonal lattice pixels inside each square of level i (in Step 1 of the subroutine). After that, in Step 4 of the subroutine, it selects $k_i m_i$ pixels and a number $x \in [k_i^2]$ from the specified distribution and then makes at most $4x$ queries for each selected pixel. Observe that $\mathbb{E}(x) = O(\log k_i)$. Thus, the expected number of queries inside a square of level i is at most $\frac{2k_i^2}{m_i} + k_i m_i \cdot 4 \cdot O(\log k_i) = O(k_i^{3/2}\sqrt{\log k_i})$. The expected total number of queries is $\sum_{i=0}^{\log(1/\epsilon)} O(k_i^{3/2}\sqrt{\log k_i}) \cdot 2^{i+1} = O(\epsilon^{-3/2}\sqrt{\log \frac{1}{\epsilon}})$.

By standard arguments, the adaptive version of Algorithm 1 can be converted to an algorithm that makes asymptotically the same number of queries in the worst case, and has the same accuracy guarantee and running time.

3.3.2 Running Time

The time complexity of Step 1a and Step 2 of Algorithm 1 is $O(\frac{1}{\epsilon})$. Therefore, the total time complexity of Algorithm 1 is $O(\frac{1}{\epsilon})$+time complexity of Step 1b. In Step 1b, Algorithm 1 uses either *Exhaustive-Square-Tester* or *Diagonal-Square-Tester*. Both of them perform a breadth first search within each sampled square. Breadth first search is linear in the sum of the number of edges and the number of nodes of the graph. Every pixel of a sampled square has at most 4 neighboring pixels. Thus, the number of edges in the image graph of every sampled square is linear in the number of pixels inside it and the time complexity of Step 1b is linear in the number of all queried pixels, i.e., $O(\frac{1}{\epsilon^2})$ for *Exhaustive-Square-Tester* and $O(\epsilon^{-3/2}\sqrt{\log(1/\epsilon)})$ for *Diagonal-Square-Tester*. This completes the proof of the theorem.

4 Lower Bound for Testing Connectedness

In this section, we give a lower bound on the query complexity of testing connectedness, proving Theorem 1.3. We use the standard set up of constructing a distribution $\mathcal{N}$ on ϵ-far inputs such that every deterministic nonadaptive algorithm that makes $q \leq \frac{c}{\epsilon}\log\frac{1}{\epsilon}$ queries (for some constant c) has probability of error greater than 1/3. By Yao's Principle [21], it is sufficient to prove Theorem 1.3.

4.1 Construction of $\mathcal{N}$

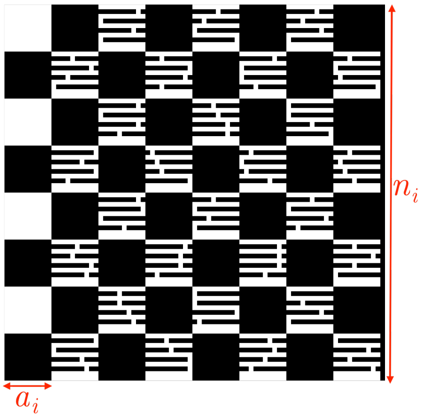


Figure 5 Our construction of $\mathcal{N}$: an interesting window together with a column of black pixels immediately to the right of it. All other pixels in the constructed image are white.

The construction is parameterized by n and the proximity parameter ϵ. It gives a distribution supported on $(n+1)\times(n+1)$ images that are ϵ-far from connectedness. We assume that ϵ is sufficiently small and that $n > \frac{1}{16}(\frac{1}{\epsilon})^{5/8}$. We also assume that n is a power of 2 and $1/\epsilon$ is an even power of 2 and that both of them are sufficiently large, so that all indices in our construction are integer. (See also Footnote 1 for the discussion of integrality issues.) Our construction starts by selecting a *level.* The indices of the levels range from the low index $\ell = \frac{1}{8}\log\frac{1}{\epsilon}$ to the high index $h = \frac{1}{4}\log\frac{1}{\epsilon}$. For all $i \in \{\ell, \ell+1, \dots, h\}$, define $a_i = 2^i$ and $n_i = 16 \cdot \sqrt{\epsilon} \cdot n \cdot a_i = \sqrt{\epsilon} \cdot n \cdot 2^{i+4}$. First, we pick a uniformly random integer $i \in \{\ell, \ell+1, \dots, h\}$. Consider the $n \times n$ image resulting from removing the last row and the last column of the $(n+1)\times(n+1)$ image. We partition this $n \times n$ image into $(\frac{n}{n_i})^2$ squares with side length n_i called *windows of level i*; that is, each window of level i is an $n_i \times n_i$ subimage. We pick one of the windows of level i uniformly at random and call it an *interesting window.* We make the n_i pixels immediately to the right of the interesting window black, representing a vertical black line segment, and all other pixels outside of the interesting window white.

Now we describe how to color the interesting window. See Figure 5 for an illustration. We fill the interesting window with a checkerboard pattern of squares of size $a_i \times a_i$ pixels. Inside each white checkerboard square that is not in the first column, number the rows starting from 0 and make every odd row, excluding the last, fully black, except for one randomly selected pixel for each row. The resulting $\frac{a_i}{2} - 1$ black lines, each containing one white pixel, are called *bridges.* The white pixels on the bridges are called *disconnecting pixels.* We refer to each checkerboard square with the bridges as a *bridge square.*

The intuition behind the construction is the following. Each black checkerboard square is in its own connected component. However, to "catch" this connected component as a witness of disconnectedness, a tester would have to query all the disconnecting pixels in the bridge squares to the left and/or to the right of the black square. Since the positions of the disconnecting pixels are random, it would have to query $\Omega(a_i^2)$ pixels in at least one relevant bridge square. Since the windows are selected at random, the algorithm would have to do it for many windows of each level. The key feature of our construction is that interesting windows of different levels are either disjoint or contained in one another, so potential witnesses for one window can't significantly help with another.

4.2 Analysis of the Construction

We start by showing that all images in the support of $\mathcal{N}$ are far from connected.

▶ **Lemma 4.1.** *Every image in the support of $\mathcal{N}$ is ϵ-far from connected.*

Proof. There are $\frac{(n_i/a_i)^2}{2} = 128\epsilon n^2$ black regions, each one in a separate connected component except for the last $\frac{n_i/a_i}{2} = 8\sqrt{\epsilon}\cdot n$ which are connected by the vertical line. Thus, the image graph has at least $128 \cdot \epsilon n^2 - 8\sqrt{\epsilon}\cdot n$ connected components. Changing one pixel from white to black corresponds to adding a node of degree at most 4 to the image graph. This decreases the number of connected components by at most 3. Removing a pixel decreases the number of connected components by at most 1. Consequently, overall, we need to change at least $\frac{1}{3}(128\epsilon n^2 - 8\sqrt{\epsilon}\cdot n)$ pixels. This is at least ϵn^2 for sufficiently large n; in particular, it holds for $n > \frac{1}{16}(\frac{1}{\epsilon})^{5/8}$, which was our assumption in the beginning of Section 4.1. ◀

Next we show that if the number of queried pixels is small, then every 1-sided error deterministic algorithm detects a violation of connectedness in an image distributed according to $\mathcal{N}$ with insufficiently small probability.

▶ **Lemma 4.2.** *Let M be an image distributed according to $\mathcal{N}$. Fix a deterministic nonadaptive 1-sided error algorithm $\mathcal{A}$ for testing connectedness of images. Let $\mathbf{Q}$ be the set of pixels queried by $\mathcal{A}$ and let $q = |\mathbf{Q}|$. For sufficiently small constant $c > 0$, if $q \le \frac{c}{\epsilon}\log\frac{1}{\epsilon}$, then $\mathcal{A}$ detects a violation of connectedness in M with probability less than $1/3$.*

Proof. We define an event E that must happen in order for the algorithm $\mathcal{A}$ to succeed in finding a witness of disconnectedness in an image distributed according to $\mathcal{N}$. Then we show that the probability of E is too small. To define E, we define a special group of pixels.

▶ **Definition 4.3** (A revealing set and event E)**.** *Let M be an image distributed according to $\mathcal{N}$. The set of all disconnecting pixels from one bridge square of M is called a* revealing set *for the window containing the bridge square. Let E denote the event that $\mathbf{Q}$ contains a revealing set for the interesting window of M.*

Since the tester $\mathcal{A}$ has 1-sided error, it can reject only if it finds a violation of connectedness. In particular, for an image distributed according to $\mathcal{N}$, in order to succeed, it must find a revealing set for the interesting window of the image.

▷ Claim 4.4. If the number of queries $q \le \frac{c}{\epsilon}\log\frac{1}{\epsilon}$ for sufficiently small constant c, then $\Pr[E] < 1/3$.

Proof. An important feature of our construction is that the largest bridge square is smaller than the smallest window. Indeed, the side length of the largest bridge square is $a_h = (\frac{1}{\epsilon})^{1/4}$, whereas the side length of the smallest window is $n_\ell = 16\sqrt{\epsilon}\cdot(\frac{1}{\epsilon})^{1/8}n = 16\epsilon^{3/8}n$. Thus, $a_h < n_\ell$ as long as $n > \frac{1}{16}(\frac{1}{\epsilon})^{5/8}$, as we assumed in the beginning of Section 4.1. The consequence of this feature and the fact that both a_i's and n_i's are powers of 2 is that each bridge square of level i is contained in one window of level j for all levels i and j.

A (potential) bridge square of level $i \in \{\ell, \ell+1, \dots, h\}$ is *covered* if $\mathbf{Q}$ contains at least $a_i^2/8$ pixels from that square. A window of level i is *good* if it contains a covered bridge square of level $j \ge i$; otherwise, it is *bad*. For each good window w, we pick a covered bridge square of the highest level contained in w and call it the covered bridge square *associated with* w. All windows of the same level are associated with different covered bridge squares, because each bridge square is contained in exactly one window of a given level.

Let G be the event that the interesting window in M is good. Then, by the law of total probability,

$$\Pr[E] = \Pr[E \mid G]\cdot\Pr[G] + \Pr[E \mid \overline{G}]\cdot\Pr[\overline{G}] \le \Pr[G] + \Pr[E \mid \overline{G}].$$

Next, we analyze event G. For every level $i \in \{\ell, \ell+1, \dots, h\}$, let g_i be the number of good windows of level i associated with covered bridge squares of level i and let t_i be the total number of good windows of level i. Then $g_h = t_h$ and $g_i = t_i - t_{i+1}$ for all $i \in [\ell, h-1]$. Observe that, for each level $i \in [\ell, h]$, the covered bridge squares associated with the good windows of level i are distinct. By definition, each of them contributes at least $a_i^2/8$ towards $\mathbf{Q}$. Therefore, the number of all pixels in $\mathbf{Q}$ satisfies

$$\begin{aligned} q &\ge \sum_{i=\ell}^{h}\frac{a_i^2 g_i}{8} = \frac{1}{8}\left(\sum_{i=\ell}^{h-1}a_i^2(t_i - t_{i+1}) + a_h^2 t_h\right) = \frac{1}{8}\left(\sum_{i=\ell}^{h-1}4^i(t_i - t_{i+1}) + 4^h t_h\right)\\ &\ge \frac{3}{32}\sum_{i=\ell}^{h}(4^i t_i). \end{aligned} \tag{5}$$

Recall that $q \le \frac{c}{\epsilon}\log\frac{1}{\epsilon}$, that the total number of levels is $h - \ell + 1 = \Theta(\log\frac{1}{\epsilon})$, and that the number of all windows for each level i is $(\frac{n}{n_i})^2 = \Theta(\frac{1}{4^i\epsilon})$. Thus,

$$\begin{aligned}\Pr[G] &= \frac{1}{h-\ell+1}\sum_{i=\ell}^{h}\frac{t_i}{(n/n_i)^2} = \frac{1}{\Theta(\log(\frac{1}{\epsilon}))}\sum_{i=\ell}^{h}(t_i\cdot\Theta(4^i\epsilon)) = \frac{1}{\Theta(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}))}\sum_{i=\ell}^{h}(4^i t_i)\\ &< \frac{q}{\Theta(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}))} < 1/6,\end{aligned}$$

where Equation (5) was used to obtain the first inequality, and the last inequality holds for sufficiently small constant c.

It remains to analyze $\Pr[E \mid \overline{G}]$, that is, the probability of E, given that the interesting window is bad. Consider a bad window of level $i \in \{\ell, \dots, h\}$. It has at most q bridge squares that contain a queried pixel. Consider one of such bridge squares. Recall that this bridge square has $a_i/2 - 1$ bridges. Number these bridges using integers $1, 2, \dots, a_i/2 - 1$. Let x_k denote the number of pixels of $\mathbf{Q}$ on the bridge number $k \in [a_i/2 - 1]$ of the bridge square. Then the probability that this bridge square has a revealing set for the window is

$$\begin{aligned}\prod_{k=1}^{a_i/2-1}\frac{x_k}{a_i} &\le \left(\frac{1}{a_i/2-1}\cdot\sum_{k=1}^{a_i/2-1}\frac{x_k}{a_i}\right)^{a_i/2-1} \le \left(\frac{a_i/8}{a_i/2-1}\right)^{a_i/2-1} \le \left(\frac{1}{2}\right)^{a_i/2-1}\\ &\le 2(\sqrt{2})^{-\sqrt[8]{\frac{1}{\epsilon}}},\end{aligned}$$

where the first inequality follows from the inequality between geometric and arithmetic means, the second inequality holds since $\sum_{k=1}^{a_i/2} x_k < a_i^2/8$, and the third inequality holds since ϵ is sufficiently small and, consequently, we can assume that the minimum value of a_i is at least 4. By a union bound over all bridge squares in this window that contain a query,

$$\Pr[E \mid \overline{G}] \le 2(\sqrt{2})^{-\sqrt[8]{\frac{1}{\epsilon}}}\cdot q \le 2(\sqrt{2})^{-\sqrt[8]{\frac{1}{\epsilon}}}\cdot\frac{c}{\epsilon}\cdot\log\frac{1}{\epsilon} < \frac{1}{6},$$

for sufficiently small ϵ and constant c.

Thus, $\Pr[E] \le \Pr[G] + \Pr[E \mid \overline{G}] < \frac{1}{6} + \frac{1}{6} = \frac{1}{3}$, as claimed. This completes the proof of Claim 4.4. ◁

This concludes the proof of Lemma 4.2. ◀

Theorem 1.3 follows from Lemma 4.2.

References

1 Noga Alon, Omri Ben-Eliezer, and Eldar Fischer. Testing hereditary properties of ordered graphs and matrices. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 848–858. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.83.

2 Omri Ben-Eliezer and Eldar Fischer. Earthmover resilience and testing in ordered structures. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, volume 102 of *LIPIcs*, pages 18:1–18:35. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CCC.2018.18.

3 Omri Ben-Eliezer, Simon Korman, and Daniel Reichman. Deleting and testing forbidden patterns in multi-dimensional arrays. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 9:1–9:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.9.

4 Petra Berenbrink, Bruce Krayenhoff, and Frederik Mallmann-Trenn. Estimating the number of connected components in sublinear time. *Inf. Process. Lett.*, 114(11):639–642, 2014. doi:10.1016/j.ipl.2014.05.008.

5 Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. The power and limitations of uniform samples in testing properties of figures. *Algorithmica*, 81(3):1247–1266, 2019.

6 Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. Testing figures under the uniform distribution. *Random Struct. Algorithms*, 54(3):413–443, 2019.

7 Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. Tolerant testers of image properties. *ACM Transactions on Algorithms (TALG)*, 18(4):1–39, 2022.

8 Piotr Berman, Sofya Raskhodnikova, and Grigory Yaroslavtsev. L_p-testing. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 – June 03, 2014*, pages 164–173. ACM, 2014. doi:10.1145/2591796.2591887.

9 Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM J. Comput.*, pages 1370–1379, 2005. doi:10.1137/S0097539702403244.

10 Eldar Fischer and Ilan Newman. Testing of matrix-poset properties. *Comb.*, 27(3):293–327, 2007. doi:10.1007/s00493-007-2154-3.

11 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.

12 Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002. doi:10.1007/s00453-001-0078-7.

13 Igor Kleiner, Daniel Keren, Ilan Newman, and Oren Ben-Zwi. Applying property testing to an image partitioning problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(2):256–265, 2011. doi:10.1109/TPAMI.2010.165.

14 Simon Korman, Daniel Reichman, and Gilad Tsur. Tight approximation of image matching. *CoRR*, abs/1111.1713, 2011. arXiv:1111.1713.

15 Simon Korman, Daniel Reichman, Gilad Tsur, and Shai Avidan. Fast-match: Fast affine template matching. In *CVPR*, pages 2331–2338. IEEE, 2013. doi:10.1109/CVPR.2013.302.

16 Marvin Minsky and Seymour A. Papert. *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, September 2017. doi:10.7551/mitpress/11301.001.0001.

17 Pritam Paral, Amitava Chatterjee, and Anjan Rakshit. Vision sensor-based shoe detection for human tracking in a human–robot coexisting environment: A photometric invariant approach using dbscan algorithm. *IEEE Sensors Journal*, 19(12):4549–4559, 2019.

18 Sofya Raskhodnikova. Approximate testing of visual properties. In Sanjeev Arora, Klaus Jansen, José D. P. Rolim, and Amit Sahai, editors, *RANDOM-APPROX*, volume 2764 of *Lecture Notes in Computer Science*, pages 370–381. Springer, 2003. doi:10.1007/978-3-540-45198-3_31.

19 Dana Ron and Gilad Tsur. Testing properties of sparse images. *ACM Trans. Algorithms*, 10(4):17:1–17:52, 2014. doi:10.1145/2635806.

20 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996.

21 Andrew C. Yao. Probabilistic computation, towards a unified measure of complexity. In *Proceedings of the Eighteenth Annual Symposium on Foundations of Computer Science*, pages 222–227, 1977.