

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

APPROX/RANDOM 2021, August 16–18, 2021,  
University of Washington, Seattle, Washington, US (Virtual  
Conference)

Edited by

Mary Wootters

Laura Sanità



*Editors*

**Mary Wootters** 

Stanford University, Departments of Computer Science and Electrical Engineering, CA, USA  
marykw@stanford.edu

**Laura Sanità** 

Eindhoven University of Technology, Department of Mathematics and Computer Science, The Netherlands  
l.sanita@tue.nl

*ACM Classification 2012*

Theory of computation

**ISBN 978-3-95977-207-5**

*Published online and open access by*

Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at <https://www.dagstuhl.de/dagpub/978-3-95977-207-5>.

*Publication date*

September, 2021

*Bibliographic information published by the Deutsche Nationalbibliothek*

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

*License*

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



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

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

The copyright is retained by the corresponding authors.

Digital Object Identifier: 10.4230/LIPIcs.APPROX/RANDOM.2021.0

ISBN 978-3-95977-207-5

ISSN 1868-8969

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

## LIPICs – Leibniz International Proceedings in Informatics

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

### *Editorial Board*

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

**ISSN 1868-8969**

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



# ■ Contents

Preface	
<i>Mary Wootters and Laura Sanità</i> .....	0:ix

## APPROX

On Approximate Envy-Freeness for Indivisible Chores and Mixed Resources	
<i>Umang Bhaskar, A. R. Sricharan, and Rohit Vaish</i> .....	1:1–1:23
Optimal Algorithms for Online $b$ -Matching with Variable Vertex Capacities	
<i>Susanne Albers and Sebastian Schubert</i> .....	2:1–2:18
Bag-Of-Tasks Scheduling on Related Machines	
<i>Anupam Gupta, Amit Kumar, and Sahil Singla</i> .....	3:1–3:16
Hardness of Approximation for Euclidean $k$ -Median	
<i>Anup Bhattacharya, Dishant Goyal, and Ragesh Jaiswal</i> .....	4:1–4:23
Online Directed Spanners and Steiner Forests	
<i>Elena Grigorescu, Young-San Lin, and Kent Quanrud</i> .....	5:1–5:25
Query Complexity of Global Minimum Cut	
<i>Arijit Bishnu, Arijit Ghosh, Gopinath Mishra, and Manaswi Paraashar</i> .....	6:1–6:15
A Constant-Factor Approximation for Weighted Bond Cover	
<i>Eun Jung Kim, Euiwoong Lee, and Dimitrios M. Thilikos</i> .....	7:1–7:14
Truly Asymptotic Lower Bounds for Online Vector Bin Packing	
<i>János Balogh, Ilan Reuven Cohen, Leah Epstein, and Asaf Levin</i> .....	8:1–8:18
Fine-Grained Completeness for Optimization in $P$	
<i>Karl Bringmann, Alejandro Cassis, Nick Fischer, and Marvin Künnemann</i> .....	9:1–9:22
An Estimator for Matching Size in Low Arboricity Graphs with Two Applications	
<i>Hossein Jowhari</i> .....	10:1–10:13
An Optimal Algorithm for Triangle Counting in the Stream	
<i>Rajesh Jayaram and John Kallaugher</i> .....	11:1–11:11
Matching Drivers to Riders: A Two-Stage Robust Approach	
<i>Omar El Housni, Vineet Goyal, Oussama Hanguir, and Clifford Stein</i> .....	12:1–12:22
Secretary Matching Meets Probing with Commitment	
<i>Allan Borodin, Calum MacRury, and Akash Rakheja</i> .....	13:1–13:23
Semi-Streaming Algorithms for Submodular Function Maximization Under $b$ -Matching Constraint	
<i>Chien-Chung Huang and François Sellier</i> .....	14:1–14:18
General Knapsack Problems in a Dynamic Setting	
<i>Yaron Fairstein, Ariel Kulik, Joseph (Seffi) Naor, and Danny Raz</i> .....	15:1–15:18
Min-Sum Clustering (With Outliers)	
<i>Sandip Banerjee, Rafail Ostrovsky, and Yuval Rabani</i> .....	16:1–16:16

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità



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

Streaming Approximation Resistance of Every Ordering CSP <i>Noah Singer, Madhu Sudan, and Santhoshini Velusamy</i> .....	17:1–17:19
Upper and Lower Bounds for Complete Linkage in General Metric Spaces <i>Anna Arutyunova, Anna Großwendt, Heiko Röglin, Melanie Schmidt, and Julian Wargalla</i> .....	18:1–18:22
On Two-Pass Streaming Algorithms for Maximum Bipartite Matching <i>Christian Konrad and Kheeran K. Naidu</i> .....	19:1–19:18
Approximation Algorithms for Demand Strip Packing <i>Waldo Gálvez, Fabrizio Grandoni, Afrouz Jabal Ameli, and Kamyar Khodamoradi</i>	20:1–20:24
Peak Demand Minimization via Sliced Strip Packing <i>Max A. Deppert, Klaus Jansen, Arindam Khan, Malin Rau, and Malte Tutas</i> ....	21:1–21:24
Tight Approximation Algorithms For Geometric Bin Packing with Skewed Items <i>Arindam Khan and Eklavya Sharma</i> .....	22:1–22:23
Approximating Two-Stage Stochastic Supplier Problems <i>Brian Brubach, Nathaniel Grammel, David G. Harris, Aravind Srinivasan, Leonidas Tsepenekas, and Anil Vullikanti</i> .....	23:1–23:22
Fast Approximation Algorithms for Bounded Degree and Crossing Spanning Tree Problems <i>Chandra Chekuri, Kent Quanrud, and Manuel R. Torres</i> .....	24:1–24:21
Hitting Weighted Even Cycles in Planar Graphs <i>Alexander Göke, Jochen Koenemann, Matthias Mnich, and Hao Sun</i> .....	25:1–25:23
Revenue Maximization in Transportation Networks <i>Kshipra Bhawalkar, Kostas Kollias, and Manish Purohit</i> .....	26:1–26:16
Connected $k$ -Partition of $k$ -Connected Graphs and $c$ -Claw-Free Graphs <i>Ralf Borndörfer, Katrin Casel, Davis Issac, Aikaterini Niklanovits, Stephan Schwartz, and Ziena Zeif</i> .....	27:1–27:14

## RANDOM

Better Pseudodistributions and Derandomization for Space-Bounded Computation <i>William M. Hoza</i> .....	28:1–28:23
On the Hardness of Average-Case $k$ -SUM <i>Zvika Brakerski, Noah Stephens-Davidowitz, and Vinod Vaikuntanathan</i> .....	29:1–29:19
Improved Hitting Set for Orbit of ROABPs <i>Vishwas Bhargava and Sumanta Ghosh</i> .....	30:1–30:23
A New Notion of Commutativity for the Algorithmic Lovász Local Lemma <i>David G. Harris, Fotis Iliopoulos, and Vladimir Kolmogorov</i> .....	31:1–31:25
From Coupling to Spectral Independence and Blackbox Comparison with the Down-Up Walk <i>Kuikui Liu</i> .....	32:1–32:21

Singularity of Random Integer Matrices with Large Entries <i>Sankeerth Rao Karingula and Shachar Lovett</i> .....	33:1–33:16
Interplay Between Graph Isomorphism and Earth Mover’s Distance in the Query and Communication Worlds <i>Sourav Chakraborty, Arijit Ghosh, Gopinath Mishra, and Sayantan Sen</i> .....	34:1–34:23
The Product of Gaussian Matrices Is Close to Gaussian <i>Yi Li and David P. Woodruff</i> .....	35:1–35:22
Fast Mixing via Polymers for Random Graphs with Unbounded Degree <i>Andreas Galanis, Leslie Ann Goldberg, and James Stewart</i> .....	36:1–36:13
Deterministic Approximate Counting of Polynomial Threshold Functions via a Derandomized Regularity Lemma <i>Rocco A. Servedio and Li-Yang Tan</i> .....	37:1–37:18
Improved Product-Based High-Dimensional Expanders <i>Louis Golowich</i> .....	38:1–38:17
Improved Bounds for Coloring Locally Sparse Hypergraphs <i>Fotis Iliopoulos</i> .....	39:1–39:16
Smoothed Analysis of the Condition Number Under Low-Rank Perturbations <i>Rikhav Shah and Sandeep Silwal</i> .....	40:1–40:21
Matroid Intersection: A Pseudo-Deterministic Parallel Reduction from Search to Weighted-Decision <i>Sumanta Ghosh and Rohit Gurjar</i> .....	41:1–41:16
On the Probabilistic Degree of an $n$ -Variate Boolean Function <i>Srikanth Srinivasan and S. Venkitesh</i> .....	42:1–42:20
The Swendsen-Wang Dynamics on Trees <i>Antonio Blanca, Zongchen Chen, Daniel Štefankovič, and Eric Vigoda</i> .....	43:1–43:15
Distance Estimation Between Unknown Matrices Using Sublinear Projections on Hamming Cube <i>Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra</i> .....	44:1–44:22
Decision Tree Heuristics Can Fail, Even in the Smoothed Setting <i>Guy Blanc, Jane Lange, Mingda Qiao, and Li-Yang Tan</i> .....	45:1–45:16
On the Structure of Learnability Beyond P/Poly <i>Ninad Rajgopal and Rahul Santhanam</i> .....	46:1–46:23
The Critical Mean-Field Chayes-Machta Dynamics <i>Antonio Blanca, Alistair Sinclair, and Xusheng Zhang</i> .....	47:1–47:15
On the Robust Communication Complexity of Bipartite Matching <i>Sepehr Assadi and Soheil Behnezhad</i> .....	48:1–48:17
L1 Regression with Lewis Weights Subsampling <i>Aditya Parulekar, Advait Parulekar, and Eric Price</i> .....	49:1–49:21
Hitting Sets for Orbits of Circuit Classes and Polynomial Families <i>Chandan Saha and Bhargav Thankey</i> .....	50:1–50:26

Sampling Multiple Edges Efficiently <i>Talya Eden, Saleet Mossel, and Ronitt Rubinfeld</i> .....	51:1–51:15
Lower Bounds for XOR of Forrelations <i>Uma Girish, Ran Raz, and Wei Zhan</i> .....	52:1–52:14
Fourier Growth of Structured $\mathbb{F}_2$ -Polynomials and Applications <i>Jarosław Błasiok, Peter Ivanov, Yaonan Jin, Chin Ho Lee, Rocco A. Servedio, and Emanuele Viola</i> .....	53:1–53:20
Candidate Tree Codes via Pascal Determinant Cubes <i>Inbar Ben Yaacov, Gil Cohen, and Anand Kumar Narayanan</i> .....	54:1–54:22
Towards a Decomposition-Optimal Algorithm for Counting and Sampling Arbitrary Motifs in Sublinear Time <i>Amartya Shankha Biswas, Talya Eden, and Ronitt Rubinfeld</i> .....	55:1–55:19
Ideal-Theoretic Explanation of Capacity-Achieving Decoding <i>Siddharth Bhandari, Prahladh Harsha, Mrinal Kumar, and Madhu Sudan</i> .....	56:1–56:21
Visible Rank and Codes with Locality <i>Omar Alrabiah and Venkatesan Guruswami</i> .....	57:1–57:18
Pseudorandom Generators for Read-Once Monotone Branching Programs <i>Dean Doron, Raghu Meka, Omer Reingold, Avishay Tal, and Salil Vadhan</i> .....	58:1–58:21
On the Power of Choice for $k$ -Colorability of Random Graphs <i>Varsha Dani, Diksha Gupta, and Thomas P. Hayes</i> .....	59:1–59:17
Memory-Sample Lower Bounds for Learning Parity with Noise <i>Sumegha Garg, Pravesh K. Kothari, Pengda Liu, and Ran Raz</i> .....	60:1–60:19
Testing Hamiltonicity (And Other Problems) in Minor-Free Graphs <i>Reut Levi and Nadav Shoshan</i> .....	61:1–61:23
Parallel Repetition for the GHZ Game: A Simpler Proof <i>Uma Girish, Justin Holmgren, Kunal Mittal, Ran Raz, and Wei Zhan</i> .....	62:1–62:19



## ■ Preface

This volume contains the papers presented at the 24th International Conference on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2021) and the 25th International Conference on Randomization and Computation (RANDOM 2021), which due to COVID-19 were organized as parallel virtual conferences during August 16–18, 2021. APPROX focuses on algorithmic and complexity issues surrounding the development of efficient approximate solutions to computationally difficult problems, and was the 24th in the series. RANDOM is concerned with applications of randomness to computational and combinatorial problems, and was the 25th in the series. Prior to 2003, APPROX took place in Aalborg (1998), Berkeley (1999), Saarbrücken (2000), Berkeley (2001), and Rome (2002), while RANDOM took place in Bologna (1997), Barcelona (1998), Berkeley (1999), Geneva (2000), Berkeley (2001), and Harvard (2002). Since 2003, APPROX and RANDOM have been co-located, taking place in Princeton (2003), Cambridge (2004), Berkeley (2005), Barcelona (2006), Princeton (2007), Boston (2008), Berkeley (2009), Barcelona (2010), Princeton (2011), Boston (2012), Berkeley (2013), Barcelona (2014), Princeton (2015), Paris (2016), Berkeley (2017), Princeton (2018), Boston (2019), and online (2020).

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

The volume contains 27 contributed papers, selected by the APPROX Program Committee out of 62 submissions; and 35 contributed papers, selected by the RANDOM Program Committee out of 84 submissions. We would like to thank all of the authors who submitted papers, the members of the Program Committees, and the external reviewers. We are grateful for the guidance of the steering committees: Jarosław Byrka, Klaus Jansen, Samir Khuller, Monaldo Mastrolili, and László Végh 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.





# On Approximate Envy-Freeness for Indivisible Chores and Mixed Resources

Umang Bhaskar ✉

Tata Institute of Fundamental Research, Mumbai, India

A. R. Sricharan ✉

Chennai Mathematical Institute, India

Rohit Vaish ✉

Tata Institute of Fundamental Research, Mumbai, India

---

## Abstract

We study the fair allocation of undesirable indivisible items, or *chores*. While the case of desirable indivisible items (or *goods*) is extensively studied, with many results known for different notions of fairness, less is known about the fair division of chores. We study envy-free allocation of chores and make three contributions. First, we show that determining the existence of an envy-free allocation is NP-complete even in the simple case when agents have *binary additive* valuations. Second, we provide a polynomial-time algorithm for computing an allocation that satisfies envy-freeness up to one chore (EF1), correcting a claim in the existing literature. A modification of our algorithm can be used to compute an EF1 allocation for *doubly monotone* instances (where each agent can partition the set of items into objective goods and objective chores). Our third result applies to a *mixed resources* model consisting of indivisible items and a divisible, undesirable heterogeneous resource (i.e., a bad cake). We show that there always exists an allocation that satisfies envy-freeness for mixed resources (EFM) in this setting, complementing a recent result of Bei et al. [22] for indivisible goods and divisible cake.

**2012 ACM Subject Classification** Theory of computation → Algorithmic game theory; Theory of computation → Exact and approximate computation of equilibria; Mathematics of computing → Combinatorial algorithms

**Keywords and phrases** Fair Division, Indivisible Chores, Approximate Envy-Freeness

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.1

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/abs/2012.06788> [27]

**Funding** UB and RV acknowledge support from project no. RTI4001 of the Department of Atomic Energy, Government of India. RV also acknowledges support from the Prof. R Narasimhan postdoctoral award.

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

## 1 Introduction

The problem of fairly dividing a set of resources among agents is of central importance in various fields including economics, computer science, and political science. Such problems arise in many settings such as settling border disputes, assigning credit among contributing individuals, rent division, and distributing medical supplies such as vaccines [58]. The theoretical study of fair division has classically focused on *divisible* resources (such as land or clean water), most prominently in the *cake-cutting* literature [28, 61, 60]; here, cake is a metaphor for a heterogeneous resource that can be fractionally allocated. A well-established concept of fairness in this setup is *envy-freeness* [40] which stipulates that no agent envies



© Umang Bhaskar, A. R. Sricharan, and Rohit Vaish;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 1; pp. 1:1–1:23



Leibniz International Proceedings in Informatics

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

another, i.e., prefers the share of another agent to its own. An envy-free division of a divisible, desirable resource (i.e., a cake) is known to exist under general settings [65, 4, 66, 15], and can be efficiently computed for a wide range of utility functions [37, 52, 18, 21].

By contrast, an envy-free solution can fail to exist when the goods are discrete or *indivisible*; important examples include the assignment of course seats at universities [57, 32] and the allocation of public housing units [24]. This has motivated relaxations such as *envy-freeness up to one good* (EF1) where pairwise envy can be eliminated by removing some good from the envied bundle [53, 31]. The EF1 notion enjoys strong theoretical and practical motivation: On the theoretical side, there exist efficient algorithms for computing an EF1 allocation under general, monotone valuations [53]. At the same time, EF1 has also found impressive practical appeal on the popular fair division website *Spliddit* [45] and in course allocation applications [31, 32].

Our focus in this work is on fair allocation of undesirable or negatively-valued indivisible items, also known as *chores*. The chore division problem, introduced by Martin Gardner [44], models scenarios such as distribution of household tasks (e.g., cleaning, cooking, etc.) or the allocation of responsibilities for controlling carbon emissions among countries [67]. For indivisible chores, too, an envy-free allocation could fail to exist, and one of our contributions is to show that determining the existence of such outcomes is NP-complete even under highly restrictive settings (Theorem 2). This negative result prompts us to explore the corresponding relaxation of *envy-freeness up to one chore*, also denoted by EF1, which addresses pairwise envy by removing some chore from the *envious* agent’s bundle.

At first glance, the chore division problem appears to be the “opposite” of the goods problem, and hence one might expect natural adaptation of algorithms that compute an EF1 allocation for goods to also work for chores. This, however, turns out to not be the case.

*Goods vs chores:* Consider the well-known *envy-cycle elimination* algorithm of Lipton et al. [53] for computing an EF1 allocation of indivisible goods. Briefly, the algorithm works by iteratively assigning a good to an agent that is not envied by anyone else. The existence of such an agent is guaranteed by means of resolving cycles in the underlying *envy graph*. (The envy graph of an allocation is a directed graph whose vertices correspond to the agents and there is an edge  $(i, j)$  if agent  $i$  envies agent  $j$ .) When adapted to the chores problem, the algorithm (Algorithm 1) assigns a chore to a “non-envious” agent that has no outgoing edge in the envy graph. Contrary to an existing claim in the literature [10], we observe that this algorithm could fail to find an EF1 allocation even when agents have additive valuations.<sup>1</sup>

► **Example 1 (Envy-cycle elimination algorithm fails EF1 for additive chores).** Consider the following instance with six chores  $c_1, \dots, c_6$  and three agents  $a_1, a_2, a_3$  with additive valuations:

	$c_1$	$c_2$	$c_3$	$c_4$	$c_5$	$c_6$
$a_1$	(-1)	-4	-2	(-3)	0	-1
$a_2$	-2	(-1)	-2	-2	(-3)	-1
$a_3$	-1	-3	(-1)	-1	-3	(-10)

Suppose the algorithm considers the chores in the increasing order of their indices (i.e.,  $c_1, c_2, \dots$ ), and breaks ties among agents in favor of  $a_1$  and then  $a_2$ . No directed cycles appear at any intermediate step during the execution of the algorithm on the above instance.

<sup>1</sup> A recent work by Bérczi et al. [26] shows that this algorithm fails to find an EF1 allocation when agents have non-monotone and non-additive valuations. We show a stronger result, in that the failure in finding an EF1 allocation persists even when agents have additive, monotone nonincreasing valuations.

■ **Algorithm 1** Naïve envy-cycle elimination algorithm.

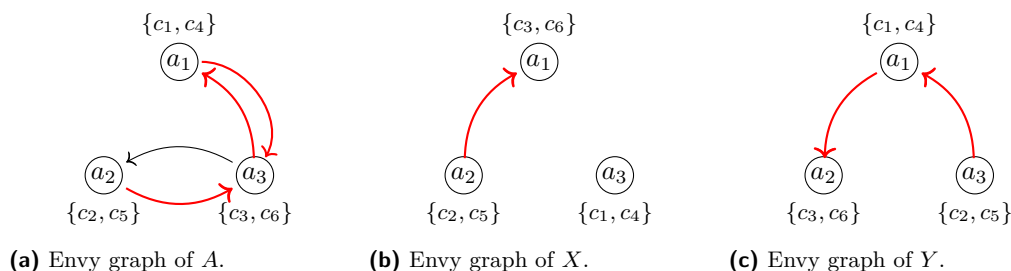
**Input:** An instance  $\langle N, M, \mathcal{V} \rangle$  with non-increasing valuations

**Output:** An allocation  $A$

```

1 Initialize  $A \leftarrow (\emptyset, \emptyset, \dots, \emptyset)$ 
2 for  $c \in M$  do
3   Choose a sink  $i$  in the envy graph  $G_A$ 
4   Update  $A_i \leftarrow A_i \cup \{c\}$ 
5   while  $G_A$  contains a directed cycle  $C$  do
6      $A \leftarrow A^C$ 
7 return  $A$ 

```



■ **Figure 1** Envy graphs of various allocations in Example 1. A red edge points to the favorite (or most envied) bundle of an agent, while a black edge points to an envied (but not the most envied) bundle.

The resulting allocation, say  $A$ , is given by  $A_1 = \{c_1, c_4\}$ ,  $A_2 = \{c_2, c_5\}$ , and  $A_3 = \{c_3, c_6\}$  (shown as circled entries in the above table). Notice that  $A$  is EF1 and its envy graph is as shown in Figure 1a.

Each node in the envy graph of  $A$  has an outgoing edge (Figure 1a). Therefore, if the algorithm were to allocate another chore after this, it would have to resolve either the cycle  $\{a_1, a_3\}$  or the cycle  $\{a_2, a_3\}$ . Let  $X$  and  $Y$  denote the allocations obtained by resolving the cycles  $\{a_1, a_3\}$  and  $\{a_2, a_3\}$ , respectively (the corresponding envy graphs are shown in Figures 1b and 1c). Although both envy graphs are *acyclic* (and thus admit a “sink” agent), only the allocation  $X$  satisfies EF1; in particular, the pair  $\{a_1, a_3\}$  violates EF1 for  $Y$ .

The above example highlights an important contrast between indivisible goods and chores: For goods, resolving arbitrary envy cycles preserves EF1, whereas for chores, the choice of which envy cycle is resolved matters. This is because when evaluating EF1 for chores, a chore is removed from the envious agent’s bundle. In the envy-cycle resolution step, if a cycle is chosen without caution, then it is possible for an agent to acquire a bundle that, although strictly more preferable, contains no chore that is large enough to compensate for the envy on its own.

A key insight of our work is that there always exists a specific envy cycle – the *top-trading envy cycle* – that can be resolved to compute an EF1 allocation of chores. Our algorithm computes EF1 allocations for *monotone* valuations, and thus provides an analogue of the result of Lipton et al. [53] for the chores setting. Furthermore, a simple modification of our algorithm computes an EF1 allocation for *doubly monotone* instances (Theorem 4), where each agent can partition the items into “objective goods” and “objective chores”, i.e., items with non-negative and negative marginal utility, respectively, for the agent [10]. This class has also been referred to as *itemwise monotone* in the literature [36].

■ **Table 1** EFM existence results for different combinations of indivisible and divisible resources. A ✓ indicates that EFM exists in that setting.

Indivisible \ Divisible	Cake	Bad Cake
Goods	✓ [22]	✓ (Theorem 16)
Chores	✓ for identical rankings (Theorem 17) ✓ for $n + 1$ items (Theorem 18)	

Motivated by this positive observation, we study a *mixed* model, consisting of both divisible as well as indivisible resources. This is a natural model in many settings, e.g., dividing an inheritance that consists of both property and money, or the simultaneous division of chores and rent among housemates. Although the use of payments in fair allocation of indivisible resources has been explored in several works [54, 3, 6, 66, 50, 55, 46, 47, 9, 30, 33], the most general formulation of a model with mixed resources, in our knowledge, is due to Bei et al. [22] who study combined allocation of a divisible *heterogeneous* resource (i.e., a cake) and a set of indivisible goods. This model and its variants are the focus of our work.

Generalizing the set of resources calls for revising the fairness benchmark. While exact envy-freeness still remains out of reach in the mixed model, EF1 can be “too permissive” when only the divisible resource is present. Bei et al. [22] propose a fairness concept called *envy-freeness for mixed goods* (EFM) for indivisible goods and divisible cake, which evaluates fairness with respect to EF1 if the envied bundle only contains indivisible goods, but switches to exact envy-freeness if the envied agent is allocated any cake. They show that an EFM allocation always exists for a mixed instance when agents have additive valuations within as well as across resource types. We note that neither the algorithm of Bei et al. [22] nor its analysis crucially depends on the valuations for the indivisible goods being additive; in fact, their results extend to monotone valuations for the indivisible goods.

We consider the problem of envy-free allocation in mixed instances with doubly monotone indivisible items and bad cake. We extend the definition of EFM naturally to this model, and show that in this model as well, an EFM allocation always exists (Theorem 16). Our work thus extends the results of Bei et al. in two ways – allowing for bad cake as well as doubly monotone indivisible items.

We also study a mixed model with indivisible chores and good cake. This turns out to be quite challenging, because unlike previous cases, one cannot start with an arbitrary EF1 allocation of the indivisible items and then allocate cake to obtain an EFM allocation. We however show the existence of an EFM allocation for two special cases in this model: when each agent has the same ranking over the chores (Theorem 17), and when the number of chores is at most one more than the number of agents (Theorem 18).

## Our Contributions

1. We first show that determining whether an envy-free allocation of chores exists is strongly NP-complete, even in the highly restricted setting when agents have *binary additive* valuations, i.e., when for all agents  $i \in [n]$  and items  $j \in [m]$ ,  $v_{i,j} \in \{-1, 0\}$  (Theorem 2). The analogous problem for indivisible goods with binary valuations is already known to be NP-complete [13, 48].

2. When the fairness goal is relaxed to envy-freeness up to one chore (EF1), we establish efficient computation for instances with chores (Theorem 3), and instances with *doubly monotone* valuations, when each agent  $i$  can partition the set of items into goods  $G_i$  (which always have nonnegative marginal value) and chores  $C_i$  (which always have nonpositive marginal value) (Theorem 4).
3. For a mixed instance consisting of doubly monotone indivisible items and bad cake, we show the existence of an allocation that satisfies the stronger fairness guarantee called *envy-freeness up to a mixed item* (EFM) (Theorem 16). Our result uses our previous theorem for indivisible chores as well as the framework of Bei et al. [22] for the allocation of the divisible item. This complements the result of Bei et al. [22] by showing existence of EFM allocations for mixed instances consisting of both desirable and undesirable items.
4. Lastly, for a mixed instance consisting of indivisible chores and (good) cake (see Section 6.4), we show the existence of an EFM allocation in two special cases: when each agent has the same preference ranking over the set of items (Theorem 17), and when the number of items is at most one more than the number of agents (Theorem 18).

Our results for mixed instances are summarized in Table 1.

## 2 Related Work

As mentioned, fair division has been classically studied for *divisible* resources. For a *heterogeneous*, desirable resource (i.e., a cake), the existence of envy-free solutions is known under mild assumptions [65, 4, 66, 15]. In addition, efficient algorithms are known for computing  $\varepsilon$ -envy-free divisions [60] and envy-free divisions under restricted preferences [37, 52, 18, 21]. For an undesirable heterogeneous resource (a bad cake), too, the existence of an envy-free division is known [59], along with a discrete and bounded procedure for finding such a division [39]. For the case of non-monotone cake (i.e., a real-valued divisible heterogeneous resource), the existence of envy-free outcomes has been shown for specific numbers of agents [62, 56, 7, 8].

Turning to the *indivisible* setting, we note that the sweeping result of Lipton et al. [53] on EF1 for indivisible *goods* has inspired considerable work on establishing stronger existence and computation guarantees in conjunction with other well-studied economic properties [34, 19, 20, 42, 25, 35, 5, 41]. The case of *indivisible chores* has been similarly well studied for a variety of solution concepts such as maximin fair share [12, 17, 14, 49], equitability [43, 2], competitive equilibria with general incomes [63], and envy-freeness [1, 29, 41].

Aziz et al. [10, 11] study a model containing both indivisible goods and chores, wherein *envy-freeness up to an item* (EF1) entails that pairwise envy is bounded by the removal of some good from the envied bundle or some chore from the envious agent's bundle. They show that a variant of the classical round-robin algorithm computes an EF1 allocation under additive utilities, and also claim that a variant of the envy-cycle elimination algorithm [53] returns such allocations for doubly monotone instances (we revisit the latter claim in Example 1). Other fairness notions such as approximate proportionality [10, 11, 16], maximin fair share [51], approximate jealousy-freeness [2], and weaker versions of EF1 [41] have also been studied in this model.

Finally, we note that the model with *mixed resources* comprising of both indivisible and (heterogeneous) divisible parts has been recently formalized by Bei et al. [22], although a special case of their model where the divisible resource is homogenous and desirable (e.g., money) has been extensively studied [54, 3, 6, 66, 50, 55, 46, 47, 9, 30, 33]. Bei et al. [22] showed that when there are indivisible goods and a divisible cake, an allocation satisfying *envy-freeness for mixed goods* (EFM) always exists. Subsequent work considers the maximin fairness notion in the mixed model [23].

### 3 Preliminaries

We consider two kinds of instances: one with purely indivisible items and the other with a mixture of divisible and indivisible items. We will present the preliminaries for instances with purely indivisible items in this section, and defer details for the mixed resources model to Section 6.

#### Problem instance

An *instance*  $\langle N, M, \mathcal{V} \rangle$  of the fair division problem is defined by a set  $N$  of  $n \in \mathbb{N}$  *agents*, a set  $M$  of  $m \in \mathbb{N}$  *indivisible items*, and a *valuation profile*  $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$  that specifies the preferences of every agent  $i \in N$  over each subset of the items in  $M$  via a *valuation function*  $v_i : 2^M \rightarrow \mathbb{R}$ .

#### Marginal valuations

For any agent  $i \in N$  and any set of items  $S \subseteq M$ , the *marginal valuation* of the set  $T \subseteq M \setminus S$  is given by  $v_i(T|S) := v_i(S \cup T) - v_i(S)$ . When the set  $T$  is a singleton (say  $T = \{j\}$ ), we will write  $v_i(j|S)$  instead of  $v_i(\{j\}|S)$  for simplicity.

#### Goods and chores

Given an agent  $i \in N$  and an item  $j \in M$ , we say that  $j$  is a *good* for agent  $i$  if for every subset  $S \subseteq M \setminus \{j\}$ ,  $v_i(j|S) \geq 0$ . We say that  $j$  is a *chore* for agent  $i$  if for every subset  $S \subseteq M \setminus \{j\}$ ,  $v_i(j|S) \leq 0$ , with one of the inequalities strict. Note that for general valuations, an item may neither be a good nor a chore for an agent.

#### Doubly monotone instances

An instance is said to be *doubly monotone* if for each agent, each item is either a good or a chore. That is, each agent  $i$  can partition the items as  $M = G_i \uplus C_i$ , where  $G_i$  are her goods, and  $C_i$  are her chores. Note that an item may be a good for one agent and a chore for another.

#### Monotone instances

A valuation function  $v$  is *monotone non-decreasing* if for any sets  $S \subseteq T \subseteq M$ , we have  $v(T) \geq v(S)$ , and *monotone non-increasing* if for any sets  $S \subseteq T \subseteq M$ , we have  $v(S) \geq v(T)$ . A *monotone goods* instance is one where all the agents have monotone non-decreasing valuations, and a *monotone chores* instance is one where all the agents have monotone non-increasing valuations. We refer to such an instance as a *monotone* if it is clear from context whether we are working with goods or chores.

#### Additive valuations

A well-studied subclass of monotone valuations is that of *additive valuations*, wherein an agent's value of any subset of items is equal to the sum of the values of individual items in the set, i.e., for any agent  $i \in N$  and any set of items  $S \subseteq M$ ,  $v_i(S) := \sum_{j \in S} v_i(\{j\})$ , where we assume that  $v_i(\emptyset) = 0$ . For simplicity, we will write  $v_i(j)$  or  $v_{i,j}$  to denote  $v_i(\{j\})$ .



### Allocation

An *allocation*  $A := (A_1, \dots, A_n)$  is an  $n$ -partition of a subset of the set of items  $M$ , where  $A_i \subseteq M$  is the *bundle* allocated to the agent  $i$  (note that  $A_i$  can be empty). An allocation is said to be *complete* if it assigns all items in  $M$ , and is called *partial* otherwise.

### Envy graph

The *envy graph*  $G_A$  of an allocation  $A$  is a directed graph on the vertex set  $N$  with a directed edge from agent  $i$  to agent  $k$  if  $v_i(A_k) > v_i(A_i)$ , i.e., if agent  $i$  prefers the bundle  $A_k$  over the bundle  $A_i$ .

### Top-trading envy graph

The *top-trading envy graph*  $T_A$  of an allocation  $A$  is a subgraph of its envy graph  $G_A$  with a directed edge from agent  $i$  to agent  $k$  if  $v_i(A_k) = \max_{j \in N} v_i(A_j)$  and  $v_i(A_k) > v_i(A_i)$ , i.e., if agent  $i$  envies agent  $k$  and  $A_k$  is the most preferred bundle for agent  $i$ .

### Cycle-swapped allocation

Given an allocation  $A$  and a directed cycle  $C$  in an envy graph or a top-trading envy graph, the *cycle-swapped allocation*  $A^C$  is obtained by reallocating bundles backwards along the cycle. For each agent  $i$  in the cycle, define  $i^+$  as the agent that she is pointing to in  $C$ . Then,  $A_i^C = A_{i^+}$  if  $i \in C$ , otherwise  $A_i^C = A_i$ .

### Envy-freeness and its relaxations

An allocation  $A$  is said to be

- *envy-free* (EF) if for every pair of agents  $i, k \in N$ , we have  $v_i(A_i) \geq v_i(A_k)$ , and
- *envy-free up to one item* (EF1) if for every pair of agents  $i, k \in N$  such that  $A_i \cup A_k \neq \emptyset$ , there exists an item  $j \in A_i \cup A_k$  such that  $v_i(A_i \setminus \{j\}) \geq v_i(A_k \setminus \{j\})$ .

## 4 Envy-Freeness for Binary Valued Chores

Our first result shows that determining the existence of an envy-free allocation is NP-complete even when agents have binary valuations, i.e., when, for all agents  $i \in N$  and items  $j \in M$ ,  $v_{i,j} \in \{-1, 0\}$  (Theorem 2). If agent valuations are not binary-valued, but are identical, the problem is still (weakly) NP-complete via a straightforward reduction from PARTITION. By contrast, our result establishes strong NP-completeness.

► **Theorem 2.** *Determining whether a given chores instance admits an envy-free allocation is NP-complete even for binary utilities.*

The proof of Theorem 2 can be found in the full version [27] of the paper.

## 5 EF1 For Doubly Monotone Instances

In light of the intractability result in the previous section, we will now explore whether one can achieve approximate envy-freeness (specifically, EF1) for indivisible chores. To this end, we note that the well-known round-robin algorithm (where, in each round, agents take turns in picking their favorite available chore) computes an EF1 allocation when agents have *additive* valuations [11]. In the following, we will provide an algorithm for computing an

■ **Algorithm 2** Top-trading envy-cycle elimination algorithm.

---

**Input:** An instance  $\langle N, M, \mathcal{V} \rangle$  with non-increasing valuations  
**Output:** An allocation  $A$

- 1 Initialize  $A \leftarrow (\emptyset, \emptyset, \dots, \emptyset)$
- 2 **for**  $c \in M$  **do**
- 3     **if** there is no sink in  $G_A$  **then**
- 4          $C \leftarrow$  any cycle in  $T_A$    ▷ if  $G_A$  has no sink, then  $T_A$  must have a cycle (Lemma 6)
- 5          $A \leftarrow A^C$
- 6     Choose a sink  $k$  in the graph  $G_A$
- 7     Update  $A_k \leftarrow A_k \cup \{c\}$
- 8 **return**  $A$

---

EF1 allocation for the much more general class of *monotone valuations*. Thus, our result establishes the analogue of the result of Lipton et al. [53] from the goods-only model for indivisible chores.

### 5.1 An Algorithm for Monotone Chores

As previously mentioned, the algorithm of Lipton et al. [53] computes an EF1 allocation for indivisible goods under monotone valuations. Recall that the algorithm works by assigning, at each step, an unassigned good to an agent who is not envied by anyone else (such an agent is a “source” agent in the underlying envy graph). The existence of such an agent is guaranteed by resolving arbitrary envy cycles in the envy graph until it becomes acyclic. Importantly, resolving an arbitrary envy cycle preserves EF1.

To design an EF1 algorithm for indivisible chores, prior work [10, 11] has proposed the following natural adaptation of this algorithm (see Algorithm 1): Instead of a “source” agent, an unassigned chore is now allocated to a “sink” (i.e., non-envious) agent in the envy graph. The existence of such an agent is once again guaranteed by means of resolving envy cycles. However, as noted in Example 1, resolving *arbitrary* envy cycles could destroy the EF1 property.

To address this gap, we propose to resolve a specific envy cycle that we call the *top-trading envy cycle*. (The nomenclature is inspired from the celebrated top-trading cycle algorithm [64] for finding a core-stable allocation that involves cyclic swaps of the most preferred objects.) Specifically, given a partial allocation  $A$ , we consider a subgraph of the envy graph  $G_A$  that we call the *top-trading envy graph*  $T_A$  whose vertices denote the agents, and an edge  $(i, k)$  denotes that agent  $i$ 's (weakly) most preferred bundle is  $A_k$ .

It is easy to observe that if the envy graph does not have a sink, then the top-trading envy graph  $T_A$  has a cycle (Lemma 6). Thus, resolving top-trading envy cycles (instead of arbitrary envy cycles) also guarantees the existence of a sink agent in the envy graph. More importantly, though, resolving a top-trading envy cycle preserves EF1. Indeed, every agent involved in the top-trading exchange receives its most preferred bundle after the swap, and therefore does not envy anyone else in the next round. The resulting algorithm is presented in Algorithm 2.

► **Theorem 3.** *For a monotone instance with indivisible chores, Algorithm 2 returns an EF1 allocation.*

In Section 5.2, we will discuss a more general result (Theorem 4) that extends the top-trading envy-cycle elimination algorithm to *doubly monotone* instances containing both indivisible goods as well as indivisible chores.

■ **Algorithm 3** An EF1 algorithm for doubly monotone indivisible instances.

---

**Input:** An instance  $\langle N, M, \mathcal{V}, \{G_i\}, \{C_i\} \rangle$  with indivisible items and doubly monotone valuations, where  $G_i$  and  $C_i$  are the set of goods and chores for agent  $i$ , respectively

**Output:** An allocation  $A$

```

1  $A \leftarrow (\emptyset, \emptyset, \dots, \emptyset)$ 
  // Goods Phase
2 for each item  $g \in \cup_i G_i$  do
3    $V^g = \{i \in N \mid g \in G_i\}$  ▷  $V^g$  contains all agents for whom  $g$  is a good
4    $G_A^g =$  the envy graph  $G_A$  restricted to the vertices  $V^g$ 
5   Choose a source  $k$  in the graph  $G_A^g$ 
6   Update  $A_k \leftarrow A_k \cup \{g\}$ 
7   while  $G_A$  contains a directed cycle  $C$  do
8      $A \leftarrow A^C$ 
  // Chores Phase
9 for each item  $c \in \cap_i C_i$  do
10  if there is no sink in  $G_A$  then
11     $C \leftarrow$  any cycle in  $T_A$  ▷ if  $G_A$  has no sink, then  $T_A$  must have a cycle
12     $A \leftarrow A^C$ 
13  Choose a sink  $k$  in the graph  $G_A$ 
14  Update  $A_k \leftarrow A_k \cup \{c\}$ 
15 return  $A$ 

```

---

## 5.2 An Algorithm for Doubly Monotone Instances

For a doubly monotone instance with indivisible items, we now give an algorithm (Algorithm 3) that returns an EF1 allocation. The algorithm runs in two phases. The first phase is for all the items that are a good for at least one agent. For these items, we run the envy-cycle elimination algorithm of Lipton et al. [53], but restricted to the subgraph of agents who consider the item a good. In the second phase, we allocate items that are chores to everybody by running the top-trading envy-cycle elimination algorithm. For a monotone chores-only instance, we recover Algorithm 2 as a special case of Algorithm 3.

► **Theorem 4.** *For a doubly monotone instance with indivisible items, Algorithm 3 returns an EF1 allocation.*

We first provide a brief sketch of the proof: At each step, we maintain the invariant that the partial allocation maintained by the algorithm is EF1. This is certainly true for the goods phase, where any envy created from agent  $i$  to agent  $j$  can always be eliminated by removing a good  $g \in A_j$  (however, unlike in the envy-cycle cancellation for goods-only instances [53], the eliminated item may not be the most recently added one since such an item could be a chore for an envious agent). In the chores phase, any new envy created by adding a chore can be removed by dropping the newly added chore. If we resolve top-trading envy cycles, then none of the agents within the cycle envy any of the agents outside it, since they now have their most preferred bundle. For any agent  $i$  outside the cycle, any envy can be removed by either removing a chore from  $i$  or a good from the envied bundle, since  $i$ 's allocation is unchanged and the bundles remain unbroken.

► **Lemma 5.** *After every step of the goods phase, the partial allocation remains EF1. Further, the goods phase terminates in polynomial time.*

## 1:10 Approximate EF for Indivisible Chores and Mixed Resources

The proof of Lemma 5 closely follows the arguments of Lipton et al. [53]; for completeness, we present a self-contained proof in Appendix A.1. We will now consider the chores phase of the algorithm, and show that if there is no sink in the envy graph  $G_A$ , then there is a cycle in the top-trading envy graph  $T_A$ .

► **Lemma 6.** *Let  $A$  be a partial allocation whose envy graph  $G_A$  does not have a sink. Then, the top-trading envy graph  $T_A$  must have a cycle. Furthermore, such a cycle can be found in polynomial time.*

**Proof.** Since  $G_A$  has no sinks, every vertex in  $G_A$  has outdegree at least one. Thus for all agents  $i$ ,  $i \notin \arg \max_k v_i(A_k)$ . So even in the top-trading envy graph  $T_A$ , each vertex has outdegree at least one. We start at an arbitrary agent and follow an outgoing edge from each successive agent. This gives us a cycle in  $T_A$ . It is easy to see that finding the cycle takes only polynomial time since we encounter each vertex at most once. ◀

We now show that resolving a cycle in the top-trading envy graph  $T_A$  gives an allocation that necessarily has a sink (the existence of such a cycle in  $T_A$  is given by Lemma 6).

► **Lemma 7.** *Let  $A$  be a partial allocation whose top-trading envy graph  $T_A$  contains a cycle. Let  $A'$  denote the allocation obtained by resolving an arbitrary cycle in  $T_A$ . Then the envy graph  $G_{A'}$  of the allocation  $A'$  must have a sink.*

**Proof.** Note that each agent points to its favorite bundle in  $T_A$ . Thus after resolving a cycle in  $T_A$ , all agents who participated in the cycle-swap now have their most preferred bundle in  $A'$  and do not envy any other agent. These agents are sinks in the graph  $G_{A'}$ . ◀

To show that the partial allocation remains EF1 throughout the chores phase, we use Lemmas 8 and 9.

► **Lemma 8.** *In the chores phase, adding a new chore to the allocation (Line 13-14) preserves EF1.*

**Proof.** Suppose at time step  $t$ , the algorithm assigns a new chore (Line 13-14). Suppose before time step  $t$ , our allocation  $A$  was EF1, and the allocation after time step  $t$  is  $A'$ . We show that  $A'$  is EF1 as well. A sink exists in the envy graph  $G_A$  at time step  $t$ , either because there were no top-trading envy cycles when we entered the loop (at Line 9) which implies the existence of a sink, or because we resolved a top-trading envy cycle  $C$  in the previous time step  $t - 1$  (Lines 10-12), in which case all the agents who were a part of the resolved top-trading envy cycle do not envy anyone after the cycle swap, and are sinks in the envy graph  $G_A$ .

Then after time  $t$ , the allocation  $A'$  will be  $A'_k = A_k \cup \{c\}$ , and  $A'_j = A_j$  for all  $j \neq k$ , where  $k$  is a sink in  $G_A$ . Pick two agents  $i$  and  $j$  such that  $i$  envies  $j$  in  $A'$ . If  $i$  did not envy  $j$  in  $A$ , then clearly  $i = k$ . In this case, removing  $c$  from  $A_i$  removes  $i$ 's envy. Suppose  $i$  envied  $j$  in  $A$  as well, and the envy was eliminated by removing  $o \in A_i \cup A_j$ . Then  $i \neq k$  since  $k$  was a sink in the graph  $G_A$ , and so  $v_i(A_i) = v_i(A'_i)$ . If  $o \in A_i$ , then  $v_i(A'_i \setminus \{o\}) \geq v_i(A_j) \geq v_i(A'_j)$ . If  $o \in A_j$ , then  $v_i(A'_i) \geq v_i(A_j \setminus \{o\}) \geq v_i(A_j \cup \{c\} \setminus \{o\})$ , since  $c$  is a chore for all agents. ◀

► **Lemma 9.** *In the chores phase, resolving a top-trading envy cycle (Lines 10-12) preserves EF1.*

**Proof.** Suppose at time step  $t$ , the algorithm resolves a top-trading envy cycle (Line 10-12). Let  $A$  be the allocation before time  $t$ ,  $C$  be the cycle along which the swap happens, and  $A' = A^C$  the allocation obtained by swapping backwards along the cycle. Pick two agents  $i$  and  $j$  such that  $i$  envies  $j$  in  $A'$ . Since every agent in the cycle obtains their favorite bundle,  $i \notin C$ . Thus  $A_i = A'_i$ . Let  $j'$  be the agent such that  $A'_j = A_{j'}$ . Since  $v_i(A'_i) = v_i(A_i)$ ,  $i$  envied  $j'$  before the swap which could be eliminated by removing  $o \in A_i \cup A_{j'}$ . If  $o \in A_i$ , then  $v_i(A'_i \setminus \{o\}) \geq v_i(A'_j)$ . If  $o \in A_{j'}$ , then  $v_i(A'_i) \geq v_i(A'_j \setminus \{o\})$ . Thus removing  $o \in A_i \cup A_{j'}$  eliminates the envy in  $A'$ . ◀

By Lemma 5, the allocation at the beginning of the chores phase is EF1. At every time step  $t$  of the chores phase, the algorithm either assigns a chore to a sink agent or resolves a top-trading envy cycle. Thus Lemma 8 and Lemma 9 together show that the allocation remains EF1 throughout the chores phase. By Lemma 6, finding a cycle in  $T_A$  takes only polynomial time. Since the while-loop executes only once for each chore, the chores phase terminates in polynomial time. This gives us the following lemma:

► **Lemma 10.** *At every step of the chores phase, the allocation remains EF1, and the chores phase terminates in polynomial time.*

The proof of Theorem 4 follows immediately, since by Lemma 10 the allocation at the end of the chores phase is EF1. Thus Algorithm 3 returns an EF1 allocation for a doubly monotone instance. Specialized to instances with only chores, we obtain Theorem 3 as a corollary.

## 6 Approximate Envy Freeness for Mixed Resources

We will now describe the setting with *mixed resources* consisting of both divisible and indivisible parts. This model was recently studied by Bei et al. [22], who introduced the notion of *envy-freeness for mixed goods* (EFM) in the context of a model consisting of indivisible goods and a divisible cake. We generalize this notion to a setting with both goods and chores.

### 6.1 Preliminaries for Instances with Divisible and Indivisible Resources

#### Mixed instance

A mixed instance  $\langle N, M, \mathcal{V}, \cdot, \mathcal{F} \rangle$  is defined by a set of  $n$  agents,  $m$  indivisible items, a valuation profile  $\mathcal{V}$  (over the indivisible items), a divisible resource represented by the interval  $[0, 1]$ , and a family  $\mathcal{F}$  of *density functions* over the divisible resource. The valuations for the indivisible items are as described in Section 3. For the divisible resource, each agent has a *density function*  $f_i : [0, 1] \rightarrow \mathbb{R}$  such that for any measurable subset  $S \subset [0, 1]$ , agent  $i$  values it at  $v_i(S) := \int_S f_i(x) dx$ . When the density function is non-negative for every agent (i.e., for all  $i \in N$ ,  $f_i : [0, 1] \rightarrow \mathbb{R}_{\geq 0}$ ), we will call the divisible resource a “cake”, and for non-positive densities (i.e., for all  $i \in N$ ,  $f_i : [0, 1] \rightarrow \mathbb{R}_{\leq 0}$ ), we will use the term “bad cake”. We do not deal with general real-valued density functions in this work.

#### Allocation

An *allocation*  $A := (A_1, \dots, A_n)$  is given by  $A_i = M_i \cup C_i$ , where  $(M_1, \dots, M_n)$  is an  $n$ -partition of the set of indivisible items  $M$ , and  $(C_1, \dots, C_n)$  is an  $n$ -partition of the divisible resource  $= [0, 1]$ . where  $A_i$  is the *bundle* allocated to the agent  $i$  (note that  $A_i$  can be empty). Given an allocation  $A$ , the *utility* of agent  $i \in N$  for the bundle  $A_i$  is  $v_i(A_i) := v_i(M_i) + v_i(C_i)$ , i.e., utility is additive across resource types.

### Perfect partition

For any  $k \in \mathbb{N}$ , a  $k$ -partition  $C = (C_1, C_2, \dots, C_k)$  of either cake or bad cake is said to be perfect if each agent values all the pieces equally, i.e., for all agents  $i \in N$  and for all pieces of the cake  $j \in [k]$ ,  $v_i(C_j) = \frac{v_i(0)}{k}$ . Note that a perfect allocation of a cake exists even when the agents' valuations are not normalized, since multiplicative scaling of agents' valuations preserves envy-freeness [4]. As in the work of Bei et al. [22], we will assume the existence of a perfect allocation oracle in our algorithmic results.

### Generalized envy graph

The *generalized envy graph*  $\overline{G}_A$  of an allocation  $A$  is a directed graph on the vertex set  $N$ , with a directed edge from agent  $i$  to agent  $k$  if  $v_i(A_k) \geq v_i(A_i)$ . If  $v_i(A_k) = v_i(A_i)$ , then we refer to the edge  $(i, k)$  as an *equality edge*, otherwise we call it an *envy edge*. A *generalized envy cycle* in this graph is a cycle  $C$  that contains at least one envy edge.

### Top-trading generalized envy graph

The *top-trading generalized envy graph*  $\overline{T}_A$  of an allocation  $A$  is a subgraph of  $\overline{G}_A$ , with a directed edge from agent  $i$  to agent  $k$  if  $i \neq k$  and  $v_i(A_k) = \max_{j \in N} v_i(A_j)$ , i.e.,  $A_k$  is one of the most preferred bundles for agent  $i$  in the allocation  $A$ . A generalized envy cycle in this graph is called a *top-trading generalized envy cycle*.

### Envy-freeness for mixed resources (EFM)

We will now discuss the notion of envy-freeness for mixed resources (EFM) that was formalized by Bei et al. [22] in the context of indivisible goods and divisible cake. Our definition extends their formulation to related settings where the indivisible part consists of chores and/or the divisible part is bad cake. The definition below is based on the following idea: Any agent who owns cake should not be envied, any agent who owns bad cake should not envy anyone else, and subject to these conditions, any pairwise envy should be EF1. Formally, an allocation  $A$  is said to be *envy-free for mixed resources* (EFM) if for any pair of agents  $i, k \in N$ , either  $i$  does not envy  $k$  (i.e.,  $v_i(A_i) \geq v_i(A_k)$ ), or all of the following hold: (a)  $i$  does not have bad cake, i.e.,  $v_i(C_i) \geq 0$ , (b)  $k$  does not have cake, i.e.,  $v_i(C_k) \leq 0$ , and (c) the envy from  $i$  to  $k$  is bounded according to EF1, i.e.,  $\exists j \in M_i \cup M_k$  such that  $v_i(A_i \setminus \{j\}) \geq v_i(A_k \setminus \{j\})$ .

## 6.2 Background: Indivisible Goods and Cake

The algorithm of Bei et al. [22] gives an EFM allocation for an instance with additive indivisible goods and cake. The algorithm initially finds an EF1 allocation of the indivisible goods using the envy-cycle elimination algorithm. It then allocates the cake in the following manner: In successive iterations, it tries to find an inclusion-wise maximal *source addable set* of agents, to which cake is then allocated.

► **Definition 11** (Source addable set). *Given a generalized envy graph, a non-empty set of agents  $S \subseteq N$  is a source addable set if (a) there is no envy edge from an agent in  $N$  to an agent in  $S$ , and (b) there is no equality edge from an agent in  $N \setminus S$  to an agent in  $S$ .*

Intuitively, to satisfy the EFM property, an agent that is envied must not get any cake, and an equality edge  $(i, j)$  implies that  $i$ 's value for the cake she gets must be at least her value for the cake that  $j$  gets.

To find a maximal source addable set of agents, the algorithm first resolves all generalized envy cycles in the generalized envy graph  $\overline{G}_A$ . It then removes all agents that are reachable from an envied agent. Bei et al. show that the remaining agents form the unique maximal source addable set. If there are  $k$  agents in this set, then the algorithm finds a perfect  $k$ -partition of the largest prefix of the cake (if  $[a, 1]$  is the remaining unallocated piece of cake, then a prefix of the cake is a piece  $[a, x]$  of the cake where  $a < x \leq 1$ ) such that giving each agent in the set a piece of this partition does not introduce envy towards any agent in the set. This continues until all the cake is allocated.

### 6.3 EFM for Doubly Monotone Indivisible Items and Bad Cake

For an instance with doubly monotone indivisible items and bad cake, we give an algorithm to obtain an EFM allocation (Algorithm 4). First, we run the doubly monotone algorithm (Algorithm 3) on the indivisible instance to obtain an EF1 allocation. We then extend it to an EFM allocation by allocating the bad cake as follows: Our algorithm always allocates prefixes of the bad cake, hence the remaining cake is always an interval  $[a, 1]$  for some  $a \geq 0$ . In each iteration, we first find an inclusion-wise maximal *sink addable set* of agents, defined analogously to the source addable set introduced earlier.

► **Definition 12** (Sink addable set). *Given a generalized envy graph, a non-empty set of agents  $S \subseteq N$  is a sink addable set if (a) there is no envy edge from an agent in  $S$  to an agent in  $N$ , and (b) there is no equality edge from an agent in  $S$  to an agent in  $N \setminus S$ .*

Since we will allocate bad cake to the agents in this set  $S$ , no agent in a sink addable set should envy another agent. Further, an equality edge  $(i, j)$  implies that if  $i$  is in the sink addable set  $S$ ,  $j$  must be in the set as well. We find a maximal sink addable set by first resolving all top-trading generalized envy cycles in the top-trading generalized envy graph  $\overline{T}_A$ , and then using the procedure in Lemma 13. The resolution of top-trading generalized envy cycles does not affect the EFM property, because of the same reasons as in the top-trading envy-cycle elimination algorithm (Section 5).

Once we find the maximal sink addable set  $S$  to which we can allocate bad cake, we need to quantify the amount of bad cake that can be allocated to the agents in  $S$  while still preserving the EFM property. We find the largest loss in utility  $\delta_i$  that an agent  $i \in S$  (who is to be given bad cake) can tolerate before she starts to envy another agent  $j \notin S$  (who is not allocated any bad cake), i.e.,

$$\delta_i = \min_{j \in N \setminus S} v_i(A_i) - v_i(A_j) \text{ for all } i \in S.$$

Note that  $\delta_i > 0$  since there are no envy or equality edges from  $S$  to  $N \setminus S$ . We then find the smallest prefix  $[a, x_{i^*}]$  of the cake, and a perfect  $|S|$ -partition of this prefix, so that if each part is allocated to an agent in  $S$ , then the utility of each agent decreases by at most  $\delta_i$ , and for a particular agent  $i^*$ , her utility goes down by exactly  $\delta_{i^*}$ . By definition of  $\delta_{i^*}$ , a new equality edge arises in the generalized envy graph  $\overline{G}_A$  from agent  $i^* \in S$  to some agent in  $N \setminus S$ . Once we allocate  $[a, x_{i^*}]$  perfectly to all agents in  $S$ , the allocation still remains EFM (Lemma 15), and we only have  $[x_{i^*}, 1]$  of the cake left to allocate. We will establish in Theorem 16 that the algorithm terminates with a polynomial number of such iterations.

We first show that the maximal sink addable set is unique if it exists (Bei et al. [22] show a similar result for source addable sets). All missing proofs can be found in the appendix.

► **Lemma 13.** *Given a partial allocation  $A$ , the maximal sink addable set (if it exists) is unique, and can be found in polynomial time.*

■ **Algorithm 4** Algorithm for EFM with doubly monotone indivisible items and bad cake.

---

**Input:** An instance  $\langle N, M, \mathcal{V}, \mathcal{F} \rangle$  with doubly monotone indivisible items  $M$ , and a divisible bad cake

**Output:** An allocation  $A$

```

1 Run the doubly monotone algorithm to obtain an EF1 allocation  $A = (A_1, A_2, \dots, A_n)$  of  $M$ 
  // Bad cake allocation phase
2 while there is still unallocated cake  $= [a, 1]$  do
3    $\bar{T}_A =$  top-trading generalized envy graph of  $A$ 
4   while there is a top-trading generalized envy cycle  $C$  in  $\bar{T}_A$  do
5      $A \leftarrow A^C$  ▷ This ensures the existence of a sink addable set
6      $S =$  maximal sink addable set for  $A$  ▷ Using Lemmas 13 and 14
7     if  $S = N$  then
8       Find an EF allocation  $(C_1, C_2, \dots, C_n)$  of
9        $A_i = A_i \cup C_i$  for all  $i \in N$ 
10       $\leftarrow \emptyset$ 
11    else
12       $\delta_i = \min_{j \in N \setminus S} v_i(A_i) - v_i(A_j)$  for all  $i \in S$ 
13      if  $v_i() \geq -|S| \cdot \delta_i$  for all  $i \in S$  then
14         $C' \leftarrow$ 
15         $\leftarrow \emptyset$ 
16      else
17         $x_i = \sup \{x \mid v_i([a, x]) \geq -|S| \cdot \delta_i\}$  for all  $i \in S$ 
18         $i^* = \arg \min_{i \in S} x_i$ 
19         $C' \leftarrow [a, x_{i^*}]$ 
20         $\leftarrow [x_{i^*}, 1]$ 
21      Obtain a perfect partition  $(C_1, C_2, \dots, C_{|S|})$  of  $C'$ 
22       $A_i \leftarrow A_i \cup C_i$  for all  $i \in S$ 
23 return  $A$ 

```

---

We now show that once we resolve all top-trading generalized envy cycles, the generalized envy graph  $\bar{G}_A$  contains a sink addable set (and thus a maximal sink addable set).

► **Lemma 14.** *If the top-trading generalized envy graph  $\bar{T}_A$  does not contain any generalized envy cycles, then the generalized envy graph  $\bar{G}_A$  has a sink addable set.*

Once we run the top-trading generalized envy cycle elimination procedure, we are thus guaranteed the existence of a sink addable set. Then, we move to the bad cake allocation procedure. The agents in the set  $S$  are then perfectly allocated a small amount of bad cake while preserving the EFM property. We now show that the partial allocation remains EFM throughout the algorithm.

► **Lemma 15.** *At each step of the bad cake allocation phase, the partial allocation in Algorithm 4 satisfies EFM.*

Finally, we will show that the algorithm terminates, assuming the existence of a perfect partition oracle.

► **Theorem 16.** *Algorithm 4 terminates after  $\mathcal{O}(n^3)$  rounds of the while-loop and returns an EFM allocation.*



## 6.4 Special Case Results with Indivisible Chores and Divisible Cake

While the approach of first allocating the indivisible resources followed by assigning the divisible resource works well for instances with indivisible goods and cake [22], and for instances with doubly monotone indivisible items and bad cake (Theorem 16), extending this approach to an instance with indivisible chores and cake is challenging for the following reason: Suppose, in such an instance, we initially allocate the indivisible chores to satisfy EF1 using the top-trading envy-cycle elimination algorithm. Then we might not be able to proceed with cake allocation, since the algorithm does not guarantee us the existence of a source in the generalized envy graph. In an effort to remedy this, we introduce the component-wise matching algorithm (refer to the full version for details [27]) to obtain an EF1 allocation of additive indivisible chores that does not have any generalized envy cycles. This algorithm ensures that the allocation at the end of indivisible chores stage is generalized envy-cycle free. However, adding even a small amount of cake might once again make the generalized envy graph sourceless, and it is unclear how to proceed at this stage.

Nevertheless, for special cases of this problem, we can prove the existence of an EFM allocation. Restricted to additive valuations of the indivisible chores, we show methods of obtaining an EFM allocation in two cases: 1) when the agents have *identical rankings* of the items (formalized in Appendix A.6), and 2) when the number of items does not exceed the number of agents by more than one, i.e.,  $m \leq n + 1$ . At a high level, the reason we are able to circumvent the aforementioned challenge in these two cases is because the particular EF1 allocation we obtain for indivisible chores in these cases has the property that we can resolve *any* generalized envy cycle that arises during the cake allocation stage. This freedom allows us to execute the algorithm of Bei et al. directly on these instances.

► **Theorem 17.** *For a mixed instance with additive indivisible chores with identical rankings and cake, an EFM allocation exists.*

► **Theorem 18.** *For a mixed instance with  $n$  agents,  $m$  additive indivisible chores and cake where  $m \leq n + 1$ , an EFM allocation exists.*

The proofs of the above two theorems can be found in Appendix A.6 and in the full version [27] of the paper respectively.

---

### References

- 1 Martin Aleksandrov. Almost Envy Freeness and Welfare Efficiency in Fair Division with Goods or Bads. *arXiv preprint arXiv:1808.00422*, 2018.
- 2 Martin Aleksandrov. Jealousy-Freeness and Other Common Properties in Fair Division of Mixed Manna. *arXiv preprint arXiv:2004.11469*, 2020.
- 3 Ahmet Alkan, Gabrielle Demange, and David Gale. Fair Allocation of Indivisible Goods and Criteria of Justice. *Econometrica: Journal of the Econometric Society*, pages 1023–1039, 1991.
- 4 Noga Alon. Splitting Necklaces. *Advances in Mathematics*, 63(3):247–253, 1987.
- 5 Georgios Amanatidis, Evangelos Markakis, and Apostolos Ntokos. Multiple Birds with One Stone: Beating 1/2 for EFX and GMMS via Envy Cycle Elimination. *Theoretical Computer Science*, 841:94–109, 2020.
- 6 Enriqueta Aragones. A Derivation of the Money Rawlsian Solution. *Social Choice and Welfare*, 12(3):267–276, 1995.
- 7 Sergey Avvakumov and Roman Karasev. Envy-Free Division Using Mapping Degree. *arXiv preprint arXiv:1907.11183*, 2019.
- 8 Sergey Avvakumov and Roman Karasev. Equipartition of a Segment. *arXiv preprint arXiv:2009.09862*, 2020.

- 9 Haris Aziz. Achieving Envy-freeness and Equitability with Monetary Transfers. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, pages 5102–5109, 2021.
- 10 Haris Aziz, Ioannis Caragiannis, Ayumi Igarashi, and Toby Walsh. Fair Allocation of Combinations of Indivisible Goods and Chores. *arXiv preprint arXiv:1807.10684 (version v3)*, 2018.
- 11 Haris Aziz, Ioannis Caragiannis, Ayumi Igarashi, and Toby Walsh. Fair Allocation of Indivisible Goods and Chores. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 53–59, 2019.
- 12 Haris Aziz, Hau Chan, and Bo Li. Weighted Maxmin Fair Share Allocation of Indivisible Chores. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 46–52, 2019.
- 13 Haris Aziz, Serge Gaspers, Simon Mackenzie, and Toby Walsh. Fair Assignment of Indivisible Objects under Ordinal Preferences. *Artificial Intelligence*, 227:71–92, 2015.
- 14 Haris Aziz, Bo Li, and Xiaowei Wu. Strategyproof and Approximately Maxmin Fair Share Allocation of Chores. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 60–66, 2019.
- 15 Haris Aziz and Simon Mackenzie. A Discrete and Bounded Envy-Free Cake Cutting Protocol for Any Number of Agents. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science*, pages 416–427, 2016.
- 16 Haris Aziz, Hervé Moulin, and Fedor Sandomirskiy. A Polynomial-Time Algorithm for Computing a Pareto Optimal and Almost Proportional Allocation. *Operations Research Letters*, 48(5):573–578, 2020.
- 17 Haris Aziz, Gerhard Rauchecker, Guido Schryen, and Toby Walsh. Algorithms for Max-Min Share Fair Allocation of Indivisible Chores. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 335–341, 2017.
- 18 Haris Aziz and Chun Ye. Cake Cutting Algorithms for Piecewise Constant and Piecewise Uniform Valuations. In *Proceedings of the 10th International Conference on Web and Internet Economics*, pages 1–14, 2014.
- 19 Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. Finding Fair and Efficient Allocations. In *Proceedings of the 19th ACM Conference on Economics and Computation*, pages 557–574, 2018.
- 20 Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. Greedy Algorithms for Maximizing Nash Social Welfare. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*, pages 7–13, 2018.
- 21 Siddharth Barman and Nidhi Rathi. Fair Cake Division Under Monotone Likelihood Ratios. In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 401–437, 2020.
- 22 Xiaohui Bei, Zihao Li, Jinyan Liu, Shengxin Liu, and Xinhang Lu. Fair Division of Mixed Divisible and Indivisible Goods. *Artificial Intelligence*, 293:103436, 2021.
- 23 Xiaohui Bei, Shengxin Liu, Xinhang Lu, and Hongao Wang. Maximin Fairness with Mixed Divisible and Indivisible Goods. *Autonomous Agents and Multi-Agent Systems*, 35(2):1–21, 2021.
- 24 Nawal Benabbou, Mithun Chakraborty, Xuan-Vinh Ho, Jakub Sliwinski, and Yair Zick. The Price of Quota-based Diversity in Assignment Problems. *ACM Transactions on Economics and Computation*, 8(3):1–32, 2020.
- 25 Nawal Benabbou, Mithun Chakraborty, Ayumi Igarashi, and Yair Zick. Finding Fair and Efficient Allocations When Valuations Don’t Add Up. In *Proceedings of the 13th International Symposium on Algorithmic Game Theory*, pages 32–46, 2020.
- 26 Kristóf Bérczi, Erika R Bérczi-Kovács, Endre Boros, Fekadu Tolessa Gedefa, Naoyuki Kamiyama, Telikepalli Kavitha, Yusuke Kobayashi, and Kazuhisa Makino. Envy-Free Relaxations for Goods, Chores, and Mixed Items. *arXiv preprint arXiv:2006.04428*, 2020.

- 27 Umang Bhaskar, AR Sricharan, and Rohit Vaish. On Approximate Envy-Freeness for Indivisible Chores and Mixed Resources. *arXiv preprint arXiv:2012.06788*, 2021.
- 28 Steven J Brams and Alan D Taylor. *Fair Division: From Cake-Cutting to Dispute Resolution*. Cambridge University Press, 1996.
- 29 Simina Brânzei and Fedor Sandomirskiy. Algorithms for Competitive Division of Chores. *arXiv preprint arXiv:1907.01766*, 2019.
- 30 Johannes Brustle, Jack Dippel, Vishnu V Narayan, Mashbat Suzuki, and Adrian Vetta. One Dollar Each Eliminates Envy. In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 23–39, 2020.
- 31 Eric Budish. The Combinatorial Assignment Problem: Approximate Competitive Equilibrium from Equal Incomes. *Journal of Political Economy*, 119(6):1061–1103, 2011.
- 32 Eric Budish, Gérard P Cachon, Judd B Kessler, and Abraham Othman. Course Match: A Large-Scale Implementation of Approximate Competitive Equilibrium from Equal Incomes for Combinatorial Allocation. *Operations Research*, 65(2):314–336, 2017.
- 33 Ioannis Caragiannis and Stavros Ioannidis. Computing Envy-Freeable Allocations with Limited Subsidies. *arXiv preprint arXiv:2002.02789*, 2020.
- 34 Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D Procaccia, Nisarg Shah, and Junxing Wang. The Unreasonable Fairness of Maximum Nash Welfare. *ACM Transactions on Economics and Computation*, 7(3):12, 2019.
- 35 Bhaskar Ray Chaudhury, Jugal Garg, and Ruta Mehta. Fair and Efficient Allocations under Subadditive Valuations. In *The 35th AAAI Conference on Artificial Intelligence*, pages 5269–5276, 2021.
- 36 Xingyu Chen and Zijie Liu. The Fairness of Leximin in Allocation of Indivisible Chores. *arXiv preprint arXiv:2005.04864*, 2020.
- 37 Yuga J Cohler, John K Lai, David C Parkes, and Ariel D Procaccia. Optimal Envy-Free Cake Cutting. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, pages 626–631, 2011.
- 38 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.
- 39 Sina Dehghani, Alireza Farhadi, MohammadTaghi HajiAghayi, and Hadi Yami. Envy-Free Chore Division For an Arbitrary Number of Agents. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2564–2583. SIAM, 2018.
- 40 Duncan Foley. Resource Allocation and the Public Sector. *Yale Economic Essays*, pages 45–98, 1967.
- 41 Rupert Freeman, Nisarg Shah, and Rohit Vaish. Best of Both Worlds: Ex-Ante and Ex-Post Fairness in Resource Allocation. In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 21–22, 2020.
- 42 Rupert Freeman, Sujoy Sikdar, Rohit Vaish, and Lirong Xia. Equitable Allocations of Indivisible Goods. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 280–286, 2019.
- 43 Rupert Freeman, Sujoy Sikdar, Rohit Vaish, and Lirong Xia. Equitable Allocations of Indivisible Chores. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 384–392, 2020.
- 44 Martin Gardner. *aha! Insight*. W. H. Freeman and Company, 1978.
- 45 Jonathan Goldman and Ariel D Procaccia. Spliddit: Unleashing Fair Division Algorithms. *ACM SIGecom Exchanges*, 13(2):41–46, 2015.
- 46 Claus-Jochen Haake, Matthias G Raith, and Francis Edward Su. Bidding for Envy-Freeness: A Procedural Approach to N-Player Fair-Division Problems. *Social Choice and Welfare*, 19(4):723–749, 2002.
- 47 Daniel Halpern and Nisarg Shah. Fair Division with Subsidy. In *Proceedings of the 12th International Symposium on Algorithmic Game Theory*, pages 374–389, 2019.

- 48 Hadi Hosseini, Sujoy Sikdar, Rohit Vaish, Hejun Wang, and Lirong Xia. Fair division through information withholding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02):2014–2021, April 2020. doi:10.1609/aaai.v34i02.5573.
- 49 Xin Huang and Pinyan Lu. An Algorithmic Framework for Approximating Maximin Share Allocation of Chores. In *The 22nd ACM Conference on Economics and Computation (forthcoming)*, 2021.
- 50 Flip Klijn. An Algorithm for Envy-Free Allocations in an Economy with Indivisible Objects and Money. *Social Choice and Welfare*, 17(2):201–215, 2000.
- 51 Rucha Kulkarni, Ruta Mehta, and Setareh Taki. Approximating Maximin Shares with Mixed Manna. In *The 22nd ACM Conference on Economics and Computation (forthcoming)*, 2021.
- 52 David Kurokawa, John K Lai, and Ariel D Procaccia. How to Cut a Cake Before the Party Ends. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, pages 555–561, 2013.
- 53 Richard J Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. On Approximately Fair Allocations of Indivisible Goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, pages 125–131, 2004.
- 54 Eric S Maskin. On the Fair Allocation of Indivisible Goods. In *Arrow and the Foundations of the Theory of Economic Policy*, pages 341–349. Palgrave Macmillan UK, 1987.
- 55 Marc Meertens, Jos Potters, and Hans Reijniere. Envy-Free and Pareto Efficient Allocations in Economies with Indivisible Goods and Money. *Mathematical Social Sciences*, 44(3):223–233, 2002.
- 56 Frédéric Meunier and Shira Zerbib. Envy-Free Cake Division Without Assuming the Players Prefer Nonempty Pieces. *Israel Journal of Mathematics*, 234(2):907–925, 2019.
- 57 Abraham Othman, Tuomas Sandholm, and Eric Budish. Finding Approximate Competitive Equilibria: Efficient and Fair Course Allocation. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 873–880, 2010.
- 58 Parag A Pathak, Tayfun Sönmez, M Utku Ünver, and M Bumin Yenmez. Fair Allocation of Vaccines, Ventilators and Antiviral Treatments: Leaving No Ethical Value Behind in Health Care Rationing. *arXiv preprint arXiv:2008.00374*, 2020.
- 59 Elisha Peterson and Francis Edward Su. N-Person Envy-Free Chore Division. *arXiv preprint arXiv:0909.0303*, 2009.
- 60 Ariel D Procaccia. Cake Cutting Algorithms. In *Handbook of Computational Social Choice, Chapter 13*. Citeseer, 2015.
- 61 Jack Robertson and William Webb. *Cake-Cutting Algorithms: Be Fair If You Can*. CRC Press, 1998.
- 62 Erel Segal-Halevi. Fairly Dividing a Cake after Some Parts were Burnt in the Oven. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1276–1284, 2018.
- 63 Erel Segal-Halevi. Competitive Equilibrium for Almost All Incomes: Existence and Fairness. *Autonomous Agents and Multi-Agent Systems*, 34(1):1–50, 2020.
- 64 Lloyd Shapley and Herbert Scarf. On Cores and Indivisibility. *Journal of Mathematical Economics*, 1(1):23–37, 1974.
- 65 Walter Stromquist. How to Cut a Cake Fairly. *The American Mathematical Monthly*, 87(8):640–644, 1980.
- 66 Francis Edward Su. Rental Harmony: Sperner’s Lemma in Fair Division. *The American Mathematical Monthly*, 106(10):930–942, 1999.
- 67 Martino Traxler. Fair Chore Division for Climate Change. *Social Theory and Practice*, 28(1):101–134, 2002.

## A Appendix

### A.1 Proof of Lemma 5

► **Lemma 5.** *After every step of the goods phase, the partial allocation remains EF1. Further, the goods phase terminates in polynomial time.*

**Proof.** Clearly the empty allocation at the beginning is EF1. Suppose before time step  $t$ , our allocation  $A$  is EF1 (i.e., any envy from agent  $i$  to agent  $j$  can be eliminated by removing an item from  $A_j$ ). Denote the allocation after time step  $t$  by  $A'$ . We will argue that  $A'$  is EF1, and any envy from agent  $i$  to agent  $j$  can be eliminated by removing an item from  $A'_j$ . At every time step, either a good is allocated or an envy cycle is resolved.

Suppose at time step  $t$ , we allocate a new item (Lines 5-6). Note that the graph  $G_A$  is acyclic at this stage. This is because it holds trivially the first time an item is allocated, and in every subsequent execution of the while-loop, we eliminate all envy cycles present (Lines 7-8) before we begin allocating the next item. Thus, the subgraph  $G_A^g$  is acyclic as well, where  $G_A^g$  is the graph  $G_A$  restricted to the agents for whom  $g$  is a good.

Then after time  $t$ , our allocation  $A'$  will be  $A'_k = A_k \cup \{g\}$ , and  $A'_j = A_j$  for all  $j \neq k$ , where  $k$  is a source in  $G_A^g$ . Pick two agents  $i$  and  $j$  such that  $i$  envies  $j$  in  $A'$ . If  $i$  did not envy  $j$  in  $A$ , then clearly  $j$  must be the agent who received the good  $g$  (i.e.,  $j = k$ ) and  $i \in V^g$ . In this case, removing  $g$  from  $A_k$  removes  $i$ 's envy as well. Suppose  $i$  envied  $j$  in  $A$  as well, and the envy was eliminated by removing  $g'$  from  $A_j$ . If  $j = k$  then  $i \notin V^g$  since  $k$  was a source in  $G_A^g$ . Then removing  $g'$  eliminates the envy in  $A'$  as well, since  $v_i(A_k \cup \{g\} \setminus \{g'\}) \leq v_i(A_k \setminus \{g'\})$ . If  $j \neq k$ , then since  $j$ 's bundle remains the same and  $v_i(A'_i) \geq v_i(A_i)$ , the envy can again be eliminated by removing  $g'$  from  $A'_j$ .

Suppose at time  $t$  we resolve an envy cycle (Lines 7-8). Let  $A$  be the allocation before time  $t$ ,  $C$  be the cycle along which the swap happens, and  $A' = A^C$  the allocation obtained by swapping backwards along the cycle. Pick two agents  $i$  and  $j$  such that  $i$  envies  $j$  in  $A'$ . Let  $i'$  and  $j'$  be the agents such that  $A'_i = A_{i'}$  and  $A'_j = A_{j'}$ . Since  $v_i(A'_i) \geq v_i(A_i)$ ,  $i$  envied  $j'$  in the allocation  $A$  before the swap. Suppose this envy was eliminated by removing  $g'$  from  $A_{j'}$ . Then  $v_i(A'_i) \geq v_i(A_i) \geq v_i(A_{j'} \setminus \{g'\})$ , and thus removing  $g'$  from  $A'_j$  eliminates the envy in  $A'$ .

To show that the algorithm terminates in polynomial time, we show that we resolve envy cycles at most a polynomial number of times for each item. Consider a single while-loop for an item, where a cycle swap occurs on the cycle  $C$ . Since the bundles remain unbroken, all agents outside the cycle have the same outdegree in  $G_{A'}$  as in  $G_A$ . An agent  $i$  inside the cycle has strictly lesser outdegree in  $G_{A'}$  compared to  $G_A$ , since the  $(i, i^+)$  edge in  $G_A$  does not translate into a  $(i, i)$  edge in  $G_{A'}$  (since  $i$  gets  $i^+$ 's bundle). Thus the number of envy edges goes down by at least  $|C|$  during each cycle swap, and the while-loop terminates in polynomial time. ◀

### A.2 Proof of Lemma 13

► **Lemma 13.** *Given a partial allocation  $A$ , the maximal sink addable set (if it exists) is unique, and can be found in polynomial time.*

**Proof.** Suppose there were two distinct maximal sink addable sets  $S_1$  and  $S_2$ . Then we show that  $S_1 \cup S_2$  is also a sink addable set, contradicting their maximality. The existence of an envy edge from an agent in  $S_1 \cup S_2$  to an agent in  $N$  contradicts either  $S_1$  or  $S_2$  being a sink addable set. Similarly, an equality edge from an agent in  $S_1 \cup S_2$  to an agent in  $N \setminus S_1 \cup S_2$  contradicts either  $S_1$  or  $S_2$  being a sink addable set. Thus  $S_1 \cup S_2$  is a sink addable set, a contradiction.

To find a maximal sink addable set, say that an agent  $i$  is an *envious* agent if there is an envy edge  $(i, j)$  in the generalized envy graph. Note that by definition a sink addable set cannot contain an envious agent. Let  $T$  be the set of all agents that have a path to an envious agent (i.e., if agent  $r$  is in  $T$ , there exists a path from  $r$  to an envious agent  $i$  in the generalized envy graph along envy and equality edges). Let  $S = N \setminus T$  be the complementary set of agents. We claim that  $S$  is a maximal sink addable set. Clearly, no agent outside  $S$  can be in any sink addable set. To see this, consider any agent  $r \notin S$ , and let  $r$  have a path to an envious agent  $i$ . By the properties of sink addable sets, if  $r$  is in  $S$ , then so must all the agents in the path including  $i$ , but this contradicts the property that a sink addable set cannot contain an envious agent. Now consider an agent  $r$  in  $S$ , and note that  $r$  does not have a path to an envious agent. Since  $r$  is not envious, it does not have an envy edge to any other agent. Further, since all agents outside  $S$  have a path to an envious agent,  $r$  cannot have an (equality or envy) edge to an outside agent. Hence, the set of agents so obtained must be a maximal sink addable set. ◀

### A.3 Proof of Lemma 14

► **Lemma 14.** *If the top-trading generalized envy graph  $\overline{T}_A$  does not contain any generalized envy cycles, then the generalized envy graph  $\overline{G}_A$  has a sink addable set.*

**Proof.** Suppose the top-trading generalized envy graph  $\overline{T}_A$  does not contain any generalized envy cycles. Then each strongly connected component  $C_i$  of the graph  $\overline{T}_A$  contains only equality edges inside it. Let  $C_1$  be a leaf component obtained by Tarjan's algorithm to find strongly connected components (see Section 22.5 of [38]). We claim that  $C_1$  is a sink addable set in the generalized envy graph  $\overline{G}_A$ .

Suppose there was an envy edge from an agent  $i$  in  $C_1$ . Since the top-trading envy graph points to an agent's favorite bundle, there would be an envy edge from  $i$  in the graph  $\overline{T}_A$  as well. Thus  $i$  would not be part of a leaf component of  $\overline{T}_A$ , contradicting that  $i \in C_1$ . Thus all agents in  $C_1$  only have equality edges in  $\overline{G}_A$ .

Suppose now that there was an equality edge from an agent  $i \in C_1$  to an agent  $j \in N \setminus C_1$ . Since  $i$  does not envy any other agent, the edge  $(i, j)$  would be present in  $\overline{T}_A$  as well, contradicting that  $C_1$  is a leaf component of  $\overline{T}_A$ . Thus  $C_1$  is a sink addable set, and thus  $\overline{G}_A$  contains a maximal sink addable set as well. ◀

### A.4 Proof of Lemma 15

► **Lemma 15.** *At each step of the bad cake allocation phase, the partial allocation in Algorithm 4 satisfies EFM.*

**Proof.** The allocation of indivisible items at the start of the algorithm is EF1 and consequently EFM as well. In the generalized top-trading envy graph, suppose we resolve a cycle  $C$ . Then the allocation remains EFM for agents outside the cycle since the bundles are unbroken. Since all agents in the cycle receive their highest valued item, they do not envy any other agent and thus satisfy EFM as well.

In the bad cake allocation stage, if  $N$  is the maximal sink addable set, then there are no envy edges inside the graph  $\overline{G}_A$  and the initial allocation is envy-free. The allocation remains envy-free on adding an EF allocation of the remaining cake to their bundles, and thus the final allocation is EFM as well.

If the maximal sink addable set  $S \subset N$  is a strict subset, then the amount of cake we allocate is chosen such that the EFM property is satisfied. Since every agent in  $S$  is given bad cake, the envy from agents in  $N \setminus S$  remains EFM. By definition of sink addable set,

none of the agents in  $S$  envy anyone before the bad cake allocation. Since we obtain a perfect allocation, none of the agents in  $S$  envy each other after the bad cake allocation as well. Note that the value of the bad cake allocated to each agent in this round is bounded below by  $\delta_i = \min_{j \in N \setminus S} v_i(A_i) - v_i(A_j)$  for all  $i \in S$ . Thus no agent in  $S$  envies an agent in  $N \setminus S$  in the final allocation as well by choice of  $\delta_i$ , and the partial allocation remains EFM throughout the algorithm.  $\blacktriangleleft$

## A.5 Proof of Theorem 16

We will break down the running time analysis into two parts: (1) The number of rounds where the algorithm resolves a top-trading generalized envy cycle, and (2) between any such consecutive rounds, the number of times when the algorithm assigns bad cake to agents in a maximal sink addable set.

Let us start with the first part. Note that assigning bad cake to a maximal sink addable set never creates new envy edges, and resolving a top-trading generalized envy cycle reduces the number of envy edges by at least one. Therefore, following the allocation of the indivisible items, the number of envy edges is a non-increasing function of time. This means that there can be at most  $\mathcal{O}(n^2)$  rounds where a top-trading generalized envy cycle is resolved by the algorithm.

Let us now consider the second part. We will argue that between any consecutive rounds where the algorithm resolves a top-trading generalized envy cycle, there can be at most  $n$  steps where the algorithm allocates bad cake. Observe that if there are no envy edges in the graph, then the maximal sink addable set is the entire set of agents (i.e.,  $S = N$ ), and the algorithm immediately terminates by allocating the entire remaining cake. Otherwise, after each round of adding bad cake, at least one new equality edge is created from  $S$  to  $N \setminus S$ . If this creates a new top-trading generalized envy cycle, then the number of envy edges strictly reduces in the next round. Else, the size of the maximal sink addable set strictly decreases in the next round, implying that there can be at most  $n$  rounds of adding bad cake before the number of envy edges strictly decreases.

Overall, we obtain that the algorithm terminates in  $\mathcal{O}(n^3)$  rounds.

► **Theorem 16.** *Algorithm 4 terminates after  $\mathcal{O}(n^3)$  rounds of the while-loop and returns an EFM allocation.*

**Proof.** By Lemma 15, the allocation returned by the algorithm is EFM. So, it suffices to show that the algorithm executes  $\mathcal{O}(n^3)$  iterations of the while-loop.

First, note that there can be at most  $\mathcal{O}(n^2)$  rounds where the algorithm resolves a top-trading generalized envy cycle. This follows from the following two observations: (a) The number of envy edges never increases during the algorithm, and (b) the number of envy edges strictly decreases by at least one whenever a top-trading generalized envy cycle is resolved. Observation (b) is straightforward. To see why (a) holds, it suffices to argue that adding bad cake does not introduce any new envy edges. Indeed, since we are adding bad cake, none of the agents in the set  $N \setminus S$  (i.e., the agents outside the maximal sink addable set) can develop new envy edges to an agent in  $S$  or  $N \setminus S$ . For each agent in  $S$ , since the amount of cake added is bounded below by  $\delta_i = \min_{j \in N \setminus S} v_i(A_i) - v_i(A_j)$  for all  $i \in S$ , none of the agents in  $S$  have new envy edges to an agent in  $N \setminus S$ . Furthermore, since the allocation is perfect, agents in  $S$  do not end up envying one another. Thus, no new envy edges are created due to the allocation of the bad cake, and therefore we have at most  $\mathcal{O}(n^2)$  rounds of top-trading generalized envy-cycle elimination.

Next, we will argue that between any consecutive cycle-elimination rounds, there can be at most  $n$  steps where the algorithm assigns bad cake to a maximal sink addable set. Note that if at any stage there are no envy edges in the generalized envy graph, then the maximal sink addable set is the entire set of agents (i.e.,  $S = N$ ), and the algorithm terminates immediately after assigning the entire remaining bad cake. So, let us assume for the remainder of the proof that there is always at least one envy edge.

If, for all agents  $i \in S$ , a perfect allocation of the remaining cake  $= [a, 1]$  preserves EFM, then the algorithm terminates in the next step. Else, we mark the point  $x_i$  on the cake for each agent such that after perfectly allocating  $[a, x_i]$ , they have a new equality edge to an agent in  $N \setminus S$ . By choice of  $i^*$  as the agent with minimum value of  $x_i$ , the agent  $i^*$  has a new equality edge to an agent  $j$  in  $N \setminus S$  after this round. If this edge creates a new top-trading generalized envy cycle, then the number of envy edges strictly reduces in the next round. Else, the size of the maximal sink addable set decreases since  $i^*$  now has a path to an envious agent and must therefore be excluded from the maximal sink addable set (additionally, no new agents are added to the sink addable set). Thus, bad cake allocation can occur for at most  $n$  consecutive rounds before the number of envy edges strictly decreases, leading to the desired  $\mathcal{O}(n^3)$  number of rounds.  $\blacktriangleleft$

## A.6 Special Case Results for EFM with Indivisible Chores and Divisible Cake

We first discuss the case of additive indivisible chores with identical rankings and cake. By identical rankings, we mean that all the agents have the same preference order on the indivisible chores, and we can order the chores as  $c_1, c_2, \dots, c_m$  such that  $v_i(c_1) \geq v_i(c_2) \geq \dots \geq v_i(c_m)$  for all agents  $i \in N$ . We assume for simplicity that the number of chores is a multiple of the number of agents. If not, we add an appropriate number of virtual chores that are valued at 0 by every agent, and remove them at the end of the algorithm. Note that this does not affect the EF1 or the EFM property. In this setting, the round-robin algorithm satisfies two desirable properties:

- Let  $(B_1, B_2, \dots, B_n)$  be a partition of the chores given by  $B_i = \{c_j \mid j \equiv i \pmod{n}\}$ . For each of the  $n!$  possible orderings of the agents, under lexicographic tiebreaking of the preferable chores, the round-robin algorithm allocates the same bundle  $B_i$  to the  $i^{\text{th}}$  agent in the ordering, and
- Resolving *any* generalized envy cycle in the allocation obtained from a round-robin instance does not violate EFM.

By lexicographic tiebreaking, we mean that if an agent has many chores of the same value to choose in round-robin, they choose the chore with the lexicographically smallest index.

► **Lemma 19.** *For an additive indivisible chores instance with identical rankings, every ordering of the agents for round-robin allocates the bundle  $B_i = \{c_j \mid j \equiv i \pmod{n}\}$  to the  $i^{\text{th}}$  agent in the ordering when tiebreaking happens lexicographically.*

**Proof.** Order the chores as  $c_1, c_2, \dots, c_m$  in non-increasing order of their value. Suppose that in the execution of the round-robin algorithm, if an agent has many chores that it has the same value for, it chooses the lexicographically smallest chore. We claim that the bundles remain the same regardless of the ordering of the agents.

Suppose the agents were ordered as  $\pi(1), \pi(2), \dots, \pi(n)$  for round-robin. In the first round of the algorithm, we claim that agent  $\pi(i)$  chooses the chore  $c_i$  during the execution. Indeed, since all the agents have the same rankings on the chores, agent  $\pi(1)$  chooses  $c_1$  in the first round (even if there were other chores with the same value,  $c_1$  is lexicographically the



smallest). Inductively, once agents  $\{\pi(1), \pi(2), \dots, \pi(i)\}$  have chosen  $\{c_1, c_2, \dots, c_i\}$ , agent  $\pi(i+1)$  weakly prefers  $c_{i+1}$  over any other chore, and adds it to their bundle because it is the lexicographically smallest chore present. Thus  $\{c_1, c_2, \dots, c_n\}$  are allocated in the first round. By a straightforward induction on the number of rounds, we see that the final allocation is  $A_{\pi(i)} = \{c_j \mid j \equiv i \pmod{n}\}$ . ◀

Note that for every position  $j$  and every agent  $i$ , there is an ordering of the agents such that agent  $i$  is in the  $j^{\text{th}}$  position during the round-robin algorithm (if  $i = j$  consider the identity permutation, else consider the permutation  $(i\ j)$ ). Since the round-robin algorithm always returns an EF1 allocation, this implies the strong property that the partition  $(B_1, B_2, \dots, B_n)$  satisfies EF1 for agent  $i$ , regardless of which bundle is allocated to that agent. Since the agent  $i$  was chosen arbitrarily at the beginning, this gives us the following lemma:

► **Lemma 20.** *For an additive indivisible chores instance with identical rankings, the allocation  $A_{\pi(i)} = B_i$  is an EF1 allocation for any ordering  $\pi$  of the agents, where  $B_i = \{c_j \mid j \equiv i \pmod{n}\}$ .*

Recall from Section 5.1 that the reason we had to restrict ourselves to resolving top-trading envy cycles when searching for an EF1 allocation of monotone indivisible chores was because resolving an arbitrary envy cycle could upset EF1 (Example 1), where an agent might obtain a bundle of higher value during the cycle swap, but the newly acquired bundle might consist of chores with low absolute value. By contrast, we can resolve *any* generalized envy cycle in the case of identical rankings, since the allocation is EF1 regardless of which agent obtains which bundle. Note that this property continues to hold even when we add cake using Bei et al.'s algorithm, since any agent with cake is never envied throughout the algorithm, and any envy present can be eliminated by the removal of one good (in fact, the same good) regardless of the identity of the agent holding a bundle (the valuation is weakly better for bundles with some portion of cake present). Thus, we have the following theorem:

► **Theorem 17.** *For a mixed instance with additive indivisible chores with identical rankings and cake, an EFM allocation exists.*

Using this result, we also show the existence of an EFM allocation when  $n - 1$  of the  $n$  agents have identical rankings. Say the agents are  $\{1, 2, \dots, n\}$ , and the agent  $n$  does not have a ranking identical to the others. Run the round-robin algorithm with the first  $n - 1$  agents and one virtual copy of one of the identical agents, say agent 1. At the end of the round-robin algorithm, allow agent  $n$  to pick their favorite bundle and arbitrarily allocate the remaining bundles between the other agents. Note that since agent  $n$  does not envy anybody at the beginning of cake allocation, and since the cake allocation stage does not create any new envy edges, the final allocation will be EFM for agent  $n$ . For all other agents, the allocation will be EF1 at the beginning of cake allocation (Lemma 20). Since resolving any generalized envy cycle still preserves EFM (Theorem 17), the final allocation will be EFM for the agents  $\{1, 2, \dots, n - 1\}$  as well. Thus we get the following corollary:

► **Corollary 21.** *For a mixed instance with additive indivisible chores with identical rankings for  $n - 1$  agents and cake, an EFM allocation exists. In particular, for two agents, an EFM allocation always exists in this setting.*

This property of being able to resolve any generalized envy cycle can also be obtained when the number of indivisible chores is at most one higher than the number of agents, i.e.,  $m \leq n + 1$ . A proof of Theorem 18 can be found in the full version [27] of the paper.



# Optimal Algorithms for Online $b$ -Matching with Variable Vertex Capacities

Susanne Albers 

Department of Computer Science, Technische Universität München, Germany

Sebastian Schubert<sup>1</sup>  

Department of Computer Science, Technische Universität München, Germany

---

## Abstract

We study the  $b$ -matching problem, which generalizes classical online matching introduced by Karp, Vazirani and Vazirani (STOC 1990). Consider a bipartite graph  $G = (S \cup R, E)$ . Every vertex  $s \in S$  is a server with a capacity  $b_s$ , indicating the number of possible matching partners. The vertices  $r \in R$  are requests that arrive online and must be matched immediately to an eligible server. The goal is to maximize the cardinality of the constructed matching. In contrast to earlier work, we study the general setting where servers may have arbitrary, individual capacities. We prove that the most natural and simple online algorithms achieve optimal competitive ratios.

As for deterministic algorithms, we give a greedy algorithm `RELATIVEBALANCE` and analyze it by extending the primal-dual framework of Devanur, Jain and Kleinberg (SODA 2013). In the area of randomized algorithms we study the celebrated `RANKING` algorithm by Karp, Vazirani and Vazirani. We prove that the original `RANKING` strategy, simply picking a random permutation of the servers, achieves an optimal competitiveness of  $1 - 1/e$ , independently of the server capacities. Hence it is not necessary to resort to a reduction, replacing every server  $s$  by  $b_s$  vertices of unit capacity and to then run `RANKING` on this graph with  $\sum_{s \in S} b_s$  vertices on the left-hand side. From a theoretical point of view our result explores the power of randomization and strictly limits the amount of required randomness. From a practical point of view it leads to more efficient allocation algorithms.

Technically, we show that the primal-dual framework of Devanur, Jain and Kleinberg cannot establish a competitiveness better than  $1/2$  for the original `RANKING` algorithm, choosing a permutation of the servers. Therefore, we formulate a new configuration LP for the  $b$ -matching problem and then conduct a primal-dual analysis. We extend this analysis approach to the vertex-weighted  $b$ -matching problem. Specifically, we show that the algorithm `PERTURBEDGREEDY` by Aggarwal, Goel, Karande and Mehta (SODA 2011), again with a sole randomization over the set of servers, is  $(1 - 1/e)$ -competitive. Together with recent work by Huang and Zhang (STOC 2020), our results demonstrate that configuration LPs can be strictly stronger than standard LPs in the analysis of more complex matching problems.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Online algorithms

**Keywords and phrases** Online algorithms, primal-dual analysis, configuration LP,  $b$ -matching, variable vertex capacities, unweighted matching, vertex-weighted matching

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.2

**Category** APPROX

**Funding** Work supported by the European Research Council, Grant Agreement No. 691672.

---

<sup>1</sup> Corresponding author

## 1 Introduction

Matching in bipartite graphs is a fundamental problem with numerous applications in computer science. We study the  $b$ -matching problem [13], where the vertices of one set of the bipartition may be matched multiple times. It generalizes the standard matching problem. Furthermore, it models capacitated allocations as well as interesting special cases of the timely AdWords problem.

More specifically, let  $G = (S \cup R, E)$  be a bipartite graph. The vertices of  $S$  are servers. Each server  $s \in S$  has an individual capacity  $b_s$ , indicating the maximum number of possible matching partners. The vertices of  $R$  are requests that have to be assigned to the servers. We consider the online problem where the set  $S$  of servers is known in advance and the requests of  $R$  arrive sequentially one by one. Whenever a new request  $r \in R$  arrives, its incident edges are revealed. The request has to be matched immediately and irrevocably to an eligible server, provided that there is one. The goal is to maximize the number of matching edges.

Prior work on  $b$ -matchings has mostly focused on the case that all servers have the same capacity, i.e.  $b_s = b$ , for all  $s \in S$ . In this paper we study the general setting of individual server capacities, as described above. This setting is particularly relevant in applications. Furthermore, we examine the *vertex-weighted* problem extension, where additionally each server  $s \in S$  has a weight  $w_s$  and the value of every matching edge incident to  $s$  is multiplied by  $w_s$ . The goal is to maximize the total weight of the constructed matching. Again this extension is interesting in allocation problems.

If  $b_s = 1$  for all  $s \in S$ , the  $b$ -matching problem is equal to classic online bipartite matching, which was introduced in a seminal paper by Karp et al. [15] and has received tremendous research interest over the last 30 years. The  $b$ -matching problem models a range of interesting applications. Naturally, the servers can be compute servers that process persistent jobs arriving over time. Furthermore, the servers can be facilities that stream content online, host web pages or store data remotely [5]. More generally, the servers can represent stations in mobile computing, queues in a network switch or even locations in a hash table [2, 5, 9]. Obviously, each server can only handle a limited number of clients.

Another relevant application are the AdWords problem and ad auctions in search engine companies [18]. There is a set of advertisers, each with a daily budget, who wish to link their ads to search keywords and issue respective bids. Queries along with their keywords arrive online and must be allocated instantly to the advertisers. The  $b$ -matching problem models the basic setting where all bids are either 0 or 1. The vertex-weighted extension captures the scenario where all the bids of an advertiser  $s \in S$  have a value of 0 or  $w_s$ .

We analyze the performance of algorithms for the  $b$ -matching problem using competitive analysis. Given an input graph  $G$ , let  $\text{ALG}(G)$  denote the size (or weight) of the matching constructed by an online algorithm  $\text{ALG}$ . Let  $\text{OPT}(G)$  be the corresponding value of an optimal offline algorithm  $\text{OPT}$ . Algorithm  $\text{ALG}$  is  $c$ -competitive if  $\text{ALG}(G) \geq c \cdot \text{OPT}(G)$  holds, for all  $G$ . If  $\text{ALG}$  is a randomized algorithm, then  $\text{ALG}(G)$  has to be replaced by the expected value  $\mathbb{E}[\text{ALG}(G)]$ .

**Related Work.** Straightforward arguments show that any algorithm that matches an incoming request to an eligible server with remaining capacity, if there exists one, is  $\frac{1}{2}$ -competitive. Kalyanasundaram and Pruhs [13] investigate the  $b$ -matching problem if all servers have equal capacity, i.e.  $b_s = b$  for all  $s \in S$ . They present a deterministic  $\text{BALANCE}$  algorithm that matches a new request to an adjacent server whose current load is smallest. Kalyanasundaram and Pruhs prove that  $\text{BALANCE}$  achieves an optimal competitive ratio

of  $1 - 1/(1 + 1/b)^b$ . As  $b$  grows, the latter expression tends to  $1 - 1/e \approx 0.63$ . Azar and Litichevsky [2] give an alternative analysis of the BALANCE algorithm. Chaudhuri et al. [5] and Grove et al. [9] study  $b$ -matchings with a different objective. At any time an algorithm must maintain a matching between the requests that have arrived so far and the servers. The goal is to minimize the total number of switches where a request is reassigned to a different server.

In a famous paper, Karp et al. [15] introduced the online bipartite matching problem. This is a  $b$ -matching problem where all servers have a capacity of 1, i.e. each vertex in the graph may be incident to at most one matching edge. Online bipartite matching has received tremendous research interest over the last years and we only mention the most important results relevant to our work. Again, any algorithm that matches an incoming request to an arbitrary available partner is  $\frac{1}{2}$ -competitive. No deterministic online algorithm can be better than  $\frac{1}{2}$ -competitive. Karp et al. [15] show that an algorithm RANDOM, which matches a request to an available partner chosen uniformly at random, does not achieve a competitiveness greater than  $1/2$ . As a main result they propose the celebrated RANKING algorithm. This strategy initially chooses a random permutation of the vertices in  $S$ . Thereby, each such vertex is assigned a priority or *rank*. Whenever a vertex of  $R$  arrives, it is matched to the eligible vertex of highest rank in  $S$ . Karp et al. prove that RANKING is  $(1 - 1/e)$ -competitive. This ratio is best possible for randomized algorithms [15].

Simplified and alternative analyses of RANKING were provided in [1, 3, 6, 7]. In particular, Devanur et al. [6] developed an elegant primal-dual analysis. Aggarwal et al. [1] defined online vertex-weighted bipartite matching, where each vertex  $s \in S$  has a weight  $w_s$ . Again, all vertices of  $S$  have a capacity of 1. The goal is to maximize the total weight of the constructed matching. Aggarwal et al. [1] devise a generalization of RANKING, named PERTURBED-GREEDY, and prove that it is  $(1 - 1/e)$ -competitive. Devanur et al. [6] analyze this strategy in their compact primal-dual framework. Further work on online bipartite matching considers different input models [8, 12, 14, 16] or refined matching models [10, 17].

The AdWords problem was formally defined by Mehta et al. [18]. They present a deterministic online algorithm that achieves a competitive ratio of  $1 - 1/e$ , under the assumption that the bids are small compared to the advertisers' budgets. No randomized algorithm can obtain a better competitive factor. Buchbinder et al. [4] develop a primal-dual algorithm that attains a competitiveness of  $(1 - 1/c)(1 - R_{\max})$ , where  $c = (1 + R_{\max})^{1/R_{\max}}$  and  $R_{\max}$  is the maximum ratio between the bid of any advertiser and its total budget. Huang et al. [11] give a 0.5016-competitive algorithm, for AdWords without the small-bids assumption.

**Our Contributions.** We present a comprehensive study of the  $b$ -matching problem with variable server capacities. As a main contribution we show that the most natural and simple online algorithms obtain optimal competitive ratios.

First, we concentrate on the unweighted setting, with the objective to maximize the cardinality of the constructed matching. In Section 2 we study deterministic algorithms. We formulate and analyze a strategy RELATIVEBALANCE that assigns an incoming request to an eligible server with minimum *relative load*. The relative load of a server  $s$  is the number requests that are currently matched with  $s$  divided by the capacity  $b_s$ . Thus the algorithm considers which fraction of a server's capacity is already used. This is the most straightforward greedy policy for the setting with variable server capacities. We show that RELATIVEBALANCE achieves a competitive ratio of  $1 - 1/(1 + 1/b_{\min})^{b_{\min}}$ , where  $b_{\min} = \min_s b_s$  is the minimum server capacity. The performance ratio is best possible for deterministic online algorithms.

In order to evaluate RELATIVEBALANCE we conduct a primal-dual analysis. We extend the framework by Devanur et al. [6], this time to analyze a deterministic algorithm different from RANKING. We remark that BALANCE by Kalyanasundaram and Pruhs [13] does not achieve a competitive ratio of  $1 - 1/(1 + 1/b_{\min})^{b_{\min}}$  when using only  $b_{\min}$  spots of each server because OPT may use the additional capacity. Moreover, we would like to add that the results by Buchbinder et al. [4] also imply a deterministic online algorithm with a competitiveness of  $1 - 1/(1 + 1/b_{\min})^{b_{\min}}$  for the  $b$ -matching problem. However, their algorithm is not equal to RELATIVEBALANCE. In fact, their strategy may assign a request to a server not having the smallest relative load and does not necessarily use the full capacity of a server, leaving requests unmatched. This leads to somewhat unintuitive assignments. We give details in Appendix A. Of course, Buchbinder et al. [4] were interested in the general AdWords problem and did not tailor their analysis to  $b$ -matchings.

In Section 3 we study randomized online algorithms. In a first step we examine the RANDOM algorithm, which assigns an incoming request to a random adjacent server with remaining capacity. We prove that the competitive factor of RANDOM is not better than  $1/2$ . The major part of Section 3 investigates the original RANKING algorithm. More specifically, RANKING initially picks a random permutation of the servers. An incoming request is matched to the eligible server of highest rank. We prove that RANKING achieves a competitive ratio of  $1 - 1/e$ , independently of the server capacities. The ratio of  $1 - 1/e$  is best possible for randomized algorithms [18]. Surprisingly, the original RANKING algorithm has an optimal competitiveness for the more complex  $b$ -matching problem. We are not aware of any other generalization of the classical online matching problem where this holds true.

Observe that we can also obtain a competitive ratio of  $1 - 1/e$  using the following reduction to standard online bipartite matching: Replace each server  $s$  with capacity  $b_s$  by exactly  $b_s$  individual vertices of capacity 1. Each request adjacent to  $s$  gets incident edges to each of these  $b_s$  vertices. On the resulting graph with  $\sum_{s \in S} b_s$  vertices on the left-hand side of the bipartition, execute the RANKING algorithm. Such a reduction can also be applied for deterministic online algorithms but only gives a competitive factor of  $1/2$ .

Our result for the original RANKING algorithm, executed on the initial input graph  $G$ , has the following implications. (1) From a theoretical point of view, an interesting question is how much randomness is needed to obtain a competitiveness of  $1 - 1/e$ . Our analysis demonstrates that a straightforward execution of the barely random RANKING strategy attains this ratio. No randomization over the server spots is necessary. (2) In practical applications a ranking of the servers leads to simple and efficient allocation algorithms. With a random permutation of a huge number of server spots, assignments might be difficult, perhaps even impossible to compute.

In our analysis we first demonstrate that the framework by Devanur et al. [6] cannot establish a competitiveness of  $1 - 1/e$  for RANKING, when executed on the original graph  $G$ . It only yields a competitiveness of  $1/2$ . Therefore, as a main technical contribution, we formulate a new configuration linear program (LP) for the  $b$ -matching problem. Using this configuration LP, we then conduct a primal-dual analysis by extending the framework of Devanur et al. [6]. We point out that, for the bipartite matchings with stochastic rewards, Huang and Zhang [10] recently were the first to employ configuration LPs. However, the concrete LPs used in [10] and in this paper are different, apart from a modeling of vertex neighborhoods. Also, the analyses differ so as to obtain the desired performance ratios.

In Section 4 we investigate vertex-weighted  $b$ -matching, with the objective to maximize the total weight of the constructed matching. We focus on randomized strategies and study PERTURBED-GREEDY [1], which was introduced for vertex-weighted online bipartite matching,

where each vertex  $s \in S$  has a capacity of 1. The algorithm, for each  $s \in S$ , computes a rank based on an initial random choice. A request is matched to the eligible vertex  $s \in S$  of highest rank. We investigate PERTURBED-GREEDY for the  $b$ -matching problem when executed on the original input graph  $G$ , without the above reduction of splitting a server  $s$  into  $b_s$  vertices of unit capacity. We extend our analysis approach based on configuration LPs and prove that the algorithm achieves an optimal competitive ratio of  $1 - 1/e$ .

In summary, simple rank-based algorithms that make initial random choices for the servers (but not for the server spots) achieve an optimal competitive ratio of  $1 - 1/e$ , independently of the server capacities. Furthermore, the paper by Huang and Zhang [10] and our work show that configuration LPs can be more powerful than standard LPs in the analysis of more advanced matching problems.

## 2 Deterministic algorithms for maximum-cardinality $b$ -matching

It is easy to verify that an online algorithm that matches a new request to an eligible server with largest remaining capacity does not achieve a competitiveness greater than  $1/2$ .

In the following we present our natural RELATIVEBALANCE algorithm. Let  $load_s$  denote the (absolute) load of a server  $s \in S$ , i.e. the number of requests assigned to  $s$  so far. We define the relative server load as  $l_s := load_s/b_s$ . RELATIVEBALANCE simply assigns incoming requests to an eligible neighbor with minimum relative server load.

### Algorithm 1 RELATIVEBALANCE.

---

```

while a new request  $r \in R$  arrives do
  Let  $N(r)$  denote the set of neighbors of  $r$  with remaining capacity;
  if  $N(r) = \emptyset$  then
    | Do not match  $r$ ;
  else
    | Match  $r$  to  $\arg \min\{l_s : s \in N(r)\}$  (break ties arbitrarily);
  end
end

```

---

We analyze RELATIVEBALANCE by conducting a primal-dual analysis. For this, consider the classical (relaxed) primal and dual LP of maximum cardinality online bipartite  $b$ -matching. Here, we use a primal variable  $m(s, r)$  for each edge  $e = \{s, r\} \in E$ , where  $s \in S$  and  $r \in R$ , indicating whether or not  $e$  is contained in the matching.

$$\begin{aligned}
 \text{Primal: } \max \quad & \sum_{\{s,r\} \in E} m(s, r) \\
 \text{s.t.} \quad & \sum_{r: \{s,r\} \in E} m(s, r) \leq b_s, & (\forall s \in S) \\
 & \sum_{s: \{s,r\} \in E} m(s, r) \leq 1, & (\forall r \in R) \\
 & m(s, r) \geq 0, & (\forall \{s, r\} \in E).
 \end{aligned}$$

$$\begin{aligned}
 \text{Dual: } \min \quad & \sum_{s \in S} b_s \cdot x(s) + \sum_{r \in R} y(r) \\
 \text{s.t.} \quad & x(s) + y(r) \geq 1, & (\forall \{s, r\} \in E) \\
 & x(s), y(r) \geq 0, & (\forall s \in S, \forall r \in R).
 \end{aligned}$$

Devanur et al. [6] developed an elegant framework that unifies the analysis of randomized online algorithms for matching problems. We will extend their framework to analyze our deterministic algorithm RELATIVEBALANCE. Whenever an online algorithm assigns a request  $r$  to a server  $s$ , the gain of 1 in the primal objective function (and thus the size of the matching) is translated into a gain of  $1/c$  in the dual objective function by splitting it across the dual variables  $x(s)$  and  $y(r)$ . Here,  $c$  is a constant that will be maximized during the analysis and will denote the competitive ratio of the algorithm,  $0 < c \leq 1$ . If an arriving request remains unmatched, the dual solution will remain unchanged as well.

It then has to be shown that this can be done in a way such that all the dual constraints are satisfied in the end. Let  $P$  and  $D$  be the value of the constructed primal and dual solution, respectively. By summing over all steps of the algorithm, we get  $P = c \cdot D$ , and thus  $P \geq c \cdot \text{OPT}$ , by weak duality. This implies that the online algorithm is  $c$ -competitive.

In the case without vertex capacities, Devanur et al. [6] show that for the known optimal randomized online algorithms that choose a random value  $x_s \in [0, 1]$  for every server  $s \in S$ , the gain of matching a request  $r$  to  $s$  can be split across  $x(s)$  and  $y(r)$  according to the function  $g(x_s) = e^{x_s - 1}$ . More precisely, in the unweighted scenario, they argue that setting

$$x(s) = \frac{g(x_s)}{c} \quad \text{and} \quad y(r) = \frac{1 - g(x_s)}{c}$$

with  $c = 1 - 1/e$  results in a dual solution that is feasible in expectation.

In our case with vertex capacities, we first have to deal with the fact that a server  $s$  may be assigned multiple requests. Therefore, we increase the value of  $x(s)$  whenever this happens. Moreover, we change the function that determines how the gain is split. Our algorithm uses the relative load of the servers for its matching decisions instead of a ranking based on the random values. Therefore, whenever RELATIVEBALANCE matches a request  $r$  to a server  $s$ , we update

$$\Delta x(s) = \frac{f(l_s)}{c \cdot b_s} \quad \text{and} \quad y(r) = \frac{1 - f(l_s)}{c},$$

where  $f : [0, 1] \rightarrow [0, 1]$  is a monotonically non-decreasing function and  $l_s$  denotes the relative load of  $s$  before the assignment. Observe that this increases the value of the dual solution by exactly  $1/c$  and guarantees  $x(s) \geq 0$  and  $y(r) \geq 0$  for all  $s \in S$  and  $r \in R$ , respectively.

Now, we have to show that  $f$  and  $c$  can be chosen such that this results in a feasible dual solution, i.e.  $x(s) + y(r) \geq 1$  holds for all edges  $\{s, r\} \in E$ . If  $r$  is not matched by RELATIVEBALANCE, then  $y(r) = 0$ . Nonetheless, we know that all of  $r$ 's neighbors had to be fully loaded when  $r$  arrived. Thus, in this case, for all  $b_s$ , we need that

$$x(s) + y(r) = \frac{1}{c \cdot b_s} \sum_{i=0}^{b_s-1} f\left(\frac{i}{b_s}\right) \geq 1. \quad (1)$$

On the other hand, if  $r$  is matched to a server  $s'$  by RELATIVEBALANCE, then we know that  $l_{s'} \leq l_s$  had to hold at the time of  $r$ 's arrival. In this case, it therefore needs to hold that

$$x(s) + y(r) = \frac{1}{c} \left( \frac{1}{b_s} \sum_{i=0}^{load_{s'}-1} f\left(\frac{i}{b_s}\right) + 1 - f\left(\frac{load_{s'}}{b_{s'}}\right) \right) \geq 1, \quad (2)$$

for all ratios  $load_{s'}/b_{s'} \leq load_s/b_s$ . Recall that  $load_{s'}$  and  $load_s$  are absolute server loads.

▷ **Claim 1.** Let  $c := 1 - 1/d$ , where  $d > 1$ . Then,  $f(l_s) = d^{l_s-1}$  satisfies both (1) and (2) if  $d \leq (1 + 1/b_s)^{b_s}$ .



Proof. First, observe that  $d \leq (1 + 1/b_s)^{b_s}$  implies  $d^{\frac{1}{b_s}} - 1 \leq 1/b_s$ . It then follows that

$$\frac{1}{c \cdot b_s} \sum_{i=0}^{b_s-1} f\left(\frac{i}{b_s}\right) = \frac{1}{c \cdot b_s \cdot d} \sum_{i=0}^{b_s-1} \left(d^{\frac{1}{b_s}}\right)^i = \frac{1}{c \cdot b_s \cdot d} \cdot \frac{d-1}{d^{\frac{1}{b_s}}-1} \geq \frac{d-1}{c \cdot b_s \cdot d \cdot \frac{1}{b_s}} = 1.$$

The last step follows from the choice of  $c$ . Moreover, we can show that

$$\begin{aligned} \frac{1}{c} \left( \frac{1}{b_s} \sum_{i=0}^{load_s-1} f\left(\frac{i}{b_s}\right) + 1 - f\left(\frac{load_s}{b_s}\right) \right) &= \frac{1}{c} \left( \frac{1}{b_s \cdot d} \sum_{i=0}^{load_s-1} \left(d^{\frac{1}{b_s}}\right)^i + 1 - d^{load_s/b_s} \right) \\ &= \frac{1}{c} \left( \frac{1}{b_s \cdot d} \cdot \frac{d^{load_s/b_s} - 1}{d^{\frac{1}{b_s}} - 1} + 1 - d^{load_s/b_s} \right) \geq \frac{1}{c} \left( \frac{d^{load_s/b_s} - 1}{d} + 1 - d^{load_s/b_s} \right) \geq \frac{1}{c} \left( 1 - \frac{1}{d} \right) = 1. \quad \triangleleft \end{aligned}$$

We have now shown that the combination of  $f(l_s) := d^{l_s-1}$  with  $c = 1 - 1/d$  yields a feasible dual solution if  $1 < d \leq (1 + 1/b_s)^{b_s}$ , for all  $s \in S$ . Here  $c$  denotes the competitiveness of RELATIVEBALANCE.  $(1 + 1/b_s)^{b_s}$  is a monotonically increasing function for  $b_s > 0$ . The largest possible value for  $d$  is therefore  $(1 + 1/b_{\min})^{b_{\min}}$ , where  $b_{\min} = \min_s b_s$  is the smallest server capacity.

► **Theorem 2.** RELATIVEBALANCE achieves a competitiveness of  $1 - 1/(1 + 1/b_{\min})^{b_{\min}}$ , where  $b_{\min} := \min_{s \in S} b_s$ .

The competitive ratio of  $1 - 1/(1 + 1/b_{\min})^{b_{\min}}$  is optimal for deterministic algorithms: Kalyanasudaram and Pruhs [13] showed that no deterministic online algorithm can achieve a competitiveness greater than  $1 - 1/(1 + 1/b)^b$  if all servers have a uniform capacity equal to  $b$ . We can take their nemesis sequence and add servers with capacity  $b' > b$  that are adjacent to few (or no) extra requests.

### 3 Randomized algorithms for maximum-cardinality $b$ -matching

Karp et al. [15] proposed an algorithm RANDOM, for online bipartite matching, which assigns a newly arriving request to a random eligible neighbor. They showed that RANDOM is not better than  $\frac{1}{2}$ -competitive. We prove that the performance ratio does not improve, for the  $b$ -matching problem, even if all servers have a uniform capacity of  $b \geq 2$ . The material on RANDOM with the proof of the following theorem is given in Appendix B.

► **Theorem 3.** RANDOM does not achieve a competitive ratio better than  $1/2$  for the maximum cardinality online  $b$ -matching problem, even if all server capacities are equal.

The remainder of this section is devoted to the RANKING algorithm. We will prove that the algorithm achieves an optimal competitiveness of  $1 - 1/e$ , for the maximum cardinality online  $b$ -matching problem. Again, we execute RANKING on the original input graph  $G$ . We will work with a version of RANKING (see Alg. 2) that is similar to that in [6]. Note that, importantly, there is a *single* random choice for each server  $s \in S$ . Initially, a  $Z_s \in [0, 1]$  is picked uniformly at random. This value is used as a rank for  $s$ . An incoming request is matched to the eligible server with smallest  $Z$ -value.

First, we argue that the classical primal-dual framework fails here, meaning that it is not able to establish a competitive ratio better than  $1/2$ . As usual, whenever RANKING assigns a request  $r$  to a server  $s$ , we increase  $x(s)$  by and set  $y(r)$  to

$$\Delta x(s) = \frac{g(Z_s)}{c \cdot b_s} \text{ and } y(r) = \frac{1 - g(Z_s)}{c},$$

---

**Algorithm 2** RANKING.
 

---

```

foreach server  $s \in S$  do
  | Pick  $Z_s \in [0, 1]$  uniformly at random;
end
while a new request  $r \in R$  arrives do
  | Let  $N(r)$  denote the set of neighbors of  $r$  with remaining capacity;
  | if  $N(r) = \emptyset$  then
  | | Do not match  $r$ ;
  | else
  | | Match  $r$  to  $\arg \min \{Z_s : s \in N(r)\}$  (break ties consistently);
  | end
end

```

---

respectively. Again,  $g : [0, 1] \rightarrow [0, 1]$  is a monotonically non-decreasing function and  $c$  is a constant that will denote the competitive ratio of the algorithm. Let  $P$  and  $D$  be random variables denoting the value of the random primal and dual solution, respectively. If we were able to show that a combination of  $g$  and  $c$  yields a dual solution that is feasible in expectation, then this would imply a competitive ratio of  $c$ . To see this, create a new (deterministic) dual solution that sets its variables to the expected value of the corresponding variable of the random solution and denote its value by  $D'$ . It then holds that the new dual solution satisfies all dual constraints and thus  $\text{OPT} \leq D' = \mathbb{E}[D]$ . Moreover, the framework yields  $P = c \cdot D$ , always, implying  $\mathbb{E}[P] = c \cdot \mathbb{E}[D] \geq c \cdot \text{OPT}$ .

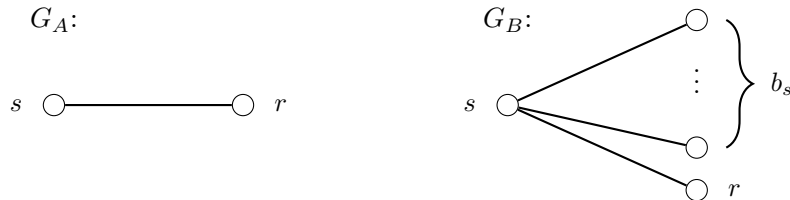
Now, consider the two input graphs  $G_A$  and  $G_B$  (see Fig. 1). If there was a combination of  $g$  and  $c$  that always yields a dual solution that is feasible in expectation, then this combination also has to satisfy the constraint for the edge  $\{s, r\}$  in both  $G_A$  and  $G_B$  in expectation. In graph  $G_A$ , this means

$$\mathbb{E}[x(s) + y(r)] = \frac{\mathbb{E}[g(Z_s)]}{c \cdot b_s} + \frac{1 - \mathbb{E}[g(Z_s)]}{c} \stackrel{!}{\geq} 1 \iff \mathbb{E}[g(Z_s)] \leq (1 - c) \cdot \frac{b_s}{b_s - 1}.$$

In  $G_B$  however, this means

$$\mathbb{E}[x(s) + y(r)] = b_s \cdot \frac{\mathbb{E}[g(Z_s)]}{c \cdot b_s} + 0 \stackrel{!}{\geq} 1 \iff \mathbb{E}[g(Z_s)] \geq c.$$

Combining these two inequalities yields  $c \leq (1 - c) \cdot \frac{b_s}{b_s - 1}$ . This is equivalent to  $c \leq \frac{b_s}{2b_s - 1}$ , which implies that the best competitive ratio that may be shown for RANKING with this framework approaches  $1/2$  for larger server capacities  $b_s$ .



**Figure 1** Two example input graphs for RANKING. Graph  $G_A$  only consists of a single edge between a server  $s$  and a request  $r$ , whereas  $G_B$  consists of a server  $s$  and its  $b_s + 1$  neighboring requests. Request  $r$  denotes the last arriving request.

Given this fact, we proceed and model the  $b$ -matching problem by a configuration LP. Let  $N_s$  denote the set of neighbors of a server  $s$ . The configuration LP differs from the classical matching LP in that it does not use a variable for every edge  $\{s, r\}$  indicating whether this edge is chosen by the algorithm. Instead it uses a variable  $m(s, N)$ , for every server  $s$  and every subset  $N \subseteq N_s$ , indicating whether this subset is the set of requests matched to  $s$ .

$$\begin{aligned}
 \text{Config LP: } & \max \sum_{s \in S} \sum_{N \subseteq N_s} \min\{|N|, b_s\} \cdot m(s, N) \\
 \text{s.t. } & \sum_{N \subseteq N_s} m(s, N) \leq 1, & (\forall s \in S) \\
 & \sum_{s \in S} \sum_{N \subseteq N_s: r \in N} m(s, N) \leq 1, & (\forall r \in R) \\
 & m(s, N) \geq 0, & (\forall s \in S, \forall N \subseteq N_s).
 \end{aligned}$$

$$\begin{aligned}
 \text{Dual CLP: } & \min \sum_{s \in S} x(s) + \sum_{r \in R} y(r) \\
 \text{s.t. } & x(s) + \sum_{r \in N} y(r) \geq \min\{|N|, b_s\}, & (\forall s \in S, \forall N \subseteq N_s) \\
 & x(s), y(r) \geq 0, & (\forall s \in S, \forall r \in R).
 \end{aligned}$$

Obviously, every valid  $b$ -matching in a graph  $G$  is captured by a solution of the configuration LP. Its optimal solution is an upper bound on the cardinality of the maximum  $b$ -matching in  $G$ . Hence the configuration LP is a suitable primal program for a primal-dual analysis.

We adapt the primal-dual analysis framework. Initially, all primal and dual variables are set to 0. Whenever a new request  $r \in R$  arrives and RANKING assigns it to a server  $s$ , we update the primal variables of  $s$  accordingly, keeping track of the set  $N$  of matching partners. The value of the primal solution increases by 1. Moreover, we update the dual variables

$$\Delta x(s) = \frac{g(Z_s)}{c} \text{ and } y(r) = \frac{1 - g(Z_s)}{c},$$

where  $g : [0, 1] \rightarrow [0, 1]$  is a monotonically non-decreasing function to be determined during the analysis and  $c$  is a constant that will denote the competitive ratio of the algorithm. Note that we now do not have to divide the gain of  $x(s)$  by  $b_s$ , since the dual objective function does not have a factor  $b_s$  before  $x(s)$ . Therefore, we still translate a gain of 1 in the primal solution to a gain of  $1/c$  in the dual solution, guaranteeing that  $P = c \cdot D$ , where  $P$  and  $D$  are the random variables denoting the value of the primal and dual solution, respectively. Similar arguments to before imply that it is sufficient to satisfy all dual constraints in expectation to show a competitive ratio of  $c$ .

Thus, it remains to show is that we can choose  $g$  and  $c$  such that

$$\mathbb{E} \left[ x(s) + \sum_{r \in N} y(r) \right] \geq \min\{|N|, b_s\},$$

for all servers  $s \in S$  and all  $N \subseteq N_s$ . For this, we will need two lemmas similar to the Dominance and Monotonicity Lemmas in [6]. We will consider two executions of RANKING on  $G$  and on  $G \setminus s$ , for some server  $s \in S$ . Here,  $G \setminus s$  denotes the graph induced by the vertex set  $S \setminus \{s\} \cup R$ . We assume that RANKING uses the same  $Z$ -values  $Z_t$  for all servers  $t \in S \setminus \{s\}$  in both executions. Further, let  $r \in R$  be any request in  $G$  and let  $z_r$  be the

$Z$ -value of its matching partner in the  $G \setminus s$  execution. If  $r$  is unmatched, we set  $z_r := 1$  and assign a dummy matching partner. Moreover, let  $y_s(r)$  be the value of  $y(r)$  in the  $G \setminus s$  execution. We impose from now on that  $g(1) = 1$ , which implies  $y_s(r) = (1 - g(z_r))/c$ . We use the idea of *server spots*. A server spot  $s_i$  of a server  $s$ ,  $1 \leq i \leq b_s$ , denotes an individual unit of a server that can accept a request. When RANKING assigns requests to a server  $s$ , we assume without loss of generality that it assigns the  $j$ -th request to the server spot  $s_j$ . A server spot is matched if it has been assigned a request, and unmatched otherwise.

► **Lemma 4.** *At any point during the parallel execution of RANKING on  $G$  and  $G \setminus s$ , the set of unmatched server spots  $U$  in the  $G$  execution forms a superset of the unmatched server spots  $\tilde{U}$  in the  $G \setminus s$  execution. For all server spots  $s'_i \in U \setminus \tilde{U}$ , it holds that  $Z_{s'_i} \geq Z_s$ . If  $Z_{s'} = Z_s$  and  $s' \neq s$ , then  $s$  has a higher priority in the tiebreaking.*

**Proof.** By induction. Initially, the properties trivially hold, since  $U \setminus \tilde{U} = \{s_1, \dots, s_{b_s}\}$  at the start. Then, whenever a new request  $r$  arrives,  $\tilde{U} \subseteq U$  can only be violated if  $r$  is assigned to a server spot  $t_i \in \tilde{U}$  in the  $G$  execution, but  $r$  is not assigned to  $t_i$  in the  $G \setminus s$  execution. There, it is either unmatched or matched to a different server spot, which leads to a contradiction in either case. Since  $t_i \in \tilde{U}$  and  $r$  is a neighbor of the server  $t$ ,  $r$  cannot be unmatched in the  $G \setminus s$  execution. If RANKING chooses a different server spot  $t_j$  for  $r$  in the  $G \setminus s$  execution, then either  $i < j$  or  $i > j$  has to hold.  $i < j$  results in a contradiction because  $t_i \in \tilde{U}$  and we defined that RANKING always chooses the unmatched server spot with smallest index. Furthermore,  $i > j$  also results in a contradiction because  $t_j \in \tilde{U} \subseteq U$  and thus RANKING would have chosen  $t_j$  in the  $G$  execution as well. Moreover, if RANKING assigns  $r$  to a server spot of a different server  $t'$  in the  $G \setminus s$  execution, then  $Z_{t'} \leq Z_t$  has to hold. However,  $\tilde{U} \subseteq U$  implies that this server spot would also be unmatched and available in the  $G$  execution. If  $Z_{t'} < Z_t$ , RANKING would not have chosen an unmatched neighbor with smallest  $Z$ -value in the  $G$  execution, and if  $Z_{t'} = Z_t$ , then the tiebreak would be inconsistent between the two executions.

Moreover, a new server spot  $t'_i$  is only added to  $U \setminus \tilde{U}$  if the matching decision for  $r$  is different in the two execution, i.e. the  $G$  execution assigns  $r$  to some server spot  $t_j \in U \setminus \tilde{U}$  and the  $G \setminus s$  execution assigns  $r$  to  $t'_i \in \tilde{U}$ . Therefore, it has to hold that either  $Z_{t'} > Z_t$  or  $Z_{t'} = Z_t$  and  $t$  has a higher tiebreak priority than  $t'$ . By induction hypothesis,  $Z_t \geq Z_s$  and thus  $Z_{t'} \geq Z_s$ . If  $t' \neq s$  and  $Z_{t'} = Z_t = Z_s$ , then by induction hypothesis  $s$  has a higher tiebreak priority than  $t$ , which in turn has a higher priority than  $t'$ . We conclude that  $s$  has a higher tiebreak priority than  $t'$ . ◀

Hence, if a request  $r$  is unmatched in the  $G$  execution, it is also unmatched in the  $G \setminus s$  execution. If  $r$  gets matched, its matching partner has a  $Z$ -value of at most  $z_r$ . Since  $g$  is non-decreasing with  $g(1) = 1$ , the following statement holds.

► **Corollary 5.** *Given  $Z_t$  for all servers  $t \in S \setminus \{s\}$ ,  $y(r) \geq y_s(r)$  holds for all possible values of  $Z_s$ .*

► **Lemma 6.** *Given  $Z_t$  for all servers  $t \in S \setminus \{s\}$ , let  $z_1 \geq \dots \geq z_k$  be the  $Z$ -values of the matching partners of the  $k = |N_s|$  neighbors of  $s$  in a  $G \setminus s$  execution in non-increasing order. Then, server  $s$  has at least  $\min\{a, b_s\}$  matching partners in an execution of RANKING on  $G$ , where  $a$  is the largest possible integer such that  $Z_s < z_a \leq \dots \leq z_1$ .*

**Proof.** Whenever a neighbor  $r_i$  of  $s$  with  $z_i > Z_s$  arrives and  $s$  still has remaining capacity, then by Lemma 4  $r_i$  will be matched to  $s$ . Among adjacent servers with remaining capacity,  $s$  has the smallest  $Z$ -value and the highest priority in case of ties. ◀

Now, we can finally show how to choose  $g$  and  $c$  such that the dual constraints are satisfied in expectation. Let  $s$  be any server in  $G$  with  $k$  neighbors. Let  $z_i$  be the  $Z$ -value of the matching partner of neighbor  $r_i \in N_s$ ,  $1 \leq i \leq k$ , in the  $G \setminus s$  execution. If  $k < b_s$ , we further define  $z_{k+1} = \dots = z_{b_s} = 0$ . Let  $z'_1 \geq \dots \geq z'_{b_s}$  then be the  $b_s$  largest values of  $\{z_1, \dots, z_{\max\{k, b_s\}}\}$  in non-increasing order. Lemma 6 implies that

$$\mathbb{E} \left[ x(s) \left| \bigwedge_{t \in S \setminus \{s\}} Z_t = z_t \right. \right] \geq \sum_{i=1}^{b_s} \int_0^{z'_i} \frac{g(t)}{c} dt.$$

Moreover, by Corollary 5, it holds for every neighbor  $r \in N_s$  of  $s$

$$\mathbb{E} \left[ y(r) \left| \bigwedge_{t \in S \setminus \{s\}} Z_t = z_t \right. \right] \geq y_s(r) = \frac{1 - g(z_r)}{c},$$

where  $z_r = z_i$  for some  $i$ ,  $1 \leq i \leq k$ . Putting everything together yields

$$\mathbb{E} \left[ x(s) + \sum_{r \in N} y(r) \left| \bigwedge_{t \in S \setminus \{s\}} Z_t = z_t \right. \right] \geq \frac{1}{c} \left( \sum_{i=1}^{b_s} \int_0^{z'_i} g(t) dt + \sum_{r \in N} (1 - g(z_r)) \right).$$

Note that  $\sum_{r \in N} (1 - g(z_r))$  is lower bounded by  $\sum_{i=1}^{\min\{|N|, b_s\}} (1 - g(z'_i))$ , since  $g$  is a non-decreasing function with  $g(1) = 1$  and the  $z'$ -values are an upper bound for the  $z_r$ -values. Plugging this in, we get

$$\begin{aligned} \mathbb{E} \left[ x(s) + \sum_{r \in N} y(r) \left| \bigwedge_{t \in S \setminus \{s\}} Z_t = z_t \right. \right] &\geq \frac{1}{c} \sum_{i=1}^{\min\{|N|, b_s\}} \left( \int_0^{z'_i} g(t) dt + 1 - g(z'_i) \right) \\ &\stackrel{\dagger}{\geq} \min\{|N|, b_s\}. \end{aligned}$$

Observe that the last inequality holds if  $g$  and  $c$  satisfy the following inequality, which is the same inequality that emerges in the analysis of RANKING without server capacities.

$$\int_0^z g(t) dt + 1 - g(z) \geq c, \quad \forall z \in [0, 1]. \quad (3)$$

It is easy to check that the combination of  $g(x) = e^{x-1}$  with  $c = 1 - 1/e$  satisfies (3) and our additional condition  $g(1) = 1$ . By applying the law of total expectation, we finish the proof:

$$\begin{aligned} \mathbb{E} \left[ x(s) + \sum_{r \in N} y(r) \right] &= \int_0^1 \dots \int_0^1 \mathbb{E} \left[ x(s) + \sum_{r \in N} y(r) \left| \bigwedge_{t \in S \setminus \{s\}} Z_t = z_t \right. \right] dz_t \dots dz_{t'} \\ &\geq \int_0^1 \dots \int_0^1 \min\{|N|, b_s\} dz_t \dots dz_{t'} = \min\{|N|, b_s\}. \end{aligned}$$

► **Theorem 7.** RANKING is  $(1 - 1/e)$ -competitive for the maximum-cardinality online  $b$ -matching problem (with variable server capacities).

#### 4 Vertex-weighted $b$ -matching

The work by Buchbinder et al. [4] implies a deterministic online algorithm with an optimal competitiveness of  $1 - 1/(1 + 1/b_{\min})^{b_{\min}}$ . We are not aware of any simpler strategy. Therefore, we focus on randomized algorithms and extend our previous result for RANKING to the vertex-weighted case. We will show that PERTURBED-GREEDY [1] achieves a competitiveness of  $1 - 1/e$  for vertex-weighted  $b$ -matching. Again, we execute the algorithm on the initial input graph  $G$ .

PERTURBED-GREEDY is similar to RANKING; only the definition of ranks differs. For each server  $s \in S$ , a *single* number  $Z_s \in [0, 1]$  is chosen uniformly at random. The rank of  $s$  is  $w_s(1 - g(Z_s))$ , where  $g : [0, 1] \rightarrow [0, 1]$  is a monotonically increasing function that will be set to  $g(x) = e^{x-1}$ .

##### Algorithm 3 PERTURBED-GREEDY.

---

```

foreach server  $s \in S$  do
  | Pick  $Z_s \in [0, 1]$  uniformly at random;
end
while a new request  $r \in R$  arrives do
  | Let  $N(r)$  denote the set of neighbors of  $r$  with remaining capacity;
  | if  $N(r) = \emptyset$  then
  | | Do not match  $r$ ;
  | else
  | | Match  $r$  to  $\arg \max\{w_s(1 - g(Z_s)) : s \in N(r)\}$  (break ties consistently);
  | end
end

```

---

We formulate the configuration LP and its dual for the vertex-weighted  $b$ -matching problem, where we take into account that each matching edge incident to a server  $s$  has a value of  $w_s$ .

$$\begin{aligned}
 \text{Config LP: } \max \quad & \sum_{s \in S} \sum_{N \subseteq N_s} w_s \cdot \min\{|N|, b_s\} \cdot m(s, N) \\
 \text{s.t.} \quad & \sum_{N \subseteq N_s} m(s, N) \leq 1, & (\forall s \in S) \\
 & \sum_{s \in S} \sum_{N \subseteq N_s: r \in N} m(s, N) \leq 1, & (\forall r \in R) \\
 & m(s, N) \geq 0, & (\forall s \in S, \forall N \subseteq N_s).
 \end{aligned}$$

$$\begin{aligned}
 \text{Dual CLP: } \min \quad & \sum_{s \in S} x(s) + \sum_{r \in R} y(r) \\
 \text{s.t.} \quad & x(s) + \sum_{r \in N} y(r) \geq w_s \cdot \min\{|N|, b_s\}, & (\forall s \in S, \forall N \subseteq N_s) \\
 & x(s), y(r) \geq 0, & (\forall s \in S, \forall r \in R).
 \end{aligned}$$

In the primal-dual analysis, we again update the primal variables as well as the dual variables  $x(s)$  and  $y(r)$  whenever PERTURBED-GREEDY matches a request  $r$  to a server  $s$ . We set

$$\Delta x(s) = \frac{w_s g(Z_s)}{c} \quad \text{and} \quad y(r) = \frac{w_s(1 - g(Z_s))}{c}$$

to ensure that the value of the dual solution is always  $1/c$  times the value of the solution for the configuration LP. Here,  $g : [0, 1] \rightarrow [0, 1]$  is a monotonically increasing function and  $c$  is a constant that will be the competitive ratio of the algorithm.

As before, it is sufficient to show that the dual constraints are satisfied in expectation. For this, we have to adapt Lemma 4 and Lemma 6. We consider two execution of PERTURBED-GREEDY on  $G$  and  $G \setminus s$  with the same  $Z$ -values  $Z_t$  for all servers  $t \in S \setminus \{s\}$ . Moreover, denote the neighbors of  $s$  in  $G$  by  $\{r_1, \dots, r_k\} = N_s$  and let  $z_i$ ,  $1 \leq i \leq k$ , be the  $Z$ -value of the matching partner  $\sigma_i$  of request  $r_i$  in the  $G \setminus s$  execution, if  $r_i$  is matched there. If  $r_i$  is unmatched, we set  $z_i := 1$  and assign a dummy matching partner  $\sigma_i$  with  $w_{\sigma_i} := 0$ . Now, further define  $\zeta_i$  as the unique value in  $[0, 1]$  such that

$$w_s (1 - g(\zeta_i)) = w_{\sigma_i} (1 - g(z_i)) ,$$

if it exists. Assuming that  $g$  is a monotonically increasing function with  $g(1) = 1$ , note that such a solution can only not exist if  $w_{\sigma_i} (1 - g(z_i)) > w_s (1 - g(0))$ , in which case we define  $\zeta_i := 0$ . It is easy to see that PERTURBED-GREEDY would prefer server  $s$  over server  $\sigma_i$  if  $Z_s < \zeta_i$ . Moreover, let  $y_s(r_i)$  be the value of  $y(r_i)$  in the  $G \setminus s$  execution. It follows that

$$y_s(r_i) = \frac{w_{\sigma_i} (1 - g(z_i))}{c} \geq \frac{w_s (1 - g(\zeta_i))}{c} .$$

We assume that PERTURBED-GREEDY assigns the  $j$ -th request matched to a server  $s$  to the server spot  $s_j$ .

► **Lemma 8.** *At any point during the parallel execution of PERTURBED-GREEDY on  $G$  and  $G \setminus s$ , it holds that the set of unmatched server spots  $U$  in the  $G$  execution forms a superset of the unmatched server spots  $\tilde{U}$  in the  $G \setminus s$  execution. For all server spots  $s'_i \in U \setminus \tilde{U}$ , it holds that  $w_{s'} (1 - g(Z_{s'})) \leq w_s (1 - g(Z_s))$ . If equality holds and  $s' \neq s$ , then  $s$  has a higher priority in the tiebreaking.*

**Proof.** By induction. The properties trivially hold initially. Then, whenever a new request  $r$  arrives,  $\tilde{U} \subseteq U$  can only be violated if  $r$  is assigned to a server spot  $t_i \in \tilde{U}$  in the  $G$  execution, but  $r$  is not assigned to  $t_i$  in the  $G \setminus s$  execution. There, it is either unmatched or matched to a different server spot, which leads to a contradiction in either case. Since  $t_i \in \tilde{U}$  and  $r$  is a neighbor of the server  $t$ ,  $r$  cannot be unmatched in the  $G \setminus s$  execution. If PERTURBED-GREEDY chooses a different server spot  $t_j$  for  $r$  in the  $G \setminus s$  execution, then either  $i < j$  or  $i > j$  has to hold.  $i < j$  results in a contradiction because  $t_i \in \tilde{U}$  and we defined that PERTURBED-GREEDY always chooses the server spot with smallest index. Furthermore,  $i > j$  also results in a contradiction because  $t_j \in \tilde{U} \subseteq U$  and thus PERTURBED-GREEDY would have chosen  $t_j$  in the  $G$  execution as well. Moreover, if PERTURBED-GREEDY assigns  $r$  to a server spot of a different server  $t'$  in the  $G \setminus s$  execution, then  $w_{t'} (1 - g(Z_{t'})) \geq w_t (1 - g(Z_t))$  has to hold. However,  $\tilde{U} \subseteq U$  implies that this server spot would also be unmatched and available in the  $G$  execution. If  $w_{t'} (1 - g(Z_{t'})) > w_t (1 - g(Z_t))$ , PERTURBED-GREEDY would not have chosen the correct neighbor in the  $G$  execution according to its definition, and if  $w_{t'} (1 - g(Z_{t'})) = w_t (1 - g(Z_t))$ , then the tiebreak would be inconsistent between the two executions.

Moreover, a new server spot  $t'_i$  is only added to  $U \setminus \tilde{U}$  if the matching decision for  $r$  is different in the two execution, i.e. the  $G$  execution assigns  $r$  to some server spot  $t_j \in U \setminus \tilde{U}$  and the  $G \setminus s$  execution assigns  $r$  to  $t'_i \in \tilde{U}$ . Therefore, it has to hold that either  $w_{t'} (1 - g(Z_{t'})) < w_t (1 - g(Z_t))$  or  $w_{t'} (1 - g(Z_{t'})) = w_t (1 - g(Z_t))$  and  $t$  has a higher tiebreak priority than  $t'$ . The induction hypothesis then finishes the proof. ◀

► **Corollary 9.** Given  $Z_t$  for all servers  $t \in S \setminus \{s\}$ ,  $y(r_i) \geq y_s(r_i) \geq w_s(1 - g(\zeta_i))/c$  holds for all  $i$ ,  $1 \leq i \leq k$ , and all possible values of  $Z_s$ .

► **Lemma 10.** Given  $Z_t$  for all servers  $t \in S \setminus \{s\}$ , let  $\zeta_1 \geq \dots \geq \zeta_k$  be the  $\zeta$ -values of the  $k = |N_s|$  neighbors of  $s$  in a  $G \setminus s$  execution in non-increasing order. Then, server  $s$  has at least  $\min\{a, b_s\}$  matching partners in an execution of PERTURBED-GREEDY on  $G$ , where  $a$  is the largest possible integer such that  $Z_s < \zeta_a \leq \dots \leq \zeta_1$ .

**Proof.** Whenever a neighbor  $r_i$  of  $s$  with  $\zeta_i > Z_s$  (note that this implies  $\zeta_i > 0$ ) arrives and  $s$  still has remaining capacity, then Lemma 8 implies that  $r_i$  will be matched to  $s$  since  $w_s(1 - g(Z_s)) > w_s(1 - g(\zeta_i)) = w_{\sigma_i}(1 - g(z_i))$  holds by definition. ◀

We finally show how to choose  $g$  and  $c$  such that the dual constraints are satisfied in expectation. Let  $s$  be any server in  $G$  with  $k$  neighbors. Let  $\zeta_i$  be the  $\zeta$ -value of neighbor  $r_i \in N_s$ ,  $1 \leq i \leq k$ , in the  $G \setminus s$  execution. If  $k < b_s$ , we further define  $\zeta_{k+1} = \dots = \zeta_{b_s} = 0$ . Let  $z'_1 \geq \dots \geq z'_{b_s}$  then be the  $b_s$  largest values of  $\{\zeta_1, \dots, \zeta_{\max\{k, b_s\}}\}$  in non-increasing order. Lemma 10 implies that

$$\mathbb{E} \left[ x(s) \mid \bigwedge_{t \in S \setminus \{s\}} Z_t = z_t \right] = \sum_{i=1}^{b_s} w_s \int_0^{z'_i} \frac{g(t)}{c} dt.$$

Moreover, by Corollary 9, it holds for every neighbor  $r \in N_s$  of  $s$

$$\mathbb{E} \left[ y(r) \mid \bigwedge_{t \in S \setminus \{s\}} Z_t = z_t \right] \geq \frac{w_s(1 - g(\zeta_r))}{c},$$

where  $\zeta_r = \zeta_i$  for some  $i$ ,  $1 \leq i \leq k$ . Putting everything together yields

$$\mathbb{E} \left[ x(s) + \sum_{r \in N} y(r) \mid \bigwedge_{t \in S \setminus \{s\}} Z_t = z_t \right] \geq \frac{w_s}{c} \left( \sum_{i=1}^{b_s} \int_0^{z'_i} g(t) dt + \sum_{r \in N} (1 - g(\zeta_r)) \right).$$

Note that  $\sum_{r \in N} (1 - g(\zeta_r))$  is lower bounded by  $\sum_{i=1}^{\min\{|N|, b_s\}} (1 - g(z'_i))$ , since  $g$  is an increasing function with  $g(1) = 1$  and the  $z'$ -values are an upper bound for the  $\zeta_r$ -values. Plugging this in, we get

$$\begin{aligned} \mathbb{E} \left[ x(s) + \sum_{r \in N} y(r) \mid \bigwedge_{t \in S \setminus \{s\}} Z_t = z_t \right] &\geq \frac{w_s}{c} \sum_{i=1}^{\min\{|N|, b_s\}} \left( \int_0^{z'_i} g(t) dt + 1 - g(z'_i) \right) \\ &\stackrel{!}{\geq} w_s \min\{|N|, b_s\}. \end{aligned}$$

Observe that this holds true if  $g$  and  $c$  fulfill the same inequality (3) as for RANKING. As argued before, it is satisfied for  $g(x) = e^{x-1}$  and  $c = 1 - 1/e$ , which also satisfies our additional constraint  $g(1) = 1$ . Therefore, using the law of total expectation, we conclude that

$$\begin{aligned} \mathbb{E} \left[ x(s) + \sum_{r \in N} y(r) \right] &= \int_0^1 \dots \int_0^1 \mathbb{E} \left[ x(s) + \sum_{r \in N} y(r) \mid \bigwedge_{t \in S \setminus \{s\}} Z_t = z_t \right] dz_t \dots dz_{t'} \\ &\geq \int_0^1 \dots \int_0^1 w_s \min\{|N|, b_s\} dz_t \dots dz_{t'} = w_s \min\{|N|, b_s\}. \end{aligned}$$

► **Theorem 11.** PERTURBED-GREEDY is  $(1 - 1/e)$ -competitive for the vertex-weighted online  $b$ -matching problem (with variable server capacities).



---

**References**

---

- 1 G. Aggarwal, G. Goel, C. Karande, and A. Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1253–1264. SIAM, 2011.
- 2 Y. Azar and A. Litichevsky. Maximizing throughput in multi-queue switches. *Algorithmica*, 45(1):69–90, 2006.
- 3 B.E. Birnbaum and C. Mathieu. On-line bipartite matching made simple. *SIGACT News*, 39(1):80–87, 2008.
- 4 N. Buchbinder, K. Jain, and J. Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *Proceedings of the 15th Annual European Symposium on Algorithms (ESA)*, volume 4698 of *Lecture Notes in Computer Science*, pages 253–264. Springer, 2007.
- 5 K. Chaudhuri, C. Daskalakis, R.D. Kleinberg, and H. Lin. Online bipartite perfect matching with augmentations. In *Proceedings of the 28th IEEE International Conference on Computer Communications (INFOCOM)*, pages 1044–1052, 2009.
- 6 N.R. Devanur, K. Jain, and R.D. Kleinberg. Randomized primal-dual analysis of RANKING for online bipartite matching. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 101–107, 2013.
- 7 A. Eden, M. Feldman, A. Fiat, and K. Segal. An economics-based analysis of RANKING for online bipartite matching. In *Proceedings of the 4th Symposium on Simplicity in Algorithms (SOSA)*, pages 107–110, 2021.
- 8 G. Goel and A. Mehta. Online budgeted matching in random input models with applications to adwords. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 982–991, 2008.
- 9 E.F. Grove, M.-Y. Kao, P. Krishnan, and J.S. Vitter. Online perfect matching and mobile computing. In *Proceedings 4th International Workshop, on Algorithms and Data Structures (WADS)*, volume 955 of *Lecture Notes in Computer Science*, pages 194–205. Springer, 1995.
- 10 Z. Huang and Q. Zhang. Online primal dual meets online matching with stochastic rewards: configuration LP to the rescue. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1153–1164, 2020.
- 11 Z. Huang, Q. Zhang, and Y. Zhang. Adwords in a panorama. In *Proceedings of the 61st IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1416–1426, 2020.
- 12 B. Jin and D.P. Williamson. Improved analysis of RANKING for online vertex-weighted bipartite matching. *CoRR*, abs/2007.12823, 2020. [arXiv:2007.12823](https://arxiv.org/abs/2007.12823).
- 13 B. Kalyanasundaram and K. Pruhs. An optimal deterministic algorithm for online b-matching. *Theor. Comput. Sci.*, 233(1-2):319–325, 2000.
- 14 C. Karande, A. Mehta, and P. Tripathi. Online bipartite matching with unknown distributions. In *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC)*, pages 587–596. ACM, 2011.
- 15 R.M. Karp, U.V. Vazirani, and V.V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 352–358, 1990.
- 16 M. Mahdian and Q. Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing LPs. In *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC)*, pages 597–606, 2011.
- 17 A. Mehta and D. Panigrahi. Online matching with stochastic rewards. In *53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 728–737, 2012.
- 18 A. Mehta, A. Saberi, U.V. Vazirani, and V.V. Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5):22, 2007.

**A Comparison of RELATIVEBALANCE and ALLOCATION**

Consider the algorithm ALLOCATION [4] specialized for maximum cardinality online  $b$ -matching. It is the other known optimal deterministic online algorithm for this problem besides RELATIVEBALANCE.

**Algorithm 4** ALLOCATION.

---

```

Initialize  $x(s) = 0, \forall s \in S$ , and  $y(r) = 0, \forall r \in R$ ;
while a new request  $r \in R$  arrives do
  Let  $N(r)$  denote the set of neighbors  $s$  of  $r$  with  $x(s) < 1$ ;
  if  $N(r) = \emptyset$  then
    Do not match  $r$ ;
  else
    Match  $r$  to  $\arg \min\{x(s) : s \in N(r)\}$  (break ties arbitrarily);
    Update  $y(r) = 1 - x(s)$ ;
    Update  $x(s) = x(s) \cdot \left(1 + \frac{1}{b_s}\right) + \frac{1}{(d-1)b_s}$ ;
  end
end

```

---

The analysis of Buchbinder et al. [4] can be extended to show that ALLOCATION constructs a feasible solution for the classical dual matching LP with its variables  $x(s)$  and  $y(r)$ . Moreover, it can be shown that the size of the constructed matching is at least  $c = 1 - 1/d$  times the value of the constructed dual solution, where  $d = (1 + 1/b_{\min})^{b_{\min}}$ . This implies that ALLOCATION achieves the optimal competitiveness of  $1 - 1/(1 + 1/b_{\min})^{b_{\min}}$ .

Recall that RELATIVEBALANCE matches a request  $r$  to an eligible neighbor with minimum relative server load. In contrast, ALLOCATION matches  $r$  to a neighbor  $s$  with minimum  $x(s)$ , if  $x(s) < 1$ . It can be proven by induction that at any point during the execution of ALLOCATION, a server  $s$  with capacity  $b_s$  and  $load_s$  assigned requests has

$$x(s) = \frac{1}{d-1} \left( \left(1 + \frac{1}{b_s}\right)^{load_s} - 1 \right). \quad (4)$$

This has two consequences: on the one hand, ALLOCATION may choose a different matching partner for  $r$  compared to RELATIVEBALANCE in certain situations, since  $load_s/b_s \leq load_{s'}/b_{s'}$  does not imply  $(1 + 1/b_s)^{load_s} \leq (1 + 1/b_{s'})^{load_{s'}}$ . On the other hand, ALLOCATION considers  $s$  to be full once  $x(s) \geq 1$ . Equation (4) implies that this is the case when

$$\left(1 + \frac{1}{b_s}\right)^{load_s} \geq \left(1 + \frac{1}{b_{\min}}\right)^{b_{\min}}.$$

Observe that this may occur before  $load_s$  becomes  $b_s$ , meaning before  $s$  actually has been assigned  $b_s$  requests. This implies that an unmodified version of ALLOCATION may leave some server spots unused and thus not create a maximal matching.

## B Analysis of RANDOM

We start with a pseudo-code description of RANDOM.

### Algorithm 5 RANDOM.

---

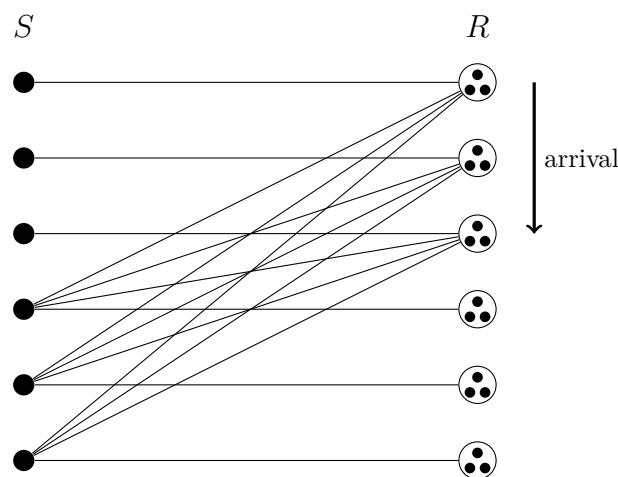
```

while a new request  $r \in R$  arrives do
  Let  $N(r)$  denote the set of neighbors of  $r$  with remaining capacity;
  if  $N(r) = \emptyset$  then
    | Do not match  $r$ ;
  else
    | Match  $r$  to a random  $s \in N(r)$ ;
  end
end

```

---

We extend RANDOM's worst case input graph of the problem without server capacities and show that RANDOM also does not achieve a competitive ratio better than  $\frac{1}{2}$  for the online  $b$ -matching problem, even if all server capacities are equal. Consider a graph with  $n$  servers  $S = \{s_1, \dots, s_n\}$  and  $n$  rounds of requests  $R = R_1 \dot{\cup} \dots \dot{\cup} R_n$ , where  $n = 2k$ . Every server  $s$  has the same capacity  $b_s := b$  and each round contains  $b$  identical requests that all have the same neighbors. The different rounds arrive one after another, such that the first request of  $R_{i+1}$  only arrives after the last request of  $R_i$  arrived,  $1 \leq i < n$ . Requests within the same round can arrive in an arbitrary order. All requests  $r \in R_i$  of the  $i$ -th round are adjacent to server  $s_i$ . This implies that there exists a perfect matching of size  $b \cdot n$  that matches  $R_i$  to  $s_i$ . Moreover, all requests  $r \in R_1 \dot{\cup} \dots \dot{\cup} R_k$  of the first half of rounds are additionally adjacent to all servers  $s_{k+1}, \dots, s_n$  of the second half (see Fig. 2).



**Figure 2** A bad input for RANDOM. There are  $n = 6$  servers, each with capacity  $b = 3$ , and  $n$  rounds of requests, each containing  $b$  identical requests. For clarity, the adjacencies of a round are depicted as a whole. Note that requests still arrive individually one after another and not together with their complete round.

Intuitively, RANDOM performs poorly on this graph since its very unlikely that it makes the correct matching decision for the requests from the first half of rounds, i.e. assigning a request from  $R_i$  to server  $s_i$ . Observe that - irrespective of the matching decision made

by RANDOM for the requests of the first half of rounds - every server of the second half will be assigned exactly  $b$  requests. Let  $X$  be a random variable indicating how many requests were matched to the first half of servers by RANDOM. The size of the constructed matching  $M$  is then  $|M| = b \cdot k + X$ . Therefore, it is possible to compute the expected size of the constructed matching by determining the expected value of  $X$ .

Let  $X_i$ ,  $1 \leq i \leq k$ , be the number of requests assigned to server  $s_i$ . It holds that  $X = \sum_{i=1}^k X_i$ . By design, only requests from  $R_i$  may be assigned to  $s_i$ , for  $1 \leq i \leq k$ . Let  $r$  be any request from such a round  $R_i$  and let  $p_i$  be the probability that RANDOM assigns  $r$  to its perfect matching partner  $s_i$ . Observe that  $p_i$  depends on the number of servers in the second half with remaining capacity. At most  $b \cdot (i - 1) + (b - 1)$  requests arrived before  $r$  ( $r$  may be the last request of  $R_i$ ). Hence at most  $(i - 1)$  of the last  $k$  servers can be full, implying at least  $(k - i + 1)$  eligible neighbors in the second half of servers for all requests from  $R_i$ . Furthermore, server  $s_i$  cannot become full before the last request of  $R_i$  arrives. RANDOM has therefore at least  $(k - i + 2)$  servers to choose from when assigning a request from round  $R_i$ . This yields

$$\mathbb{E}[X] = \sum_{i=1}^k \mathbb{E}[X_i] \leq \sum_{i=1}^k b \cdot p_i \leq b \sum_{i=1}^k \frac{1}{k + 2 - i} = b \sum_{j=2}^{k+1} \frac{1}{j} = b(H_{k+1} - 1) \leq b \ln(k + 1),$$

where  $H_n$  denotes the  $n$ -th harmonic number and the inequality  $H_n \leq \ln(n) + 1$  is used. The size of the perfect matching in this graph is  $b \cdot n$ . Thus, RANDOM achieves a competitive ratio of

$$\frac{\mathbb{E}[|M|]}{b \cdot n} \leq \frac{b \cdot k + b \ln(k + 1)}{b \cdot n} = \frac{1}{2} + \frac{\ln(n/2 + 1)}{n} \xrightarrow{n \rightarrow \infty} \frac{1}{2}.$$

This finishes the proof of Theorem 3.

# Bag-Of-Tasks Scheduling on Related Machines

Anupam Gupta ✉

Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

Amit Kumar ✉

Computer Science and Engineering Department, Indian Institute of Technology, Delhi, India

Sahil Singla ✉

Department of Computer Science, Princeton University, NJ, USA

---

## Abstract

We consider online scheduling to minimize weighted completion time on related machines, where each *job* consists of several *tasks* that can be concurrently executed. A job gets completed when all its component tasks finish. We obtain an  $O(K^3 \log^2 K)$ -competitive algorithm in the *non-clairvoyant* setting, where  $K$  denotes the number of distinct machine speeds. The analysis is based on dual-fitting on a precedence-constrained LP relaxation that may be of independent interest.

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

**Keywords and phrases** approximation algorithms, scheduling, bag-of-tasks, related machines

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.3

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/abs/2107.06216>

**Funding** *Anupam Gupta*: Supported in part by NSF awards CCF-1907820, CCF1955785, and CCF-2006953.

## 1 Introduction

Scheduling to minimize the weighted completion time is a fundamental problem in scheduling. Many algorithms have been developed in both the online and offline settings, and for the cases where machines are identical, related, or unrelated. Most of the work, however, focuses on the setting where each job is a monolithic entity, and has to be processed in a sequential manner.

In this work, we consider the online setting with multiple related machines, where each *job* consists of several *tasks*. These tasks are independent of each other, and can be executed concurrently on different machines. (Tasks can be preempted and migrated.) A job is said to have *completed* when all its component tasks finish processing. We consider the *non-clairvoyant* setting where the algorithm does not know the size of a task up-front, but only when the task finishes processing. Such instances arise in operating system schedulers, where a job and its tasks correspond to a process and its threads that can be executed in parallel. This setting is sometimes called a “*bag of tasks*” (see e.g. [2, 10, 4]).

The bag-of-tasks model can be modeled using precedence constraints. Indeed, each job is modeled as a star graph, where the tasks correspond to the leaves (and have zero weight), and the root is an auxiliary task with zero processing requirement but having weight  $w_j$ . Hence the root can be processed only after all leaf tasks have completed processing. The goal is to minimize total *weighted completion time*. Garg et al. [7] gave a constant-competitive algorithm for this problem for *identical machines*, in a more general setting where tasks form arbitrary precedence DAGs.



© Anupam Gupta, Amit Kumar, and Sahil Singla;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 3; pp. 3:1–3:16



Leibniz International Proceedings in Informatics

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

### 3:2 Bag-Of-Tasks Scheduling on Related Machines

We extend this result to the setting of *related machines* where machine  $i$  has speed  $s_i$ . By losing a constant factor, we assume that all speeds are powers of some constant  $C$ . Let  $K$  denote the number of distinct machine speeds. In §2, we show that this problem is strictly more challenging than in the identical machines setting:

► **Theorem 1 (Lower Bound).** *Any online non-clairvoyant algorithm has  $\Omega(K)$  competitive ratio for bags-of-tasks on related machines.*

The lower bound arises because we want to process larger tasks on fast machines, but we have no idea about the sizes of the tasks, so we end up clogging the fast machines with small tasks: this issue did not arise when machines were identical. Given the lower bound, we now look for a non-clairvoyant scheduling algorithm with a competitive ratio that depends on  $K$ , the number of distinct speeds. This number may be small in many settings, e.g., when we use commodity hardware of a limited number of types (say, CPUs and GPUs). Our main result is a positive answer to this question:

► **Theorem 2 (Upper Bound).** *The online non-clairvoyant algorithm for bags-of-tasks on related machines has a competitive ratio of §3 is  $O(\min\{K^3 \log^2 K, K + \log n\})$ .*

Our algorithm uses a greedy strategy. Instead of explicitly building a schedule, it assigns (*processing*) rates to tasks at each time  $t$ . Such a rate assignment is called feasible if for every  $k$ , the rate assigned to any subset of  $k$  tasks is at most the total speed of the  $k$  fastest machines. Using an argument based on Hall’s matching theorem, a schedule exists if and only if such a rate assignment can be found. To assign these rates, each alive task gets a “priority”, which is the ratio of the weight of the job containing it to the number of alive tasks of this job. In other words, a task with low weight or with many tasks gets a low priority. We assign feasible rates to alive tasks in a “fair manner”, i.e., we cannot increase the rate of a high priority task by decreasing the rate of a lower priority task. To efficiently find such feasible rates, we use a water-filling procedure.

The analysis proceeds using the popular dual-fitting approach, but we need new ideas: (i) we adapt the precedence-constrained LP relaxation for completion time in [5] to our setting. A naive relaxation would define the completion time of a task as the maximum of the (fractional) completion times of each of the tasks, where the fractional completion time of a task is the sum over times  $t$  of the fraction of the task remaining at this time. Instead, we define  $U_{j,t}$ , for a job  $j$  and time  $t$  as the maximum over all tasks  $v$  for  $j$  of the fraction of  $v$  which remains to be completed at time  $t$ , the completion time of  $j$  as  $\sum_t U_{jt}$ . (See §4 for details.) (ii) Although it is natural to divide the machines into classes based on their speeds, we need a finer partitioning, which drives our setting of dual variables. Indeed, the usual idea of dividing up the job’s weight equally among the tasks that are still alive only leads to an  $O(\log n)$ -competitiveness (see §5). To do better, we first preprocess the instance so that distinct machine speeds differ by a constant factor, but the total processing capacity of a slower speed class is far more than that of all faster machines. Now, at each time, we divide the machines into *blocks*. A constant fraction of the blocks have the property that either the average speed of the machines in the block is close to one of the speed classes, or the total processing capacity of a block is close to that of *all* the machines of a speed class. It turns out that our dual-fitting approach works for accounting the weight of jobs which get processed by such blocks; proving this constitutes the bulk of technical part of the analysis. Finally, we show that most jobs (in terms of weight) get processed by such blocks, and hence we are able to bound the overall weighted completion time. We present the proofs in stages, giving intuition for the new components in each of the sections.

## 1.1 Related Work

Minimizing weighted completion time on parallel machines with precedence constraints has  $O(1)$ -approximation in the *offline* setting: Li [9] improves on [8, 11] to give a  $3.387 + \varepsilon$ -approximation. For *related* machines the precedence constraints make the problem harder: there is an  $O(\log m / \log \log m)$ -approximation [9] improving on a prior  $O(\log K)$  result [5], and an  $\omega(1)$  hardness under certain complexity assumptions [3]. Here  $m$  denotes the number of machines. These results are for offline and hence clairvoyant settings, and do not apply to our setting of non-clairvoyant scheduling.

In the setting of parallel machines, there has been recent work on minimizing weighted completion time in DAG scheduling, where each job consists of a set of tasks with precedence constraints between them given by a DAG [13, 1]. [7] generalized this to the non-clairvoyant setting and gave an  $O(1)$ -competitive algorithm. Our algorithm for the related case is based on a similar water-filling rate assignment idea. Since the machines have different speeds, a set of rates assigned to tasks need to satisfy a more involved feasibility condition. Consequently, its analysis becomes much harder; this forms the main technical contribution of the paper. Indeed, even for the special case considered in this paper where every DAG is a star, we can show a lower bound of  $\Omega(K)$  on the competitive ratio of any non-clairvoyant algorithm. In the full version, we show that any non-clairvoyant algorithm for related machines DAG scheduling must have  $\Omega(\frac{\log m}{\log \log m})$ -competitive ratio.

Our problem also has similarities to open shop scheduling. In open shop scheduling, each job consists of several tasks, where each task  $v$  (for job  $j$ ) needs to be processed on a distinct machine for  $p_{vj}$  amount of time. However, unlike our setting, two tasks for a job cannot be processed simultaneously on different machines. [12] considered open shop scheduling in the offline setting for related machines and gave a  $(2 + \varepsilon)$ -approximation. [6] considered a further generalization of our problem to unrelated machines, where the tasks corresponding to distinct jobs need not be disjoint. They gave a constant-factor approximation algorithm, again offline.

## 1.2 Paper Organization

In this extended abstract, we first give the algorithm in §3, and the linear program in §4. A simpler proof of  $O(K + \log n)$ -competitiveness is in §5. We show  $\text{poly}(K)$ -competitiveness for the case of a single job (which corresponds to makespan minimization) in §6, and then give the complete proof for the general case in §7.

## 2 Problem Statement and the $\Omega(K)$ Hardness

Each job  $j$  has a *weight*  $w_j$  and consists of *tasks*  $T(j) = \{(j, 1), (j, 2), \dots, (j, k_j)\}$  for some  $k_j$ . Each task  $v = (j, \ell)$  has an associated *processing requirement/size*  $p_v = p_{(j, \ell)}$ . The job  $j$  completes when all its associated tasks finish processing. We use letters  $j, j'$ , etc. to denote jobs, and  $v, v'$ , etc. to denote tasks  $(j, \ell)$ .

There are  $m$  machines with speeds  $s_1 \geq s_2 \geq \dots \geq s_m$ . The goal is to minimize the weighted completion time of the jobs. We allow task *preemption* and *migration*, and different tasks of a job can be processed concurrently on different machines. However, a task itself can be processed on at most one machine at any time. In this extended abstract we consider the special case when all release dates are 0, but our results also extend to the more general setting of arbitrary release dates (details in the full version). Let  $S_k := s_1 + \dots + s_k$  denote the total speed of the fastest  $k$  machines. Since we care about the number of *distinct* speeds, we assume there are  $K$  *speed classes*, with speeds  $\sigma_1 > \sigma_2 > \dots > \sigma_K$ . There are  $m_i$  machines having speed  $\sigma_i$ , where  $\sum_i m_i = m$ .

### 3:4 Bag-Of-Tasks Scheduling on Related Machines

► **Assumption 3** (Increasing Capacity Assumption). For parameter  $\gamma \geq 1$ :

- (1) (Falling Speeds.) For each  $\ell$ , we have  $\sigma_i/\sigma_{i+1} \geq 64$ .
- (2) (Increasing Capacity.) For each  $\ell$ , the total processing capacity of speed class  $\ell$  is at least twice that of the previous (faster) speed classes. I.e.,  $m_\ell \sigma_\ell \geq 2(m_1 \sigma_1 + \dots + m_{\ell-1} \sigma_{\ell-1})$ .
- (3) (Speed-up.) The algorithm uses machines that are  $\gamma$  times faster than the adversary's machines.

► **Proposition 4.** An arbitrary instance can be transformed into one satisfying Assumption 3 by losing a factor  $O(\gamma K)$  in the competitive ratio.

**Proof.** (Sketch) For the first part, we round down the speed of each machine to a power of 64. This changes the completion time by at most a factor of 64. The second increasing capacity assumption is not without loss of generality – we greedily find a subset of speed classes by losing  $O(K)$  factor in competitive ratio (see details in Appendix A). Finally, the  $\gamma$ -speedup can only change the competitive ratio by  $\gamma$  factor. ◀

Next we show that any online algorithm has to be  $\Omega(K)$ -competitive even for a single job with the machines satisfying increasing capacity Assumption 3.

► **Proposition 5.** Any online algorithm is  $\Omega(K)$ -competitive even for a single job under increasing capacity Assumption 3.

**Proof.** (Sketch) Consider a single job  $j$  with  $m$  tasks, where  $m$  is the number of machines. For every speed class  $\ell$ , there are  $m_\ell$  tasks of size  $\sigma_\ell$  – call these tasks  $T_\ell(j)$ . Since there is only one job, the objective is to minimize the makespan. The offline (clairvoyant) objective is 1, since all tasks can be assigned to machines with matching speeds. However, any online algorithm incurs a makespan of  $\Omega(K)$ . Here is an informal argument, which can be proved even for randomized algorithms against oblivious adversaries: since there is no way to distinguish between the tasks, the algorithm can at best run all the alive tasks at the same speed. The tasks in  $T_K(j)$  will be the first to finish by time  $\frac{m_K \sigma_K}{\sum_\ell m_\ell \sigma_\ell} \geq \frac{1}{2}$ , where the inequality follows from the increasing capacity assumption. At this time, the processing on tasks from  $T_\ell(j)$  for  $\ell < K$  has been very small, and so tasks in  $T_{K-1}(j)$  will require about 1/2 more units of time to finish, and so on. ◀

## 3 The Scheduling Algorithm

The scheduling algorithm assigns, at each time  $t$ , a rate  $L_v^t$  to each unfinished task  $v$ . The following lemma (whose proof is deferred to the appendix) characterizes rates that correspond to schedules:

► **Lemma 6.** A schedule  $\mathcal{S}$  is feasible if for every time  $t$  and every value of  $k$ :

- (\*) the total rate assigned to any subset of  $k$  tasks is at most  $\gamma \cdot S_k$ .

For each time  $t$ , we now specify the rates  $L_v^t$  assigned to each unfinished task  $v$ . For job  $j$ , let  $T^t(j)$  be the set of tasks in  $T(j)$  which are alive at time  $t$ . Initially all tasks are *unfrozen*. We raise a parameter  $\tau$ , starting at zero, at a uniform speed. The values taken by  $\tau$  will be referred to as *moments*. For each job  $j$  and each task  $v \in T^t(j)$  that is unfrozen, define a *tentative rate* at  $\tau$  to be

$$L_v^t := \frac{w_j}{|T^t(j)|} \cdot \tau. \tag{1}$$



Hence the tentative rates of these unfrozen tasks increase linearly, as long as condition  $(\star)$  is satisfied. However, if  $(\star)$  becomes tight for some subset  $V$  of alive tasks, i.e.,  $\sum_{v \in V} L_v^t = \gamma \cdot S_{|V|}$ , pick a *maximal* set of such tasks and *freeze* them, fixing their rates at their current tentative values. (Observe the factor of  $\gamma$  appears on the right side because we assume the machines in the algorithm to have a speedup of  $\gamma$ .) Now continue the algorithm this way, raising  $\tau$  and the  $L_v^t$  values of remaining unfrozen tasks  $v$  until another subset gets tight, etc., stopping when all jobs are frozen. This defines the  $L_v^t$  rates for each task  $v$  for time  $t$ . By construction, these rates satisfy  $(\star)$ .

### 3.1 Properties of the Rate Assignment

The following claim shows that all alive tasks corresponding to a job get frozen simultaneously.

► **Lemma 7** (Uniform Rates). *For any time  $t$  and any job  $j$ , all its alive tasks (i.e., those in  $T^t(j)$ ) freeze at the same moment  $\tau$ , and hence get the same rate.*

**Proof.** For the sake of contradiction, consider the first moment  $\tau$  where a maximal set  $V$  of tasks contains  $v$  but not  $v'$ , for some job  $j$  with  $v, v' \in T(j)$ . Both  $v, v'$  have been treated identically until now, so  $L_v^t = L_{v'}^t$ . Also, by the choice of  $\tau$ ,  $\sum_{u \in V: u \neq v} L_u^t + L_v^t = \gamma S_{|V|}$ . Since we maintain feasibility at all moments,

$$\sum_{u \in V: u \neq v} L_u^t + L_v^t + L_{v'}^t \leq \gamma S_{|V|+1} \quad \text{and} \quad \sum_{u \in V: u \neq v} L_u^t \leq \gamma S_{|V|-1}.$$

This implies  $L_v^t \geq \gamma s_{|V|}$  and  $L_{v'}^t \leq \gamma s_{|V|+1}$ . Since  $L_v^t = L_{v'}^t$  and  $s_{|V|} \geq s_{|V|+1}$ , all of these must be equal. In that case, by the maximality of set  $V$ , the algorithm should have picked  $V \cup \{v'\}$  instead of  $V$ . ◀

For a task  $v \in T^t(j)$ , define  $\tilde{w}^t(v) := w_j / |T^t(j)|$  to be task  $v$ 's “share” of the weight of job  $j$  at time  $t$ . So if task  $v$  freezes at moment  $\tau$ , then its rate is  $L_v^t = \tilde{w}^t(v) \cdot \tau$ . Let us relate this share for  $v$  to certain averages of the weight. (Proof in Appendix B)

► **Corollary 8.** *Fix a time  $t$ . Let  $V$  be the set of tasks frozen by some moment  $\tau$ . For a task  $v \in V$ ,*

- (i) *if  $V' \subseteq V$  is any subset of tasks which freeze either at the same moment as  $v$ , or after it, then  $\frac{\tilde{w}^t(v)}{s_{|V|}} \geq \frac{w(V')}{S_{|V|}}$ .*
- (ii) *if  $V'' \subseteq V$  is any subset of tasks which freeze either at the same moment as  $v$ , or before it, then  $\frac{\tilde{w}^t(v)}{L_v^t} \leq \frac{\tilde{w}^t(V'')}{\sum_{v' \in V''} L_{v'}^t}$ .*

### 3.2 Defining the Blocks

The rates for tasks alive at any time  $t$  are defined by a sequence of freezing steps, where some group of tasks are frozen: we call these groups *blocks*. By Lemma 7, all tasks in  $T^t(j)$  belong to the same block. The weight  $w(B)$  of block  $B$  is the total weight of jobs whose tasks belong to  $B$ . Let  $B_1^t, B_2^t, \dots$  be the blocks at time  $t$  in the order they were frozen, and  $\tau_1^t, \tau_2^t, \dots$  be the moments at which they froze. Letting  $b_r^t := |B_1^t \cup \dots \cup B_r^t|$ , we get that any task  $v \in B_r^t$  satisfies  $\tau_v^t \cdot w(B_r^t) = \gamma(S_{b_{r+1}^t} - S_{b_r^t})$ .

Each block  $B_r^t$  has an associated set of machines, namely the machines on which the tasks in this block are processed – i.e., the machines indexed  $b_{r-1}^t + 1, \dots, b_r^t$ . We use  $m(B)$  to denote the set of machines associated with a block  $B$ . Since  $|B| = |m(B)|$  and the jobs in  $B$  are processed on  $m(B)$  in a pre-emptive manner at time  $t$ , the rate assigned to any job is at least the slowest speed (and at most the fastest speed) of the machines in  $m(B)$ .

## 4 The Analysis and Intuition

We prove the competitiveness by a dual-fitting analysis: we give a primal-dual pair of LPs, use the algorithm above to give a feasible primal, and then exhibit a feasible dual with value within a small factor of the primal cost.

In the primal LP, we have variables  $x_{ivt}$  for each task  $v$ , machine  $i$ , and time  $t$  denoting the extent of processing done on task  $v$  at machine  $i$  during the interval  $[t, t + 1]$ . Here  $U_{j,t}$  denotes fraction of job  $j$  finished at or after time  $t$ , and  $C_j$  denotes the completion time of job  $j$ .

$$\begin{aligned} \min \sum_j w_j C_j + \sum_{j,t} w_j U_{j,t} \\ U_{j,t} &\geq \sum_{t' \geq t} \sum_i \frac{x_{ivt'}}{p_v} && \forall j, \forall v \in T(j), \forall t \end{aligned} \quad (2)$$

$$C_j \geq \sum_t \sum_i \frac{x_{ivt}}{s_i} \quad \forall j, \forall v \in T(j) \quad (3)$$

$$\sum_i \sum_t \frac{x_{ivt}}{p_v} \geq 1 \quad \forall j, \forall v \in T(j) \quad (4)$$

$$\sum_v \frac{x_{ivt}}{s_i} \leq 1 \quad \forall i, \forall t \quad (5)$$

The constraint (2) is based on precedence-constrained LP relaxations for completion time. Indeed, each job can be thought of as a star graph with a zero size task at the root preceded by all the actual tasks at the leaf. In our LP, for each time  $t$ , we define  $U_{j,t} \in [0, 1]$  to be the maximum over all tasks  $v \in T(j)$  of the fraction of  $v$  that remains (the RHS of (2)), and the completion time of  $j$  is at least the total sum over times  $t$  of  $U_{j,t}$  values. Since we do not explicitly enforce that a task cannot be processed simultaneously on many machines, the first term  $\sum_j w_j C_j$  is added to avoid a large integrality gap. We show feasibility of this LP relaxation (up to factor 2) in §C.

▷ **Claim 9.** For any schedule  $\mathcal{S}$ , there is a feasible solution to the LP of objective value at most  $2 \text{cost}(\mathcal{S})$ .

The linear programming dual has variables  $\alpha_{j,v}, \delta_{j,v}, \delta_{j,v,t}$  corresponding to constraints (4),(3),(2) for every job  $j$  and task  $v \in T(j)$ , and  $\beta_{i,t}$  corresponding to constraints (5) for every machine  $i$  and time  $t$ :

$$\begin{aligned} \max. \sum_{j,v} \alpha_{j,v} - \sum_{i,t} \beta_{i,t} \\ \frac{\alpha_{j,v}}{p_v} \leq \frac{\beta_{i,t}}{s_i} + \sum_{t' \leq t} \frac{\delta_{j,v,t'}}{p_v} + \frac{\delta_{j,v,t}}{s_i} & \quad \forall j, \forall i, \forall t, \forall v \in T(j) \end{aligned} \quad (6)$$

$$\sum_{v \in T(j)} \delta_{j,v} \leq w_j \quad \forall j \quad (7)$$

$$\sum_{v \in T(j)} \delta_{j,v,t} \leq w_j \quad \forall j, t \quad (8)$$

We now give some intuition about these dual variables. The quantity  $\delta_{j,v,t}$  should be thought of the contribution (at time  $t$ ) towards the weighted flow-time of  $j$ . Similarly,  $\delta_{j,v}$  is *global* contribution of  $v$  towards the flow-time of  $v$ . (In the integral case,  $\delta_{j,v}$  would be  $w_j$  for the task which finishes last. If there are several such tasks,  $\delta_{j,v}$  would be non-zero only for such tasks only and would add up to  $w_j$ ). The quantity  $\alpha_{j,v}$  can be thought of as  $v$ 's contribution towards the total weighted flow-time, and  $\beta_{i,t}$  is roughly the queue size at time  $t$  on machine  $i$ . Constraint (6) upper bounds  $\alpha_{j,v}$  in terms of the other dual variables. More intuition about these variables can be found in §4.2.

## 4.1 Simplifying the dual LP

Before interpreting the dual variables, we rewrite the dual LP and add some additional constraints. Define additional variables  $\alpha_{j,v,t}$  for each job  $j$  and task  $v \in T(j)$  and time  $t$ , such that variable  $\alpha_{j,v} = \sum_t \alpha_{j,v,t}$ . We add a new constraint:

$$\sum_{v \in T(j)} \alpha_{j,v,t} \leq w_j. \quad (9)$$

This condition is not a requirement in the dual LP, but we will set  $\alpha_{j,v,t}$  to satisfy it. Assuming this, we set  $\delta_{j,v,t} := \alpha_{j,v,t}$  for all jobs  $j$ , tasks  $v \in T(j)$  and times  $t$ ; feasibility of (9) implies that of (8). Moreover, (6) simplifies to

$$\sum_{t' \geq t} \frac{\alpha_{j,v,t'}}{p_v} \leq \frac{\beta_{i,t}}{s_i} + \frac{\delta_{j,v}}{s_i}.$$

Observe that we can write  $p_v$  as the sum of the rates, and hence as  $p_v = \sum_{t'} L_v^{t'}$ . Since this is at least  $\sum_{t' \geq t} L_v^{t'}$  for any  $t$ , we can substitute above, and infer that it suffices to verify the following condition for all tasks  $v \in T(j)$ , time  $t$ , and time  $t' \geq t$ :

$$\alpha_{j,v,t'} \leq \frac{\beta_{i,t} \cdot L_v^{t'}}{s_i} + \frac{\delta_{j,v} \cdot L_v^{t'}}{s_i}. \quad (10)$$

Henceforth, we ensure that our duals (including  $\alpha_{j,v,t}$ ) satisfy (9),(10) and (7).

## 4.2 Interpreting the Duals and the High-Level Proof Idea

We give some intuition about the dual variables, which will be useful for understanding the subsequent analysis. We set dual variables  $\alpha_{j,v}$  such that for any job  $j$ , the sum  $\sum_{v \in T(j)} \alpha_{j,v}$  is (approximately) the weighted completion of job  $j$ . This ensures that  $\sum_{j,v} \alpha_{j,v}$  is the total weighted completion of the jobs. One way of achieving this is as follows: for every time  $t$  and task-job pair  $(j,v)$  we define  $\alpha_{j,v,t}$  variables such that they add up to be  $w_j$  if job  $j$  is unfinished at time  $t$  (i.e., (9) is satisfied with equality). If  $\alpha_{j,v}$  is set to  $\sum_t \alpha_{j,v,t}$ , then these  $\alpha_{j,v}$  variables would add up to the weighted completion time of  $j$ .

The natural way of defining  $\alpha_{j,v,t}$  is to evenly distribute the weight of  $j$  among all the alive tasks at time  $t$ , i.e., to set  $\alpha_{j,v,t} = \frac{w_j}{T^t(j)}$ . This idea works if we only want to show that the algorithm is  $O(\log n)$ -competitive, but does not seem to generalize if we want to show  $O(K)$ -competitiveness. The reason for this will be clearer shortly, when we discuss the  $\delta_{j,v}$  variables.

Now we discuss  $\beta_{i,t}$  dual variables. We set these variables so that  $\sum_t \beta_{i,t}$  is a constant (less than 1) times the total weighted completion time. This ensures that the objective value of the dual LP is also a constant times the total weighted completion time. A natural idea (ignoring constant factors for now) is to set  $\beta_{i,t} = \frac{w(A^t)}{K m_\ell}$ , where  $A^t$  is the set of alive jobs at time  $t$  and  $\ell$  is the speed class of machine  $i$ . Since we have put an  $\Omega(K)$  term in the denominator of  $\beta_{i,t}$  (and no such term in the definition of  $\alpha_{j,v}$ ), ensuring the feasibility of (6) would require a speed augmentation of  $\Omega(K)$ .

Finally, consider the  $\delta_{j,v}$  dual variables. As (7) suggests, setting  $\delta_{j,v}$  is the same as deciding how to distribute the weight  $w_j$  among the tasks in  $T(j)$ . Notice, however, that this distribution cannot depend on time (unlike  $\alpha_{j,v,t}$  where we were distributing  $w_j$  among all the alive tasks at time  $t$ ). In the ideal scenario, tasks finishing later should get high  $\delta_{j,v}$  values. Since we are in the non-clairvoyant setting, we may want to set  $\delta_{j,v} = \frac{w_j}{|T(j)|}$ . We now argue this can lead to a problem in satisfying (10).

Consider the setting of a single unit-weight job  $j$  initially having  $n$  tasks, and so we set  $\delta_{j,v} = \frac{1}{n}$  for all  $v$ . Say that  $n = m_\ell$  for a large value of  $\ell$ : by the increasing capacity assumption,  $m_\ell \approx m_1 + \dots + m_\ell$ . Now consider a later point in time  $t$  when only  $n'$  tasks

remain, where  $n' = m_{\ell'}$  for some speed class  $\ell' \ll \ell$ . At this time  $t$ , each of the  $n'$  surviving tasks have  $\alpha_{j,v,t} = \frac{1}{n'}$ . But look at the RHS of (10), with machine  $i$  of speed class  $\ell$ . The rate  $L_v^t$  will be very close to  $\sigma_{\ell'}$  (again, by the increasing capacity assumption), and so both the terms would be about  $\frac{\sigma_{\ell'}}{m_{\ell}\sigma_{\ell}}$ . However,  $m_{\ell}\sigma_{\ell}$  could be much larger than  $m_{\ell'}\sigma_{\ell'}$ , and so this constraint will not be satisfied. In fact, we can hope to satisfy (10) at some time  $t$  only if  $n'$  is close to  $n$ , say at least  $n/2$ . When the number of alive tasks drops below  $n/2$ , we need to *redistribute* the weight of  $j$  among these tasks, i.e., we need to increase the  $\delta_{j,v}$  value for these tasks, to about  $\frac{1}{n/2}$ . Since these halving can happen for  $\log n$  steps, we see that (3) is violated by a factor of  $\log n$ . These ideas can be extended to give an  $O(\log n + K)$ -competitive algorithm for arbitrary inputs; see §5 for details. To get a better bound, we need a more careful setting of the dual variables, which we talk about in §6 and §7.

## 5 Analysis I: A Weaker $O(K + \log n)$ Guarantee

We start with a simpler analysis which yields an  $O(K + \log n)$ -competitiveness. This argument will not use the increasing capacity assumption from Assumption 3; however, the result gives a competitiveness of  $O(\max(K, \log n))$  which is logarithmic when  $K$  is small, whereas our eventual result will be  $O(\min(K^{O(1)}, K + \log n))$ , which can be much smaller when  $K \ll \log n$ .

► **Theorem 10.** *The scheduling algorithm in §3 is  $O(K + \log n)$ -competitive.*

**Proof.** For each job  $j$ , we arrange the tasks in  $T(j)$  in descending order of their processing requirements. (This is the opposite of the order in which they finish, since all alive tasks of a job are processed at the same rate.) Say the sequence of the tasks for a job  $j$  is  $v_1, \dots, v_r$ . We partition these tasks into *groups* with exponentially increasing cardinalities:  $T_1(j) := \{v_1\}$ ,  $T_2(j) := \{v_2, v_3\}$ , and  $T_h(j) := \{v_{2^{h-1}}, \dots, v_{2^h-1}\}$  has  $2^{h-1}$  tasks. (Assume w.l.o.g. that  $r + 1$  is a power of 2 by adding zero-sized tasks to  $T(j)$ ). Now we define the dual variables.

**Dual Variables.** Define  $\gamma := 2 \max\{K, \log_2 n\}$ .

- For a time  $t$  and machine  $i$  of speed class  $\ell$ , let  $A^t$  denote the set of active (unfinished) jobs at time  $t$ , and define  $\beta_{i,t} := \frac{w(A^t)}{m_{\ell} \cdot \gamma}$ .
- For job  $j$  and a task  $v \in T_h(j)$  in the  $h$ -th group, define  $\delta_{j,v} := \frac{w_j}{2^{h-1} \cdot \gamma}$ .
- In order to define  $\alpha_{j,v}$ , we first define quantities  $\alpha_{j,v,t}$  for every time  $t$ , and then set  $\alpha_{j,v} := \sum_t \alpha_{j,v,t}$ . At time  $t$ , recall that  $T^t(j)$  is the set of alive tasks of job  $j$ , and define

$$\alpha_{j,v,t} := \frac{w_j}{|T^t(j)|} \cdot \mathbf{1}_{(v \text{ alive at time } t)} = \frac{w_j}{|T^t(j)|} \cdot \mathbf{1}_{(v \in T^t(j))}.$$

This “spreads” the weight of  $j$  equally among its alive tasks.

Having defined the dual variables, we first argue that they are feasible.

► **Lemma 11** (Dual feasibility). *The dual variables defined above always satisfy the constraints (9), (7) and (10) for a speed-up factor  $\gamma \geq 2 \max\{K, \log_2 n\}$ .*

**Proof.** To check feasibility of (7), consider a job  $j$  and observe that

$$\sum_{v \in T(j)} \delta_{j,v} = \sum_h \sum_{v \in T_h(j)} \delta_{j,v} = \sum_h \sum_{v \in T_h(j)} \frac{w_j}{2^{h-1} \cdot \gamma} = \sum_h \frac{w_j}{\gamma} \leq w_j,$$

because  $|T_h(j)| = 2^{h-1}$  and there are at most  $\log_2 n \leq \gamma$  distinct groups. Feasibility of (9) also follows easily. It remains to check (10) for a job  $j$ , task  $v$ , machine  $i$  and times  $t' \leq t$ .

If  $v$  is not alive at time  $t'$ , then  $\alpha_{j,v,t'}$  is 0, and (10) follows trivially. Else,  $v \in T^{t'}(j)$ , and suppose  $v \in T_h(j)$ . This means the jobs in  $T_1(j), \dots, T_{h-1}(j)$  are also alive at time  $t'$ , so  $|T^{t'}(j)| \geq 1 + 2 + \dots + 2^{h-2} + 1 = 2^{h-1}$ . Furthermore, suppose the tasks in  $T^{t'}(j)$  belong to block  $B$  (defined in §3.1), and let  $\ell^*$  be the speed class with the slowest machines among the associated machines  $m(B)$ . Let  $\ell$  denote the speed class of machine  $i$  (considered in (10)). Two cases arise: the first is when  $\ell \geq \ell^*$ , where  $L_v^{t'} \geq \gamma\sigma_{\ell^*} \geq \gamma\sigma_\ell = \gamma s_i$ , so (10) holds because

$$\alpha_{j,v,t'} = \frac{w_j}{|T^{t'}(j)|} \leq \frac{w_j}{2^{h-1}} = \gamma \cdot \delta_{j,v} \leq \frac{\delta_{j,v} \cdot L_v^{t'}}{s_i}.$$

The second case is  $\ell < \ell^*$ : Let  $V \subseteq A^{t'}$  be the set of jobs which are frozen by the moment  $v$  freezes. In other words,  $V$  contains tasks in block  $B$  and the blocks before it. Applying the second statement in Corollary 8 with  $V'' = V$ ,

$$\frac{w_j}{|T^{t'}(v)|L_v^{t'}} \leq \frac{\tilde{w}^t(V)}{\sum_{v' \in V} L_{v'}^{t'}} \leq \frac{w(V)}{\sum_{v' \in V} L_{v'}^{t'}} \leq \frac{w(A^{t'})}{\gamma \cdot m_\ell \sigma_\ell},$$

where the last inequality uses the fact that all machines of speed class  $\ell$  are busy processing jobs in  $V$ . Therefore,

$$\alpha_{j,v,t'} = \frac{w_j}{|T^{t'}(j)|} \leq \frac{w(A^{t'}) \cdot L_v^{t'}}{\gamma \cdot m_\ell \sigma_\ell} \leq \frac{\beta_{i,t} \cdot L_v^{t'}}{s_i},$$

the last inequality using the definition of  $\beta_{i,t}$  and that  $w(A^t) \geq w(A^{t'})$ .  $\blacktriangleleft$

Finally, we show that the dual objective value for this setting of dual variables is close to the primal value. It is easy to check that  $\sum_j \alpha_j = \sum_t w(A^t)$ , which is the total weighted completion time of the jobs. Moreover,

$$\sum_{i,t} \beta_{i,t} = \sum_t \sum_\ell \sum_{i:s_i=\sigma_\ell} \frac{w(A^t)}{m_\ell \gamma} = \frac{K}{\gamma} \cdot \sum_t w(A^t).$$

Since we chose speedup  $\gamma = 2 \max\{K, \log_2 n\}$ , we have  $K \leq \gamma/2$  and the dual objective value  $\sum_{j,v} \alpha_{j,v} - \sum_{i,t} \beta_{i,t}$  is at least half of the total weighted completion time (primal value). This completes the proof of Theorem 10.  $\blacktriangleleft$

## 6 Analysis II: An Improved Guarantee for a Single Job

We want to show that the competitiveness of our algorithm just depends on  $K$ , the number of speed classes. To warm up, in this section we consider the special case of a single job; in §7 we consider the general case. As was shown in Proposition 5, any algorithm has competitive ratio  $\Omega(K)$  even in the case of a single job. We give a matching upper bound using dual fitting for an instance *with a single job*  $j$ , say of weight 1, when the machines satisfy Assumption 3.

► **Theorem 12.** *If the machines satisfy Assumption 3, the scheduling algorithm in §3 is  $O(K^2)$ -competitive for a single job.*

### 6.1 The Intuition Behind the Improvement

The analysis in §5 incurred  $\Omega(\log n)$ -competitive ratio because we divided the execution of the tasks of each job into  $O(\log n)$  *epochs*, where each epoch ended when the number of tasks halved. In each such epoch, we set the  $\delta_{j,v}$  variables by distributing the job's weight evenly among all tasks alive at the beginning of the epoch. A different way to define epochs would

be to let them correspond to the time periods when the number of alive tasks falls in the range  $(m_\ell, m_{\ell+1})$ . This would give us only  $K$  epochs. There is a problem with this definition: as the number of tasks vary in the range  $(m_\ell, m_{\ell+1})$ , the rate assigned to tasks varies from  $\sigma_\ell$  to  $\sigma_{\ell+1}$ . Indeed, there is a transition point  $\tilde{m}_\ell$  in  $(m_\ell, m_{\ell+1})$  such that the rate assigned to the tasks stays close to  $\sigma_{\ell+1}$  as long as the number of tasks lie in the range  $(\tilde{m}_\ell, \sigma_{\ell+1})$ ; but if the number of tasks lie in the range  $(m_\ell, \tilde{m}_\ell)$ , the assigned rate may not stay close to any fixed value. However, in this range, the *total* processing rate assigned to all the tasks stays close to  $m_\ell \sigma_\ell$ .

It turns out that our argument for an epoch (with minor modifications) works as long as one of these two facts hold during an epoch: (i) the total rate assigned to the tasks stays close to  $m_\ell \sigma_\ell$  for some speed class  $\ell$  (even though the number of tasks is much larger than  $m_\ell$ ), or (ii) the actual rate assigned to the tasks stays close to  $\sigma_\ell$ . Thus we can divide the execution of the job into  $2K$  epochs, and get an  $O(K)$ -competitive algorithm. In this section, we prove this for a single job; we extend to the case of multiple jobs in §7 (with a slightly worse competitiveness).

## 6.2 Defining the New Epochs

Before defining the dual variables, we begin with a definition. For each speed class  $\ell$ , define the *threshold*  $\tilde{m}_\ell$  to be the following:

$$\tilde{m}_\ell := \frac{1}{\sigma_{\ell+1}} (\sigma_1 m_1 + \dots + \sigma_\ell m_\ell). \quad (11)$$

The parameter  $\tilde{m}_\ell$  is such that the processing capacity of  $\tilde{m}_\ell$  machines of class  $\ell + 1$  equals the combined processing capacity of machines of class at most  $\ell$ . The increasing capacity assumption implies  $m_\ell < \tilde{m}_\ell < m_{\ell+1}$ , as formalized below:

▷ **Claim 13.** Define  $M_\ell := m_1 + \dots + m_\ell$  and  $\tilde{M}_\ell := M_\ell + \tilde{m}_\ell$ . Under the increasing capacity Assumption 3 and  $\kappa = 2$ , for any speed class  $\ell$ , we have

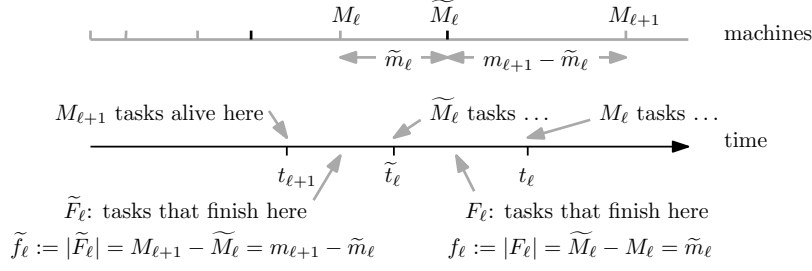
- (a)  $2\tilde{m}_\ell \leq m_{\ell+1}$  and so,  $\tilde{M}_\ell \leq M_{\ell+1}$ ,
- (b)  $\tilde{M}_\ell \geq 2M_\ell$ ,
- (c)  $m_\ell \sigma_\ell \geq \frac{1}{2} \tilde{m}_\ell \sigma_{\ell+1}$ , and
- (d)  $\tilde{m}_\ell \geq 2m_\ell$ .

*Proof.* Fact (a) follows from the increasing capacity assumption and the definition of the threshold, since  $2\tilde{m}_\ell \sigma_{\ell+1} \leq \sigma_{\ell+1} m_{\ell+1}$ . This implies  $\tilde{M}_\ell = M_\ell + \tilde{m}_\ell \leq M_\ell + m_{\ell+1} \leq M_{\ell+1}$ . Proving (b) is equivalent to showing  $\tilde{m}_\ell \geq M_\ell$ , which follows from the definition of  $\tilde{m}_\ell$  and the fact that  $\sigma_{\ell+1} < \sigma_i$  for all  $i \leq \ell$ . The last two statements also follow from the increasing capacity assumption. ◁

We identify a set of  $2K$  *break-points* as follows: for each speed class  $\ell$ , let  $t_\ell$  denote the first time when  $M_\ell$  alive tasks remain. Similarly, let  $\tilde{t}_\ell$  be the first time when exactly  $\tilde{M}_\ell$  alive tasks remain. Note that  $t_{\ell+1} < \tilde{t}_\ell < t_\ell$ . Let  $\tilde{F}_\ell$  be the tasks which finish during  $[t_{\ell+1}, \tilde{t}_\ell]$ , and  $F_\ell$  be those which finish during  $[\tilde{t}_\ell, t_\ell]$ . Let  $\tilde{f}_\ell$  and  $f_\ell$  denote the cardinality of  $\tilde{F}_\ell$  and  $F_\ell$  respectively. Note that  $\tilde{f}_\ell = M_{\ell+1} - \tilde{M}_\ell = m_{\ell+1} - \tilde{m}_\ell$ ,  $f_\ell = \tilde{M}_\ell - M_\ell = \tilde{m}_\ell$ .

▷ **Claim 14.** For any speed class  $\ell$ , we have  $f_\ell \leq \tilde{f}_\ell \leq f_{\ell+1}$ .

*Proof.* The first statement requires that  $\tilde{m}_\ell \leq m_{\ell+1} - \tilde{m}_\ell$ . This is the same as  $2\tilde{m}_\ell \leq m_{\ell+1}$ , which follows from Claim 13 (a). The second statement requires that  $m_{\ell+1} - \tilde{m}_\ell \leq \tilde{m}_{\ell+1}$ , i.e.,  $m_{\ell+1} \leq \tilde{m}_\ell + \tilde{m}_{\ell+1}$ . But  $m_{\ell+1} \leq \tilde{m}_{\ell+1}$  (by Claim 13 (d)), hence the proof. ◁



■ **Figure 1** Defining breakpoints.

Next we set the duals. Although it is possible to directly argue that the delay incurred by the job in each epoch is at most (a constant times) the optimal objective value, the dual fitting proof generalizes to the arbitrary set of jobs.

### 6.3 Setting the Duals

Define the speed-up  $\gamma \geq 2K$ . We set the duals as:

- Define  $\delta_{j,v} := \begin{cases} \frac{1}{2K \cdot f_\ell} & \text{if } v \in F_\ell \\ \frac{1}{2K \cdot \tilde{f}_\ell} & \text{if } v \in \tilde{F}_\ell \end{cases}$ .
- For machine  $i$  of class  $\ell$ , define  $\beta_{i,t} := \frac{1}{2K \cdot m_\ell} \cdot \mathbf{1}_{(\text{not all tasks finished})}$ .
- Finally, as in §5, we define  $\alpha_{j,v,t}$  for each task  $v$  of job  $j$ , and then set  $\alpha_{j,v} := \sum_t \alpha_{j,v,t}$ . To define  $\alpha_{j,v,t}$ , we consider two cases (we use  $n_t$  to denote the number of alive tasks at time  $t$ ):
  1.  $n_t \in [M_\ell, \tilde{M}_\ell]$  for some  $\ell$ : Then  $\alpha_{j,v,t} := (1/n_t) \cdot \mathbf{1}_{(v \text{ alive at time } t)}$ .
  2.  $n_t \in [\tilde{M}_\ell, M_{\ell+1}]$  for some  $\ell$ : Then  $\alpha_{j,v,t} := (1/f_\ell) \cdot \mathbf{1}_{(v \in F_\ell)}$ .

Note the asymmetry in the definition. It arises because in the first case, the *total* speed of machines processing a task is (up to a constant)  $m_\ell \sigma_\ell$ , whereas in the second case the *average* speed of such machines is about  $\sigma_{\ell+1}$ .

► **Lemma 15** (Dual feasibility). *The dual variables defined above always satisfy the constraints (7) and (9), and satisfy constraint (10) for speed-up  $\gamma \geq 2K$ .*

**Proof.** It is easy to check from the definition of  $\delta_{j,v}$  and  $\alpha_{j,v,t}$  that the dual constraints (7) and (9) are satisfied. It remains to verify constraint (10) (re-written below) for any task  $v$ , machine  $i$ , times  $t$  and  $t' \geq t$ .

$$\alpha_{j,v,t'} \leq \frac{L_v^{t'}}{s_i} \cdot (\beta_{i,t} + \delta_{j,v}). \quad ((10) \text{ repeated})$$

As in the definition of  $\alpha_{j,v,t'}$ , there are two cases depending on where  $n_{t'}$  lies. First assume that there is class  $\ell^*$  such that  $M_{\ell^*} \leq n_{t'} < \tilde{M}_{\ell^*}$ . Assume that  $v$  is alive at time  $t'$  (otherwise  $\alpha_{j,v,t'}$  is 0), so  $\alpha_{j,v,t'} = \frac{1}{n_{t'}}$ , where  $n_{t'}$  is the number of alive tasks at time  $t'$ . Being alive at this time  $t'$ , we know that  $v$  will eventually belong to some  $F_\ell$  with  $\ell \leq \ell^*$ , or in some  $\tilde{F}_\ell$  with  $\ell < \ell^*$ . So by Claim 14,  $\delta_{j,v} \geq \frac{1}{2K \cdot f_{\ell^*}}$ . Moreover, let  $i$  be a machine of some class  $\ell$ , so  $s_i = \sigma_\ell$ . Hence, it is enough to verify the following in order to satisfy (10):

$$\frac{1}{n_{t'}} \leq \frac{L_v^{t'}}{\sigma_\ell} \cdot \left( \frac{1}{2K \cdot m_\ell} + \frac{1}{2K \cdot f_{\ell^*}} \right). \quad (12)$$

Two subcases arise, depending on how  $\ell$  and  $\ell^*$  relate – in each we show that just one of the terms on the right is larger than the left.

### 3:12 Bag-Of-Tasks Scheduling on Related Machines

- $\ell^* \geq \ell$ : Since at least  $M_{\ell^*}$  tasks are alive at this time, the total speed assigned to all the alive tasks at time  $t'$  is at least  $\gamma \cdot \sigma_{\ell^*} m_{\ell^*}$ . Therefore,  $L_v^{t'} \geq \frac{\gamma \cdot m_{\ell^*} \sigma_{\ell^*}}{n_{t'}}$ . Now using  $\gamma \geq 2K$ , we get

$$\frac{L_v^{t'}}{2K \cdot m_{\ell} \sigma_{\ell}} \geq \frac{m_{\ell^*} \sigma_{\ell^*}}{m_{\ell} \sigma_{\ell}} \cdot \frac{1}{n_{t'}} \geq \frac{1}{n_{t'}},$$

where the last inequality follows from the increasing capacity assumption.

- $\ell^* \leq \ell - 1$ : The quantity  $L_v^{t'}$  is the total speed of the machines which are busy at time  $t'$ , which is at least  $\gamma(m_1 \sigma_1 + \dots + m_{\ell^*} \sigma_{\ell^*}) = \gamma \cdot \tilde{m}_{\ell^*} \sigma_{\ell^*+1}$ . Again, using  $\gamma \geq 2K$ , we get

$$\frac{L_v^{t'} \cdot n_{t'}}{2K \cdot f_{\ell^*} \sigma_{\ell}} \geq \frac{\tilde{m}_{\ell^*} \sigma_{\ell^*+1}}{f_{\ell^*} \sigma_{\ell}} \geq 1$$

because  $\sigma_{\ell^*+1} \geq \sigma_{\ell}$  and  $\tilde{m}_{\ell^*} = f_{\ell^*}$ .

Thus, (12) is satisfied in both the above subcases.

Next we consider the case when there is a speed class  $\ell^*$  such that  $\tilde{M}_{\ell^*} < n_{t'} \leq M_{\ell^*+1}$ . We can assume that  $v \in F_{\ell^*}$ , otherwise  $\alpha_{j,v,t'}$  is 0; this means  $\delta_{v,j} = \frac{1}{2K \cdot f_{\ell^*}}$ . Since  $\alpha_{j,v,t'} = \frac{1}{f_{\ell^*}} = \frac{1}{m_{\ell^*}}$ , and  $L_v^{t'} \geq \gamma \cdot \sigma_{\ell^*+1}$ , the expression (10) follows from showing

$$\frac{1}{\tilde{m}_{\ell^*}} \leq \frac{\gamma}{\sigma_{\ell}} \cdot \left( \frac{1}{2K \cdot m_{\ell}} + \frac{1}{2K \cdot f_{\ell^*}} \right) \cdot \sigma_{\ell^*+1}. \quad (13)$$

Since  $\gamma \geq 2K$ , we can drop those terms. Again, two cases arise:

- $\ell^* \geq \ell$ : By definition,  $\sigma_{\ell^*+1} \cdot \tilde{m}_{\ell^*} \geq \sigma_{\ell^*} m_{\ell^*} \geq \sigma_{\ell} m_{\ell}$  (by the increasing capacity assumption).
- $\ell^* \leq \ell - 1$ : Since  $f_{\ell^*} = \tilde{m}_{\ell^*}$  and  $\sigma_{\ell} \leq \sigma_{\ell^*+1}$ , this case also follows easily. ◀

**Proof of Theorem 12.** Having checked dual feasibility in Lemma 15, consider now the objective function. For any time  $t$  when at least one task is alive,  $\sum_v \alpha_{j,v,t} = 1$ . Therefore,  $\sum_v \alpha_{j,v}$  is the makespan. Also,  $\sum_i \beta_{i,t} = 1/2$  as long as there are unfinished tasks, so  $\sum_{i,t} \beta_{i,t}$  is half the makespan, and the objective function  $\sum_v \alpha_{j,v} - \sum_{i,t} \beta_{i,t}$  also equals half the makespan. Since we had assumed  $\gamma = O(K)$ -speedup, the algorithm is  $O(K)$ -competitive. ◀

## 7 Analysis III: Proof for $\tilde{O}(K^3)$ Guarantee

We now extend the ideas from the single job case to the general case. We only discuss the proof outline here, and refer the readers to the full version for details. For time  $t$ , let  $A^t$  be the set of alive jobs at time  $t$ . Unlike the single job case where we had only one block, we can now have multiple blocks. While defining  $\alpha_{j,v,t}$  in the single job case, we had considered two cases: (i) the rate assigned to each task stayed close to  $\sigma_{\ell}$  for some class  $\ell$  (this corresponded to  $n_t \in [\tilde{M}_{\ell-1}, M_{\ell})$ ), and (ii) the total rate assigned to each task was close to  $m_{\ell} \sigma_{\ell}$  for speed class  $\ell$  (this corresponded to  $n_t \in [M_{\ell}, \tilde{M}_{\ell})$ ). We extend these notions to blocks as follows: *Simple blocks*: A block  $B$  is said to be *simple* w.r.t. to a speed class  $\ell$  if the average rate assigned to the tasks in  $B$  is close to  $\sigma_{\ell}$ . Similarly a job  $j$  is said to be simple w.r.t. a speed class  $\ell$  if all the alive tasks in it are assigned rates close to  $\sigma_{\ell}$  (recall that all alive tasks in a job are processed at the same rate). All the jobs in a simple block  $B$  may not be simple (w.r.t. the same speed class  $\ell$ ), but we show that a large fraction of jobs (in terms of weight) in  $B$  will be simple. Thus, it is enough to account for the weight of simple jobs in  $B$ . This is analogous to case (i) mentioned above (when there is only one job and tasks in it receive rate



close to  $\sigma_\ell$ ). In §6, we had defined  $\alpha_{j,v,t}$  for such time  $t$  as follows: we consider only those tasks which survive in  $F_\ell$ , and then evenly distribute  $w_j$  among these tasks. The analogous definition here would be as follows: let  $\tau_{\ell,j}$  be the *last* time when  $j$  is simple w.r.t. the speed class  $\ell$ . We define  $\alpha_{j,v,t}$  by evenly distributing  $w_j$  among those tasks in  $v$  which are alive at  $\tau_{\ell,j}$ . We give details in the full version.

*Long blocks:* The total speed of the machines in this block stays close to  $m_\ell \sigma_\ell$  for some speed class  $\ell$ . Again, inspired by the definitions in §6, we assign  $\alpha_{j,v,t}$  for tasks  $v \in B$  by distributing  $w(B)$  to these tasks (in proportion to the rate assigned to them). From the perspective of a job  $j$  which belongs to a long block  $B$  w.r.t. a speed class  $\sigma_\ell$  at a time  $t$ , the feasibility of (6) works out provided for *all* subsequent times  $t'$  when  $j$  again belongs to such a block  $B'$ , we have  $w(B')$  and  $w(B)$  remain close to each other. If  $w(B')$  exceeds (say)  $2w(B)$ , we need to reassign a new set of  $\delta_{j,v}$  values for  $v$ . To get around this problem we require that long blocks (at a time  $t$ ) also have weight at least  $w(A^t)/(10K)$ . With this requirement, the doubling step mentioned above can only happen  $O(\log K)$  times (and so we incur an additional  $O(\log K)$  in the competitive ratio). The details are given in the full version. Blocks which were cheaper than  $w(A^t)/(10K)$  do not create any issue because there can be at most  $K$  of them, and so their total weight is small in comparison to  $w(A^t)$ .

*Short blocks:* Such blocks  $B$  straddle two speed classes, say  $\ell$  and  $\ell + 1$ , but do not contain too many machines of either class (otherwise they will fall into one of the two categories above). We show in the full version that the total weight of such blocks is small compared to  $w(A^t)$ . The intuitive reason is as follows: for any two consecutive short blocks  $B_1$  and  $B_2$ , there must be blocks in between them whose span is much longer than  $B_2$ . Since these blocks freeze before  $B_2$ , their total weight would be large compared to  $w(B_2)$ .

In the overall analysis, we charge short blocks to simple and long blocks, and use dual fitting as indicated above to handle simple and long blocks.

## 8 Discussion

Several interesting problems remain open. (i) Can we close the gap between lower bound of  $\Omega(K)$  and upper bound of  $O(K^3 \log^2 K)$ ? (ii) Can we prove an analogous result for weighted *flow-time* (with speed augmentation)? (iii) Can we generalize this result to the unrelated machines setting? (iv) Our lower bound of  $\Omega(K)$ -competitive ratio relies on non-clairvoyance; can we prove a better bound if the processing times of tasks are known at their arrival times?

---

### References

- 1 Kunal Agrawal, Jing Li, Kefu Lu, and Benjamin Moseley. Scheduling parallel DAG jobs online to minimize average flow time. In *Proceedings of SODA*, pages 176–189, 2016.
- 2 C. Anglano and M. Canonico. Scheduling algorithms for multiple bag-of-task applications on desktop grids: A knowledge-free approach. In *2008 IEEE International Symposium on Parallel and Distributed Processing*, pages 1–8, 2008.
- 3 Abbas Bazzi and Ashkan Norouzi-Fard. Towards tight lower bounds for scheduling problems. In *Proceedings of ESA*, pages 118–129, 2015.
- 4 Anne Benoit, Loris Marchal, Jean-Francois Pineau, Yves Robert, and Frédéric Vivien. Scheduling concurrent bag-of-tasks applications on heterogeneous platforms. *IEEE Trans. Computers*, 59(2):202–217, 2010.
- 5 Fabián A. Chudak and David B. Shmoys. Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds. *J. Algorithms*, 30(2):323–343, 1999.

- 6 José R. Correa, Martin Skutella, and José Verschae. The power of preemption on unrelated machines and applications to scheduling orders. In *Proceedings of APPROX/RANDOM*, pages 84–97, 2009.
- 7 Naveen Garg, Anupam Gupta, Amit Kumar, and Sahil Singla. Non-clairvoyant precedence constrained scheduling. In *Proceedings of ICALP*, pages 63:1–63:14, 2019.
- 8 Leslie A. Hall, Andreas S. Schulz, David B. Shmoys, and Joel Wein. Scheduling to minimize average completion time: off-line and on-line approximation algorithms. *Math. Oper. Res.*, 22(3):513–544, 1997.
- 9 Shi Li. Scheduling to minimize total weighted completion time via time-indexed linear programming relaxations. In *Proceedings of FOCS*, pages 283–294, 2017.
- 10 Ioannis A. Moschakis and Helen D. Karatza. Multi-criteria scheduling of bag-of-tasks applications on heterogeneous interlinked clouds with simulated annealing. *J. Syst. Softw.*, 101:1–14, 2015.
- 11 Alix Munier, Maurice Queyranne, and Andreas S. Schulz. Approximation bounds for a general class of precedence constrained parallel machine scheduling problems. In *Proceedings of IPCO*, volume 1412, pages 367–382, 1998.
- 12 Maurice Queyranne and Maxim Sviridenko. A  $(2+\epsilon)$ -approximation algorithm for generalized preemptive open shop problem with minsum objective. In *Proceedings of IPCO*, volume 2081, pages 361–369, 2001.
- 13 Julien Robert and Nicolas Schabanel. Non-clairvoyant scheduling with precedence constraints. In *Proceedings of SODA*, pages 491–500, 2008.

## A Missing Proofs of Section 2

► **Proposition 4.** *An arbitrary instance can be transformed into one satisfying Assumption 3 by losing a factor  $O(\gamma K)$  in the competitive ratio.*

**Proof.** We show how to transform the instance so that it satisfies the increasing capacity assumption, while losing only  $O(K)$ -factor in the competitive ratio. For sake of brevity, let  $\kappa$  denote the constant 64.

For a speed class  $\ell$ , let  $C_\ell$  denote  $m_\ell \sigma_\ell$ , i.e., the total processing capacity of the machines in this speed class. Starting from speed class 1, we construct a subset  $X$  of speed classes as follows: if  $\ell$  denotes the last speed class added to  $X$ , then let  $\ell' > \ell$  be the smallest class such that  $C_{\ell'} \geq 2\kappa C_\ell$ . We add  $\ell'$  to  $X$  and continue this process till we have exhausted all the speed classes.

Consider the instance  $\mathcal{I}'$  in which the set of jobs is the same as those in  $\mathcal{I}$ , but there are  $K m_\ell$  machines of speed class  $\ell$  for each  $\ell \in X$ . For a speed class  $\ell \in X$ , let  $C'_\ell$  denote  $2\kappa K m_\ell \sigma_\ell$ , which is at most the total capacity of the speed class  $\ell$  machines in  $\mathcal{I}'$ . Let us now consider the optimal solutions of the two instances. We first observe that  $\text{opt}(\mathcal{I}') \leq \text{opt}(\mathcal{I})$ . Consider two consecutive speed classes  $\ell_1 < \ell_2$  in  $X$ . From the definition of  $X$ , we see that  $C'_{\ell_1} \geq \sum_{l=\ell_1}^{\ell_2-1} C_l$ . Therefore all the processing done by a solution to  $\mathcal{I}$  on machines of speed class  $[\ell_1, \ell_2)$  during a timeslot  $[t, t + 1]$  can be performed on machines of speed class  $\ell_1$  in  $\mathcal{I}'$  during the same timeslot. Therefore,  $\text{opt}(\mathcal{I}') \leq \text{opt}(\mathcal{I})$ .

For the converse statement, it is easy to see that if we give  $2\kappa K$  speedup to each machine in  $\mathcal{I}$ , then the processing capacity of each speed class in  $\mathcal{I}$  is at least that in  $\mathcal{I}'$ . Therefore,  $\text{opt}(\mathcal{I}) \leq 2\kappa K \text{opt}(\mathcal{I}')$ . Therefore, replacing  $\mathcal{I}$  by  $\mathcal{I}'$  will result in  $O(\kappa K)$  loss in competitive ratio. It is also easy to check that  $\mathcal{I}'$  satisfies increasing capacity assumption.

Observe that the conversion from  $\mathcal{I}$  to  $\mathcal{I}'$  can be easily done at the beginning – we just need to identify the index set  $X$ , and use only these for processing. The factor  $K$  loss in competitive ratio is also tight for the instance  $\mathcal{I}$  where all speed classes have the same capacity. ◀

## B

 Missing proofs of Section 3

► **Lemma 6.** *A schedule  $S$  is feasible if for every time  $t$  and every value of  $k$ :*

( $\star$ ) *the total rate assigned to any subset of  $k$  tasks is at most  $\gamma \cdot S_k$ .*

**Proof.** The rates assigned to tasks change only when one of these events happen: (i) a new job  $j$  arrives, (ii) an existing task finishes. Assuming that the job sizes, release dates are integers, we can find a suitable  $\delta > 0$  (which will also depend on the speeds of the machines) such that all the above events happen at integral multiples of  $\delta$ .

Consider an interval  $[t, t + \delta)$ , where  $t$  is an integral multiple of  $\delta$ . We need to show that if  $L_v^t$ 's satisfy the condition ( $\star$ ), then we can build a feasible schedule during  $[t, t + \delta)$ . By feasibility, we mean that each task  $v$  can be processed to an extent of  $\bar{p}_v := L_v^t \cdot \delta$  extent and at any point of time, it gets processed on at most one machine.

We follow a greedy strategy to build the schedule. Suppose we have built the schedule till time  $t' \in [t, t + \delta)$ . At time  $t'$ , we order the tasks in descending order of the remaining processing requirement for this slot (at time  $t$ , each task  $v$  has processing requirement of  $\bar{p}_v$ ). Let the ordered tasks at time  $t'$  be  $v_1, \dots, v_n$ . We schedule  $v_i$  on machine  $i$ .

Suppose for the sake of contradiction, a task  $v^*$  is not able to complete  $\bar{p}_{v^*}$  amount of processing. We first make the following observation:

▷ **Claim 16.** Let  $v$  and  $v'$  be two tasks such that at some time  $t' \in [t, t + \delta)$ , we prefer  $v$  to  $v'$  in the ordering at time  $t'$ . Then if  $v'$  does not complete  $\bar{p}_{v'}$  amount of processing during  $[t, t + \delta)$ , then neither does  $v$ .

*Proof.* Since we prefer  $v$  at time  $t'$ ,  $v$  has more remaining processing time. If we never prefer  $v'$  to  $v$  after time  $t'$ , then  $v$  always has more remaining processing requirement than  $v'$  during this interval. If we prefer  $v'$  to  $v$  at some point of time during  $(t', t + \delta)$ , then it is easy to check that the remaining processing requirements for both  $v$  and  $v'$  will remain the same. The result follows easily from this observation. ◀

Starting from  $\{v^*\}$ , we build a set  $S$  of tasks which has the following property: if  $v \in S$ , then we add to  $S$  all the tasks  $v'$  such that  $v'$  was preferred over  $v$  at some point of time during  $[t, t + \delta)$ . Repeating application of Claim 16 shows that none of these tasks  $v$  complete  $\bar{p}_v$  amount of processing during  $[t, t + \delta)$ . Let  $\bar{m}$  denote  $|S|$ . We note that only tasks in  $S$  would have been processed on the first  $\bar{m}$  machines during  $[t, t + \delta)$  – otherwise, we can add more tasks to  $S$ . Since none of these tasks finish their desired amount of processing during this interval, it follows that

$$\sum_{v \in S} \bar{p}_v \geq \gamma \delta \cdot S_{\bar{m}}.$$

Since  $\bar{p}_v = \delta L_v^t$ , we see that the set of tasks in  $S$  violates ( $\star$ ). This is a contradiction, and so such a task  $v^*$  cannot exist. ◀

► **Corollary 8.** *Fix a time  $t$ . Let  $V$  be the set of tasks frozen by some moment  $\tau$ . For a task  $v \in V$ ,*

- (i) *if  $V' \subseteq V$  is any subset of tasks which freeze either at the same moment as  $v$ , or after it, then  $\frac{\tilde{w}^t(v)}{s|V'|} \geq \frac{w(V')}{S|V'|}$ .*
- (ii) *if  $V'' \subseteq V$  is any subset of tasks which freeze either at the same moment as  $v$ , or before it, then  $\frac{\tilde{w}^t(v)}{L_v^t} \leq \frac{\tilde{w}^t(V'')}{\sum_{v' \in V''} L_{v'}^t}$ .*

### 3:16 Bag-Of-Tasks Scheduling on Related Machines

**Proof.** For any task  $v'$ , let  $\tau_{v'}$  be the value of  $\tau$  at which  $v'$  freezes. We know that

$$\sum_{v' \in V} \tilde{w}^t(v') \cdot \tau_{v'} = \gamma S_{|V|}. \quad (14)$$

Since  $\sum_{v' \in V \setminus \{v\}} \tilde{w}^t(v') \cdot \tau_{v'} \leq \gamma S_{|V|-1}$  by feasibility, it follows that

$$\tilde{w}^t(v) \cdot \tau_v \geq \gamma S_{|V|}. \quad (15)$$

Now for all  $v' \in V'$ , we have  $\tau_{v'} \geq \tau_v$ , so

$$\tilde{w}^t(V') \cdot \tau_v = \sum_{v' \in V'} \tilde{w}^t(v') \cdot \tau_v \leq \sum_{v' \in V'} \tilde{w}^t(v') \cdot \tau_{v'} \leq \sum_{v' \in V} \tilde{w}^t(v') \cdot \tau_{v'} \stackrel{(14)}{=} \gamma S_{|V|}.$$

Hence, the first claim follows:

$$\frac{\tilde{w}^t(v)}{s_{|V|}} \stackrel{(15)}{\geq} \frac{\gamma}{\tau_v} \geq \frac{\tilde{w}^t(V')}{S_{|V|}}.$$

For the second claim,

$$\sum_{v' \in V''} L_{v'}^t = \sum_{v' \in V''} \tilde{w}_{v'}^t \cdot \tau_{v'} \leq \tilde{w}^t(V'') \cdot \tau_v.$$

The claim now follows by the definition  $L_v^t = \tilde{w}^t(v) \cdot \tau_v$ . ◀

## C Missing Proofs of Section 4

▷ **Claim 9.** For any schedule  $\mathcal{S}$ , there is a feasible solution to the LP of objective value at most  $2 \text{cost}(\mathcal{S})$ .

**Proof.** Consider a schedule  $\mathcal{S}$ , and let  $x_{ivt}$  be the extent of processing done on a task  $v$  (belonging to job  $j$ ) during  $[t, t + 1]$  on machine  $i$ . More formally, if the task is processed for  $\varepsilon$  units of time on machine  $i$  during this time slot, then we set  $x_{ivt}$  to  $\varepsilon \cdot s_i$ . Constraint (4) states that every task  $v$  needs to be processed to an extent of  $p_v$ , whereas (5) requires that we cannot do more than  $s_i$  unit of processing in a unit time slot on machine  $i$ . Now we verify (3). Consider a task job  $j$  and a task  $v$  belonging to it. The total processing time of  $v$  is

$$\sum_{i,t} \frac{x_{ivt}}{s_i}. \quad (16)$$

The completion time  $F_j$  of  $j$  is at least the processing time of each of the tasks in it. Finally, we check (2). Define  $F_{j,t}$  to be 1 if  $j$  is alive at time  $t$ . The RHS of this constraint is the fraction of  $v$  which is done after time  $t$ ; and so if this is non-zero, then  $F_{j,t}$  is 1. This shows the validity of this constraint.

In the objective function, the first term is the total weighted completion time of all the jobs. The second term is also the same quantity, because  $F_j$  is equal to  $\sum_{t \geq r_j} F_{j,t}$ . ◁

# Hardness of Approximation for Euclidean $k$ -Median

Anup Bhattacharya ✉

School of Computer Sciences, National Institute of Science Education and Research (NISER),  
Khurda, India

Dishant Goyal ✉

Indian Institute of Technology Delhi, India

Ragesh Jaiswal ✉

Indian Institute of Technology Delhi, India

---

## Abstract

The Euclidean  $k$ -median problem is defined in the following manner: given a set  $\mathcal{X}$  of  $n$  points in  $d$ -dimensional Euclidean space  $\mathbb{R}^d$ , and an integer  $k$ , find a set  $C \subset \mathbb{R}^d$  of  $k$  points (called centers) such that the cost function  $\Phi(C, \mathcal{X}) \equiv \sum_{x \in \mathcal{X}} \min_{c \in C} \|x - c\|_2$  is minimized. The Euclidean  $k$ -means problem is defined similarly by replacing the distance with squared Euclidean distance in the cost function. Various hardness of approximation results are known for the Euclidean  $k$ -means problem [7, 29, 17]. However, no hardness of approximation result was known for the Euclidean  $k$ -median problem. In this work, assuming the *unique games conjecture* (UGC), we provide the hardness of approximation result for the Euclidean  $k$ -median problem in  $O(\log k)$  dimensional space. This solves an open question posed explicitly in the work of Awasthi et al. [7].

Furthermore, we study the hardness of approximation for the Euclidean  $k$ -means/ $k$ -median problems in the bi-criteria setting where an algorithm is allowed to choose more than  $k$  centers. That is, bi-criteria approximation algorithms are allowed to output  $\beta k$  centers (for constant  $\beta > 1$ ) and the approximation ratio is computed with respect to the optimal  $k$ -means/ $k$ -median cost. We show the hardness of bi-criteria approximation result for the Euclidean  $k$ -median problem for any  $\beta < 1.015$ , assuming UGC. We also show a similar hardness of bi-criteria approximation result for the Euclidean  $k$ -means problem with a stronger bound of  $\beta < 1.28$ , again assuming UGC.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Facility location and clustering; Theory of computation  $\rightarrow$  Approximation algorithms analysis

**Keywords and phrases** Hardness of approximation, bicriteria approximation, approximation algorithms,  $k$ -median,  $k$ -means

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.4

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/pdf/2011.04221.pdf> [9]

**Acknowledgements** Dishant Goyal would like to thank TCS Research Scholar Program.

## 1 Introduction

We start by giving the definition of the Euclidean  $k$ -median problem.

► **Definition 1** ( $k$ -median). *Given a set  $\mathcal{X}$  of  $n$  points in  $d$ -dimensional Euclidean space  $\mathbb{R}^d$ , and a positive integer  $k$ , find a set of centers  $C \subset \mathbb{R}^d$  of size  $k$  such that the cost function  $\Phi(C, \mathcal{X}) \equiv \sum_{x \in \mathcal{X}} \min_{c \in C} \|x - c\|$  is minimized.*

The Euclidean  $k$ -means problem is defined similarly by replacing the distance with squared Euclidean distance in the cost function (i.e., replacing  $\|x - c\|$  with  $\|x - c\|^2$ ). These problems are also studied in the discrete setting where the centers are restricted to be chosen from a specific set  $L \subset \mathbb{R}^d$ , also given as input. This is known as the *discrete* version whereas the former version (with  $L = \mathbb{R}^d$ ) is known as the *continuous* version. In the approximation



© Anup Bhattacharya, Dishant Goyal, and Ragesh Jaiswal;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 4; pp. 4:1–4:23



Leibniz International Proceedings in Informatics

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

setting, the continuous version is not harder than its discrete counterpart since it is known (e.g., [22, 36]) that an  $\alpha$ -approximation for the discrete problem gives an  $\alpha + \varepsilon$  approximation for the continuous version, for arbitrary small constant  $\varepsilon > 0$ , in polynomial time. In this work, we only study the hardness of approximation for continuous version of the problem. The hardness of approximation for the discrete version thus follows from the hardness of approximation of continuous version. In the rest of the paper, we use  $k$ -means/median to implicitly mean *continuous Euclidean  $k$ -means/median* unless specified otherwise <sup>1</sup>.

The relevance of the  $k$ -means and  $k$ -median problems in various computational domains such as resource allocation, big data analysis, pattern mining, and data compression is well known. A significant amount of work has been done to understand the computational aspects of the  $k$ -means/median problems. The  $k$ -means problem is known to be NP-hard even for fixed  $k$  or  $d$  [4, 20, 33, 40]. Similar NP hardness result is also known for the  $k$ -median problem [37]. Even the 1-median problem, popularly known as the *Fermat-Weber* problem [21], is a hard problem and designing efficient algorithms for this problem is a separate line of research in itself – see for e.g. [27, 42, 12, 10, 15]. These hardness barriers motivate approximation algorithms for these problems and a lot of progress have been made in this area. For example, there are various polynomial time approximation schemes (PTASs) known for  $k$ -means and  $k$ -median when  $k$  is fixed (or constant) [36, 28, 22, 14, 25]. Similarly, various PTASs are known for fixed  $d$  [19, 23, 16]. A number of constant factor approximation algorithms are also known for  $k$ -means and  $k$ -median when  $k$  and  $d$  are considered as part of the input. For the  $k$ -means problem, constant approximation algorithms have been given [26, 2], the best being a 6.357 approximation algorithm by Ahmadian *et al.* [2]. On the negative side, there exists a constant  $\varepsilon > 0$  such that there does not exist an efficient  $(1 + \varepsilon)$ -approximation algorithm for the  $k$ -means problem, assuming  $P \neq NP$  [7, 29, 17]. The best-known hardness of approximation result for the  $k$ -means problem is 1.07 due to Cohen-Addad and Karthik [17].

The constant factor approximation algorithms for the  $k$ -median problem are also known [13, 5, 32, 11, 2]. The best known approximation guarantee for  $k$ -median is 2.633 due to Ahmadian *et al.* [2]. On the hardness side, it was known that for general metric spaces, the discrete  $k$ -median problem is hard to approximate within a factor of  $1 + 2/e$  [24]. However, unlike the Euclidean  $k$ -means problem, no hardness of approximation result was known for the Euclidean  $k$ -median problem. Resolving the hardness of approximation for the Euclidean  $k$ -median problem was left as an open problem in the work of Awasthi *et al.* [7]. They asked whether their techniques for proving the inapproximability results for Euclidean  $k$ -means can be used to prove the hardness of approximation result for the Euclidean  $k$ -median problem. From their paper,

*“It would also be interesting to study whether our techniques give hardness of approximation results for the Euclidean  $k$ -median problem.”*

In this work, assuming UGC, we solve this open problem by obtaining the hardness of approximation result for the Euclidean  $k$ -median problem. Following is one of the main results of this work.

► **Theorem 2 (Main Theorem).** *There exists a constant  $\varepsilon > 0$  such that the Euclidean  $k$ -median problem in  $O(\log k)$  dimensional space cannot be approximated to a factor better than  $(1 + \varepsilon)$ , assuming the Unique Games Conjecture.*

<sup>1</sup> In some literature, the Euclidean space implicitly means the dimension is bounded, but in our case the dimension  $d$  can be arbitrarily large

Having established the hardness of approximation results for  $k$ -means and  $k$ -median, the next natural step in the discussion is to allow more flexibility to the algorithm. One possible relaxation is to allow an approximation algorithm to choose more than  $k$  centers, say,  $\beta k$  centers (for some constant  $\beta > 1$ ) and produce a solution that is close to the optimal solution with respect to  $k$  centers. This is known as bi-criteria approximation and the following definition formalizes this notion.

► **Definition 3** ( $(\alpha, \beta)$ -approximation algorithm). *An algorithm  $\mathcal{A}$  is called an  $(\alpha, \beta)$  approximation algorithm for the Euclidean  $k$ -means/ $k$ -median problem if given any instance  $\mathcal{I} = (\mathcal{X}, k)$  with  $\mathcal{X} \subset \mathbb{R}^d$ ,  $\mathcal{A}$  outputs a center set  $F \subset \mathbb{R}^d$  of size  $\beta k$  that has the cost at most  $\alpha$  times the optimal cost with  $k$  centers. That is,*

$$\sum_{x \in \mathcal{X}} \min_{f \in F} \{D(x, f)\} \leq \alpha \cdot \min_{\substack{C \subset \mathbb{R}^d \\ |C|=k}} \left\{ \sum_{x \in \mathcal{X}} \min_{c \in C} \{D(x, c)\} \right\}$$

For the Euclidean  $k$ -means problem,  $D(p, q) \equiv \|p - q\|^2$  and for the  $k$ -median problem  $D(p, q) \equiv \|p - q\|$ .

One expects that as  $\beta$  grows, there would exist efficient  $(\alpha, \beta)$ -approximation algorithms with smaller values of  $\alpha$ . This is indeed observed in the work of Makarychev et al. [35]. For example, their algorithm gives a  $(9 + \varepsilon)$  approximation for  $\beta = 1$ ; 2.59 approximation for  $\beta = 2$ ; 1.4 approximation for  $\beta = 3$ . In other words, the approximation factor of their algorithm decreases as the value of  $\beta$  increases. Furthermore, their algorithm gives a  $(1 + \varepsilon)$ -approximation guarantee with  $O(k \log(1/\varepsilon))$  centers. Bandyapadhyay and Varadarajan [8] gave a  $(1 + \varepsilon)$  approximation algorithm that outputs  $(1 + \varepsilon)k$  centers in constant dimension. There are various other bi-criteria approximation algorithms that use distance-based sampling techniques and achieve better approximation guarantees than their non bi-criteria counterparts [3, 1, 41]. Unfortunately in these bi-criteria algorithms, at least one of  $\alpha, \beta$  is large. Ideally, we would like to obtain a PTAS with a small violation of the number of output centers. More specifically, we would like to address the following question:

*Does the Euclidean  $k$ -means or Euclidean  $k$ -median problem admit an efficient  $(1 + \varepsilon, 1 + \varepsilon)$ -approximation algorithm?*

Note that such type of bi-criteria approximation algorithms that outputs  $(1 + \varepsilon)k$  centers have been extremely useful in obtaining a constant approximation for the *capacitated*  $k$ -median problem [30, 31] for which no true constant approximation is known yet.<sup>2</sup> Therefore, the above question is worth exploring. Note that here we are specifically aiming for a PTAS since the  $k$ -means and  $k$ -median problems already admit a constant factor approximation algorithm. In this work, we give a negative answer to the above question by showing that there exists a constant  $\varepsilon > 0$  such that an efficient  $(1 + \varepsilon, 1 + \varepsilon)$ -approximation algorithm for the  $k$ -means and  $k$ -median problems does not exist assuming the Unique Games Conjecture. The following two theorems state this result more formally.

► **Theorem 4** ( $k$ -median). *For any constant  $1 < \beta < 1.015$ , there exists a constant  $\varepsilon > 0$  such that there is no  $(1 + \varepsilon, \beta)$ -approximation algorithm for the Euclidean  $k$ -median problem in  $O(\log k)$  dimensional space assuming the Unique Games Conjecture.*

<sup>2</sup> In the capacitated  $k$ -median/ $k$ -means problem there is an additional constraint on each center that it cannot serve more than a specified number of clients (or points).

► **Theorem 5** ( $k$ -means). *For any constant  $1 < \beta < 1.28$ , there exists a constant  $\varepsilon > 0$  such that there is no  $(1 + \varepsilon, \beta)$ -approximation algorithm for the Euclidean  $k$ -means problem in  $O(\log k)$  dimensional space assuming the Unique Games Conjecture. Moreover, the same result holds for any  $1 < \beta < 1.1$  under the assumption that  $P \neq NP$ .*

For simplicity, we present the proof of our results in  $O(n)$  dimensional space. However, the results easily extend to  $O(\log k)$  dimensional space using dimensionality reduction techniques of Makarychev *et al.* [34].

**Important note:** We would like to note that assuming  $P \neq NP$ , a similar hardness of approximation result for the Euclidean  $k$ -median problem using different techniques has been obtained independently by Vincent Cohen-Addad, Karthik C. S., and Euiwoong Lee. We came to know about their results through personal communication with the authors. Since their manuscript has not been published online yet, we are not able to add a citation to their work.

In the next subsection, we discuss the known results on hardness of approximation of the  $k$ -means and  $k$ -median problems in more detail.

## 1.1 Related Works

Guha and Khuller proved a  $(1 + \frac{2}{\varepsilon})$  hardness of approximation result for the discrete  $k$ -median problem for the general metric spaces [24]. The first hardness of approximation result for the Euclidean  $k$ -means problem was given by Awasthi *et al.* [7]. They obtained their result using a reduction from Vertex Cover on triangle-free graphs of bounded degree  $\Delta$  to the Euclidean  $k$ -means instances. Their reduction yields a  $(1 + \frac{\varepsilon}{\Delta})$  hardness factor for the  $k$ -means problem for some constant  $\varepsilon > 0$ . Lee *et al.* [29] showed the hardness of approximation of Vertex Cover on triangle-free graphs of bounded degree four. Using  $\Delta = 4$ , they obtained a 1.0013 hardness of approximation for the Euclidean  $k$ -means problem. Subsequently, Cohen-Addad and Karthik [17] improved the hardness of approximation to 1.07 using a modified reduction from the vertex coverage problem instead of a reduction from the vertex cover problem. Moreover, they also gave several improved hardness results for the discrete  $k$ -means/ $k$ -median problems in general and  $\ell_p$  metric spaces. In their more recent work, they also improved the hardness of approximation results for the continuous  $k$ -means/ $k$ -median problem in general metric spaces [18].

Unlike the Euclidean  $k$ -means problem, no hardness of approximation result was known for the Euclidean  $k$ -median problem. In this work, we give hardness of approximation result for the Euclidean  $k$ -median problem assuming the Unique Game Conjecture. As mentioned earlier, in an unpublished work communicated to us through personal communication, Vincent Cohen-Addad, Karthik C. S., and Euiwoong Lee have independently obtained hardness of approximation result for the Euclidean  $k$ -median problem using different set of techniques and under the assumption that  $P \neq NP$ . They also gave bi-criteria hardness of approximation results in  $\ell_\infty$ -metric for the  $k$ -means and  $k$ -median problems [18]. We would like to point out that in the bi-criteria setting, our result is the first hardness of approximation result for the Euclidean  $k$ -means/ $k$ -median problem to the best of our knowledge.

## 1.2 Technical Overview and Contributions

Awasthi *et al.* [7] proved the first hardness of approximation result for the Euclidean  $k$ -means problem. Given any instance  $\mathcal{I} = (\mathcal{X}, k)$  for the Euclidean  $k$ -means problem, they showed that there exists an  $\epsilon > 0$  such that obtaining  $(1 + \epsilon)$ -approximation for Euclidean  $k$ -means



is NP-hard. In this work we build on their techniques to prove the inapproximability result for the Euclidean  $k$ -median problem. First, we describe the reduction employed by Awasthi et al. for the Euclidean  $k$ -means problem and some related results.

*Construction of  $k$ -means instance:* Let  $(G, k)$  be a hard **Vertex Cover** instance where the graph  $G$  has bounded degree  $\Delta$ . Let  $n$  and  $m$  denote respectively the number of vertices and the number of edges in the graph. A  $k$ -means instance  $\mathcal{I} := (\mathcal{X}, k)$  with  $\mathcal{X} \subset \mathbb{R}^n$  is constructed as follows. For every vertex  $i \in V$ , we have an  $n$ -dimensional vector  $x_i \in \{0, 1\}^n$ , which has a 1 at  $i^{\text{th}}$  coordinate and 0 everywhere else. For each edge  $e = (i, j) \in E$ , a point  $x_e := x_i + x_j$  is defined in  $\{0, 1\}^n$ . The set  $\mathcal{X} := \{x_e \mid e \in E\}$  with  $m$  points in  $\mathbb{R}^n$  and parameter  $k$  define the  $k$ -means instance.

Awasthi et al. proved the following theorem based on the above construction [7].

► **Theorem 6** (Theorem 4.1 [7]). *There is an efficient reduction from vertex cover on bounded degree triangle-free graphs to the Euclidean  $k$ -means problem that satisfies the following properties:*

1. *If vertex cover of the instance is  $k$ , then there is a  $k$ -means clustering of cost at most  $(m - k)$ .*
2. *If vertex cover of the instance is at least  $(1 + \varepsilon)k$ , then the cost of optimal  $k$ -means clustering is at least  $(m - k + \delta k)$ .*

Here,  $\varepsilon$  is some fixed constant  $> 0$  and  $\delta = \Omega(\varepsilon)$ .

Awasthi et al. [7] used the following hardness result for the vertex cover problem on bounded degree triangle-free graphs.

► **Theorem 7** (Corollary 5.3 [7]). *Given any unweighted bounded degree triangle-free graph  $G$ , it is NP-hard to approximate **Vertex Cover** within any factor smaller than 1.36.*

Theorem 6 and Theorem 7 together imply that the Euclidean  $k$ -means problem is APX-hard. A formal statement for the same is given as follows (see Section 4 of [7] for the proof of this result).

► **Corollary 8.** *There exists a constant  $\varepsilon' > 0$  such that it is NP-hard to approximate the Euclidean  $k$ -means problem to any factor better than  $(1 + \varepsilon')$ .*

We would like to obtain a similar gap-preserving reduction for the Euclidean  $k$ -median problem. The first obstacle one encounters in this direction is that unlike the 1-mean problem, there does not exist a closed form expression for the 1-median problem, and hence we don't have an exact expression for the optimal 1-median cost. We overcome this barrier by obtaining good upper and lower bounds on the optimal 1-median cost and showing that these bounds suffice for our purpose. More concretely, to upper bound the optimal 1-median cost, we use the centroid as the 1-median and compute the 1-median cost with respect to the centroid. To obtain a lower bound on the 1-median cost of a cluster, we use a decomposition technique to break a cluster into smaller sub-clusters<sup>3</sup> for which we can compute exact or good approximate lower bounds on the 1-median cost. Here we use a simple observation that the optimal 1-median cost of a cluster is at least the sum of the optimal 1-median costs of the sub-clusters. For any sub-cluster that corresponds to a star graph, one can compute the

<sup>3</sup> Since a set of edges in a graph form a cluster of points in the reduction, we use the terms sub-graphs and sub-clusters interchangeably.

exact 1-median cost using our reduction. In order to bound the 1-median cost for sub-clusters that correspond to non-star graphs, we use the following observation crucially: the optimal 1-median cost is preserved under any transformation that preserves the pairwise distances. For non-star graphs, we first employ such a transformation that preserves the 1-median cost and then compute this cost exactly in the projected space. Note that this technique does not give exact 1-median cost for any arbitrary non-star graph, but works only for some *special families* of non-star graphs. The main idea of the decomposition technique is to ensure that only these kinds of non-star graphs are created in the decomposition process. The upper and lower bounds on the 1-median cost, as constructed in the above manner, are used in the completeness and soundness steps of the proof of the reduction, respectively.

The analysis for the completeness part of the reduction is relatively straightforward. If the vertex cover of a graph is  $k$ , then the edges of the graph can be divided into  $k$  star sub-graphs, each of which results in a star cluster in the  $k$ -median instance. The cost for this clustering with  $k$  star clusters can be found using the reduction easily.

In the proof for the soundness part of our reduction, we prove the contrapositive statement that assumes the  $k$ -median clustering cost to be bounded and proves that the vertex cover of the graph is not too large. Our analysis crucially depends on the relation between the vertex cover of a subgraph and the 1-median cost for that subgraph. More specifically, we need to answer the following question. Given a graph with  $r$  edges having vertex cover  $z$ , how does the optimal 1-median cost for that graph behave with respect to  $z$ . For example, for star graphs,  $z = 1$  and the optimal 1-median cost of a star graph on  $r$  edges is exactly  $\sqrt{r(r-1)}$ . For any non-star graph with  $r$  edges, we first show that the optimal 1-median cost of the non-star graph is at least the optimal 1-median cost of a star graph with  $r$  edges. For any non-star graph  $F$  with  $r$  edges, we denote by  $\delta(F)$  the *extra cost* of  $F$ , defined as the difference of the optimal 1-median cost of  $F$  and the optimal 1-median cost of a star graph with  $r$  edges. If we can figure out non-trivial lower bounds for  $\delta(F)$  for different non-star graphs  $F$ , then we would be done. But, figuring out these non-trivial lower bounds that work for any non-star graph is quite a daunting prospect. The way we overcome this in our work is as follows. We characterize the non-star graphs as having maximum matching of size two or more than two, and for each, we relate the *extra cost* of 1-median clustering of that graph with the vertex cover of that graph. We show that the extra cost of a non-star sub-graph is proportional to the number of vertex-disjoint edges in the sub-graph. And since we assume the  $k$ -median cost to be bounded, the number of vertex disjoint edges is also bounded, giving a small vertex cover.

We need one more idea to finish the proof for the soundness part of the reduction. We call a cluster “singleton” if there is only one point in the cluster. Note that any such cluster would cost zero in a  $k$ -median clustering. If there are a large number of singleton clusters, say  $t < k$ , then they pay zero to the cost of the solution, even though those edges have vertex cover  $t$ . We prove a key lemma showing that for any *hard* instance of the vertex cover, the vertex cover of the sub-graph spanned by  $t$  singleton edges is at most  $\frac{2t}{3}$ . We combine these ideas to prove that if  $k$ -median clustering cost is bounded, the vertex cover of the graph cannot be too large.

We also prove the hardness of bi-criteria approximation results for Euclidean  $k$ -means and  $k$ -median problems. The hardness of bi-criteria approximation for Euclidean  $k$ -median is obtained by extending the proof for the hardness of approximation for the Euclidean  $k$ -median problem. We use the same reduction from the vertex cover problem and show that the soundness guarantees hold even if one is allowed to use  $\beta k$  centers, for some  $\beta > 1$ . We also show that similar techniques give the hardness of bi-criteria approximation results for the Euclidean  $k$ -means problem.

## 2 Useful Facts and Inequalities

In this section, we discuss some basic facts and inequalities that we will frequently use in our proofs. First, we note that the Fermat-Weber problem is not difficult for all 1-median instances. We can efficiently obtain 1-median for some special instances. For example, for a set of equidistant points, the 1-median is simply the centroid of the point set. We give a proof of this statement in the next section. Most importantly, we use the following fact and lemma to compute the 1-median cost.

► **Fact 1** ([38]). *For a set of non-collinear points the optimal 1-median is unique.*

► **Lemma 9.** *Let  $A = \{a_1, \dots, a_n\}$  and  $B = \{b_1, \dots, b_n\}$  be any two sets of  $n$  points in  $\mathbb{R}^d$ . If the pairwise distances between points within  $A$  is the same as pairwise distance between points within  $B$ . That is, for all  $i, j \in \{1, \dots, n\}$ ,  $\|a_i - a_j\| = \|b_i - b_j\|$ . Then the optimal 1-median cost of  $A$  is the same as the optimal 1-median cost of  $B$ .*

The proof of Lemma 9 is deferred to Appendix A. We use the above lemma, in vector spaces where it is tricky to compute the optimal 1-median exactly. In such cases, we transform the space to a different vector space, where computing the 1-median is relatively simpler. More specifically, we employ a rigid transformation since it preserves pairwise distances. Next, we give a simple lemma, that is used to prove various bounds related to the quantity  $\sqrt{m(m-1)}$ .

► **Lemma 10.** *Let  $m$  and  $t$  be any positive real numbers greater than one. If  $m \geq t$ , the following bound holds:*

$$m - (t - \sqrt{t(t-1)}) \leq \sqrt{m(m-1)} \leq m - 1/2.$$

**Proof.** The upper bound follows from the sequence of inequalities:  $\sqrt{m(m-1)} < \sqrt{m^2 - m + 1/4} = \sqrt{(m-1/2)^2} = m - 1/2$ . The lower bound follows from the following sequence of inequalities:

$$\sqrt{m(m-1)} = m + m \cdot \left( \sqrt{\frac{m-1}{m}} - 1 \right) \geq m + t \cdot \left( \sqrt{\frac{t-1}{t}} - 1 \right) = m - (t - \sqrt{t(t-1)}).$$

The second inequality holds because  $\frac{a+1}{b+1} \geq \frac{a}{b}$  for  $b \geq a$ . This completes the proof of the lemma. ◀

### 2.1 Preliminaries

Recall that a point in  $\mathcal{X}$  corresponds to an edge of the graph. Therefore, a sub-graph  $S$  of  $G$  corresponds to a subset of points  $\mathcal{X}(S) := \{x_e \mid e \in E(S)\}$  of  $\mathcal{X}$ . We define the 1-median cost of  $\mathcal{X}(S)$  with respect to a center  $c \in \mathbb{R}^n$  as  $\Phi(c, S) \equiv \sum_{x \in \mathcal{X}(S)} \|x - c\|$ . Furthermore, we define the **optimal** 1-median cost of  $\mathcal{X}(S)$  as  $\Phi^*(S)$ . That is,  $\Phi^*(S) \equiv \min_{c \in \mathbb{R}^n} \Phi(c, S)$ . We often use these statements interchangeably, “optimal 1-median cost of a graph  $S$ ” to mean “optimal 1-median cost of the cluster  $\mathcal{X}(S)$ ”.

## 3 Inapproximability of Euclidean k-Median

In this section, we show the inapproximability result of the Euclidean  $k$ -median problem. We obtain this result by showing a gap preserving reduction from Vertex Cover on bounded degree triangle-free graphs to the Euclidean  $k$ -median. For Vertex Cover on bounded degree triangle-free graphs, the inapproximability result is stated in Corollary 13. The corollary simply follows from the following two results of Austrin et al. [6] and Awasthi et al. [7].

► **Theorem 11** (Austrin et al. [6]). *Given any unweighted bounded degree graph  $G = (V, E)$  of maximum degree  $\Delta$ , Vertex Cover can not be approximated within any factor smaller than  $2 - \varepsilon$ , for  $\varepsilon = (2 + o_\Delta(1)) \cdot \frac{\log \log \Delta}{\log \Delta}$  assuming the Unique Games Conjecture.*

In the above theorem,  $\varepsilon$  can be set to arbitrarily small value by taking sufficiently large value of  $\Delta$ .

► **Theorem 12** (Awasthi et al. [7]). *There is a  $(1 + \varepsilon)$ -approximation-preserving reduction from Vertex Cover on bounded degree graphs to Vertex Cover on triangle-free graphs of bounded degree.*

► **Corollary 13.** *Given any unweighted triangle-free graph  $G$  of bounded degree, Vertex Cover can not be approximated within a factor smaller than  $2 - \varepsilon$ , for any constant  $\varepsilon > 0$ , assuming the Unique Games Conjecture.*

Earlier, in Section 1.2, we described the reduction used by Awasthi et al. [7] to construct instances for Euclidean  $k$ -means from a Vertex Cover instance. We use the same construction for the Euclidean  $k$ -median instances. Let  $G = (V, E)$  denote a triangle-free graph of bounded degree  $\Delta$ . Let  $\mathcal{I} = (\mathcal{X}, k)$  denote the Euclidean  $k$ -median instance constructed from  $G$ . We establish the following theorem based on this construction.

► **Theorem 14.** *There is an efficient reduction from Vertex Cover on bounded degree triangle-free graphs with  $m$  edges to the Euclidean  $k$ -median problem that satisfies the following properties:*

1. *If the graph has a vertex cover of size  $k$ , then the  $k$ -median instance has a solution of cost at most  $m - k/2$ .*
2. *If the graph has no vertex cover of size at most  $(2 - \varepsilon) \cdot k$ , then the cost of any  $k$ -median solution on the instance is at least  $m - k/2 + \delta k$ .*

*Here,  $\varepsilon$  is some fixed constant,  $\delta = \Omega(\varepsilon)$ , and  $k \geq$  the size of maximum matching of the graph.*

The graphs with a vertex cover of size at most  $k$  are said to be “Yes” instances and the graphs with no vertex cover of size at most  $(2 - \varepsilon)k$  are said to be “No” instances. Now, the above theorem gives the following inapproximability result for the Euclidean  $k$ -median problem.

► **Corollary 15.** *There exists a constant  $\varepsilon' > 0$  such that the Euclidean  $k$ -median problem can not be approximated to a factor better than  $(1 + \varepsilon')$ , assuming the Unique Games Conjecture.*

**Proof.** Since the hard Vertex Cover instances have bounded degree  $\Delta$ , the maximum matching of such graphs is at least  $\lceil \frac{m}{2\Delta} \rceil$ . First, let us prove this statement. Suppose  $M$  be a matching, that is initially empty, i.e.,  $M = \emptyset$ . We construct  $M$  in an iterative manner. First, we pick an arbitrary edge from the graph and add it to  $M$ . Then, we remove this edge and all the edges incident on it. We repeat this process for the remaining graph until the graph becomes empty. In each iteration, we remove at most  $2\Delta$  edges. Therefore, the matching size of the graph is at least  $\lceil \frac{m}{2\Delta} \rceil$ .

Now, suppose  $k < \frac{m}{2\Delta}$ . Then, the graph does not have a vertex cover of size  $k$  since matching size is at least  $\lceil \frac{m}{2\Delta} \rceil$ . Therefore, such graph instances can be classified as “No” instances in polynomial time. So, they are not the hard Vertex Cover instances. Therefore, we can assume  $k \geq \frac{m}{2\Delta}$  for all the hard Vertex Cover instances. In that case, the second property of Theorem 14, implies that the cost of  $k$ -median instance is  $(m - \frac{k}{2}) + \delta k \geq (1 + \frac{\delta}{2\Delta}) \cdot (m - \frac{k}{2})$ . Thus, the  $k$ -median problem can not be approximated within any factor smaller than  $1 + \frac{\delta}{2\Delta} = 1 + \Omega(\varepsilon)$ . ◀

### 3.1 Completeness

Let  $W = \{v_1, \dots, v_k\}$  be a vertex cover of  $G$ . Let  $S_i$  denote the set of edges covered by  $v_i$ . If an edge is covered by two vertices  $v_i$  and  $v_j$ , then we arbitrarily keep the edge either in  $S_i$  or  $S_j$ . Let  $m_i$  denote the number of edges in  $S_i$ . We define  $\{\mathcal{X}(S_1), \dots, \mathcal{X}(S_k)\}$  as a clustering of the point set  $\mathcal{X}$ . Now, we show that the cost of this clustering is at most  $m - k/2$ . Note that each  $S_i$  forms a star graph centered at  $v_i$ . Moreover, the point set  $\mathcal{X}(S_i)$  forms a regular simplex of side length  $\sqrt{2}$ . We compute the optimal cost of  $\mathcal{X}(S_i)$  using the following lemma.

► **Lemma 16.** *For a regular simplex on  $r$  vertices and side length  $s$ , the optimal 1-median is the centroid of the simplex. Moreover, the optimal 1-median cost is  $s \cdot \sqrt{\frac{r(r-1)}{2}}$ .*

**Proof.** The statement is easy to see for  $r = 1$ . For  $r = 2$ , there are two points  $s$  distance apart. Therefore, the optimal center lies on the line segment joining the two points and the optimal 1-median cost is trivially  $s$ . So, for the rest of the proof, we assume that  $r > 2$ . Suppose  $A = \{a_1, a_2, \dots, a_r\}$  denote the vertex set of a regular simplex. Let  $s$  be the side length of the simplex. Using Lemma 9, we can represent each point  $a_i$  in an  $r$ -dimensional space as follows; we use the same notation to denote the points after such transformations.

$$a_1 := \left( \frac{s}{\sqrt{2}}, 0, \dots, 0 \right), \quad a_2 := \left( 0, \frac{s}{\sqrt{2}}, \dots, 0 \right), \quad \dots, \quad a_r := \left( 0, 0, \dots, \frac{s}{\sqrt{2}} \right)$$

Note that the distance between any  $a_i$  and  $a_j$  is  $s$ , which is the side length of the simplex. Let  $c^* = (c_1, \dots, c_r)$  be an optimal 1-median of point set  $A$ . Then, the 1-median cost is the following:

$$\Phi(c^*, A) = \sum_{i=1}^r \|a_i - c^*\| = \sum_{i=1}^r \left( \sum_{j=1}^r c_j^2 - c_i^2 + \left( \frac{s}{\sqrt{2}} - c_i \right)^2 \right)^{1/2}$$

Suppose  $c_i \neq c_j$  for any  $i \neq j$ . Then, we can swap  $c_i$  and  $c_j$  to create a different median, while keeping the 1-median cost the same. It contradicts the fact that there is only one optimal 1-median, by Fact 1. Therefore, we can assume  $c^* = (c, c, \dots, c)$ . Now, the optimal 1-median cost is:

$$\Phi^*(A) = \Phi(c^*, A) := r \cdot \sqrt{\left( c - \frac{s}{\sqrt{2}} \right)^2 + (r-1) \cdot c^2}$$

The function  $\Phi(c^*, A)$  is strictly convex and attains minimum at  $c = \frac{s}{m \cdot \sqrt{2}}$ , which is the centroid of  $A$ . The optimal 1-median clustering cost is  $\Phi(c^*, A) = s \cdot \sqrt{\frac{r(r-1)}{2}}$ . This completes the proof of the lemma. ◀

The following corollary establishes the cost of a star graph  $S_i$ .

► **Corollary 17.** *Any star graph  $S_i$  with  $r$  edges has the optimal 1-median cost of  $\sqrt{r(r-1)}$*

Using this corollary, we bound the optimal  $k$ -median cost of  $\mathcal{X}$  as follows. Let  $OPT(\mathcal{X}, k)$  denote the optimal  $k$ -median cost of  $\mathcal{X}$ . The following sequence of inequalities proves the first property of Theorem 14.

$$OPT(\mathcal{X}, k) \leq \sum_{i=1}^k \Phi^*(S_i) \stackrel{\text{(Corollary 17)}}{=} \sum_{i=1}^k \sqrt{m_i(m_i - 1)} \stackrel{\text{(Lemma 10)}}{\leq} \sum_{i=1}^k \left( m_i - \frac{1}{2} \right) = m - \frac{k}{2}.$$

### 3.2 Soundness

Now, we prove the second property of Theorem 14. For this, we prove the equivalent contrapositive statement: If the optimal  $k$ -median clustering of  $\mathcal{X}$  has cost at most  $(m - \frac{k}{2} + \delta k)$ , for some constant  $\delta > 0$ , then  $G$  has a vertex cover of size at most  $(2 - \varepsilon)k$ , for some constant  $\varepsilon > 0$ . Let  $\mathcal{C}$  denote an optimal  $k$ -median clustering of  $\mathcal{X}$ . We classify its optimal clusters into two categories: (1) *star* and (2) *non-star*. Let  $F_1, F_2, \dots, F_t$  denote the non-star clusters, and  $S_1, \dots, S_{k-t}$  denote the star clusters. For any star cluster, the vertex cover size is exactly one. Moreover, using Corollary 17, the optimal 1-median cost of any star cluster with  $r$  edges is  $\sqrt{r(r-1)}$ . On the other hand, it may be tricky to exactly compute the vertex cover or the optimal cost of any non-star cluster. Suppose the optimal 1-median cost of a non-star cluster  $F$  on  $r$  edges is given as  $\sqrt{r(r-1)} + \delta(F)$ , where  $\delta(F)$  denotes the *extra-cost* due to a non-star cluster  $F$ . Using this, we define  $\delta(F)$  as the following:

$$\delta(F) \equiv \Phi^*(F) - \sqrt{|F|(|F| - 1)}$$

The following lemmas bound the vertex cover of  $F$  in terms of  $\delta(F)$ .

► **Lemma 18.** *Any non-star cluster  $F$  with a maximum matching of size two has a vertex cover of size at most  $1.62 + (\sqrt{2} + 1)\delta(F)$ .*

► **Lemma 19.** *Any non-star cluster  $F$  with a maximum matching of size at least three has a vertex cover of size at most  $1.8 + (\sqrt{2} + 1)\delta(F)$ .*

These lemmas are the key to proving the main result. We discussed the main proof ideas earlier in Section 1.2; however, due to page-limit, the complete proof is deferred to the full version of the paper [9]. Now, let us see how these lemmas give a vertex cover of size at most  $(2 - \varepsilon)k$ . Let us classify the star clusters into the following two sub-categories:

- (a) Clusters composed of exactly one edge. Let these clusters be:  $P_1, P_2, \dots, P_{t_1}$ .
- (b) Clusters composed of at least two edges. Let these clusters be:  $S_1, S_2, \dots, S_{t_2}$ .

Similarly, we classify the non-star clusters into the following two sub-categories:

- (i) Clusters with a maximum matching of size two. Let these clusters be:  $W_1, W_2, \dots, W_{t_3}$
- (ii) Clusters with a maximum matching of size at least three. Let these clusters be:  $Y_1, Y_2, \dots, Y_{t_4}$

Note that  $t_1 + t_2 + t_3 + t_4$  equals  $k$ . Now, consider the following strategy of computing the vertex cover of  $G$ . Suppose, we compute the vertex cover for every cluster separately. Let  $C_i$  be any cluster, and  $|VC(C_i)|$  denote the vertex cover size of  $C_i$ . Then, the vertex cover of  $G$  can be simply bounded in the following manner:

$$|VC(G)| \leq \sum_{i=1}^{t_1} |VC(P_i)| + \sum_{i=1}^{t_2} |VC(S_i)| + \sum_{i=1}^{t_3} |VC(W_i)| + \sum_{i=1}^{t_4} |VC(Y_i)|$$

However, we can obtain a vertex cover of smaller size using a slightly different strategy. In this strategy, we first compute a minimum vertex cover of all the clusters except single edge clusters  $P_1, P_2, \dots, P_{t_1}$ . Suppose that vertex cover is  $VC'$ . Then we compute a vertex cover for  $P_1, P_2, \dots, P_{t_1}$ . Now, let us see why this strategy gives a vertex cover of smaller size than before. Note that some vertices in  $VC'$  may also cover the edges in  $P_1, \dots, P_{t_1}$ . Suppose there are  $t'_1$  clusters in  $P_1, \dots, P_{t_1}$  that remain uncovered by  $VC'$ . Without loss of generality, assume these clusters to be  $P_1, \dots, P_{t'_1}$ . Now, the vertex cover of  $G$  is bounded in

the following manner:

$$\begin{aligned}
|VC(G)| &\leq |VC\left(\bigcup_{i=1}^{t'_1} P_i\right)| + |VC'| \\
&= |VC\left(\bigcup_{i=1}^{t'_1} P_i\right)| + |VC\left(\left(\bigcup_{j=1}^{t_2} S_j\right) \cup \left(\bigcup_{k=1}^{t_3} W_k\right) \cup \left(\bigcup_{l=1}^{t_4} Y_l\right)\right)| \\
&\leq |VC\left(\bigcup_{i=1}^{t'_1} P_i\right)| + \sum_{i=1}^{t_2} |VC(S_i)| + \sum_{i=1}^{t_3} |VC(W_i)| + \sum_{i=1}^{t_4} |VC(Y_i)|
\end{aligned}$$

Now, we will try to bound the size of the vertex cover of  $P_1 \cup \dots \cup P_{t'_1}$ . Note that we can cover all these single-edge clusters with  $t'_1$  vertices by choosing one vertex per cluster. However, it may be possible to obtain a vertex cover of smaller size if we collectively consider all these clusters. Suppose  $E_P$  denote the set of all edges in  $P_1, \dots, P_{t'_1}$  and  $V_P$  denote the vertex set spanned by them. We define a graph  $G_P = (V_P, E_P)$ . Further, suppose that  $M_P$  is a maximal matching of  $G_P$ . Then, it is easy to see that if  $|M_P| \leq t'_1/3 + 4\delta k$  for some  $\delta > 0$ , we can simply pick both end-points of every edge in  $M_P$ , and it would give a vertex cover of  $G_P$  of size at most  $2t'_1/3 + 8\delta k$ . On the other hand, if  $|M_P| > t'_1/3 + 4\delta k$ , we show that the graph  $G$  admits a vertex cover of size at most  $(2k - 2\delta k)$ . This fact is mentioned in the following lemma. Due to page limit, the proof is deferred to the full version of the paper [9].

► **Lemma 20.** *Let  $\delta > 0$  be any constant and  $G_P$  be as defined above. If  $G_P$  does not have a vertex cover of size  $\leq (\frac{2t'_1}{3} + 8\delta k)$ , then  $G$  has a vertex cover of size at most  $(2k - 2\delta k)$ .*

Based on the above lemma, we will assume that all single edge clusters can be covered with  $(\frac{2t'_1}{3} + 8\delta k) \leq (\frac{2t_1}{3} + 8\delta k)$  vertices; otherwise the graph has a vertex cover of size at most  $(2k - 2\delta k)$  and the soundness proof would be complete. Now, we bound the vertex cover of the entire graph in the following manner.

$$\begin{aligned}
|VC(G)| &\leq |VC\left(\bigcup_{i=1}^{t'_1} P_i\right)| + |VC'| \\
&= |VC\left(\bigcup_{i=1}^{t'_1} P_i\right)| + |VC\left(\left(\bigcup_{j=1}^{t_2} S_j\right) \cup \left(\bigcup_{k=1}^{t_3} W_k\right) \cup \left(\bigcup_{l=1}^{t_4} Y_l\right)\right)| \\
&\leq \sum_{i=1}^{t'_1} |VC(P_i)| + \sum_{i=1}^{t_2} |VC(S_i)| + \sum_{i=1}^{t_3} |VC(W_i)| + \sum_{i=1}^{t_4} |VC(Y_i)| \\
&\leq \left(\frac{2t_1}{3} + 8\delta k\right) + t_2 + \sum_{i=1}^{t_3} \left((\sqrt{2} + 1) \delta(W_i) + 1.62\right) + \sum_{i=1}^{t_4} \left((\sqrt{2} + 1) \delta(Y_i) + 1.8\right), \\
&\hspace{20em} \text{(using Lemmas 18, 19, and 20)} \\
&= (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 + (\sqrt{2} + 1) \left(\sum_{i=1}^{t_3} \delta(W_i) + \sum_{i=1}^{t_4} \delta(Y_i)\right)
\end{aligned}$$

Since the optimal cost  $OPT(\mathcal{X}, k) = \sum_{j=1}^k \sqrt{m_j(m_j - 1)} + \sum_{i=1}^{t_3} \delta(W_i) + \sum_{i=1}^{t_4} \delta(Y_i) \leq m - k/2 + \delta k$ ,

we get  $\sum_{i=1}^{t_3} \delta(W_i) + \sum_{i=1}^{t_4} \delta(Y_i) \leq m - k/2 + \delta k - \sum_{j=1}^k \sqrt{m_j(m_j - 1)}$ . We substitute this value in the previous equation, and get the following inequality:

$$|VC(G)| \leq (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 + (\sqrt{2} + 1) \cdot \left(m - k/2 - \sum_{j=1}^k \sqrt{m_j(m_j - 1)} + \delta k\right)$$

## 4:12 Hardness of Approximation for Euclidean $k$ -Median

Using Lemma 10, we obtain the following inequalities:

1. For any cluster  $P_j$  with  $|P_j| = 1$ , we have  $\sqrt{|P_j|(|P_j| - 1)} \geq |P_j| - 1$
  2. For any cluster  $S_j$  with  $|S_j| \geq 2$ , we have  $\sqrt{|S_j|(|S_j| - 1)} \geq |S_j| - (2 - \sqrt{2})$
  3. For any cluster  $W_j$  with  $|W_j| \geq 2$ , we have  $\sqrt{|W_j|(|W_j| - 1)} \geq |W_j| - (2 - \sqrt{2})$
  4. For any cluster  $Y_j$  with  $|Y_j| \geq 3$ , we have  $\sqrt{|Y_j|(|Y_j| - 1)} \geq |Y_j| - (3 - \sqrt{6})$
- We substitute these values in the previous equation, and get the following inequality:

$$|VC(G)| \leq (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 + (\sqrt{2} + 1) \cdot \left( m - k/2 - \sum_{j=1}^{t_1} (|P_j| - 1) \right. \\ \left. - \sum_{j=1}^{t_2} (|S_j| - (2 - \sqrt{2})) - \sum_{j=1}^{t_3} (|W_j| - (2 - \sqrt{2})) - \sum_{j=1}^{t_4} (|Y_j| - (3 - \sqrt{6})) + \delta k \right)$$

Since the number of edges  $m = \sum_{j=1}^{t_1} |P_j| + \sum_{j=1}^{t_2} |S_j| + \sum_{j=1}^{t_3} |W_j| + \sum_{j=1}^{t_4} |Y_j|$ , we get the following inequality:

$$|VC(G)| \leq (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 + (\sqrt{2} + 1) \cdot \left( -k/2 + t_1 + t_2 \cdot (2 - \sqrt{2}) + \right. \\ \left. + t_3 \cdot (2 - \sqrt{2}) + t_4 \cdot (3 - \sqrt{6}) + \delta k \right)$$

We substitute  $k = t_1 + t_2 + t_3 + t_4$ , and obtain the following inequality:

$$|VC(G)| \leq (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 + (\sqrt{2} + 1) \cdot \left( \frac{t_1}{2} + \frac{t_2}{10} + \frac{t_3}{10} + \frac{3t_4}{50} + \delta k \right) \\ = (1.88)t_1 + (1.25)t_2 + (1.87)t_3 + (1.95)t_4 + (\sqrt{2} + 9) \delta k \\ < (1.95)k + (\sqrt{2} + 9) \delta k \quad (\text{using } t_3 + t_4 + t_1 + t_2 = k) \\ \leq (2 - \varepsilon)k, \quad \text{for appropriately small constants } \varepsilon, \delta > 0$$

This proves the soundness condition and it completes the proof of Theorem 14. Note that the result holds under the Unique Games Conjecture. To prove the result in a weaker assumption of  $P \neq NP$ , it would require to show that  $|VC(G)| < (1.36)k$  instead of  $|VC(G)| < (1.95)k$ . That would require tighter analysis of the cost of  $k$ -median instances than the one done in this work.

In the next section, we extend the above techniques to give the bi-criteria inapproximability results for the Euclidean  $k$ -median and  $k$ -means problems.

### 4 Bi-criteria Hardness of Approximation

In the previous section, we showed that the  $k$ -median problem cannot be approximated to any factor smaller than  $(1 + \varepsilon)$ , where  $\varepsilon$  is some positive constant. The next step in the *beyond worst-case* discussion is to study the bi-criteria approximation algorithms. That is, we allow the algorithm to choose more than  $k$  centers and analyse whether it produces a solution that is close to the optimal solution with respect to  $k$  centers? Since the algorithm is allowed to output more than  $k$  centers we can hope to get a better approximate solution. An interesting question in this regard would be: *Does there exist a PTAS (polynomial time approximation scheme) for the  $k$ -median/ $k$ -means problem when the algorithm is allowed to choose  $\beta k$  centers for some constant  $\beta > 1$ ?* In other words, is there an  $(1 + \varepsilon, \beta)$ -approximation algorithm? Note that here we compare the cost of  $\beta k$  centers with the optimal cost with respect to  $k$  centers. See Definition 3 in Section 1 for formal definition of  $(\alpha, \beta)$  bi-criteria approximation algorithms.



In this section, we show that even with  $\beta k$  centers, the  $k$ -means/ $k$ -median problems cannot be approximated within any factor smaller than  $(1 + \varepsilon')$ , for some constant  $\varepsilon' > 0$ . The following theorem state this result formally.

► **Theorem 21** (*k*-median). *For any constant  $1 < \beta < 1.015$ , there exists a constant  $\varepsilon > 0$  such that there is no  $(1 + \varepsilon, \beta)$ -approximation algorithm for the Euclidean  $k$ -median problem assuming the Unique Games Conjecture.*

► **Theorem 22** (*k*-means). *For any constant  $1 < \beta < 1.28$ , there exists a constant  $\varepsilon > 0$  such that there is no  $(1 + \varepsilon, \beta)$ -approximation algorithm for the Euclidean  $k$ -means problem assuming the Unique Games Conjecture. Moreover, the same result holds for any  $1 < \beta < 1.1$  under the assumption that  $P \neq NP$ .*

First, let us prove the bi-criteria inapproximability result for the  $k$ -median problem.

### 4.1 Bi-criteria Inapproximability: $k$ -Median

In this subsection, we give a proof of Theorem 21. Let us define a few notations. Suppose  $\mathcal{I} = (\mathcal{X}, k)$  be some  $k$ -median instance. Then,  $OPT(\mathcal{X}, k)$  denote the optimal  $k$ -median cost of  $\mathcal{X}$ . Similarly,  $OPT(\mathcal{X}, \beta k)$  denote the optimal  $\beta k$ -median cost of  $\mathcal{X}$  (or the optimal cost of  $\mathcal{X}$  with  $\beta k$  centers). We use the same reduction as we used in the previous section for showing the hardness of approximation of the  $k$ -median problem. Based on the reduction, we establish the following theorem.

► **Theorem 23.** *There is an efficient reduction from Vertex Cover on bounded degree triangle-free graphs  $G$  (with  $m$  edges) to Euclidean  $k$ -median instances  $\mathcal{I} = (\mathcal{X}, k)$  that satisfies the following properties:*

1. *If  $G$  has a vertex cover of size  $k$ , then  $OPT(\mathcal{X}, k) \leq m - k/2$*
2. *For any constant  $1 < \beta < 1.015$ , there exists constants  $\varepsilon, \delta > 0$  such that if  $G$  has no vertex cover of size  $\leq (2 - \varepsilon) \cdot k$ , then  $OPT(\mathcal{X}, \beta k) \geq m - k/2 + \delta k$ .*

**Proof.** Since the reduction is the same as we discussed in Section 1.2 and 3, we keep all notations the same as before. Also, note that Property 1 in this theorem is the same as Property 1 of Theorem 14. Therefore, the proof is also the same as we did in Section 3.1. Now, we directly move to the proof of Property 2.

The proof is almost the same as we gave in Section 3.2. However, it has some minor differences since we consider the optimal cost with respect to  $\beta k$  centers instead of  $k$  centers. Now, we prove the following contrapositive statement: “For any constants  $1 < \beta < 1.015$  and  $\varepsilon > 0$ , there exists constants  $\varepsilon, \delta > 0$  such that if  $OPT(\mathcal{X}, \beta k) < (m - k/2 + \delta k)$  then  $G$  has a vertex cover of size at most  $(2 - \varepsilon)k$ ”. Let  $\mathcal{C}$  denote an optimal clustering of  $\mathcal{X}$  with  $\beta k$  centers. We classify its optimal clusters into two categories: (1) *star* and (2) *non-star*. Further, we sub-classify the star clusters into the following two sub-categories:

- (a) Clusters composed of exactly one edge. Let these clusters be:  $P_1, P_2, \dots, P_{t_1}$ .
- (b) Clusters composed of at least two edges. Let these clusters be:  $S_1, S_2, \dots, S_{t_2}$ .

Similarly, we sub-classify the non-star clusters into the following two sub-categories:

- (i) Clusters with a maximum matching of size two. Let these clusters be:  $W_1, W_2, \dots, W_{t_3}$
- (ii) Clusters with a maximum matching of size at least three. Let these clusters be:  $Y_1, Y_2, \dots, Y_{t_4}$

Note that  $t_1 + t_2 + t_3 + t_4$  equals  $\beta k$ . Suppose, we first compute a vertex cover of all the clusters except the single edge clusters:  $P_1, \dots, P_{t_1}$ . Let that vertex cover be  $VC'$ . Now, some vertices in  $VC'$  might also cover the edges in  $P_1, \dots, P_{t_1}$ . Suppose there are  $t'_1$  single edge clusters

#### 4:14 Hardness of Approximation for Euclidean k-Median

that remain uncovered by  $VC'$ . Without loss of generality, we assume that these clusters are  $P_1, \dots, P_{t_1}$ . By Lemma 20, we can cover these clusters with  $(\frac{2t_1}{3} + 8\delta k) \leq (\frac{2t_1}{3} + 8\delta k)$  vertices; otherwise the graph would have a vertex cover of size at most  $(2k - \delta k)$ , and the proof of Property 2 would be complete. Now, we bound the vertex cover of the entire graph in the following manner.

$$\begin{aligned} |VC(G)| &\leq \sum_{i=1}^{t_1} |VC(P_i)| + \sum_{i=1}^{t_2} |VC(S_i)| + \sum_{i=1}^{t_3} |VC(W_i)| + \sum_{i=1}^{t_4} |VC(Y_i)| \\ &\leq \left(\frac{2t_1}{3} + 8\delta k\right) + t_2 + \sum_{i=1}^{t_3} ((\sqrt{2} + 1)\delta(W_i) + 1.62) + \sum_{i=1}^{t_4} ((\sqrt{2} + 1)\delta(Y_i) + 1.8), \\ &\hspace{15em} \text{(using Lemmas 18, 19, and 20)} \\ &= (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 + (\sqrt{2} + 1) \left( \sum_{i=1}^{t_3} \delta(W_i) + \sum_{i=1}^{t_4} \delta(Y_i) \right) \end{aligned}$$

Since the optimal cost  $OPT(\mathcal{X}, \beta k) = \sum_{j=1}^{\beta k} \sqrt{m_j(m_j - 1)} + \sum_{i=1}^{t_3} \delta(W_i) + \sum_{i=1}^{t_4} \delta(Y_i) \leq m - k/2 + \delta k$ ,

we get  $\sum_{i=1}^{t_3} \delta(W_i) + \sum_{i=1}^{t_4} \delta(Y_i) \leq m - k/2 + \delta k - \sum_{j=1}^{\beta k} \sqrt{m_j(m_j - 1)}$ . We substitute this value in the previous equation, and get the following inequality:

$$|VC(G)| \leq (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 + (\sqrt{2} + 1) \cdot \left( m - k/2 - \sum_{j=1}^{\beta k} \sqrt{m_j(m_j - 1)} + \delta k \right)$$

Using Lemma 10, we obtain the following inequalities:

1. For  $P_j$ ,  $\sqrt{m(P_j)(m(P_j) - 1)} \geq m(P_j) - 1$  since  $m(P_j) = 1$
2. For  $S_j$ ,  $\sqrt{m(S_j)(m(S_j) - 1)} \geq m(S_j) - (2 - \sqrt{2})$  since  $m(S_j) \geq 2$
3. For  $W_j$ ,  $\sqrt{m(W_j)(m(W_j) - 1)} \geq m(W_j) - (2 - \sqrt{2})$  since  $m(W_j) \geq 2$
4. For  $Y_j$ ,  $\sqrt{m(Y_j)(m(Y_j) - 1)} \geq m(Y_j) - (3 - \sqrt{6})$  since  $m(Y_j) \geq 3$

We substitute these values in the previous equation, and get the following inequality:

$$\begin{aligned} |VC(G)| &\leq (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 + (\sqrt{2} + 1) \cdot \left( m - k/2 - \sum_{j=1}^{t_1} (m(P_j) - 1) + \right. \\ &\quad \left. - \sum_{j=1}^{t_2} (m(S_j) - (2 - \sqrt{2})) - \sum_{j=1}^{t_3} (m(W_j) - (2 - \sqrt{2})) - \sum_{j=1}^{t_4} (m(Y_j) - (3 - \sqrt{6})) + \delta k \right) \end{aligned}$$

Since  $m = \sum_{j=1}^{t_1} m(P_j) + \sum_{j=1}^{t_2} m(S_j) + \sum_{j=1}^{t_3} m(W_j) + \sum_{j=1}^{t_4} m(Y_j)$ , we get the following inequality:

$$\begin{aligned} |VC(G)| &\leq (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 + (\sqrt{2} + 1) \cdot \left( -k/2 + t_1 + t_2 \cdot (2 - \sqrt{2}) + \right. \\ &\quad \left. + t_3 \cdot (2 - \sqrt{2}) + t_4 \cdot (3 - \sqrt{6}) + \delta k \right) \\ &= (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 + (\sqrt{2} + 1) \cdot \left( \frac{(\beta - 1)k}{2} - \frac{\beta k}{2} + t_1 + t_2 \cdot (2 - \sqrt{2}) + \right. \\ &\quad \left. + t_3 \cdot (2 - \sqrt{2}) + t_4 \cdot (3 - \sqrt{6}) + \delta k \right) \end{aligned}$$

Now, we substitute  $\beta k = t_1 + t_2 + t_3 + t_4$ , and obtain the following inequality:

$$\begin{aligned}
|VC(G)| &\leq (0.67)t_1 + 8\delta k + t_2 + (1.62)t_3 + (1.8)t_4 + (\sqrt{2} + 1) \cdot \left( \frac{(\beta - 1)k}{2} + \frac{t_1}{2} + \frac{t_2}{10} + \frac{t_3}{10} + \frac{3t_4}{50} + \delta k \right) \\
&= (1.88)t_1 + (1.25)t_2 + (1.87)t_3 + (1.95)t_4 + (\sqrt{2} + 1) \cdot \frac{(\beta - 1)k}{2} + (\sqrt{2} + 9) \delta k \\
&< (1.95)\beta k + (\sqrt{2} + 1) \cdot \frac{(\beta - 1)k}{2} + (\sqrt{2} + 9) \delta k && \text{(using } t_3 + t_4 + t_1 + t_2 = \beta k) \\
&< (3.16)\beta k - (1.21)k + (\sqrt{2} + 9) \delta k \\
&\leq (2 - \varepsilon)k, \quad \text{for any } \beta < 1.015 \text{ and appropriately small constants } \varepsilon, \delta > 0
\end{aligned}$$

This proves Property 2 and it completes the proof of Theorem 23.  $\blacktriangleleft$

The following corollary states the main bi-criteria inapproximability result for the  $k$ -median problem.

**► Corollary 24.** *There exists a constant  $\varepsilon' > 0$  such that for any constant  $1 < \beta < 1.015$ , there is no  $(1 + \varepsilon', \beta)$ -approximation algorithm for the  $k$ -median problem assuming the Unique Games Conjecture.*

**Proof.** In the proof of Corollary 15, we showed that  $k \geq \frac{m}{2\Delta}$  for all the hard Vertex Cover instances. Therefore, the second property of Theorem 23, implies that  $OPT(\mathcal{X}, \beta k) \geq (m - \frac{k}{2}) + \delta k \geq (1 + \frac{\delta}{2\Delta}) \cdot (m - \frac{k}{2})$ . Thus, the  $k$ -median problem can not be approximated within any factor smaller than  $1 + \frac{\delta}{2\Delta} = 1 + \Omega(\varepsilon)$ , with  $\beta k$  centers for any  $\beta < 1.015$ .  $\blacktriangleleft$

The proof for the bi-criteria inapproximability of the  $k$ -means problem works in a similar manner. We defer its proof to Appendix B.

---

## References

- 1 Ankit Aggarwal, Amit Deshpande, and Ravi Kannan. Adaptive sampling for  $k$ -means clustering. In Irit Dinur, Klaus Jansen, Joseph Naor, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop, APPROX 2009, and 13th International Workshop, RANDOM 2009, Berkeley, CA, USA, August 21-23, 2009. Proceedings*, volume 5687 of *Lecture Notes in Computer Science*, pages 15–28. Springer, 2009. doi:10.1007/978-3-642-03685-9\_2.
- 2 S. Ahmadian, A. Norouzi-Fard, O. Svensson, and J. Ward. Better guarantees for  $k$ -means and euclidean  $k$ -median by primal-dual algorithms. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 61–72, October 2017. doi:10.1109/FOCS.2017.15.
- 3 Nir Ailon, Ragesh Jaiswal, and Claire Monteleoni. Streaming  $k$ -means approximation. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 10–18. Curran Associates, Inc., 2009. URL: <http://papers.nips.cc/paper/3812-streaming-k-means-approximation.pdf>.
- 4 Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat.  $\text{Np}$ -hardness of euclidean sum-of-squares clustering. *Mach. Learn.*, 75(2):245–248, May 2009. doi:10.1007/s10994-009-5103-0.
- 5 Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for  $k$ -median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004. doi:10.1137/S0097539702416402.
- 6 P. Austrin, S. Khot, and M. Safra. Inapproximability of vertex cover and independent set in bounded degree graphs. In *2009 24th Annual IEEE Conference on Computational Complexity*, pages 74–80, 2009. doi:10.1109/CCC.2009.38.

- 7 Pranjali Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The Hardness of Approximation of Euclidean  $k$ -Means. In Lars Arge and János Pach, editors, *31st International Symposium on Computational Geometry (SoCG 2015)*, volume 34 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 754–767, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.SOCG.2015.754.
- 8 Sayan Bandyapadhyay and Kasturi Varadarajan. On Variants of  $k$ -means Clustering. In Sándor Fekete and Anna Lubiw, editors, *32nd International Symposium on Computational Geometry (SoCG 2016)*, volume 51 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:15, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.SOCG.2016.14.
- 9 Anup Bhattacharya, Dishant Goyal, and Ragesh Jaiswal. Hardness of approximation of euclidean  $k$ -median. *CoRR*, abs/2011.04221, 2020. arXiv:2011.04221.
- 10 Mihai Bundeinedoiu, Sarel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, page 250–257, New York, NY, USA, 2002. Association for Computing Machinery. doi:10.1145/509907.509947.
- 11 Jarosław Byrka, Thomas Pensch, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for  $k$ -median and positive correlation in budgeted optimization. *ACM Trans. Algorithms*, 13(2), 2017. doi:10.1145/2981561.
- 12 Ramaswamy Chandrasekaran and Arie Tamir. Open questions concerning weiszfeld’s algorithm for the ferat-weber location problem. *Math. Program.*, 44(1-3):293–295, 1989. doi:10.1007/BF01587094.
- 13 Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the  $k$ -median problem (extended abstract). In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, STOC '99, page 1–10, New York, NY, USA, 1999. Association for Computing Machinery. doi:10.1145/301250.301257.
- 14 Ke Chen. On coresets for  $k$ -median and  $k$ -means clustering in metric and euclidean spaces and their applications. *SIAM Journal on Computing*, 39(3):923–947, 2009. doi:10.1137/070699007.
- 15 Michael B. Cohen, Yin Tat Lee, Gary Miller, Jakub Pachocki, and Aaron Sidford. *Geometric Median in Nearly Linear Time*, page 9–21. Association for Computing Machinery, New York, NY, USA, 2016. doi:10.1145/2897518.2897647.
- 16 Vincent Cohen-Addad. A fast approximation scheme for low-dimensional  $k$ -means. 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 430–440. SIAM, 2018. doi:10.1137/1.9781611975031.29.
- 17 Vincent Cohen-Addad and Karthik C. S. Inapproximability of clustering in  $l_p$  metrics. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 519–539. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00040.
- 18 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.
- 19 Vincent Cohen-Addad, Philip N. Klein, and Claire Mathieu. Local search yields approximation schemes for  $k$ -means and  $k$ -median in euclidean and minor-free metrics. *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, 00:353–364, 2016. doi:doi.ieeecomputersociety.org/10.1109/FOCS.2016.46.
- 20 Sanjoy Dasgupta. The hardness of  $k$ -means clustering. Technical Report CS2008-0916, Department of Computer Science and Engineering, University of California San Diego, 2008.
- 21 Zvi Drezner and Horst W Hamacher. *Facility location: applications and theory*. Springer Science & Business Media, 2001.

- 22 Dan Feldman, Morteza Monemizadeh, and Christian Sohler. A PTAS for  $k$ -means clustering based on weak coresets. In *Proceedings of the twenty-third annual symposium on Computational geometry*, SCG '07, pages 11–18, New York, NY, USA, 2007. ACM. doi:10.1145/1247069.1247072.
- 23 Zachary Friggstad, Mohsen Rezapour, and Mohammad R. Salavatipour. Local search yields a PTAS for  $k$ -means in doubling metrics. *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, 00:365–374, 2016. doi:doi.ieeecomputersociety.org/10.1109/FOCS.2016.47.
- 24 Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *J. Algorithms*, 31(1):228–248, 1999. doi:10.1006/jagm.1998.0993.
- 25 Ragesh Jaiswal, Amit Kumar, and Sandeep Sen. A simple  $D^2$ -sampling based PTAS for  $k$ -means and other clustering problems. *Algorithmica*, 70(1):22–46, 2014. doi:10.1007/s00453-013-9833-9.
- 26 Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for  $k$ -means clustering. In *Proceedings of the Eighteenth Annual Symposium on Computational Geometry*, SCG '02, page 10–18, New York, NY, USA, 2002. Association for Computing Machinery. doi:10.1145/513400.513402.
- 27 J. KRARUP and S. VAJDA. On torricelli's geometrical solution to a problem of fermat. *IMA Journal of Management Mathematics*, 8(3):215–224, 1997. doi:10.1093/imaman/8.3.215.
- 28 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear-time approximation schemes for clustering problems in any dimensions. *J. ACM*, 57(2):5:1–5:32, 2010. doi:10.1145/1667053.1667054.
- 29 Euiwoong Lee, Melanie Schmidt, and John Wright. Improved and simplified inapproximability for  $k$ -means. *Information Processing Letters*, 120:40–43, 2017. doi:10.1016/j.ipl.2016.11.009.
- 30 Shi Li. Approximating capacitated  $k$ -median with  $(1 + \epsilon)k$  open facilities. 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 786–796. SIAM, 2016. doi:10.1137/1.9781611974331.ch56.
- 31 Shi Li. On uniform capacitated  $k$ -median beyond the natural  $l_p$  relaxation. *ACM Trans. Algorithms*, 13(2), 2017. doi:10.1145/2983633.
- 32 Shi Li and Ola Svensson. Approximating  $k$ -median via pseudo-approximation. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, page 901–910, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2488608.2488723.
- 33 Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar  $k$ -means problem is  $np$ -hard. *Theoretical Computer Science*, 442:13–21, 2012. Special Issue on the Workshop on Algorithms and Computation (WALCOM 2009). doi:10.1016/j.tcs.2010.05.034.
- 34 Konstantin Makarychev, Yury Makarychev, and Ilya Razenshteyn. Performance of johnson-lindenstrauss transform for  $k$ -means and  $k$ -medians clustering. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 1027–1038, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3313276.3316350.
- 35 Konstantin Makarychev, Yury Makarychev, Maxim Sviridenko, and Justin Ward. A Bi-Criteria Approximation Algorithm for  $k$ -Means. In Klaus Jansen, Claire Mathieu, José D. P. Rolim, and Chris Umans, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2016)*, volume 60 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:20, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.APPROX-RANDOM.2016.14.
- 36 Jirí Matousek. On approximate geometric  $k$ -clustering. *Discret. Comput. Geom.*, 24(1):61–84, 2000. doi:10.1007/s004540010019.

- 37 Nimrod Megiddo and Kenneth J. Supowit. On the complexity of some common geometric location problems. *SIAM Journal on Computing*, 13(1):182–196, 1984. doi:10.1137/0213014.
- 38 P. Milasevic and G. R. Ducharme. Uniqueness of the spatial median. *Ann. Statist.*, 15(3):1332–1333, September 1987. doi:10.1214/aos/1176350511.
- 39 Stanislav Minsker. Geometric median and robust estimation in Banach spaces. *Bernoulli*, 21(4):2308–2335, 2015. doi:10.3150/14-BEJ645.
- 40 Andrea Vattani. The hardness of k-means clustering in the plane. Technical report, Department of Computer Science and Engineering, University of California San Diego, 2009.
- 41 Dennis Wei. A constant-factor bi-criteria approximation guarantee for k-means++. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 604–612. Curran Associates, Inc., 2016. URL: <http://papers.nips.cc/paper/6309-a-constant-factor-bi-criteria-approximation-guarantee-for-k-means.pdf>.
- 42 E. WEISZFELD. Sur le point pour lequel la somme des distances de n points donnes est minimum. *Tohoku Mathematical Journal, First Series*, 43:355–386, 1937.

## A Proof of Lemma 9

► **Lemma 25.** Let  $A = \{a_1, \dots, a_n\}$  and  $B = \{b_1, \dots, b_n\}$  be any two sets of  $n$  points in  $\mathbb{R}^d$ . If the pairwise distances between points within  $A$  is the same as pairwise distance between points within  $B$ . That is, for all  $i, j \in \{1, \dots, n\}$ ,  $\|a_i - a_j\| = \|b_i - b_j\|$ . Then the optimal 1-median cost of  $A$  is the same as the optimal 1-median cost of  $B$ .

Let  $co(A)$  and  $co(B)$  denote the convex hulls of  $A$  and  $B$ , respectively. We split the proof of Lemma 25 in two parts. In the first part (Lemma 26), we show that there exists a distance preserving transformation  $\mathcal{R}$  from  $co(A)$  to  $co(B)$  such that  $\mathcal{R}(a_i) = b_i$  for every  $i \in \{1, \dots, n\}$ . By distance preserving transformation, we mean that for any two points  $x, y \in co(A)$ , the distance  $\|x - y\|$  is preserved after applying the transformation  $\mathcal{R}$ , i.e.,  $\|x - y\| = \|\mathcal{R}(x) - \mathcal{R}(y)\|$ . In the second part (Lemma 27), we show that applying the transformation  $\mathcal{R}$  preserves the optimal 1-median cost of  $A$ .

► **Lemma 26.** Given two sets of points  $A = \{a_1, a_2, \dots, a_n\}$  and  $B = \{b_1, b_2, \dots, b_n\}$  in  $\mathbb{R}^d$  such that  $\|a_i - a_j\| = \|b_i - b_j\|$  for all  $i, j \in \{1, \dots, n\}$ . Then there exists a distance preserving transformation  $\mathcal{R}: co(A) \rightarrow co(B)$  such that  $\mathcal{R}(a_i) = b_i$  for every  $i \in \{1, \dots, n\}$ .

**Proof.** Let  $\mathbf{X}_i$  be a vector<sup>4</sup> defined as  $\mathbf{a}_i - \mathbf{a}_1$  for every  $\mathbf{a}_i \in A$ . Similarly, we define a vector  $\mathbf{Y}_i := \mathbf{b}_i - \mathbf{b}_1$  for every  $\mathbf{b}_i \in B$ . We will use these vectors to define the transformation  $\mathcal{R}$ . For now, note the following property of inner product of  $\mathbf{X}_i$  and  $\mathbf{X}_j$ .

$$\langle \mathbf{X}_i, \mathbf{X}_j \rangle = \langle \mathbf{Y}_i, \mathbf{Y}_j \rangle \quad \text{for every } i, j \in \{1, \dots, n\} \quad (1)$$

The proof of the above property follows from the following sequence of inequalities:

$$\begin{aligned} 2 \cdot \langle \mathbf{X}_i, \mathbf{X}_j \rangle &= \|\mathbf{X}_i\|^2 + \|\mathbf{X}_j\|^2 - \|\mathbf{X}_i - \mathbf{X}_j\|^2 \\ &= \|\mathbf{Y}_i\|^2 + \|\mathbf{Y}_j\|^2 - \|\mathbf{X}_i - \mathbf{X}_j\|^2, & \because \|\mathbf{X}_i\| &= \|\mathbf{a}_i - \mathbf{a}_1\| = \|\mathbf{b}_i - \mathbf{b}_1\| = \|\mathbf{Y}_i\| \\ & & & \text{for every } 1 \leq i \leq n \\ &= \|\mathbf{Y}_i\|^2 + \|\mathbf{Y}_j\|^2 - \|\mathbf{Y}_i - \mathbf{Y}_j\|^2, & \because \|\mathbf{X}_i - \mathbf{X}_j\| &= \|\mathbf{a}_i - \mathbf{a}_j\| = \|\mathbf{b}_i - \mathbf{b}_j\| = \|\mathbf{Y}_i - \mathbf{Y}_j\| \\ &= 2 \cdot \langle \mathbf{Y}_i, \mathbf{Y}_j \rangle \end{aligned}$$

In other words, the triangles  $(a_1, a_i, a_j)$  and  $(b_1, b_i, b_j)$  are congruent for all  $i, j \in \{1, \dots, n\}$ . Therefore, the inner product  $\langle \mathbf{X}_i, \mathbf{X}_j \rangle$  is the same as  $\langle \mathbf{Y}_i, \mathbf{Y}_j \rangle$ .

<sup>4</sup> For better readability, we boldfaced the vector symbols to distinguish them from any scalar quantity.

Now, we describe the transformation  $\mathcal{R}$  from  $co(A)$  to  $co(B)$ . By the definition of  $co(A)$ , any point  $\mathbf{x} \in co(A)$  can be expressed in the form  $\sum_{i=1}^n \lambda_i \cdot \mathbf{a}_i$  for some  $0 \leq \lambda_i \leq 1$  and  $\sum_{i=1}^n \lambda_i = 1$ . Equivalently,  $\mathbf{x}$  can be expressed as  $\mathbf{a}_1 + \sum_{i=2}^n \lambda_i \cdot \mathbf{X}_i$ . For  $\mathbf{x} \in co(A)$ , we define the transformation  $\mathcal{R}$  as  $\mathcal{R}(\mathbf{x}) := \sum_{i=1}^n \lambda_i \cdot \mathbf{b}_i$ . Again,  $\mathcal{R}(\mathbf{x})$  can be equivalently expressed as  $\mathbf{b}_1 + \sum_{i=2}^n \lambda_i \cdot \mathbf{Y}_i$ . It is easy to see that  $\lambda_i \cdot \mathbf{b}_i$  indeed belongs to  $co(B)$  since  $0 \leq \lambda_i \leq 1$  and  $\sum_{i=1}^n \lambda_i = 1$ . Now, we show that  $\mathcal{R}$  is a distance preserving transformation. Let  $\mathbf{x} := \mathbf{a}_1 + \sum_{i=2}^n \lambda_i \cdot \mathbf{X}_i$  and  $\mathbf{y} := \mathbf{a}_1 + \sum_{i=2}^n \gamma_i \cdot \mathbf{X}_i$  be any two points in  $co(A)$ . The following sequence of inequalities prove that  $\|\mathbf{x} - \mathbf{y}\| = \|\mathcal{R}(\mathbf{x}) - \mathcal{R}(\mathbf{y})\|$ .

$$\begin{aligned}
\|\mathbf{x} - \mathbf{y}\|^2 &= (\mathbf{x} - \mathbf{y})^T (\mathbf{x} - \mathbf{y}) \\
&= \left( \sum_{i=2}^n (\lambda_i - \gamma_i) \cdot \mathbf{X}_i \right)^T \cdot \left( \sum_{i=2}^n (\lambda_i - \gamma_i) \cdot \mathbf{X}_i \right) \\
&= \sum_{i=2}^n \sum_{j=2}^n (\lambda_i - \gamma_i) \cdot (\lambda_j - \gamma_j) \cdot \langle \mathbf{X}_i, \mathbf{X}_j \rangle \\
&= \sum_{i=2}^n \sum_{j=2}^n (\lambda_i - \gamma_i) \cdot (\lambda_j - \gamma_j) \cdot \langle \mathbf{Y}_i, \mathbf{Y}_j \rangle, && \text{(using Equation 1)} \\
&= \left( \sum_{i=2}^n (\lambda_i - \gamma_i) \cdot \mathbf{Y}_i \right)^T \cdot \left( \sum_{i=2}^n (\lambda_i - \gamma_i) \cdot \mathbf{Y}_i \right) \\
&= (\mathcal{R}(\mathbf{x}) - \mathcal{R}(\mathbf{y}))^T (\mathcal{R}(\mathbf{x}) - \mathcal{R}(\mathbf{y})) \\
&= \|\mathcal{R}(\mathbf{x}) - \mathcal{R}(\mathbf{y})\|^2
\end{aligned}$$

This proves that  $\mathcal{R}$  is a distance preserving transformation from  $co(A)$  to  $co(B)$ . Moreover, note that  $\mathcal{R}$  is a bijective function. It is possible that a vector  $\mathbf{x} \in co(A)$  has multiple forms, say  $\sum_{i=1}^n \lambda_i \cdot \mathbf{a}_i$  and  $\sum_{i=1}^n \Lambda_i \cdot \mathbf{a}_i$ . Therefore, it appears that  $\mathbf{x}$  maps to different vectors in  $co(B)$ . However, it always maps to the same vector. For the sake of contradiction, assume that  $\mathbf{x}$  maps to two different vectors  $\mathbf{p} := \sum_{i=1}^n \lambda_i \cdot \mathbf{b}_i$  and  $\mathbf{q} := \sum_{i=1}^n \Lambda_i \cdot \mathbf{b}_i$  in  $co(B)$ . Then  $\|\mathbf{p} - \mathbf{q}\| \neq 0$ . It contradicts the fact that  $\mathcal{R}$  is a distance preserving transformation. Similarly, we can show that any two different vectors  $\mathbf{x}, \mathbf{y} \in co(A)$  can not map to the same vector in  $co(B)$ . This proves that  $\mathcal{R}$  is a bijective function.

Furthermore, note that  $\mathcal{R}(\mathbf{a}_i) = \mathbf{b}_i$  for every  $i \in \{1, \dots, n\}$ . To see this, consider  $\lambda_i = 1$  and  $\lambda_j = 0$  for all  $j \in \{1, \dots, n\} \setminus \{i\}$ . Then  $\mathbf{a}_i = \sum_{j=1}^n \lambda_j \cdot \mathbf{a}_j$  and therefore  $\mathcal{R}(\mathbf{a}_i) = \sum_{j=1}^n \lambda_j \cdot \mathbf{b}_j = \mathbf{b}_i$ . This completes the proof of the lemma.  $\blacktriangleleft$

Similar to  $\mathcal{R}$ , we can also define a distance preserving transformation  $\mathcal{R}^{-1}$  from  $co(B)$  to  $co(A)$ . The transformation  $\mathcal{R}^{-1}$  is defined such that for any  $\mathbf{x} = \sum_{i=1}^n \lambda_i \cdot \mathbf{b}_i \in co(B)$ ,

$\mathcal{R}^{-1}(x) = \sum_{i=1}^n \lambda_i \cdot \mathbf{a}_i \in \text{co}(A)$ . Furthermore, as per this definition of  $\mathcal{R}^{-1}$ ,  $\mathcal{R}^{-1}(\mathbf{b}_i) = \mathbf{a}_i$  for every  $i \in \{1, \dots, n\}$ . Now, we show that applying the transformation  $\mathcal{R}$  on  $A$  preserves the optimal 1-median cost of  $A$ .

► **Lemma 27.** *If there exists distance preserving transformations  $\mathcal{R}: \text{co}(A) \rightarrow \text{co}(B)$  and  $\mathcal{R}^{-1}: \text{co}(B) \rightarrow \text{co}(A)$  such that  $\mathcal{R}(\mathbf{a}_i) = \mathbf{b}_i$  and  $\mathcal{R}^{-1}(\mathbf{b}_i) = \mathbf{a}_i$  for every  $i \in \{1, \dots, n\}$ . Then the optimal 1-median cost of  $A$  is the same as the optimal 1-median cost of  $B$ .*

**Proof.** Recall that 1-median cost of an instance  $A$  with respect to a center  $\mathbf{c} \in \mathbb{R}^d$  is denoted by  $\Phi(\mathbf{c}, A) \equiv \sum_{\mathbf{a}_i \in A} \|\mathbf{a}_i - \mathbf{c}\|$ . Let  $\mathbf{c}_1^*$  be the optimal 1-median of  $A$ . Furthermore, we can assume that  $\mathbf{c}_1^* \in \text{co}(A)$  since the optimal 1-median lies in the convex hull of  $A$  (see *e.g.* Remark 2.1 in [39]). Similarly, let  $\mathbf{c}_2^* \in \text{co}(B)$  be the optimal 1-median of  $B$ . Now, we show that  $\Phi(\mathbf{c}_1^*, A) \geq \Phi(\mathbf{c}_2^*, B)$  and  $\Phi(\mathbf{c}_1^*, A) \leq \Phi(\mathbf{c}_2^*, B)$  using the following sequence of inequalities:

$$\begin{aligned} \Phi(\mathbf{c}_1^*, A) &= \sum_{\mathbf{a}_i \in A} \|\mathbf{a}_i - \mathbf{c}_1^*\| \\ &= \sum_{\mathbf{a}_i \in A} \|\mathcal{R}(\mathbf{a}_i) - \mathcal{R}(\mathbf{c}_1^*)\|, && \because \mathcal{R} \text{ preserves the pairwise distances} \\ &= \sum_{\mathbf{b}_i \in B} \|\mathbf{b}_i - \mathcal{R}(\mathbf{c}_1^*)\|, && \because \mathcal{R}(\mathbf{a}_i) = \mathbf{b}_i \\ &\geq \sum_{\mathbf{b}_i \in B} \|\mathbf{b}_i - \mathbf{c}_2^*\|, && \because \mathbf{c}_2^* \text{ is the optimal 1-median of } B \\ &= \Phi(\mathbf{c}_2^*, B) \end{aligned}$$

Similarly, we show that  $\Phi(\mathbf{c}_2^*, B) \geq \Phi(\mathbf{c}_1^*, A)$  as follows:

$$\begin{aligned} \Phi(\mathbf{c}_2^*, B) &= \sum_{\mathbf{b}_i \in B} \|\mathbf{b}_i - \mathbf{c}_2^*\| \\ &= \sum_{\mathbf{b}_i \in B} \|\mathcal{R}^{-1}(\mathbf{b}_i) - \mathcal{R}^{-1}(\mathbf{c}_2^*)\|, && \because \mathcal{R}^{-1} \text{ preserves the pairwise distances} \\ &= \sum_{\mathbf{a}_i \in A} \|\mathbf{a}_i - \mathcal{R}^{-1}(\mathbf{c}_2^*)\|, && \because \mathcal{R}^{-1}(\mathbf{b}_i) = \mathbf{a}_i \\ &\geq \sum_{\mathbf{a}_i \in A} \|\mathbf{a}_i - \mathbf{c}_1^*\|, && \because \mathbf{c}_1^* \text{ is the optimal 1-median of } A \\ &= \Phi(\mathbf{c}_1^*, A) \end{aligned}$$

This proves that  $\Phi(\mathbf{c}_1^*, A) = \Phi(\mathbf{c}_2^*, B)$ . Hence it proves the lemma. ◀

Therefore, Lemma 26 and 27 together proves Lemma 25.

## **B** Bi-criteria Inapproximability: k-means

Here, we again use the same reduction that we used earlier for the  $k$ -median problem in Sections 1.2, 3, and 4.1. Using this, we establish the following theorem.

► **Theorem 28.** *There is an efficient reduction from Vertex Cover on bounded degree triangle-free graphs  $G$  (with  $m$  edges) to Euclidean  $k$ -means instances  $\mathcal{I} = (\mathcal{X}, k)$  that satisfies the following properties:*



1. If  $G$  has a vertex cover of size  $k$ , then  $OPT(\mathcal{X}, k) \leq m - k$
2. For any  $1 < \lambda \leq 2$  and  $\beta < \frac{2}{7} \cdot \left(\lambda + \frac{5}{2}\right)$ , there exists constants  $\varepsilon, \delta > 0$  such that if  $G$  has no vertex cover of size  $\leq (\lambda - \varepsilon) \cdot k$ , then  $OPT(\mathcal{X}, \beta k) \geq m - k + \delta k$ .

This theorem is simply an extension of the result of Awasthi *et al.* [7] to the bi-criteria setting. Now, let us prove this theorem.

## B.1 Completeness

Note that the proof of completeness is already given in [7]. Therefore, we just describe the main components of the proof for the sake of clarity. To understand the proof, let us define some notations used in [7]. Suppose  $F$  is a subgraph of  $G$ . For a vertex  $v \in V(F)$ , let  $d_F(v)$  denote the number of edges in  $F$  that are incident on  $v$ . Note that, the optimal center for 1-means problem is simply the centroid of the point set. Therefore, we can compute the optimal 1-means cost of  $F$ . The following lemma states the optimal 1-means cost of  $F$ .

► **Lemma 29** (Claim 4.3 [7]). *Let  $F$  be a subgraph of  $G$  with  $r$  edges. Then, the optimal 1-means cost of  $F$  is  $\sum_v d_F(v) \left(1 - \frac{d_F(v)}{r}\right)$*

The following corollary bounds the optimal 1-means cost of a star cluster. This corollary is implicitly stated in the proof of Claim 4.4 of [7].

► **Corollary 30.** *The optimal 1-means cost of a star cluster with  $r$  edges is  $r - 1$ .*

Using the above corollary, we give the proof of completeness. Let  $V = \{v_1, \dots, v_k\}$  be a vertex cover of  $G$ . Let  $S_i$  denote the set of edges covered by  $v_i$ . If an edge is covered by two vertices  $i$  and  $j$ , then we arbitrarily keep the edge either in  $S_i$  or  $S_j$ . Let  $m_i$  denote the number of edges in  $S_i$ . We define  $\{\mathcal{X}(S_1), \dots, \mathcal{X}(S_k)\}$  as a clustering of the point set  $\mathcal{X}$ . Now, we show that the cost of this clustering is at most  $m - k$ . Note that each  $S_i$  forms a star graph with its edges sharing the common vertex  $v_i$ . The following sequence of inequalities bound the optimal  $k$ -means cost of  $\mathcal{X}$ .

$$OPT(\mathcal{X}, k) \leq \sum_{i=1}^k \Phi^*(S_i) \stackrel{\text{(Corollary 30)}}{=} \sum_{i=1}^k (m(S_i) - 1) = m - k.$$

## B.2 Soundness

For the proof of soundness, we prove the following contrapositive statement: “For any constant  $1 < \lambda \leq 2$  and  $\beta < \frac{2}{7} \cdot \left(\lambda + \frac{5}{2}\right)$ , there exists constants  $\varepsilon, \delta > 0$  such that if  $OPT(\beta k) \leq (m - k + \delta k)$  then  $G$  has a vertex cover of size at most  $(\lambda - \varepsilon)k$ , for  $\varepsilon = \Omega(\delta)$ .” Let  $\mathcal{C}$  denote an optimal clustering of  $\mathcal{X}$  with  $\beta k$  centers. We classify its optimal clusters into two categories: (1) *star* and (2) *non-star*. Suppose there are  $t_1$  star clusters:  $S_1, \dots, S_{t_1}$ , and  $t_2$  non-star clusters:  $F_1, F_2, \dots, F_{t_2}$ . Note that  $t_1 + t_2$  equals  $\beta k$ . The following lemma bounds the optimal 1-means cost of a non-star cluster.

► **Lemma 31** (Lemma 4.8 [7]). *The optimal 1-means cost of any non-star cluster  $F$  with  $m$  edges is at least  $m - 1 + \delta(F)$ , where  $\delta(F) \geq \frac{2}{3}$ . Furthermore, there is an edge  $(u, v) \in E(F)$  such that  $d_F(u) + d_F(v) \geq m + 1 - \delta(F)$ .*

In the original statement of the lemma in [7], the authors mentioned a weak bound of  $\delta(F) > 1/2$ . However, in the proof of their lemma they have shown  $\delta(F) > 2/3 > 1/2$ . This difference does not matter when we consider inapproximability of the  $k$ -means problem. However, this difference improves the  $\beta$  value in bi-criteria inapproximability of the  $k$ -means problem.

## 4:22 Hardness of Approximation for Euclidean k-Median

► **Corollary 32** ([7]). *Any non-star cluster  $F$  has a vertex cover of size at most  $1 + \frac{5}{2} \cdot \delta(F)$ .*

**Proof.** Suppose  $(u, v)$  be an edge in  $F$  that satisfies the property:  $d_F(u) + d_F(v) \geq m + 1 - \delta(F)$ , by Lemma 31. This means that  $u$  and  $v$  covers at least  $m(F) - \delta(F)$  edges of  $F$ . We pick  $u$  and  $v$  in the vertex cover, and for the remaining  $\delta(F)$  edges we pick one vertex per edge. Therefore,  $F$  has a vertex cover of size at most  $2 + \delta(F)$ . Since  $\delta(F) \geq \frac{2}{3}$ , by Lemma 31, we get  $2 + \delta(F) \leq 1 + \frac{5}{2} \cdot \delta(F)$ . Hence,  $F$  has a vertex cover of size at most  $1 + \frac{5}{2} \cdot \delta(F)$ . This proves the corollary. ◀

Now, the following sequence of inequalities bound the vertex cover size of the entire graph  $G$ .

$$\begin{aligned} |VC(G)| &\leq \sum_{i=1}^{t_1} |VC(S_i)| + \sum_{i=1}^{t_2} |VC(F_i)| \\ &\leq t_1 + \sum_{i=1}^{t_2} \left(1 + \frac{5}{2} \cdot \delta(F_i)\right) \quad (\text{using Corollary 32}) \\ &= t_1 + t_2 + \frac{5}{2} \cdot \sum_{i=1}^{t_2} \delta(F_i) \end{aligned}$$

Since the optimal  $k$ -means cost  $OPT(\mathcal{X}, \beta k) = \sum_{i=1}^{t_1} (m(S_i) - 1) + \sum_{i=1}^{t_2} (m(F_i) - 1 + \delta(F_i)) \leq m - k + \delta k$ , and  $t_1 + t_2 = \beta k$ . Therefore,  $\sum_{i=1}^{t_2} \delta(F_i) \leq (\beta - 1)k + \delta k$ . On substituting this value in the previous equation, we get the following inequality:

$$\begin{aligned} |VC(G)| &\leq t_1 + t_2 + \frac{5}{2} \cdot (\beta - 1)k + \frac{5}{2} \cdot \delta k \\ &= \beta k + \frac{5}{2} \cdot (\beta - 1)k + \frac{5}{2} \cdot \delta k, \quad (\because t_1 + t_2 = \beta k) \\ &\leq (\lambda - \varepsilon)k, \quad \text{for any } \beta < \frac{2}{7} \cdot \left(\lambda + \frac{5}{2}\right) \text{ and appropriately small constants } \varepsilon, \delta > 0 \end{aligned}$$

This proves the soundness condition and thus completes the proof of Theorem 28.

Next, we state a corollary of Theorem 28 that gives the main bi-criteria inapproximability result for the  $k$ -means problem.

► **Corollary 33.** *For any constant  $1 < \beta < 1.28$ , there exists a constant  $\varepsilon' > 0$  such that there is no  $(1 + \varepsilon', \beta)$ -approximation algorithm for the  $k$ -means problem assuming the Unique Games Conjecture. Moreover, the same result holds for any  $1 < \beta < 1.1$  under the assumption that  $P \neq NP$ .*

**Proof.** Suppose Vertex Cover can not be approximated to any factor smaller than  $\lambda - \varepsilon$ , for some constants  $\varepsilon, \lambda > 0$ . In the proof of Corollary 15, we showed that  $k \geq \frac{m}{2\Delta}$  for all the hard Vertex Cover instances. In that case, the second property of Theorem 28 implies that  $OPT(\mathcal{X}, \beta k) \geq (m - k) + \delta k \geq (1 + \frac{\delta}{2\Delta}) \cdot (m - k)$ . Thus, the  $k$ -means problem can not be approximated within any factor smaller than  $1 + \frac{\delta}{2\Delta} = 1 + \Omega(\varepsilon)$ , with  $\beta k$  centers. Now, let us compute the value of  $\beta$  using the value of  $\lambda$ . We know that  $\beta < \frac{2}{7} \cdot \left(\lambda + \frac{5}{2}\right)$ . Consider the following two cases:

- By Corollary 13, Vertex Cover is hard to approximate within any factor smaller than  $2 - \varepsilon$  on bounded degree triangle-free graphs assuming UGC. Hence  $\lambda = 2$  and thus  $\beta < 1.28$  assuming UGC.

- By Theorem 7, Vertex Cover is hard to approximate within any factor smaller than 1.36 on bounded degree triangle-free graphs assuming  $P \neq NP$ . Hence  $\lambda = 1.36$  and thus  $\beta < 1.1$  assuming  $P \neq NP$ .

This completes the proof of the corollary. ◀





# Online Directed Spanners and Steiner Forests

Elena Grigorescu   

Department of Computer Science, Purdue University, West Lafayette, IN, USA

Young-San Lin<sup>1</sup>   

Department of Computer Science, Purdue University, West Lafayette, IN, USA

Kent Quanrud  

Department of Computer Science, Purdue University, West Lafayette, IN, USA

---

## Abstract

We present online algorithms for directed spanners and directed Steiner forests. These are well-studied network connectivity problems that fall under the unifying framework of online covering and packing linear programming formulations. This framework was developed in the seminal work of Buchbinder and Naor (Mathematics of Operations Research, 34, 2009) and is based on primal-dual techniques. Specifically, our results include the following:

- For the *pairwise spanner* problem, in which the pairs of vertices to be spanned arrive online, we present an efficient randomized algorithm with competitive ratio  $\tilde{O}(n^{4/5})$  for graphs with general edge lengths, where  $n$  is the number of vertices of the given graph. For graphs with uniform edge lengths, we give an efficient randomized algorithm with competitive ratio  $\tilde{O}(n^{2/3+\epsilon})$ , and an efficient deterministic algorithm with competitive ratio  $\tilde{O}(k^{1/2+\epsilon})$ , where  $k$  is the number of terminal pairs. To the best of our knowledge, these are the first online algorithms for directed spanners. In the offline version, the current best approximation ratio for uniform edge lengths is  $\tilde{O}(n^{3/5+\epsilon})$ , due to Chlamtáč, Dinitz, Kortsarz, and Laekhanukit (SODA 2017, TALG 2020).
- For the *directed Steiner forest* problem with uniform costs, in which the pairs of vertices to be connected arrive online, we present an efficient randomized algorithm with competitive ratio  $\tilde{O}(n^{2/3+\epsilon})$ . The state-of-the-art online algorithm for general costs is due to Chakrabarty, Ene, Krishnaswamy, and Panigrahi (SICOMP 2018) and is  $\tilde{O}(k^{1/2+\epsilon})$ -competitive. In the offline version, the current best approximation ratio with uniform costs is  $\tilde{O}(n^{26/45+\epsilon})$ , due to Abboud and Bodwin (SODA 2018).

To obtain *efficient* and *competitive* online algorithms, we observe that a small modification of the online covering and packing framework by Buchbinder and Naor implies a polynomial-time implementation of the primal-dual approach with separation oracles, which a priori might perform exponentially many calls to the oracle. We convert the online spanner problem into an online covering problem and complete the rounding-step analysis in a problem-specific fashion.

**2012 ACM Subject Classification** Theory of computation → Online algorithms; Theory of computation → Packing and covering problems; Theory of computation → Routing and network design problems; Theory of computation → Rounding techniques

**Keywords and phrases** online directed pairwise spanners, online directed Steiner forests, online covering/packing linear programming, primal-dual approach

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.5

**Category** APPROX

**Related Version** *Full Version:* <https://arxiv.org/abs/2103.04543> [52]

**Funding** E.G and Y.L. were supported in part by NSF CCF-1910659 and NSF CCF-1910411.

**Acknowledgements** We thank the anonymous reviewers for comments and suggestions that helped improve the presentation. We thank Anupam Gupta and Greg Bodwin for bringing to our attention references that we missed in previous versions of the writeup.

---

<sup>1</sup> Corresponding author



## 1 Introduction

We study online variants of directed network optimization problems. In an online problem, the input is presented sequentially, one item at a time, and the algorithm is forced to make irrevocable decisions in each step, without knowledge of the remaining part of the input. The performance of the algorithm is measured by its *competitive ratio*, which is the ratio between the value of the online solution and that of an optimal offline solution.

Our main results focus on *directed spanners*, which are sparse subgraphs that approximately preserve pairwise distances between vertices. Spanners are fundamental combinatorial objects with a wide range of applications, such as distributed computation [9, 69], data structures [5, 75], routing schemes [35, 67, 70, 72], approximate shortest paths [18, 41, 42], distance oracles [18, 31, 68], and property testing [8, 22]. For a comprehensive account of the literature, we refer the reader to the excellent survey [2].

We also study related network *connectivity* problems, and in particular on *directed Steiner forests*, which are sparse subgraphs that maintain connectivity between target terminal vertex pairs. Steiner forests are ubiquitously used in a heterogeneous collection of areas, such as multicommodity network design [49, 53], mechanism design and games [30, 63, 64, 73], computational biology [62, 71], and computational geometry [19, 24].

Our approaches are based on covering and packing linear programming (LP) formulations that fall into the unifying framework developed by Buchbinder and Naor [26], using the powerful primal-dual technique [51]. This unifying framework extends across widely different domains, and hence provides a general abstraction that captures the algorithmic essence of all online covering and packing formulations. In our case, to obtain *efficient* competitive algorithms for solving the LPs online, we observe that the algorithms in [26] can be slightly modified to significantly speed up the setting of our applications, in which the algorithm might otherwise make exponentially many calls to a separation oracle. This component is not tailored to the applications studied here and may be of independent interest. In particular, previous approaches solving online covering and packing problems either focus on the competitiveness of the algorithm [4, 12, 16], or manage to leverage the specific structure of the problem for better time efficiency in a somewhat ad-hoc manner [3, 11, 15, 21, 25, 59], while here the solution may be viewed as a more *unified* framework that is also *efficient*.

### 1.1 Our contributions

#### 1.1.1 Directed spanners

Let  $G = (V, E)$  be a directed simple graph with  $n$  vertices. Each edge is associated with its *length*  $\ell : E \rightarrow \mathbb{R}_{\geq 0}$ . The edge lengths are *uniform* if  $\ell(e) = 1$  for all  $e \in E$ . In spanner problems, the goal is to compute a minimum cardinality (number of edges) subgraph in which the distance between terminals is preserved up to some prescribed factor. In the most well-studied setting, called the *directed  $s$ -spanner* problem, there is a fixed value  $s \geq 1$  called the *stretch*, and the goal is to find a minimum cardinality subgraph in which *every* pair of vertices has distance within a factor of  $s$  in the original graph. For low stretch spanners, when  $s = 2$ , there is a tight  $\Theta(\log n)$ -approximation algorithm [44, 65]; when  $s = 3, 4$  both with uniform edge lengths, there are  $\tilde{O}(n^{1/3})$ -approximation algorithms [20, 40]. When  $s > 4$ , the best known approximation factor is  $\tilde{O}(n^{1/2})$  [20]. The problem is hard to approximate within an  $O(2^{\log^{1-\varepsilon} n})$  factor for  $3 \leq s = O(n^{1-\delta})$  and any  $\varepsilon, \delta \in (0, 1)$ , unless  $NP \subseteq DTIME(n^{\text{polylog } n})$  [45].

A more general setting, called the *pairwise spanner* problem [34], and the *client-server* model [22, 44], considers an arbitrary set of terminals  $D = \{(s_i, t_i) \mid i \in [k]\} \subseteq V \times V$ . Each terminal pair  $(s_i, t_i)$  has its own target distance  $d_i$ . The goal is to compute a minimum cardinality subgraph in which for each  $i$ , the distance from  $s_i$  to  $t_i$  is at most  $d_i$ . For the pairwise spanner problem with uniform edge lengths, [34] obtains an  $\tilde{O}(n^{3/5+\epsilon})$ -approximation.

In the online version, the graph is known ahead of time, and the terminal pairs and the corresponding target distances are received one by one in an online fashion. The distance requirement of the arriving terminal pair is satisfied by irrevocably including edges. There are no online algorithms for the pairwise spanner problem that we are aware of, even in the simpler and long-studied case of stretch  $s$  or graphs with uniform lengths.

For graphs with uniform edge lengths, we present the proof outline of the following theorem in Section 2 and refer the reader to the full version [52] for the complete proof.

► **Theorem 1.** *For the online pairwise spanner problem with uniform edge lengths, there exists a deterministic polynomial time algorithm with competitive ratio  $\tilde{O}(k^{1/2+\delta})$  for any constant  $\delta > 0$ .*

Next, we turn to graphs with general edge lengths and derive online algorithms with competitive ratios in terms of  $n$ . We present a generic algorithm (Algorithm 3) used for Theorems 2, 3, 4, 5, 6, and 7. Due to space limitations, we refer the reader to the full version [52] for the complete proof of Theorems 3, 4, and 5.

For graphs with general edge lengths, we show the following in Section 3.1.

► **Theorem 2.** *For the online pairwise spanner problem, there is a randomized polynomial time algorithm with competitive ratio  $\tilde{O}(n^{4/5})$ .*

In one special case, the given graph might have uniform edge lengths, and the diameter is bounded or it is guaranteed that the distances between the terminal pairs are bounded. Let  $d = \max_{i \in [k]} \{d_i\}$  that is known offline be the maximum allowed distance of any pair of terminals in the input. This setting is equivalent to the *d-diameter spanning subgraph* problem introduced in [22].

► **Theorem 3.** *For the online pairwise spanner problem with uniform edge lengths and maximum allowed distance  $d$ , there is a randomized polynomial time algorithm with competitive ratio  $\tilde{O}(d^{1/3}n^{2/3})$ .*

Another special case is where the edge lengths are *quasimetric*. That is, they satisfy the following directed form of the triangle inequality. For any two edges  $u \rightarrow v$  and  $v \rightarrow w$ , there is also an edge  $u \rightarrow w$  such that  $\ell(u, w) \leq \ell(u, v) + \ell(v, w)$ . This setting includes the class of transitive-closure graphs with uniform edge lengths, in which each pair of vertices connected by a directed path is also connected by a directed edge. The offline version of the *transitive-closure spanner* problem was formally defined in [22].

► **Theorem 4.** *For the online pairwise spanner problem where edge lengths are quasimetric, there is a randomized polynomial time algorithm with competitive ratio  $\tilde{O}(n^{2/3})$ .*

In a special case on graphs with uniform edge lengths, for each terminal pair  $(s_i, t_i)$ , there is also an edge  $s_i \rightarrow t_i$  in the given graph. This setting is equivalent to the *all-server spanner* problem introduced in [44].

► **Theorem 5.** *For the online all-server spanner problem with uniform edge lengths, there is a randomized polynomial time algorithm with competitive ratio  $\tilde{O}(n^{2/3})$ .*

## 5:4 Online Directed Spanners and Steiner Forests

For graphs with uniform edge lengths without further assumptions, we use Theorem 1 and the generic algorithm to prove the following theorem in Section 3.2.

► **Theorem 6.** *For the online pairwise spanner problem with uniform edge lengths, there is a randomized polynomial time algorithm with competitive ratio  $\tilde{O}(n^{2/3+\varepsilon})$  for any constant  $\varepsilon \in (0, 1/3)$ .*

### 1.1.2 Directed Steiner forests

In the *directed Steiner forest* problem, we are given a directed graph  $G = (V, E)$  with edge costs  $w : E \rightarrow \mathbb{R}_{\geq 0}$ , and a set of terminals  $D = \{(s_i, t_i) \mid i \in [k]\} \subseteq V \times V$ . The goal is to find a subgraph  $H = (V, E')$  which includes an  $s_i \rightsquigarrow t_i$  path for each terminal pair  $(s_i, t_i)$ , and the total cost  $\sum_{e \in E'} w(e)$  is minimized. The costs are *uniform* when  $w(e) = 1$  for all  $e \in E$ .

In the offline setting with general costs, the best known approximations are  $O(k^{1/2+\varepsilon})$  by Chekuri et al. [32] and  $O(n^{2/3+\varepsilon})$  by Berman et al. [20]. For the special case of uniform costs, there is an improved approximation factor of  $\tilde{O}(n^{26/45+\varepsilon})$  by Abboud and Bodwin [1]. In the online setting, Chakrabarty et al. [28] give an  $\tilde{O}(k^{1/2+\varepsilon})$  approximation for general costs. Their algorithm also extends to the more general buy-at-bulk version. We prove the following in Section 3.3..

► **Theorem 7.** *For the online directed Steiner forest problem with uniform costs, there is a randomized polynomial time algorithm with competitive ratio  $\tilde{O}(n^{2/3+\varepsilon})$  for any constant  $\varepsilon \in (0, 1/3)$ .*

We essentially improve the competitive ratio when the number of terminal pairs is  $\omega(n^{4/3})$ .

### 1.1.3 Summary

We summarize our main results for online pairwise spanners and directed Steiner forests in Table 1 by listing the competitive ratios and contrast them with the corresponding known competitive and approximation ratios. We note that offline  $\tilde{O}(n^{4/5})$ -approximate pairwise spanners for graphs with general edge lengths and offline  $\tilde{O}(k^{1/2+\varepsilon})$ -approximate pairwise spanners for graphs with uniform edge lengths can be obtained by our online algorithms.

■ **Table 1** Summary of the competitive and approximation ratios. Here,  $n$  refers to the number of vertices and  $k$  refers to the number of terminal pairs. We include the known results for comparison. The text in gray refers to known results while the text in black refers to our contributions.

Setting	Pairwise Spanners	Directed Steiner Forests
Offline	$\tilde{O}(n^{4/5})$ (implied by Thm 2)	$\tilde{O}(n^{26/45+\varepsilon})$ (uniform costs) [1]
	$\tilde{O}(n^{3/5+\varepsilon})$ (uniform lengths) [34]	$O(n^{2/3+\varepsilon})$ [20]
	$\tilde{O}(k^{1/2+\varepsilon})$ (uniform lengths, implied by Thm 1)	$O(k^{1/2+\varepsilon})$ [32]
Online	$\tilde{O}(n^{4/5})$ (Thm 2)	$\tilde{O}(k^{1/2+\varepsilon})$ [28]
	$\tilde{O}(n^{2/3+\varepsilon})$ (uniform lengths, Thm 6)	$\tilde{O}(n^{2/3+\varepsilon})$ (uniform costs, Thm 7)
	$\tilde{O}(k^{1/2+\varepsilon})$ (uniform lengths, Thm 1)	

## 1.2 An efficient online covering and packing framework

Before presenting our modification to the unified framework in [26] to obtain efficient online covering and packing LP solvers, we give an overview of the well-known primal-dual framework



for approximating covering and packing LP's online. This framework is the main engine of our application and it is important to establish some context before getting into the application for spanners and Steiner forests. We also introduce a discussion of certain technical nuances that arise for our application, and the small modification we propose to address it. A more formal description, including proofs and fully parameterized theorem statements, is fairly technical and therefore deferred to Appendix C *after* we have used these tools in the context of spanners and Steiner forests.

The primal-dual framework was first developed for the online set cover problem in the seminal work of [4]. The approach was extended to network optimization problems in undirected graphs in [3], then abstracted and generalized to a broad LP-based primal-dual framework in [26]. Our discussion primarily centers around the abstract framework in [26]. A number of previous results in online algorithms, such as ski rental [61] and paging [16], can be recovered from this approach and many new important applications have since been developed, such as the  $k$ -server problem [76]. We refer the reader to the excellent survey by Buchbinder and Naor [27].

These works develop a clean two-step approach to online algorithms based on 1) solving the LP online, and 2) rounding the LP online. Solving the LP online can be done in a generic fashion, while rounding tends to be problem-specific. The setting for the *covering* LP is the following.

$$\text{minimize } \langle \mathbf{c}, x \rangle \text{ over } x \in \mathbb{R}_{\geq 0}^n \text{ s.t. } Ax \geq \mathbf{b}. \quad (1)$$

Here  $A \in \mathbb{R}_{\geq 0}^{m \times n}$  consists of  $m$  covering constraints,  $\mathbf{b} \in \mathbb{R}_{> 0}^m$  is a positive lower bound of the covering constraints, and  $\mathbf{c} \in \mathbb{R}_{> 0}^n$  denotes the positive coefficients of the linear cost function. Each constraint can be normalized, so we focus on covering LP's in the following form.

$$\text{minimize } \langle \mathbf{c}, x \rangle \text{ over } x \in \mathbb{R}_{\geq 0}^n \text{ s.t. } Ax \geq \mathbf{1} \quad (2)$$

where  $\mathbf{1}$  is a vector of all ones.

In the online covering problem, the cost vector  $\mathbf{c}$  is given offline, and each of these covering constraints is presented one by one in an online fashion, that is,  $m$  can be unknown. The goal is to update  $x$  in a non-decreasing manner such that all the covering constraints are satisfied and the objective value  $\langle \mathbf{c}, x \rangle$  is approximately optimal. An important idea in this line of work is to simultaneously consider the dual *packing* problem:

$$\text{maximize } \langle \mathbf{1}, y \rangle \text{ over } y \in \mathbb{R}_{\geq 0}^m \text{ s.t. } A^T y \leq \mathbf{c} \quad (3)$$

where  $A^T$  consists of  $n$  packing constraints with an upper bound  $\mathbf{c}$  given offline.

In the online packing problem, the *columns* of  $A^T$  and the corresponding variables are presented online with value zero, one can either let the arriving variable remain zero, or irrevocably assign a positive value to the arriving variable. The goal is to approximately maximize the objective value  $\langle \mathbf{1}, y \rangle$  with each constraint approximately satisfied.

### 1.2.1 Separation oracles in the online setting

The primal-dual framework in [26] simultaneously solves both LP (2) and LP (3), and crucially uses LP-duality and strong connections between the two solutions to argue that they are both nearly optimal. Here we give a sketch of the LP solving framework for reference in the subsequent discussion. We maintain solutions  $x$  and  $y$  for LP (2) and LP (3), respectively, in an online fashion. The covering solution  $x$  is a function of the packing solution  $y$ . In particular, each coordinate  $x_j$  is exponential in the *load* of the corresponding packing constraint in LP

(3). Both  $x$  and  $y$  are monotonically increasing. The algorithm runs in phases, where each phase corresponds to an estimate for OPT revised over time. Within a phase we have the following. If the new covering constraint  $i \in [m]$ , presented online, is already satisfied, then there is nothing to be done. Otherwise, increase the corresponding coordinate  $y_i$ , which simultaneously increases the  $x_j$ 's based on the magnitude of the coordinate  $a_{ij}$ , where  $a_{ij}$  is the  $i$ -th row  $j$ -th column entry of  $A$ . The framework in [26] increases  $y_i$  until the increased  $x_j$ 's satisfy the new constraint. This naturally extends to the setting when the problem relies on a *separation oracle* to retrieve an unsatisfied covering constraint where the number of constraints can be unbounded [26]. However, while this approach will fix all violating constraints, each individual fix may require a diminishingly small adjustment that cannot be charged off from a global perspective. Consequently the algorithm may have to address exponentially many constraints.

### 1.2.2 A primal-dual bound on separation oracles

Our goal is to adjust the framework to ensure that we only address a polynomial number of constraints (per phase). For many concrete problems in the literature, this issue can be addressed directly based on the problem at hand (discussed in greater detail in Section A). In our setting, we start with a combinatorially defined LP that is not a pure covering problem, and convert it to a covering LP. While having a covering LP is conducive to the online LP framework, the machinery generates a large number of covering constraints that are very unstructured. For example, we have little control over the coefficients of these constraints. This motivates us to develop a more generic argument to bound the number of queries to the separation oracle, based on the online LP framework, more so than the exact problem at hand. Here, when addressing a violated constraint  $i$ , we instead increase the dual variable  $y_i$  until the increased primal variables  $x$  (over-)satisfy the new constraint by a factor of 2. This forces at least one  $x_j$  to be doubled – and in the dual, this means we used up a substantial amount of the corresponding packing constraint. Since the packing solution is already guaranteed to be feasible in each phase by the overall framework, this leads us to conclude that we only ever encounter polynomially many violating constraints.

For our modified online covering and packing framework, we show that 1) the approximation guarantees are identical to those in [26], 2) the framework only encounters polynomially many violating constraints for the online covering problem, and 3) only polynomially many updates are needed for the online packing problem.

► **Theorem 8 (Informal).** *There exists an  $O(\log n)$ -competitive online algorithm for the covering LP (2) which encounters polynomially many violating constraints.*

► **Theorem 9 (Informal).** *Given any parameter  $B > 0$ , there exists a  $1/B$ -competitive online algorithm for the packing LP (3) which updates  $y$  polynomially many times, and each constraint is violated within an  $O(f(A)/B)$  factor ( $f(A)$  is a logarithmic function that depends on the entries in  $A$ ).*

We note that the competitive ratios given in [26] are tight, which also implies the tightness of the modified framework. The number of violating constraints depends not only on the number of covering variables and packing constraints  $n$ , but also on the number of bits used to present the entries in  $A$  and  $\mathbf{c}$ . The formal proof for Theorem 8 is provided in Appendix C, while the formal proof for Theorem 9 provided in the full version [52] is not directly relevant to this work, but may be of independent interest.

### 1.3 High-level technical overview for online network optimization problems

#### 1.3.1 Online pairwise spanners

For this problem, a natural starting point is the flow-based LP approach for offline  $s$ -stretch directed spanners, introduced in [38]. The results of [34] adopt a slight tweak for this approach to achieve an  $\tilde{O}(n/\sqrt{\text{OPT}})$ -approximation, where  $\text{OPT}$  is the size of the optimal solution. With additional ideas, the  $\tilde{O}(n/\sqrt{\text{OPT}})$ -approximation is converted into an  $\tilde{O}(n^{3/5+\varepsilon})$ -approximation for pairwise spanners. One technical obstacle in the online setting is the lack of a useful lower bound for  $\text{OPT}$ . Another challenge is solving the LP for the spanner problem and rounding the solution in an online fashion, particularly as the natural LP is not a covering LP. We address these technical obstacles as discussed below in Section 3. Ultimately we obtain an  $\tilde{O}(n^{4/5})$  competitive ratio for the online setting. The strategy here is to convert the LP for spanners into a covering LP, where the constraints are generated by an *internal* LP. The covering LP previously appeared in [38] implicitly, and in [39] explicitly.

#### 1.3.2 Online pairwise spanners with uniform edge lengths

For the special case of uniform edge lengths, [34] obtains an improved bound of  $\tilde{O}(n^{3/5+\varepsilon})$ . It is natural to ask if the online bound of  $\tilde{O}(n^{4/5})$  mentioned above can be improved as well. Indeed, we obtain an improved bound of  $\tilde{O}(n^{2/3+\varepsilon})$  by replacing the greedy approach in the small  $\text{OPT}$  regime by using the  $\tilde{O}(k^{1/2+\varepsilon})$ -competitive online algorithm discussed in Section 2. This algorithm leverages ideas from [34] in reducing to label cover problems with ideas from the online network design algorithms of [28]. Some additional ideas are required to combine the existing tools and among others we had to formulate a new pure covering LP that can be solved online, to facilitate the transition.

#### 1.3.3 Online Steiner forests with uniform costs

This problem is a special case of the online pairwise spanner problem where the distance requirement for each terminal pair is infinity and the edge lengths are uniform. The online algorithm for this problem has a similar structure to the one for pairwise spanners and similar obstacles to overcome. Before small value of  $\text{OPT}$  gets large, we can leverage the  $\tilde{O}(k^{1/2+\varepsilon})$ -competitive online algorithm or the online buy-at-bulk framework [28] for a better bound than a greedy approach would give, improving the competitive ratio to  $\tilde{O}(n^{2/3+\varepsilon})$ .

## 1.4 Organization

Since the proof of Theorem 1 is the most involved contribution of this work, we start by presenting the proof outline in Section 2 and refer the reader to the full version [52] for the technical proof details. In Section 3, we prove Theorems 2, 6, and 7 by designing a generic online algorithm, which is also used for proving Theorems 3, 4, and 5 in the full version [52]. We show the modified online covering framework in Appendix C, while the modified online packing framework is presented in the full version [52]. We refer the reader to Appendix A for a detailed description of related work and an exposition situating our work in the expansive literature of directed spanners, Steiner forests, and covering and packing problems.

## 2 Online Pairwise Spanners with Uniform Lengths

In this section, we present the proof outline of Theorem 1, namely we design an online algorithm for the pairwise spanner problem with uniform edge lengths with competitive ratio  $\tilde{O}(k^{1/2+\delta})$  for any constant  $\delta > 0$ . We recall that in the *pairwise spanner* problem, we are given a directed graph  $G = (V, E)$  with edge length  $\ell : E \rightarrow \mathbb{R}_{\geq 0}$ , a general set of  $k$  terminals  $D = \{(s_i, t_i) \mid i \in [k]\} \subseteq V \times V$ , and a target distance  $d_i$  for each terminal pair  $(s_i, t_i)$ , the goal is to output a subgraph  $H = (V, E')$  of  $G$  such that for every pair  $(s_i, t_i) \in D$  it is the case that  $d_H(s_i, t_i) \leq d_i$ , i.e. the length of a shortest  $s_i \rightsquigarrow t_i$  path is at most  $d_i$  in the subgraph  $H$ , and we want to minimize the number of edges in  $E'$ . The edge lengths are uniform if  $\ell(e) = 1$  for all  $e \in E$ . In the online setting, the directed graph  $G$  is given offline, while the vertex pairs in  $D \subseteq V \times V$  arrive online one at a time. In the beginning,  $E' = \emptyset$ . Suppose  $(s_i, t_i)$  and its target distance  $d_i$  arrive in round  $i$ , we select some edges from  $E$  and irrevocably add them to  $E'$ , such that in the subgraph  $H = (V, E')$ ,  $d_H(s_i, t_i) \leq d_i$ .

### 2.1 Outline of the proof of Theorem 1

We start by describing the high-level approach of our proof of Theorem 1. While the proof combines ideas of the online buy-at-bulk framework in [28] and of the reduction from the pairwise spanner problem to a connectivity problem in [34], implementing the details require several new ideas. Specifically, we introduce a useful extension of the Steiner problem, called the *Steiner label cover* problem, and our main contribution is an online covering LP formulation for this problem. This approach allows us to not only capture the global approximation property in an online setting, as in [28], but also to handle distance constraints, as in [34]. The entire proof consists of three main ingredients:

1. We first show that there exists an  $O(\sqrt{k})$ -approximate solution consisting of *junction trees*. A junction tree is a subgraph consisting of an in-arborescence and out-arborescence rooted at the same vertex (see also Definition 10).
2. We then show a reduction from the online pairwise spanner problem to the online Steiner label cover problem on a forest with a loss of an  $O(k^{1/2+\delta})$  factor. More precisely, an  $O(\sqrt{k})$  factor comes from the junction tree approximation and an extra  $O(k^\delta)$  factor comes from the *height reduction* technique introduced in [32, 58]. The height reduction technique allows us to focus on low-cost trees of height  $O(1/\delta)$  in order to recover a junction tree approximation.
3. Finally, we show a reduction from the online Steiner label cover problem to the online undirected Steiner forest problem, with a loss of a  $\text{polylog}(n)$  factor. More precisely, we first formulate an online covering LP for the online Steiner label cover instance, then construct an online undirected Steiner forest instance from the LP solution, with a loss of a factor of 2. By [21], the online undirected Steiner forest problem can then be solved deterministically with competitive ratio  $\text{polylog}(n)$ .

Combining these three ingredients results in an  $\tilde{O}(k^{1/2+\delta})$ -competitive algorithm. We provide further intuition below. The detailed description these three ingredients are presented in the full version [52].

#### 2.1.1 Junction tree approximation

Many connectivity problems, including Steiner forests, buy-at-bulk, and spanner problems, are usually solved using *junction trees* introduced in [33].

► **Definition 10.** A junction tree rooted at  $r \in V$  is a directed graph  $G = (V, E)$ , by taking the union of an in-arborescence rooted at  $r$  and an out-arborescence rooted at  $r^2$ . A junction tree solution is a collection of junction trees rooted at different vertices, that satisfies all the terminal distance constraints.

► **Lemma 11.** There exists an  $O(\sqrt{k})$ -approximate junction tree solution for pairwise spanners.

At a high level, the proof of Lemma 11 follows by a standard *density* argument. A *partial solution* is a subgraph that connects a subset of the terminal pairs within the required distances. The density of a partial solution is the ratio between the number of edges used and the number of terminal pairs connected within the required distances. This argument is used for solving offline problems including the Steiner forest problem [20, 32, 46], the buy-at-bulk problem [6], the Client-Server  $s$ -spanner [22] problem, and the pairwise spanner problem [34], by greedily removing low density partial solutions in an iterative manner. Fortunately, this iterative approach also guarantees a nice *global* approximation that consists of junction trees rooted at different vertices, which is amenable in the online setting. The online buy-at-bulk algorithm in [28] is an online version of the junction tree framework for connectivity problems. Our main technical contribution is further modifying the online version of the junction tree framework for problems with distance constraints.

### 2.1.2 Reduction to Steiner label cover

We reduce the pairwise spanner problem to the following extension of Steiner problem termed *Steiner label cover*.

► **Definition 12.** In the Steiner label cover problem, we are given a (directed or undirected) graph  $G = (V, E)$ , non-negative edge costs  $w : E \rightarrow \mathbb{R}_{\geq 0}$ , and a collection of  $k$  disjoint vertex subset pairs  $(S_i, T_i)$  for  $i \in [k]$  where  $S_i, T_i \subseteq V$  and  $S_i \cap T_i = \emptyset$ . Each pair is associated with a relation (set of permissible pairs)  $R_i \subseteq S_i \times T_i$ . The goal is to find a subgraph  $F = (V, E')$  of  $G$ , such that 1) for each  $i \in [k]$ , there exists  $(s, t) \in R_i$  such that there is an  $s \rightsquigarrow t$  path in  $F$ , and 2) the cost  $\sum_{e \in E'} w(e)$  is minimized.

For the online Steiner label cover problem,  $(S_i, T_i)$  and  $R_i$  arrive online, and the goal is to irrevocably select edges to satisfy the first requirement and also approximately minimize the cost.

To reduce to the online Steiner label cover problem, we construct a directed graph  $G'$  that consists of disjoint layered graphs from the given graph  $G = (V, E)$ . Each vertex in  $G'$  is labelled by the distance to (from) the root of a junction tree. This allows us to capture distance constraints by a Steiner label cover instance with distance-based relations. From  $G'$ , we further construct an undirected graph  $H$  which is a forest by the *height reduction* technique [32, 58]. In  $H$ , we define the corresponding Steiner label cover instance, where the terminal vertex sets consist of the leaves, and the solution is guaranteed to be a forest. The Steiner label cover instance on the forest  $H$  has a nice property. For each tree in  $H$ , the terminal vertices can be ordered in a way such that if an *interval* belongs to the relation, then any *subinterval* also belongs to the relation.

<sup>2</sup> A junction tree does not necessarily have a tree structure in directed graphs, i.e. edges in the in-arborescence and edges in the out-arborescence may overlap. Nevertheless, we continue using this term because of historical reasons. A similar notion can also be used for undirected graphs, where a junction tree is indeed a tree.

► **Definition 13.** The ordered Steiner label cover problem on a forest is defined as a special case of the Steiner label cover problem (see Definition 12) with the following properties.

1.  $G$  is an undirected graph consisting of disjoint union of trees  $H_1, H_2, \dots, H_n$  each of which has a distinguished root vertex  $r_j$  where  $j \in [n]$ .
2. For each  $(S_i, T_i)$  and  $R_i$ , and each tree  $H_j$ , the input also includes the orderings  $\prec_{i,j}$  such that:
  - a. For  $S_i^j := S_i \cap H_j$  and  $T_i^j := T_i \cap H_j$ , the ordering  $\prec_{i,j}$  is defined on  $S_i^j \cup T_i^j$ .
  - b. The root  $r_j$  separates  $S_i^j$  from  $T_i^j$ .
  - c. If  $s \in S_i^j$  and  $t \in T_i^j$  are such that  $(s, t) \in R_i$ , then for any  $s' \in S_i^j$  and  $t' \in T_i^j$  such that  $s \preceq_{i,j} s' \prec_{i,j} t' \preceq_{i,j} t$ , we have that  $(s', t') \in R_i$ .

We note that for the online ordered Steiner label cover problem on a forest, besides  $(S_i, T_i)$  and  $R_i$ , the orderings  $\{\succ_{i,j}\}_{j \in [n]}$  also arrive online.

We employ a well-defined mapping between junction trees in  $G$  and forests in  $H$  by paying an  $\tilde{O}(k^{1/2+\delta})$  factor for competitive online solutions. A crucial step for showing Theorem 1 is the following theorem.

► **Theorem 14.** For any constant  $\delta > 0$ , an  $\alpha$ -competitive polynomial time algorithm for online ordered Steiner label cover on a forest implies an  $O(\alpha k^{1/2+\delta})$ -competitive polynomial time algorithm for the online pairwise spanner problem on a directed graph with uniform edge lengths.

At a high level, the online pairwise spanner problem on a directed graph  $G = (V, E)$  with uniform edge lengths reduces to an instance of online Steiner label cover on the forest  $H$  with the following properties.

1.  $H$  consists of disjoint trees  $H_r$  rooted at  $r'$  for each vertex  $r \in V$ .
2.  $|V(H)| = n^{O(1/\delta)}$ ,  $E(H) = n^{O(1/\delta)}$ , and each tree  $H_r$  has depth  $O(1/\delta)$  with respect to  $r'$ .
3. For each arriving terminal pair  $(s_i, t_i)$  with distance requirement  $d_i$ , there is a corresponding pair of terminal sets  $(\hat{S}_i, \hat{T}_i)$  and relation  $\hat{R}_i$  with  $|\hat{R}_i| = n^{O(1/\delta)}$ , where  $\hat{S}_i$  and  $\hat{T}_i$  are disjoint subsets of leaves in  $H$ . Furthermore, we can generate orderings  $\prec_{i,r}$  based on the distance-based relations  $\hat{R}_i$  such that the Steiner label cover instance is an ordered instance on the forest  $H$ .

This technique closely follows the one for solving offline pairwise spanners in [34]. The intermediary problem considered in [34] is the *minimum density Steiner label cover problem*. In this framework, the solution is obtained by selecting the partial solution with the lowest density among the junction trees rooted at different vertices and repeat. In the online setting, to capture the global approximation for pairwise spanners, we construct a forest  $H$  and consider all the possible roots simultaneously.

### 2.1.3 An online algorithm for Steiner label cover on $H$

The goal is to prove the following lemma.

► **Lemma 15.** For the online ordered Steiner label cover problem on a forest (see definition 13), there is a deterministic polynomial time algorithm with competitive ratio  $\text{polylog}(n)$ .

We derive an LP formulation for the Steiner label cover instance on  $H$ . At a high level, the LP minimizes the total edge weight by selecting edges that cover paths with endpoint pairs which belong to the distance-based relation. We show that the LP for Steiner label cover can be converted into an online covering problem, which is efficiently solvable by Theorem 8.

The online rounding is based on the online LP solution for Steiner label cover. Given the online LP solution and the orderings in round  $i$  generated by the distance-based relation  $\hat{R}_i$ , we extract the representative vertex sets  $\tilde{S}_i$  and  $\tilde{T}_i$  from the terminal sets  $\hat{S}_i$  and  $\hat{T}_i$ , respectively, according to orderings  $\prec_{i,r}$  and the contribution of the terminal vertex to the objective of the Steiner label cover LP. We show that the union of cross-products over partitions of  $\tilde{S}_i$  and  $\tilde{T}_i$  (based on the trees in  $H$ ) is a subset of the distance-based relation  $\hat{R}_i$ . This allows us to reduce the online ordered Steiner label cover problem to the online undirected Steiner forest problem by connecting a super source to  $\tilde{S}_i$  and a super sink to  $\tilde{T}_i$ .

This technique closely follows the one for solving offline pairwise spanners in [34]. The main difference is that in the offline pairwise spanner framework, the LP formulation is density-based and considers only one (fractional) junction tree. To globally approximate the online pairwise spanner solution, our LP formulation is based on the forest  $H$  and its objective is the total weight of a (fractional) forest.

The LP for the undirected Steiner forest problem is roughly in the following form.

$$\begin{aligned} \min_x \quad & \sum_{e \in E(H)} w'(e)x_e \\ \text{subject to} \quad & x \text{ supports an } \tilde{S}_i\text{-}\tilde{T}_i \text{ flow of value } 1 \quad \forall i \in [k], \\ & x_e \geq 0 \quad \forall e \in E(H). \end{aligned} \tag{4}$$

Here  $w'$  denotes the edge weights in  $H$ . We show that a solution of the undirected Steiner forest LP (4) recovers a solution for the Steiner label cover LP by a factor of 2. The integrality gap of the undirected Steiner forest LP is  $\text{polylog}(n)$  because the instance can be decomposed into single source Steiner forest instances by the structure of  $H$  [28, 50]. This implies that the online rounding for the Steiner label cover LP can be naturally done via solving the undirected Steiner forest instance online, by using the  $\text{polylog}(n)$ -competitive framework [21].

### 2.1.3.1 Putting it all together

We summarize the overall  $\tilde{O}(k^{1/2+\delta})$ -competitive algorithm for online pairwise spanners when the given graph has uniform edge lengths. The reduction strategy is as follows:

1. Reduce the online pairwise spanner problem on the original graph  $G$  to the online Steiner label cover problem on the directed graph  $G'$  which consists of disjoint layered graphs.
2. Reduce the online Steiner label cover problem on  $G'$  to an online ordered Steiner label cover problem on  $H$ , where  $H$  is a forest.
3. In the forest  $H$ , reduce the online ordered Steiner label cover problem to the online undirected Steiner forest problem.

We note that  $G'$  and  $H$  are constructed offline, while the graph for the final undirected Steiner forest instance is partially constructed online, by adding super sources and sinks and connecting incident edges to the representative leaf vertices online in  $H$ . The pairwise spanner in  $G$  is  $O(\sqrt{k})$ -approximated by junction trees according to Lemma 11. The graph  $G'$  preserves the same cost of the pairwise spanner (junction tree solution) in  $G$ . The solution of the ordered Steiner label cover problem in graph  $H$  is a forest. One can map a forest in  $H$  to junction trees in  $G'$ , via the height reduction technique by losing an  $O(k^\delta)$  factor. Finally, in the forest  $H$ , we solve the undirected Steiner forest instance online and recover an ordered Steiner label cover solution by losing a  $\text{polylog}(n)$  factor. The overall competitive ratio is therefore  $\tilde{O}(k^{1/2+\delta})$ .

### 3 Online Pairwise Spanners

We recall that in the general *pairwise spanner* problem, we are given a directed graph  $G = (V, E)$  with edge length  $\ell : E \rightarrow \mathbb{R}_{\geq 0}$ , a general set of  $k$  terminals  $D = \{(s_i, t_i) \mid i \in [k]\} \subseteq V \times V$ , and a target distance  $d_i$  for each terminal pair  $(s_i, t_i)$ , the goal is to output a subgraph  $H = (V, E')$  of  $G$  such that for every pair  $(s_i, t_i) \in D$  it is the case that  $d_H(s_i, t_i) \leq d_i$ , i.e. the length of a shortest  $s_i \rightsquigarrow t_i$  path is at most  $d_i$  in the subgraph  $H$ , and we want to minimize the number of edges in  $E'$ .

#### 3.1 An $\tilde{O}(n^{4/5})$ -competitive online algorithm for pairwise spanners

In this section, we prove Theorem 2. Recall that in the online setting of the problem, the directed graph  $G$  is given offline, while the vertex pairs in  $D \subseteq V \times V$  arrive online one at a time. In the beginning,  $E' = \emptyset$ . Suppose  $(s_i, t_i)$  and its target distance  $d_i$  arrive in round  $i$ , we select some edges from  $E$  and irrevocably add them to  $E'$ , such that in the subgraph  $H = (V, E')$ ,  $d_H(s_i, t_i) \leq d_i$ . The goal is to approximately minimize the total number of edges added to  $E'$ .

We start with a high-level sketch of an offline algorithm, which we will build on for the online setting. The randomized rounding framework in [20, 34] has two main steps. One step is to solve and round an LP for the spanner problem. The second is to uniformly sample vertices and add the shortest path in-arborescences and out-arborescences rooted at each of the sampled vertices. Terminal pairs are classified as either *thin* or *thick* and are addressed by one of the two steps above accordingly.

In the first step, the rounding scheme based on an LP solution for spanners ensures with high probability that for all thin terminal pairs the distance requirements are met. The second step ensures with high probability that for all thick terminal pairs the distance requirements are met. By selecting an appropriate threshold for classifying the thin and thick pairs, this leads to an  $O(n/\sqrt{\text{OPT}})$ -approximation, where  $\text{OPT}$  is the number of edges in the optimal solution.

The main challenges in adapting this approach to the online setting are as follows: 1)  $\text{OPT}$  can be very small, and 2) the LP for spanners is not naturally a pure covering LP, which makes it difficult to solve online. In the previous work in the offline setting, the small- $\text{OPT}$  case is addressed by sophisticated strategies that appear difficult to emulate online. Instead, we show that the optimal value (however small) is at least the square root of the number of terminal pairs that have arrived. Thus, if  $\text{OPT}$  is small and not many pairs have arrived, we can greedily add a path with the fewest edges subject to the distance requirement for each pair. To overcome the second challenge, we convert the LP for spanners into an equivalent covering LP as in [39], where exponentially many covering constraints are generated by an auxiliary LP. Having transformed the LP into a purely covering one, we can solve the LP online, treating the auxiliary LP as a separation oracle.

##### 3.1.1 A simple $\tilde{O}(n^{4/5})$ -approximate offline algorithm based on [34]

For ease of exposition, we first design a simpler offline algorithm (slightly weaker than the state-of-the-art) that is more amenable to the online setting. This allows us to establish the main ingredients governing the approximation factor in a simpler setting, and then address the online aspects separately. The algorithm leverages the framework developed in [20, 34].



Let  $\mathcal{P}_i$  denote the collection of  $s_i \rightsquigarrow t_i$  paths of length at most  $d_i$  consisting of edges in  $E$ . Let the *local graph*  $G^i = (V^i, E^i)$  be the union of all vertices and edges in  $\mathcal{P}_i$ . A pair  $(s_i, t_i) \in D$  is *t-thick* if  $|V^i| \geq t$ , otherwise  $(s_i, t_i)$  is *t-thin*. Consider the following standard LP relaxation (essentially the one in [38]).

$$\begin{aligned}
& \min_{x,y} && \sum_{e \in E} x_e \\
& \text{subject to} && \sum_{P \in \mathcal{P}_i} y_P \geq 1 && \forall i \in [k], \\
& && \sum_{P|e \in P \in \mathcal{P}_i} y_P \leq x_e && \forall e \in E, \forall i \in [k], \\
& && x_e \geq 0 && \forall e \in E, \\
& && y_P \geq 0 && \forall P \in \mathcal{P}_i \quad \forall i \in [k].
\end{aligned} \tag{5}$$

Herein,  $x_e$  is an indicator of edge  $e$  and  $y_P$  is an indicator of path  $P$ . Suppose we have an integral feasible solution. Then the first set of constraints ensures that there is at least one  $s_i \rightsquigarrow t_i$  path of length at most  $d_i$  selected, and the second set of constraints ensures that if a path  $P$  is selected, then all its edges are selected.

We say that a pair  $(s_i, t_i) \in D$  is *settled* if the selection of edges is such that there exists an  $s_i \rightsquigarrow t_i$  path of length at most  $d_i$ . Applying a simple rounding scheme based on a solution of LP (5) settles the thin pairs with high probability, while sampling enough vertices and adding shortest path in-arborescences and out-arborescences rooted at each sampled vertex ensures with high probability that thick pairs are settled. Let  $\text{OPT}$  be the optimum value of the given pairwise spanner instance. Without loss of generality, we may assume that we know  $\text{OPT}$  since we can guess every value of  $\text{OPT}$  in  $[|E|]$  in the offline setting. Now we are ready to describe Algorithm 1 in [34].

► **Lemma 16.** ([34]) *Algorithm 1 is  $\tilde{O}(n/\sqrt{\text{OPT}})$ -approximate.*

■ **Algorithm 1** Offline pairwise spanner.

- 
- 1:  $E' \leftarrow \emptyset$  and  $t \leftarrow n/\sqrt{\text{OPT}}$ .
  - 2: Solve LP (5) and add each edge  $e \in E$  to  $E'$  with probability  $\min\{1, x_e t \ln n\}$  independently.
  - 3: Obtain a vertex set  $W \subseteq V$  by sampling  $(3n \ln n)/t$  vertices from  $V$  independently and uniformly at random. Add the edges of shortest path in-arborescences and out-arborescences rooted at  $w$  for each  $w \in W$ .
- 

In the all-pairs spanner problem where  $\text{OPT}$  is  $\Omega(n)$ , Algorithm 1 is  $\tilde{O}(\sqrt{n})$ -approximate which matches the state-of-the-art approximation ratio given in [20]. For the pairwise spanner problem, the main challenge is the lack of a nice lower bound for  $\text{OPT}$ . In the offline setting, [34] achieves an  $\tilde{O}(n^{3/5+\epsilon})$ -approximate solution by a careful case analysis when edges have uniform lengths. We give an alternative approach that is amenable to the online setting by considering two cases, where one resolves the issue when  $\text{OPT}$  does not have a nice lower bound, and the other uses a variant of Algorithm 1 given that  $\text{OPT}$  has a nice lower bound. This approach relies on the following observation.

► **Lemma 17.**  $\text{OPT} \geq \sqrt{k}$ .

**Proof.** We observe that when the spanner has  $\ell$  edges, there are at most  $\ell$  source vertices and  $\ell$  sink vertices, so there are at most  $\ell^2$  terminal pairs. Therefore, when the spanner has  $\text{OPT}$  edges, there are at most  $\text{OPT}^2$  terminal pairs, so  $\text{OPT} \geq \sqrt{k}$ . ◀

Now we specify the simple offline algorithm given in Algorithm 2. In the beginning, we set two parameters  $T$  and  $t$  (which we will describe later), and set  $E' = \emptyset$ . An  $s_i \rightsquigarrow t_i$  path is *cheapest feasible* if it meets the distance requirement  $d_i$  by using the minimum number of edges from  $E$ . We note that cheapest feasible paths can be found by Bellman-Ford algorithm.

■ **Algorithm 2** Simple offline pairwise spanner.

- 
- 1: **if**  $k < T$  **then**
  - 2:     Add the edges of a cheapest feasible  $s_i \rightsquigarrow t_i$  path to  $E'$  for each  $i \in [k]$ .
  - 3: **else**
  - 4:     Solve LP (5) and add each edge  $e \in E$  to  $E'$  with probability  $\min\{1, x_e t \ln n\}$  independently.
  - 5:     Obtain a vertex set  $W \subseteq V$  by sampling  $(3n \ln n)/t$  vertices from  $V$  independently and uniformly at random. Add the edges of shortest path in-arborescences and out-arborescences rooted at each vertex  $w \in W$  to  $E'$ .
- 

► **Lemma 18.** *Algorithm 2 is  $\tilde{O}(n^{4/5})$ -approximate when  $T = t = n^{4/5}$ .*

**Proof.** If  $k < n^{4/5}$ , we add the edges of a cheapest feasible  $s_i \rightsquigarrow t_i$  path for each  $(s_i, t_i) \in D$ . Each cheapest feasible  $s_i \rightsquigarrow t_i$  path contains at most  $\text{OPT}$  edges, so the ratio between this solution and  $\text{OPT}$  is  $n^{4/5}$ . If  $k \geq n^{4/5}$ , then  $\text{OPT} \geq n^{2/5}$  by Lemma 17. Let  $\text{LP}^*$  be the optimal objective value of LP (5). The approximation guarantee is

$$\frac{\tilde{O}(t\text{LP}^*) + \tilde{O}(n^2/t)}{\text{OPT}} \leq \frac{\tilde{O}(n^{4/5}\text{OPT}) + \tilde{O}(n^{6/5})}{\text{OPT}} = \tilde{O}(n^{4/5})$$

since the number of edges retained from the rounding scheme is at most  $\tilde{O}(t)\text{LP}^*$  and the number of edges retained by adding arborescences is at most  $2n \cdot 3n \ln n/t$ . This summarizes the simple offline  $\tilde{O}(n^{4/5})$ -approximation algorithm. ◀

### 3.1.2 An $\tilde{O}(n^{4/5})$ -competitive online algorithm

It remains to convert the simple offline algorithm to an online algorithm. We address the two main modifications.

1. We have to (approximately) solve LP (5) online, which is not presented as a covering LP.
2. We have to round the solution of LP (5) online.

For the first modification, LP (5) is converted to an equivalent covering LP (9) (which we show in Appendix B) and approximately solved in an online fashion. For the second modification, we use an online version of the rounding scheme in Algorithm 2, such that the overall probability (from round 1 to the current round) for the edge selection is consistent with the probability based on the online solution of LP (5), by properly scaling the probability based on a conditional argument.

The online algorithm in round  $i$  is given in Algorithm 3. The same structure is used for other variants of the online pairwise spanner problem. In the beginning, we pick a threshold parameter  $T$  and a thickness parameter  $t$ , and set  $E' = \emptyset$ . Let  $x_e^i$  denote the value of  $x_e$  in the approximate solution of LP (9) obtained in round  $i$ . Let  $p_e^i := \min\{1, x_e^i t \ln n\}$ . Algorithm 3 is the online version of Algorithm 2. A key insight is that when we add the arborescences in round  $T$ , it also settles the *future* thick terminal pairs with high probability. With the outline structure of the online algorithm, we prove Theorem 2 in Appendix B.

► **Theorem 2.** *For the online pairwise spanner problem, there is a randomized polynomial time algorithm with competitive ratio  $\tilde{O}(n^{4/5})$ .*

■ **Algorithm 3** Online pairwise spanner.

---

```

1: for an arriving pair  $(s_i, t_i)$  do
2:   Convert the spanner LP (5) to the covering LP (9) and solve LP (9) online.
3:   if  $i < T$  then
4:     Add the edges of a cheapest feasible  $s_i \rightsquigarrow t_i$  path to  $E'$ .
5:   else if  $i = T$  then
6:     Obtain a vertex set  $W \subseteq V$  by sampling  $(3n \ln n)/t$  vertices from  $V$  independently
       and uniformly at random. Add the edges of shortest path in-arborescences and out-
       arborescences rooted at each vertex  $w \in W$  to  $E'$ .
7:     Add each edge  $e$  to  $E'$  independently with probability  $p_e^i$  for each edge  $e \in E \setminus E'$ .
8:   else ▷  $i > T$ 
9:     Add each edge  $e$  to  $E'$  independently with probability  $(p_e^i - p_e^{i-1})/(1 - p_e^{i-1})$  for
       each edge  $e \in E \setminus E'$ .

```

---

### 3.2 Online pairwise spanners with uniform edge lengths

In this section, we prove Theorem 6.

► **Theorem 6.** *For the online pairwise spanner problem with uniform edge lengths, there is a randomized polynomial time algorithm with competitive ratio  $\tilde{O}(n^{2/3+\varepsilon})$  for any constant  $\varepsilon \in (0, 1/3)$ .*

**Proof.** We employ Algorithm 3 with a slight tweak and set  $T = \lfloor n^{4/3-4\varepsilon} \rfloor$  and  $t = n^{2/3+\varepsilon}$ .

If  $k < T$ , instead of adding edges of a shortest  $s_i \rightsquigarrow t_i$  path, we use Theorem 1 to find an  $\tilde{O}(n^{2/3+\varepsilon})$ -competitive solution.

► **Theorem 1.** *For the online pairwise spanner problem with uniform edge lengths, there exists a deterministic polynomial time algorithm with competitive ratio  $\tilde{O}(k^{1/2+\delta})$  for any constant  $\delta > 0$ .*

For any  $\varepsilon \in (0, 1/3)$ , there exists  $\delta$  such that  $4\delta/(9 + 12\delta) = \varepsilon$ . By picking this  $\delta$ , we have

$$\begin{aligned} \left(\frac{4}{3} - 4\varepsilon\right)\left(\frac{1}{2} + \delta\right) &= \left(\frac{4}{3} - \frac{16\delta}{9 + 12\delta}\right)\left(\frac{1}{2} + \delta\right) = \frac{2}{3} + \frac{4\delta}{3} - \frac{8\delta + 16\delta^2}{9 + 12\delta} \\ &= \frac{2}{3} + \frac{12\delta + 16\delta^2 - 8\delta - 16\delta^2}{9 + 12\delta} = \frac{2}{3} + \frac{4\delta}{9 + 12\delta} = \frac{2}{3} + \varepsilon. \end{aligned}$$

Hence, the ratio between the solution obtained by Theorem 1 and OPT is

$$k^{1/2+\delta} \leq n^{(4/3-4\varepsilon)(1/2+\delta)} = n^{2/3+\varepsilon} = \tilde{O}(n^{2/3+\varepsilon}).$$

By Lemma 17, if  $k \geq T$ , then  $\text{OPT} \geq n^{2/3-2\varepsilon}$ . Let LP be the online integral solution of LP (5) obtained by Algorithm 3. The approximation guarantee is

$$\frac{\tilde{O}(t\text{LP}) + \tilde{O}(n^2/t)}{\text{OPT}} \leq \frac{\tilde{O}(n^{2/3+\varepsilon}\text{OPT}) + \tilde{O}(n^{4/3-\varepsilon})}{\text{OPT}} = \tilde{O}(n^{2/3+\varepsilon}). \quad (6)$$

◀

### 3.3 Online directed Steiner forests with uniform costs

We recall that in this problem, we are given a directed graph  $G = (V, E)$  and a set of terminals  $D = \{(s_i, t_i) \mid i \in [k]\} \subseteq V \times V$ . The goal is to find a minimum cardinality subgraph which includes an  $s_i \rightsquigarrow t_i$  path for each terminal pair  $(s_i, t_i)$ . We show the following theorem.

► **Theorem 7.** *For the online directed Steiner forest problem with uniform costs, there is a randomized polynomial time algorithm with competitive ratio  $\tilde{O}(n^{2/3+\varepsilon})$  for any constant  $\varepsilon \in (0, 1/3)$ .*

**Proof.** In this problem, for each terminal pair  $(s_i, t_i)$ , it suffices to have an  $s_i \rightsquigarrow t_i$  path. Therefore, this problem reduces to the pairwise spanner problem by setting  $d_i = \infty$  for each  $i \in [k]$ . The structure of the online algorithm is the same as that for online pairwise spanners with uniform lengths. We employ Algorithm 3 with a slight tweak and set  $T = \lfloor n^{4/3-4\varepsilon} \rfloor$  and  $t = n^{2/3+\varepsilon}$ . If  $k < T$ , instead of adding edges of a shortest  $s_i \rightsquigarrow t_i$  path, we use Theorem 1 to find an  $\tilde{O}(n^{2/3+\varepsilon})$  competitive solution<sup>3</sup>. If  $k \geq T$ , then the algorithm is  $\tilde{O}(n^{2/3+\varepsilon})$ -competitive by (6). ◀

## 4 Conclusions and Open Problems

In this work, we present the first online algorithm for pairwise spanners with competitive ratio  $\tilde{O}(n^{4/5})$  for general lengths and  $\tilde{O}(n^{2/3+\varepsilon})$  for uniform lengths, and improve the competitive ratio for the online directed Steiner forest problem with uniform costs to  $\tilde{O}(n^{2/3+\varepsilon})$  when  $k = \omega(n^{4/3})$ . We also show an efficient modified framework for online covering and packing. Our work raises several open questions that we state below.

An intriguing open problem is improving the competitive ratio for online pairwise spanners. For graphs with uniform edge lengths, there is a small polynomial gap between the state-of-the-art offline approximation ratio  $\tilde{O}(n^{3/5+\varepsilon})$  and the online competitive ratio  $\tilde{O}(n^{2/3+\varepsilon})$ . For graphs with general edge lengths, we are not aware of any studies about the pairwise spanner problem. Our  $\tilde{O}(n^{4/5})$ -competitive online algorithm intrinsically suggests an  $\tilde{O}(n^{4/5})$ -approximate offline algorithm. As the approach in [34] achieves an  $\tilde{O}(n/\sqrt{\text{OPT}})$ -approximation, we believe that the approximation ratio can be improved for the offline pairwise spanner problem, by judicious case analysis according to the cardinality of OPT.

The state-of-the-art online algorithm for Steiner forests with general costs is  $\tilde{O}(k^{1/2+\varepsilon})$ -competitive [28]. A natural open question is designing an  $o(n)$ -competitive online algorithm when  $k$  is large, and potentially extend this result to the more general buy-at-bulk network design problem. The currently best known offline approximation for Steiner forests with general costs is  $O(n^{2/3+\varepsilon})$  [20], by case analysis that settles thick and thin terminal pairs separately. However, the approach in [20] for settling thin pairs is essentially a greedy procedure which is inherently offline. Our approach utilizes the uniform cost assumption to obtain a useful lower bound for the optimal solution, which is incompatible with general costs. It would be interesting to resolve the aforementioned obstacles and have an  $o(n)$ -competitive online algorithm for directed Steiner forests with general edge costs. One open problem for uniform costs is to improve the competitive ratio, as there is a polynomial gap between the state-of-the-art offline approximation ratio  $\tilde{O}(n^{26/45+\varepsilon})$  and the online competitive ratio  $\tilde{O}(n^{2/3+\varepsilon})$ .

<sup>3</sup> One can also use the  $\tilde{O}(k^{1/2+\delta})$ -competitive online algorithm for graphs with general costs in [28].

---

**References**


---

- 1 Amir Abboud and Greg Bodwin. Reachability preservers: New extremal bounds and approximation algorithms. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, 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, 2019. [arXiv:1909.03152](https://arxiv.org/abs/1909.03152).
- 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, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. The online set cover problem. *SIAM J. Comput.*, 39(2):361–370, 2009.
- 5 Noga Alon and Baruch Schieber. Optimal preprocessing for answering on-line product queries, 1987.
- 6 Spyridon Antonakopoulos. Approximating directed buy-at-bulk network design. In *International Workshop on Approximation and Online Algorithms*, pages 13–24. Springer, 2010.
- 7 Esther M Arkin, Joseph SB Mitchell, and Christine D Piatko. Bicriteria shortest path problems in the plane. In *Proc. 3rd Canad. Conf. Comput. Geom.*, pages 153–156. Citeseer, 1991.
- 8 Pranjal Awasthi, Madhav Jha, Marco Molinaro, and Sofya Raskhodnikova. Testing lipschitz functions on hypergrid domains. *Algorithmica*, 74(3):1055–1081, 2016.
- 9 Baruch Awerbuch. Communication-time trade-offs in network synchronization. In *Proceedings of the Fourth Annual ACM Symposium on Principles of Distributed Computing*, PODC '85, page 272–276, New York, NY, USA, 1985. Association for Computing Machinery.
- 10 Baruch Awerbuch and Yossi Azar. Buy-at-bulk network design. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 542–547. IEEE, 1997.
- 11 Baruch Awerbuch, Yossi Azar, and Yair Bartal. On-line generalized steiner problem. *Theoretical Computer Science*, 324(2-3):313–324, 2004.
- 12 Baruch Awerbuch, Yossi Azar, and Serge Plotkin. Throughput-competitive on-line routing. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pages 32–40. IEEE, 1993.
- 13 Yossi Azar, Umang Bhaskar, Lisa Fleischer, and Debmalya Panigrahi. Online mixed packing and covering. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 85–100. SIAM, 2013.
- 14 Yossi Azar, Niv Buchbinder, TH Hubert Chan, Shahar Chen, Ilan Reuven Cohen, Anupam Gupta, Zhiyi Huang, Ning Kang, Viswanath Nagarajan, Joseph Naor, et al. Online algorithms for covering and packing problems with convex objectives. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 148–157. IEEE, 2016.
- 15 Nikhil Bansal, Niv Buchbinder, and Joseph Naor. Randomized competitive algorithms for generalized caching. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 235–244, 2008.
- 16 Nikhil Bansal, Niv Buchbinder, and Joseph Naor. A primal-dual randomized algorithm for weighted paging. *Journal of the ACM (JACM)*, 59(4):1–24, 2012.
- 17 Surender Baswana. Streaming algorithm for graph spanners - single pass and constant processing time per edge. *Inf. Process. Lett.*, 2008.
- 18 Surender Baswana and Telikepalli Kavitha. Faster algorithms for all-pairs approximate shortest paths in undirected graphs. *SIAM J. Comput.*, 39(7):2865–2896, 2010.
- 19 MohammadHossein Bateni and MohammadTaghi Hajiaghayi. Euclidean prize-collecting steiner forest. *Algorithmica*, 62(3-4):906–929, 2012.
- 20 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.

- 21 Piotr Berman and Chris Coulston. On-line algorithms for steiner tree problems. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 344–353, 1997.
- 22 Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P Woodruff. Transitive-closure spanners. *SIAM Journal on Computing*, 41(6):1380–1425, 2012.
- 23 Greg Bodwin and Virginia Vassilevska Williams. Better distance preservers and additive spanners. In Robert Krauthgamer, editor, *SODA*, pages 855–872. SIAM, 2016.
- 24 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.
- 25 Niv Buchbinder and Joseph Naor. Improved bounds for online routing and packing via a primal-dual approach. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 293–304. IEEE, 2006.
- 26 Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing. *Mathematics of Operations Research*, 34(2):270–286, 2009.
- 27 Niv Buchbinder and Joseph Seffi Naor. The design of competitive online algorithms via a primal-dual approach. *Foundations and Trends® in Theoretical Computer Science*, 3(2–3):93–263, 2009.
- 28 Deeparnab Chakrabarty, Alina Ene, Ravishankar Krishnaswamy, and Debmalya Panigrahi. Online buy-at-bulk network design. *SIAM J. Comput.*, 47(4):1505–1528, 2018.
- 29 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.
- 30 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.
- 31 Shiri Chechik. Approximate distance oracles with improved bounds. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *STOC*, pages 1–10. ACM, 2015.
- 32 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.
- 33 Chandra Chekuri, Mohammad Taghi Hajiaghayi, Guy Kortsarz, and Mohammad R Salavatipour. Approximation algorithms for nonuniform buy-at-bulk network design. *SIAM Journal on Computing*, 39(5):1772–1798, 2010.
- 34 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.
- 35 Lenore Cowen and Christopher G. Wagner. Compact roundtrip routing in directed networks. *J. Algorithms*, 50(1):79–95, 2004.
- 36 Bilel Derbel, Cyril Gavoille, and David Peleg. Deterministic distributed construction of linear stretch spanners in polylogarithmic time. In Andrzej Pelc, editor, *DISC*, volume 4731 of *Lecture Notes in Computer Science*, pages 179–192. Springer, 2007.
- 37 Bilel Derbel, Cyril Gavoille, David Peleg, and Laurent Viennot. On the locality of distributed sparse spanner construction. In Rida A. Bazzi and Boaz Patt-Shamir, editors, *PODC*, pages 273–282. ACM, 2008.
- 38 Michael Dinitz and Robert Krauthgamer. Directed spanners via flow-based linear programs. In *STOC*, pages 323–332, 2011.
- 39 Michael Dinitz, Yasamin Nazari, and Zeyu Zhang. Lasserre integrality gaps for graph spanners and related problems. *arXiv preprint arXiv:1905.07468*, 2019.
- 40 Michael Dinitz and Zeyu Zhang. Approximating low-stretch spanners. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 821–840. SIAM, 2016.

- 41 Dorit Dor, Shay Halperin, and Uri Zwick. All-pairs almost shortest paths. *SIAM J. Comput.*, 29(5):1740–1759, 2000.
- 42 Michael Elkin. Computing almost shortest paths. *ACM Trans. Algorithms*, 1(2):283–323, 2005.
- 43 Michael Elkin. Streaming and fully dynamic centralized algorithms for constructing and maintaining sparse spanners. *ACM Trans. Algorithms*, 7(2):20:1–20:17, 2011.
- 44 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.
- 45 Michael Elkin and David Peleg. The hardness of approximating spanner problems. *Theory Comput. Syst.*, 41(4):691–729, 2007.
- 46 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.
- 47 Manuel Fernandez, David P. Woodruff, and Taisuke Yasuda. Graph spanners in the message-passing model. In Thomas Vidick, editor, *ITCS*, volume 151 of *LIPICs*, pages 77:1–77:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
- 48 Arnold Filtser, Michael Kapralov, and Navid Nouri. Graph spanners by sketching in dynamic streams and the simultaneous communication model. *arXiv preprint arXiv:2007.14204*, 2020.
- 49 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 thirty-eighth annual ACM symposium on Theory of computing*, pages 663–670, 2006.
- 50 Naveen Garg, Goran Konjevod, and Ramamoorthi Ravi. A polylogarithmic approximation algorithm for the group steiner tree problem. *Journal of Algorithms*, 37(1):66–84, 2000.
- 51 Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995.
- 52 Elena Grigorescu, Young-San Lin, and Kent Quanrud. Online directed spanners and steiner forests. *CoRR*, 2021. [arXiv:2103.04543](https://arxiv.org/abs/2103.04543).
- 53 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, 2003. Proceedings.*, pages 606–615. IEEE, 2003.
- 54 Anupam Gupta and Viswanath Nagarajan. Approximating sparse covering integer programs online. *Mathematics of Operations Research*, 39(4):998–1011, 2014.
- 55 Anupam Gupta, R Ravi, Kunal Talwar, and Seun William Umboh. Last but not least: Online spanners for buy-at-bulk. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 589–599. SIAM, 2017.
- 56 Anupam Gupta, Kunal Talwar, and Udi Wieder. Changing bases: Multistage optimization for matroids and matchings. In *International Colloquium on Automata, Languages, and Programming*, pages 563–575. Springer, 2014.
- 57 Refael Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations research*, 17(1):36–42, 1992.
- 58 Christopher S Helvig, Gabriel Robins, and Alexander Zelikovsky. An improved approximation scheme for the group steiner problem. *Networks: An International Journal*, 37(1):8–20, 2001.
- 59 Makoto Imase and Bernard M Waxman. Dynamic steiner tree problem. *SIAM Journal on Discrete Mathematics*, 4(3):369–384, 1991.
- 60 Michael Kapralov and David P. Woodruff. Spanners and sparsifiers in dynamic streams. In Magnús M. Halldórsson and Shlomi Dolev, editors, *PODC*, pages 272–281. ACM, 2014.
- 61 Anna R. Karlin, Mark S. Manasse, Lyle A. McGeoch, and Susan S. Owicki. Competitive randomized algorithms for non-uniform problems. *Algorithmica*, 11(6):542–571, 1994.

- 62 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  $\alpha$ -synuclein through specific molecular pathways. *Cell systems*, 4(2):157–170, 2017.
- 63 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.
- 64 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.
- 65 Guy Kortsarz. On the hardness of approximating spanners. *Algorithmica*, 30:432–450, 2001.
- 66 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.
- 67 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, *SODA*, pages 1374–1392. SIAM, 2018.
- 68 Mihai Patrascu and Liam Roditty. Distance oracles beyond the Thorup-Zwick bound. *SIAM J. Comput.*, 43(1):300–311, 2014.
- 69 David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989. doi:10.1002/jgt.3190130114.
- 70 David Peleg and Jeffrey D. Ullman. An optimal synchronizer for the hypercube. *SIAM J. Comput.*, 18(4):740–747, 1989.
- 71 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.
- 72 Liam Roditty, Mikkel Thorup, and Uri Zwick. Roundtrip spanners and roundtrip routing in directed graphs. *ACM Trans. Algorithms*, 4(3):29:1–29:17, 2008.
- 73 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.
- 74 Xiangkun Shen and Viswanath Nagarajan. Online covering with  $l_q$ -norm objectives and applications to network design. *Mathematical Programming*, 184, 2020.
- 75 Andrew Chi-Chih Yao. Space-time tradeoff for answering range queries (extended abstract). In *STOC '82*, 1982.
- 76 Neal Young. The  $k$ -server dual and loose competitiveness for paging. *Algorithmica*, 11(6):525–541, 1994.
- 77 Alexander Zelikovsky. A series of approximation algorithms for the acyclic directed steiner tree problem. *Algorithmica*, 18(1):99–110, 1997.

## **A** Additional background and related work

A model related to online algorithms is that of streaming algorithms. In the streaming model an input is also revealed sequentially, but the algorithm is only allowed to use some small amount of space, which is sublinear in the length of the stream, and is supposed to maintain an approximate solution. For this model, several papers consider spanner variants, such as undirected or weighted graphs, and additive or multiplicative stretch approximations, and the aim is to build spanners with small size or distortion [17, 48, 60]. In a related direction, spanners have also been studied in the setting of dynamic data structures, where the edges of a graph are inserted or removed one at a time and the goal is to maintain an approximate solution with small update time and space [23, 43]. A relevant model is that of distributed



computation where nodes in the network communicate efficiently to build a solution [36,37,47]. As mentioned earlier, the survey by Ahmed et al. [2] gives a comprehensive account of the vast literature on spanners, and we refer the reader to the references within.

In the buy-at-bulk network design problem [10], each edge is associated with a sub-additive cost function of its *load*. Given a set of terminal demands, the goal is to route integral flows from each source to each sink concurrently to minimize the total cost of the routing. This problem is a generalization of various single-source or multicommodity network connectivity problems, including Steiner trees and Steiner forests, in which the cost function of each edge is a fixed value once allocated. While most problems admit polylogarithmic approximations in either the online or offline setting for undirected networks [11,21,32,55], the problems are much harder for directed networks. In the offline setting, the current best approximation ratio is  $O(k^\epsilon)$  for the directed Steiner tree problem [29,77],  $O(\min\{k^{1/2+\epsilon}, n^{2/3+\epsilon}\})$  for the directed Steiner forest problem [20,32], and  $O(\min\{k^{1/2+\epsilon}, n^{4/5+\epsilon}\})$  for the directed buy-at-bulk problem [6]. In the online setting for directed networks, [28] showed that compared to offline, it suffices to pay an extra polylogarithmic factor, where the polylogarithmic term was later improved by [74]. The main contribution of [28] is essentially bringing the *junction-tree-based* approach into the online setting for connectivity problems. This is the main ingredient that improves the competitive ratio of our online algorithm from  $\tilde{O}(n^{4/5+\epsilon})$  for pairwise spanners to  $\tilde{O}(n^{2/3+\epsilon})$  for Steiner forests. Our approach for online pairwise spanners with uniform edge length combines this ingredient and the offline pairwise spanner framework [34] which tackles hard distance requirements.

As previously mentioned, generating separating constraints with an oracle in the online setting is not new. For example, this arises implicitly in early work on network optimization [3] and the oracle is discussed explicitly in [26]. As a recent example, [56] develops online algorithms for the multistage matroid maintenance problem, which requires solving a covering LP with box constraints online. [56] adjusts the separation oracle to only identify constraints that are violated by at least some constant. Because of the  $\{0,1\}$ -incidence structure of their LP, the sum of primal variables has to increase by a constant to satisfy such a constraint. Meanwhile the box constraints limit the total sum of primal variables to  $O(n)$ . This leads to an  $O(n)$  bound on the number of separating constraints. While there are strong similarities to our approach, one difference is the use of the  $\{0,1\}$ -structure and box constraints to obtain their bound. Our comparably unstructured setting required us to develop an argument independent of concrete features such as these.

Beyond linear objectives, there are other variants of online covering and packing problems, which focus on different objectives with linear constraints. This includes optimizing convex objectives [14] and  $\ell_q$ -norm objectives [74]. Other online problem-dependent variants include for instance mixed covering and packing programs [13], and sparse integer programs [54]. All these frameworks utilize the primal-dual technique, which updates the covering and packing solutions simultaneously with some judiciously selected growth rate, to guarantee nice competitive ratio. Instead, our modified framework focuses on the efficiency of online algorithms for fundamental covering and packing problems, which is amenable to applications with exponential or unbounded number of constraints, where a violating one can be searched by an efficient separation oracle.

## **B** Proof of Theorem 2

► **Theorem 2.** *For the online pairwise spanner problem, there is a randomized polynomial time algorithm with competitive ratio  $\tilde{O}(n^{4/5})$ .*

**Proof.** Suppose in the online setting, there are  $k$  rounds where  $k$  may be unknown. In round  $i \in [k]$ , the pair  $(s_i, t_i)$  and the distance requirement  $d_i$  arrive and we select some new edges from  $E$  to settle  $(s_i, t_i)$ . We run Algorithm 3 by setting  $T = t = \lfloor n^{4/5} \rfloor$ . It suffices to show that 1) LP (5) can be solved online by losing a polylogarithmic factor, and 2) the overall probability of edge selection is consistent with the probability based on the online solution of LP (5).

### B.1 Converting and solving LP (5) online

The goal is to update  $x$  in a non-decreasing manner upon the arrival of the pair  $(s_i, t_i)$  to satisfy all its corresponding constraints, so that the objective value is still approximately optimal. We convert LP (5) into a covering LP as follows.

First, we check in round  $i$ , given the *edge capacity*  $x$ , if there is a (fractional)  $s_i \rightsquigarrow t_i$  path of length at most  $d_i$ . This can be captured by checking the optimum of the following LPs,

$$\max_y \sum_{P \in \mathcal{P}_i} y_P \text{ over } y : \mathcal{P}_i \rightarrow \mathbb{R}_{\geq 0} \text{ s.t. } \sum_{P|e \in P \in \mathcal{P}_i} y_P \leq x_e \text{ for all } e \in E \quad (7)$$

and its dual

$$\min_z \sum_{e \in E} x_e z_e \text{ over } z : E \rightarrow \mathbb{R}_{\geq 0} \text{ s.t. } \sum_{e \in P} z_e \geq 1 \text{ for all } P \in \mathcal{P}_i. \quad (8)$$

We say that  $x$  is *good* if the optimum of LP (7) and LP (8) is at least 1, and it is *bad* otherwise. Namely,  $x$  is good if there is at least one (fractional)  $s_i \rightsquigarrow t_i$  path of length at most  $d_i$ . In LP (8), the feasibility problem is equivalent to the following problem. Given the local graph  $G^i$  and edge weight  $z$ , is there an  $s_i \rightsquigarrow t_i$  path of length at most  $d_i$  and weight less than 1? We note that with uniform lengths, this problem can be solved by Bellman-Ford algorithm with  $d_i$  iterations, which computes the smallest weight among all  $s_i \rightsquigarrow t_i$  paths of length at most  $d_i$  in the local graph  $G^i$ .

Although this bicriteria path problem in general is NP-hard [7], an FPTAS is known to exist [57,66], which gives an approximate separation oracle. We can verify in polynomial time that if there is a path of length at most  $d_i$  and weight less than  $1 - \varepsilon$ . We obtain a solution  $z'$  for LP (8) where each constraint is satisfied by a factor of  $1 - \varepsilon$  and set  $z := z'/(1 - \varepsilon)$  as the solution.

To solve LP (5), suppose in round  $i$ , we are given  $x$ . First, we check if  $x$  is good or bad by approximately solving LP (8). If  $x$  is good, then there exists  $y$  such that  $\sum_{P \in \mathcal{P}_i} y_P \geq 1$ , i.e. the solution is feasible for LP (5), so we move on to the next round. Otherwise,  $x$  is bad, so we increment  $x$  until it becomes good, which implies  $\sum_{e \in E} x_e z_e \geq 1$  for all feasible  $z$  in LP (8). Let  $Z_i$  be the feasible polyhedron of LP (8) in round  $i$ . We derive the following LP (essentially the one in [38,39]) which is equivalent to LP (5), by considering all the constraints of LP (8) from round 1 to round  $k$ .

$$\min_x \sum_{e \in E} x_e \text{ over } x : E \rightarrow \mathbb{R}_{\geq 0} \text{ s.t. } \sum_{e \in E} z_e x_e \geq 1 \text{ for all } i \in [k] \text{ and } z \in Z_i. \quad (9)$$

In round  $i \in [k]$ , the subroutine that approximately solves LP (8) and checks if the optimum is good or not, is the separation oracle used for solving LP (9) online. Here we use Theorem 20 (the formal version of Theorem 8) to show that LP (9) can be solved online in polynomial time by paying an  $O(\log n)$  factor. This requires that both  $\log(1/z_e)$  and  $\log \text{LP}^*$  are polynomial in the number of bits used for the edge lengths, where  $\text{LP}^*$  is the

optimum of LP (9). Clearly,  $\log \text{LP}^* \leq \log |E|$  is in  $\text{poly}(n)$ . For  $\log(1/z_e)$ , the subroutine that approximately solves LP (8) returns an approximate solution  $z$  which is represented by polynomial number of bits used for the edge lengths [57, 66]. By Theorem 20, we have the following Lemma.

► **Lemma 19.** *There exists a polynomial time  $O(\log n)$ -competitive online algorithm for LP (5).*

## B.2 Conditional edge selection

After having a fractional solution of LP (5) in round  $i$  where  $i \geq T = \lfloor n^{4/5} \rfloor$ , we independently pick  $e \in E \setminus E'$  with some scaled probability so that the edge selection is consistent with the probability based on the online solution of LP (5). More specifically, let  $p_e := \min\{1, x_e t \ln n\}$  and let  $p_e^i$  be the value of  $p_e$  in round  $i$ . Let  $\tilde{E}$  be the set of edges where each edge is neither selected while adding cheapest feasible paths prior to round  $T$  nor selected while adding in-arborescences and out-arborescences in round  $T$ . We show that each edge  $e \in \tilde{E}$  has already been selected with probability  $p_e^i$  in round  $i$ . This can be proved by induction. According to Algorithm 3, the base case is round  $T$ , where  $e \in \tilde{E}$  is selected with probability  $p_e^T$ . Now suppose  $i > T$ , if  $e \in \tilde{E}$  has been selected, it is either selected prior to round  $i$  or in round  $i$ . For the former case,  $e$  must have already been selected in round  $i-1$ , with probability  $p_e^{i-1}$  by inductive hypothesis. For the later case, conditioned on  $e$  has not been selected in round  $i-1$ ,  $e$  is selected with probability  $(p_e^i - p_e^{i-1})/(1 - p_e^{i-1})$ . Therefore, in round  $i$ ,  $e$  has been selected with probability  $p_e^{i-1} + (1 - p_e^{i-1}) \cdot \frac{p_e^i - p_e^{i-1}}{1 - p_e^{i-1}} = p_e^i$ , which completes the proof. Intuitively, when  $i > T$ , conditioned on  $e \in \tilde{E}$  was not picked from round 1 to round  $i-1$ , we pick  $e$  with probability  $(p_e^i - p_e^{i-1})/(1 - p_e^{i-1})$  at round  $i$ , so that the overall probability that  $e$  is picked from round 1 to round  $i$  is  $p_e^i$ .

## B.3 Summary

We conclude the proof as follows. The overall algorithm is given in Algorithm 3. For the initialization,  $x$  is a zero vector,  $E'$  is an empty set, and  $T = t = \lfloor n^{4/5} \rfloor$ . The set  $E'$  is the solution. We pay an extra logarithmic factor for solving LP (5) online by Lemma 19. The competitive ratio remains  $\tilde{O}(n^{4/5})$ . ◀

## C Online Covering in Polynomial Time

This section is devoted to proving the formal version of Theorem 8. We recall that the problem of interest is to solve the covering LP (2) online:

$$\text{minimize } \langle \mathbf{c}, x \rangle \text{ over } x \in \mathbb{R}_{\geq 0}^n \text{ s.t. } Ax \geq \mathbf{1}$$

where  $A \in \mathbb{R}_{\geq 0}^{m \times n}$  consists of  $m$  covering constraints,  $\mathbf{1} \in \mathbb{R}_{> 0}^m$  is a vector of all ones treated as the lower bound of the covering constraints, and  $\mathbf{c} \in \mathbb{R}_{> 0}^n$  denotes the positive coefficients of the linear cost function.

In the online covering problem, the cost vector  $\mathbf{c}$  is given offline, and each of these covering constraints is presented one by one in an online fashion, that is,  $m$  can be unknown. In round  $i \in [m]$ ,  $\{a_{ij}\}_{j \in [n]}$  (where  $a_{ij}$  denotes the  $i$ -th row  $j$ -th column entry of  $A$ ) is revealed, and we have to monotonically update  $x$  so that the constraint  $\sum_{j \in [n]} a_{ij} x_j \geq 1$  is satisfied. We always assume that there is at least one positive entry  $a_{ij}$  in each round  $i$ , otherwise constraint  $i$  cannot be satisfied since all the row entries are zeros. The goal is to update  $x$  in a non-decreasing manner and approximately minimize the objective value  $\langle \mathbf{c}, x \rangle$ .

We recall that an important idea in this line of work is to simultaneously consider the dual packing problem:

$$\text{maximize } \langle \mathbf{1}, y \rangle \text{ over } y \in \mathbb{R}_{\geq 0}^m \text{ s.t. } A^T y \leq \mathbf{c}$$

where  $A^T$  consists of  $n$  packing constraints with an upper bound  $\mathbf{c}$  given offline.

The primal-dual framework in [26] simultaneously solves both LP (2) and LP (3), and crucially uses LP-duality and strong connections between the two solutions to argue that they are both nearly optimal. The modified framework closely follows the *guess-and-double* scheme in [26]. Specifically, the scheme runs in phases where each phase estimates a lower bound for the optimum. When the first constraint arrives, the scheme generates the first lower bound

$$\alpha(1) \leftarrow \min_{j \in [n]} \left\{ \frac{c_j}{a_{1j}} \mid a_{1j} > 0 \right\} \leq \text{OPT}$$

where  $c_j$  is the  $j$ -th entry of  $\mathbf{c}$  and OPT is the optimal value of LP (2).

During phase  $r$ , we always assume that the lower bound of the optimum is  $\alpha(r)$  until the online objective  $\langle \mathbf{c}, x \rangle$  exceeds  $\alpha(r)$ . Once the online objective exceeds  $\alpha(r)$ , we start the new phase  $r + 1$  from the current violating constraint<sup>4</sup> (let us call it constraint  $i_{r+1}$ , in particular,  $i_1 = 1$ ), and double the estimated lower bound, i.e.  $\alpha(r + 1) \leftarrow 2\alpha(r)$ . We recall that  $x$  must be updated in a non-decreasing manner, so the algorithm maintains  $\{x_j^r\}$ , which denotes the value of each variable  $x_j$  in each phase  $r$ , and the value of each variable  $x_j$  is actually set to  $\max_r \{x_j^r\}$ .

In Algorithm 4, we describe one round of the modified scheme in phase  $r$ . When a covering constraint  $i$  arrives, we introduce a packing variable  $y_i = 0$ . If the constraint is violated, we increment each  $x_j$  according to an exponential function of  $y_i$  until the constraint is satisfied *by a factor of 2*. This is the main difference between the modified framework and [26], which increments the variables until the constraint is satisfied.

■ **Algorithm 4** Online Covering.

- 
- 1: **for** arriving covering constraint  $i$  **do**
  - 2:      $y_i \leftarrow 0$ . ▷ the packing variable  $y_i$  is used for the analysis
  - 3:     **if**  $\sum_{j=1}^n a_{ij} x_j^r < 1$  **then** ▷ if constraint  $i$  is not satisfied
  - 4:         **while**  $\sum_{j=1}^n a_{ij} x_j^r < 2$  **do** ▷ update until constraint  $i$  is satisfied by a factor of 2
  - 5:             Increase  $y_i$  continuously.
  - 6:             Increase each variable  $x_j^r$  by the following increment function:

$$x_j^r \leftarrow \frac{\alpha(r)}{2nc_j} \exp \left( \frac{\ln(2n)}{c_j} \sum_{k=i_r}^i a_{kj} y_k \right).$$


---

Although the augmentation is in a continuous fashion, it is not hard to implement it in a discrete way for any desired precision by binary search. Therefore, to show that the modified framework is efficient, it suffices to bound the number of violating constraints it will encounter. The performance of the modified scheme is analyzed in Theorem 20 (the formal version of Theorem 8).

---

<sup>4</sup> In [26], the scheme starts all over again from the first constraint. We start from the current violating constraint because it is more amenable when violating constraints are generated by a separation oracle. There is no guarantee for the order of arriving violating constraints in such settings.

► **Theorem 20.** *There exists an  $O(\log n)$ -competitive online algorithm for the covering LP (2) which encounters  $\text{poly}(n, \log \text{OPT}, \log(1/\alpha(1)))$  violating constraints.*

**Proof.** The proof for the  $O(\log n)$ -competitiveness closely follows the one in [26]. Let  $X(r)$  and  $Y(r)$  be the covering and packing objective values, respectively, generated during phase  $r$ . The following claims are used to show that Algorithm 4 is  $O(\log n)$ -competitive.

1.  $x$  is feasible.
2. For each *finished* phase  $r$ ,  $\alpha(r) \leq 4 \ln(2n) \cdot Y(r)$ .
3.  $y$  generated during phase  $r$  is feasible.
4. The sum of the covering objective generated from phase 1 to  $r$  is at most  $2\alpha(r)$ .
5. Let  $r'$  be the last phase, then the covering objective  $\langle \mathbf{c}, x \rangle \leq 2\alpha(r')$ .

From these five claims together with weak duality, we conclude that

$$\langle \mathbf{c}, x \rangle \leq 2\alpha(r') = 4\alpha(r' - 1) \leq 16 \ln(2n) \cdot Y(r' - 1) \leq 16 \ln(2n) \cdot \text{OPT}.$$

Now we show that Algorithm 4 encounters  $\text{poly}(n, \log \text{OPT}, \log(1/\alpha(1)))$  violating constraints. We first show that there are  $O(\log \log n + \log \text{OPT} + \log(1/\alpha(1)))$  phases. The estimated lower bound  $\alpha$  doubles when we start a new phase. Suppose there are  $r'$  phases, then  $\alpha(1) \cdot 2^{r'-1} = O(\log n) \text{OPT}$  because Algorithm 4 is  $O(\log n)$ -competitive. This implies that  $r' = O(\log \log n + \log \text{OPT} + \log(1/\alpha(1)))$ .

In each phase, when a violating constraint arrives, we increment  $x$  so that the constraint is satisfied by a factor of 2. This implies that at least one variable  $x_j$  is doubled.  $x_j = O(\log n) \text{OPT}/c_j$  because  $c_j x_j \leq \langle \mathbf{c}, x \rangle = O(\log n) \text{OPT}$ . At the start of phase  $r$ ,  $x_j = \alpha(r)/(2nc_j) \geq \alpha(1)/(2nc_j)$ . Suppose  $x_j$  has been doubled  $t$  times in phase  $r$ , then

$$\frac{\alpha(1)}{2nc_j} \cdot 2^t \leq \frac{\alpha(r)}{2nc_j} \cdot 2^t \leq x_j = O(\log n) \frac{\text{OPT}}{c_j}$$

which indicates that  $t = O(\log n + \log \text{OPT} + \log(1/\alpha(1)))$ .

There are  $n$  variables and  $r'$  phases, and in each phase, each variable is doubled at most  $t$  times. Therefore, Algorithm 4 encounters  $\text{poly}(n, \log \text{OPT}, \log(1/\alpha(1)))$  violating constraints. ◀



# Query Complexity of Global Minimum Cut

Arijit Bishnu ✉

Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, India

Arijit Ghosh ✉

Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, India

Gopinath Mishra ✉

Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, India

Manaswi Paraashar ✉

Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, India

---

## Abstract

In this work, we resolve the query complexity of global minimum cut problem for a graph by designing a randomized algorithm for approximating the size of minimum cut in a graph, where the graph can be accessed through local queries like DEGREE, NEIGHBOR, and ADJACENCY queries.

Given  $\epsilon \in (0, 1)$ , the algorithm with high probability outputs an estimate  $\hat{t}$  satisfying the following  $(1 - \epsilon)t \leq \hat{t} \leq (1 + \epsilon)t$ , where  $t$  is the size of minimum cut in the graph. The expected number of local queries used by our algorithm is  $\min\{m + n, \frac{m}{\epsilon}\} \text{poly}(\log n, \frac{1}{\epsilon})$  where  $n$  and  $m$  are the number of vertices and edges in the graph, respectively. Eden and Rosenbaum showed that  $\Omega(m/t)$  local queries are required for approximating the size of minimum cut in graphs, but no local query based algorithm was known. Our algorithmic result coupled with the lower bound of Eden and Rosenbaum [APPROX 2018] resolve the query complexity of the problem of estimating the size of minimum cut in graphs using local queries.

Building on the lower bound of Eden and Rosenbaum, we show that, for all  $t \in \mathbb{N}$ ,  $\Omega(m)$  local queries are required to decide if the size of the minimum cut in the graph is  $t$  or  $t - 2$ . Also, we show that, for any  $t \in \mathbb{N}$ ,  $\Omega(m)$  local queries are required to find all the minimum cut edges even if it is promised that the input graph has a minimum cut of size  $t$ . Both of our lower bound results are randomized, and hold even if we can make RANDOM EDGE queries in addition to local queries.

**2012 ACM Subject Classification** Mathematics of computing → Probabilistic algorithms; Theory of computation → Streaming, sublinear and near linear time algorithms

**Keywords and phrases** Query complexity, Global mincut

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.6

**Category** APPROX

**Related Version** *Previous Version:* <https://arxiv.org/abs/2007.09202>

*Previous Version:* <https://ecc.weizmann.ac.il/report/2020/108/>

## 1 Introduction

The global minimum cut (denoted MINCUT) of a connected, unweighted, undirected and simple graph  $G = (V, E)$ ,  $|V| = n$  and  $|E| = m$ , is a partition of the vertex set  $V$  into two sets  $S$  and  $V \setminus S$  such that the number of edges between  $S$  and  $V \setminus S$  is minimized. Let  $\text{CUT}(G)$  denote this edge set corresponding to a minimum cut in  $G$ , and  $t$  denote  $|\text{CUT}(G)|$ . The problem is so fundamental that researchers keep coming back to it again and again across different models [23, 22, 27, 25, 28, 1, 29, 14, 12, 13, 21]. The algorithmic landscape for the minimum cut problem has been heavily influenced by Karger and Stein's work [22, 23] and algorithmic solutions for minimum cut across different models [27, 25, 28, 1, 29, 14, 12,



© Arijit Bishnu, Arijit Ghosh, Gopinath Mishra, and Manaswi Paraashar;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 6; pp. 6:1–6:15



Leibniz International Proceedings in Informatics

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

13, 21] \* have revisited their approach [22, 23]. Fundamental graph parameter estimation problems, like estimation of the number of edges [11, 17], triangles [7], cliques [8], stars [18], etc. have been solved in the local and bounded query models [16, 17, 24]. Estimation of the size of MINCUT is also in the league of such fundamental problems to be solved in the model of local queries.

In property testing [15], a graph can be accessed at different granularities – the query oracle can answer properties about graph that are local or global in nature. Local queries involve the relation of a vertex with its immediate neighborhood, whereas, global queries involve the relation between sets of vertices. Recently using a global query, named CUT QUERY [29], the problem of estimating and finding MINCUT was solved, but the problem of estimating or finding MINCUT using local queries has not been solved. The fundamental contribution of our work is to resolve the query complexity of MINCUT using local queries. We resolve both the estimation and finding variants of the problem. To start with, we formally define the query oracle models we would be needing for discussions that follow. Before proceeding further, we note that all the upper and lower bound results in this paper are randomized unless otherwise stated.

**The query oracle models.** We start with the most studied local queries and the random edge query for a graph  $G = (V, E)$  where the vertex set  $V$  is known but the edge set  $E$  is unknown.

- Local Query
  - DEGREE query: given  $u \in V$ , the oracle reports the degree of  $u$  in  $V$ ;
  - NEIGHBOR query: given  $u \in V$  and a positive integer  $i$ , the oracle reports the  $i$ -th neighbor of  $u$ , if it exists; otherwise, the oracle reports  $\perp$ ;
  - ADJACENCY query: given  $u, v \in V$ , the oracle reports whether  $\{u, v\} \in E$ .
- RANDOM EDGE query: The query outputs an uniformly random edge of  $G$ .

Apart from the local queries mentioned, in the last few years, researchers have also used the RANDOM EDGE query [2, 3]. Notice that the randomness will be over the probability space of all edges, and hence, a random edge query is not a local query. This fact is also evident from the work of Eden and Rosenbaum [10]. We use RANDOM EDGE query in conjunction with local queries only for lower bound purposes. The other query oracle relevant for our discussion will be a *global query* called the CUT QUERY proposed by Rubinfeld et al. [29] that was motivated by submodular function minimization. The query takes as input a subset  $S$  of the vertex set  $V$  and returns the size of the cut between  $S$  and  $V \setminus S$  in the graph  $G$ .

**Prologue.** Our motivation for this work is twofold – MINCUT is a fundamental graph estimation problem that needs to be solved in the local query oracle model and the lower bound of Eden and Rosenbaum [9] who extended the seminal work of Blais et al. [5] to develop a technique for proving query complexity lower bounds for graph properties via reductions from communication complexity. Using those techniques, for graphs that can be accessed by only local queries like DEGREE, NEIGHBOR, ADJACENCY and RANDOM EDGE, Eden and Rosenbaum [9] showed that MINCUT admits a lower bound of  $\Omega(m/t)$  in general graphs, where  $m$  and  $t$  are the number of edges and the size of the minimum cut, respectively, in the graph. However, the query complexity of estimating MINCUT (in general graphs)

---

\* The list is to name a few and it is not exhaustive.



has remained elusive as there is no matching upper bound. It is surprising that the query complexity of a fundamental graph problem like MINCUT has not been addressed before Eden and Rosenbaum [9].

In this work, we prove an upper bound of  $\min\{m + n, \frac{m}{t}\} \text{poly}(\log n, \frac{1}{\epsilon})$  for estimating MINCUT using local queries only (and not RANDOM EDGE query). Observe that for  $t \geq 1$ , our upper bound almost matches with the lower bound given by Eden and Rosenbaum [9] if we ignore  $\text{poly}(\log n, \frac{1}{\epsilon})$  term. Note that the case of  $t = 0$  is the CONNECTIVITY problem, where the objective is to decide whether the graph is connected. We build on the lower bound result for *estimating* MINCUT by Eden et al. [9] to show that  $\Omega(m)$  local queries are required if we want to determine the exact size of a MINCUT or find a MINCUT. This result implies that there is a separation between the problem of estimating the size of MINCUT and the problem of finding a MINCUT using local queries. On the other hand, Babai et al. [4] showed that CONNECTIVITY testing has a randomized communication complexity of  $\Omega(n)$ <sup>†</sup>. This implies that any algorithm that solves CONNECTIVITY requires  $\Omega(n/\log n)$  local queries. This is because Alice and Bob can deterministically simulate each local query by communicating at most  $\log n$  bits. We have already discussed that  $\Omega(m)$  local queries are needed to solve CONNECTIVITY. From the lower bounds of  $\Omega(m)$  and  $\Omega(n/\log n)$  for CONNECTIVITY along with the lower bound result of Eden and Rosenbaum [9] for estimating MINCUT, our upper bound result on estimating MINCUT (ignoring  $\text{poly}(\log n, \frac{1}{\epsilon})$  term) is tight - this settles the query complexity of MINCUT using local queries.

Prior to our work, no local query based algorithm existed for MINCUT. But it was Rubinfeld et al. [29] who studied MINCUT for the first time using CUT QUERY, a global query. They showed that there exists a randomized algorithm for finding a MINCUT in  $G$  using  $\tilde{O}(n)$ <sup>‡</sup> CUT QUERY. The deterministic lower bound of  $\Omega(n \log n)$  by Hajnal et al. [20], for CONNECTIVITY in communication complexity, implies that  $\Omega(n)$  CUT QUERY are required by any deterministic algorithm to estimate MINCUT. The randomized lower bound of  $\Omega(n)$  by Babai et al. [4], for CONNECTIVITY in communication complexity, says that  $\Omega(n/\log n)$  CUT QUERY are necessary for any randomized algorithm to estimate MINCUT<sup>§</sup>. So, if we ignore polylogarithmic factors, the upper bound result by Rubinfeld et al. for finding a MINCUT along with the above discussed lower bound result for finding a MINCUT, imply that there is no separation between the problem of estimating size of MINCUT and the problem of finding a MINCUT using CUT QUERY. On a different note, Gaur et al. [19] showed a deterministic lower bound of  $3n/2$  on the number of CUT QUERY for estimating MINCUT.

**Problem statements and results.** We focus on two problems in this work.

Minimum Cut Estimation

**Input:** A parameter  $\epsilon \in (0, 1)$ , and access to an unknown graph  $G$  via local queries

**Output:** A  $(1 \pm \epsilon)$ -approximation to  $|\text{CUT}(G)|$ .

Minimum Cut Finding

**Input:** Access to an unknown graph  $G$  via local queries

**Output:** Find a set  $\text{CUT}(G)$ .

<sup>†</sup> Here the edge set of the graph is partitioned among Alice and Bob. The objective of Alice and Bob is to determine whether the graph is connected by communicating.

<sup>‡</sup>  $\tilde{O}(n)$  hides polylogarithmic terms in  $n$ .

<sup>§</sup> We would like to thank Troy Lee for pointing us to the papers of Hajnal et al. [20] and Babai et al. [4]

Our results are the following.

► **Theorem 1.1** (Minimum cut estimation using local queries). *There exists an algorithm, with DEGREE and NEIGHBOR query access to an unknown graph  $G = (V, E)$ , that solves the minimum cut estimation problem with high probability. The expected number of queries used by the algorithm is  $\min\{m + n, \frac{m}{t}\} \text{poly}(\log n, \frac{1}{\epsilon})$ .*

Building on the lower bound construction of Eden and Rosenbaum [9], we show that no nontrivial query algorithm exists for finding a minimum cut or even estimating the exact size of a minimum cut in graphs.

► **Theorem 1.2** (Lower bound for minimum cut finding, i.e.,  $\text{CUT}(G)$ ). *Let  $m, n, t \in \mathbb{N}$  with  $t \leq n - 1$  and  $2nt \leq m \leq \binom{n}{2}$ <sup>¶</sup>. Any algorithm that has access to DEGREE, NEIGHBOR, ADJACENCY and RANDOM EDGE queries to an unknown graph  $G = (V, E)$  must make at least  $\Omega(m)$  queries in order to find all the edges in a minimum cut of  $G$  with probability  $2/3$ .*

► **Theorem 1.3** (Lower bound for finding the exact size of the minimum cut, i.e.,  $|\text{CUT}(G)|$ ). *Let  $m, n, t \in \mathbb{N}$  with  $2 \leq t \leq n - 2$  and  $2nt \leq m \leq \binom{n}{2}$ . Any algorithm that has access to DEGREE, NEIGHBOR, ADJACENCY and RANDOM EDGE queries to an unknown graph  $G = (V, E)$  must make at least  $\Omega(m)$  queries in order to decide whether  $|\text{CUT}(G)| = t$  or  $|\text{CUT}(G)| = t - 2$  with probability  $2/3$ .*

► **Remark 1.** Local queries show a clear separation in its power in finding MINCUT as opposed to the estimation problem. This is established by using the tight lower bound of minimum cut estimation (viz.  $\Omega(m/t)$  lower bound of Eden and Rosenbaum and our Theorem 1.1) vis-a-vis minimum cut finding as mentioned in our Theorems 1.2 and 1.3 on lower bound for finding  $\text{CUT}(G)$ . As noted earlier, there is no such separation between estimating and finding MINCUT when CUT QUERY is used.

**Notations.** In this paper, we denote the set  $\{1, \dots, n\}$  by  $[n]$ . For ease of notation, we sometimes use  $[n]$  to denote the set of vertices of a graph. We say  $x \geq 0$  is an  $(1 \pm \epsilon)$ -approximation to  $y \geq 0$  if  $|x - y| \leq \epsilon y$ .  $V(G)$  and  $E(G)$  would denote the vertex and edge sets when we want to make the graph  $G$  explicit, else we use  $V$  and  $E$ . For a graph  $G$ ,  $\text{CUT}(G)$  denotes the set of edges in a minimum cut of  $G$ . Let  $A_1, A_2$  be a partition of  $V$ , i.e.,  $V = A_1 \cup A_2$  with  $A_1 \cap A_2 = \emptyset$ . Then,  $\mathcal{C}_G(A_1, A_2) = \{\{u, v\} \in E : u \in A_1 \text{ and } v \in A_2\}$ . The statement *with high probability* means that the probability of success is at least  $1 - \frac{1}{n^c}$ , where  $c$  is a positive constant.  $\tilde{\Theta}(\cdot)$  and  $\tilde{O}(\cdot)$  hides a  $\text{poly}(\log n, \frac{1}{\epsilon})$  term in the upper bound.

**Organization of the paper.** Section 2 discusses the query algorithm for estimating the MINCUT while Section 3 proves lower bounds on finding the MINCUT. Section 4 concludes with a few observations.

## 2 Estimation algorithm

In this Section, we will prove Theorem 1.1. In Section 2.1, we discuss about the intuitions and give the overview of our algorithm. We formalize the intuitions in Section 2.2.

<sup>¶</sup> As mentioned at the beginning of the introduction,  $n, m$  and  $t$  denote the number of vertices, number of edges and the size of MINCUT in  $G$ , respectively.

## 2.1 Overview of our algorithm

We start by assuming that a lower bound  $\hat{t}$  on  $t = |\text{CUT}(G)|$  is known. Later, we discuss how to remove this assumption.

We generate a random subgraph  $H$  of  $G$  by sampling each edge of the graph  $G$  independently with probability  $p = \Theta(\log n / \epsilon^2 \hat{t})$ <sup>||</sup>. Using Chernoff bound, we can show that any particular cut of size  $k$ ,  $k \geq t$ , in  $G$  is *well approximated* in  $H$  with probability at least  $n^{-\Omega(k/\hat{t})}$ . With this idea, consider the following Algorithm, stated informally, for minimum cut estimation.

### Algorithm-Sketch (works with $\hat{t} \leq t$ )

**Step-1:** Generate a random subgraph  $H$  of  $G$  by sampling each edge in  $G$  independently with probability  $p = \Theta(\log n / \epsilon^2 \hat{t})$ . Note that  $H$  can be generated by using  $\tilde{O}(m/\hat{t})$  DEGREE and NEIGHBOR queries in expectation. We will discuss it in Algorithm 1 in Section 2.2.

**Step-2** Determine  $|\text{CUT}(H)|$  and report  $\tilde{t} = \frac{|\text{CUT}(H)|}{p}$  as a  $(1 \pm \epsilon)$ -approximation of  $|\text{CUT}(G)|$ . The number of queries made by the above algorithm is  $\tilde{O}(m/\hat{t})$  in expectation. But it produces correct output only when the vertex partition corresponding to  $\text{CUT}(G)$  and  $\text{CUT}(H)$  are the same. This is not the case always. If we can show that all cuts in  $G$  are approximately preserved in  $H$ , then Algorithm-Sketch produces correct output with high probability. The main bottleneck to prove it is that the total number of cuts in  $G$  can be exponential. A result of Karger (stated in the following lemma) will help us to make Algorithm-Sketch work.

► **Lemma 2.1** (Karger [22]). *For a given graph  $G$  the number of cuts in  $G$  of size at most  $j \cdot |\text{CUT}(G)|$  is at most  $n^{2j}$ .*

Using the above lemma along with Chernoff bound, we can show the following.

► **Lemma 2.2.** *Let  $G$  be a graph,  $\hat{t} \leq t = |\text{CUT}(G)|$  and  $\epsilon \in (0, 1)$ . If  $H(V(G), E_p)$  is a subgraph of  $G$  where each edge in  $E(G)$  is included in  $E_p$  with probability  $p = \min\left\{\frac{200 \log n}{\epsilon^2 \hat{t}}, 1\right\}$  independently, then every cut of size  $k$  in  $G$  has size  $pk(1 \pm \epsilon)$  in  $H$  with probability at least  $1 - \frac{1}{n^{10}}$ .*

The above lemma implies the correctness of Algorithm-Sketch, which is for minimum cut estimation when we know a lower bound  $\hat{t}$  of  $|\text{CUT}(G)|$ . But in general we do not know any such  $\hat{t}$ . To get around the problem, we start guessing  $\hat{t}$  starting from  $\frac{n}{2}$  each time reducing  $\hat{t}$  by a factor of 2. The guessing scheme gives the desired solution due to Lemma 2.2 coupled with the following intuition when  $\hat{t} = \Omega(t \log n / \epsilon^2)$  – if we generate a random subgraph  $H$  of  $G$  by sampling each edge with probability  $p = \Theta(\log n / \epsilon^2 \hat{t})$ , then  $H$  is disconnected with at least a constant probability. So, it boils down to a connectivity check in  $H$ . The intuition is formalized in the following lemma that can be proved using Markov's inequality.

► **Lemma 2.3.** *Let  $G$  be a graph with  $|V(G)| = n$ ,  $\hat{t} \geq \frac{2000 \log n}{\epsilon^2} |\text{CUT}(G)|$  and  $\epsilon \in (0, 1)$ . If  $H(V(G), E_p)$  be a subgraph of  $G$  where each edge in  $E(G)$  is included in  $E_p$  independently with probability  $p = \min\left\{\frac{200 \log n}{\epsilon^2 \hat{t}}, 1\right\}$ , then  $H$  is connected with probability at most  $\frac{1}{10}$ .*

Before moving to the next section, we prove Lemmas 2.2 and 2.3 here.

<sup>||</sup> Though  $p$  can be more than 1 here, we will make it explicit in the formal description

## 6:6 Query Complexity of Global Minimum Cut

**Proof of Lemma 2.2.** If  $p = 1$ , we are done as the graph  $H$  is exactly the same as that of  $G$ . So, without loss of generality assume that the graph  $G$  is connected. Otherwise, the lemma holds trivially as  $|\text{CUT}(G)| = 0$ , i.e.,  $\hat{t} = 0$  and  $p = 1$ . Hence, for the rest of the proof we will assume that  $p = \frac{200 \log n}{\epsilon^2 \hat{t}}$ .

Consider a cut  $\mathcal{C}_G(A_1, A_2)$  of size  $k$  in  $G$ . As we are sampling each edge with probability  $p$ , the expected size of the cut  $\mathcal{C}_H(A_1, A_2)$  is  $pk$ . Using Chernoff bound (see Lemma B.1 in Section B), we get

$$\mathbb{P}(|\mathcal{C}_H(A_1, A_2) - pk| \geq \epsilon pk) \leq e^{-\epsilon^2 pk/3\hat{t}} = n^{-\frac{100k}{3\hat{t}}} \quad (1)$$

Note that here we want to show that every cut in  $G$  is approximately preserved in  $H$ . To do so, we will use Lemma 2.1 along with Equation (1) as follows. Let  $Z_1, Z_2, \dots, Z_\ell$  be the partition of the set of all cuts in  $G$  such that each cut in  $Z_j$  has the number of edges between  $[j \cdot |\text{CUT}(G)|, (j+1) \cdot |\text{CUT}(G)|]$ , where  $\ell \leq \frac{n}{|\text{CUT}(G)|}$  and  $j \leq \ell - 1$ . From Lemma 2.1,  $|Z_j| \leq n^{2j}$ . Consider a particular  $Z_j, j \in [\ell]$ . Using the union bound along with Equation 1, the probability that there exists a cut in  $Z_j$  that is not approximately preserved in  $H$  is at most  $\frac{1}{n^{11}}$ . Taking union bound over all  $Z_j$ 's, the probability that there exists a cut in  $G$  that is not approximately preserved is at most  $\frac{1}{n^{10}}$ .  $\blacktriangleleft$

**Proof of Lemma 2.3.** Let  $\mathcal{C}_G(A_1, A_2)$  be a minimum cut in  $G$ . Observe,  $\mathbb{E}[|\mathcal{C}_H(A_1, A_2)|] = p|\mathcal{C}_G(A_1, A_2)| = p|\text{CUT}(G)|$ . From Markov's inequality, we get  $\mathbb{P}(G \text{ is connected}) \leq \mathbb{P}(|\mathcal{C}_G(A_1, A_2)| \geq 1) \leq \mathbb{E}[|\mathcal{C}_G(A_1, A_2)|] \leq \frac{1}{10}$ .  $\blacktriangleleft$

## 2.2 Formal Algorithm (Proof of Theorem 1.1)

In this Section, the main algorithm for minimum cut estimation is described in Algorithm 3 (ESTIMATOR) that makes multiple calls to Algorithm 2 (VERIFY-GUESS). The VERIFY-GUESS subroutine in turn calls Algorithm 1 (SAMPLE) multiple times.

Given degree sequence of the graph  $G$ , that can be obtained using degree queries, we will first show how to independently sample each edge of  $G$  with probability  $p$  using only NEIGHBOR queries.

■ **Algorithm 1** SAMPLE( $D, p$ ).

---

**Input:**  $D = \{d(i) : i \in [n]\}$ , where  $d(i)$  denotes the degree of the  $i$ -th vertex in the graph  $G$ , and  $p \in (0, 1]$ .

**Output:** Return a subgraph  $H(V, E_p)$  of  $G(V, E)$  where each edge in  $E(G)$  is included in  $E_p$  with probability  $p$ .

Set  $q = 1 - \sqrt{1 - p}$  and  $m = \frac{\sum_{i=1}^n d_i}{2}$ ;

**for** (each  $i \in [n]$ ) **do**

**for** (each  $j \in [d(i)]$  with  $d(i) > 0$ ) **do**

        // Let  $r_j$  be the  $j$ -th neighbor of the  $i$ -th vertex;

        Add the edge  $(i, r_j)$  to the set  $E_p$  with probability  $q$ ;

**end**

**end**

Return the graph  $H(V, E_p)$ .

---

The following lemma proves the correctness of the above algorithm SAMPLE( $D, p$ ).

► **Lemma 2.4.**  $\text{SAMPLE}(D, p)$  returns a random subgraph  $H(V(G), E_p)$  of  $G$  such that each edge  $e \in E$  is included in  $E_p$  independently with probability  $p$ . Moreover, in expectation, the number of NEIGHBOR queries made by  $\text{SAMPLE}(D, p)$  is at most  $2pm$ .

**Proof.** From the description of  $\text{SAMPLE}(D, p)$ , it is clear that the probability that a particular edge  $e \in E(G)$  is added to  $E_p$  with probability  $1 - (1 - p)^2 = p$ .

Observe,  $\mathbb{E}[|E_p|] = pm$ . The bound on the number of NEIGHBOR queries now follows from the fact that  $\text{SAMPLE}(D, p)$  makes at most  $2|E_p|$  many NEIGHBOR queries. ◀

One of the core ideas behind the proof of Theorem 1.1 is that, given an estimate  $\hat{t}$  of  $t$ , we want to efficiently (in terms of number of local queries used by the algorithm) decide if  $\hat{t} \leq t$  or if  $\hat{t} \gtrsim \frac{\log n}{\epsilon^2} \times t$ . Using Algorithm 2, we will show that this can be done using  $\tilde{O}\left(\frac{m}{\hat{t}}\right)$  many NEIGHBOR queries in expectation. Another interesting feature of Algorithm 2 is that, if estimate  $\hat{t} \leq t$ , then Algorithm 2 outputs an estimate which is a  $(1 \pm \epsilon)$ -approximation of  $t$ .

■ **Algorithm 2**  $\text{VERIFY-GUESS}(D, \hat{t}, \epsilon)$ .

---

**Input:**  $D = \{d(i) : i \in [n]\}$ , where  $d(i)$  denotes the degree of the  $i$ -th vertex in the graph  $G$  and  $m = \frac{1}{2} \sum_{i=1}^n d(i) \geq n - 1$ . Also, a guess  $\hat{t}$ , with  $1 \leq \hat{t} \leq \frac{n}{2}$ , for the size of the global minimum cut in  $G$ , and  $\epsilon \in (0, 1)$ .

**Output:** The algorithm should “ACCEPT” or “REJECT”  $\hat{t}$ , with high probability, depending on the following

- If  $\hat{t} \leq |\text{CUT}(G)|$ , then ACCEPT  $\hat{t}$  and also output a  $(1 \pm \epsilon)$ -approximation of  $|\text{CUT}(G)|$
- If  $\hat{t} \geq \frac{200 \log n}{\epsilon^2} |\text{CUT}(G)|$ , then REJECT  $\hat{t}$

Set  $p = \min \left\{ \frac{200 \log^2 n}{\epsilon^2 \hat{t}}, 1 \right\}$ ;

Set  $\Gamma = 100 \log n$  and Call  $\text{SAMPLE}(D, p)$   $\Gamma$  times;

// Let  $H_i(V, E_p^i)$  be the output of  $i$ -th call to  $\text{SAMPLE}(D, p)$ , where  $i \in [\Gamma]$

**if** (at least  $\Gamma/2$  many  $H_i$ 's are disconnected) **then**

  | REJECT  $\hat{t}$

**end**

**else if** (all  $H_i$ 's are connected) **then**

  | ACCEPT  $\hat{t}$ , find  $\text{CUT}(H_i)$  for any  $i \in [\Gamma]$ , and return  $\tilde{t} = \frac{|\text{CUT}(H_i)|}{p}$ .

**end**

**else**

  | Return FAIL.

  | // When we cannot decide between “REJECT” or “ACCEPT” it will return FAIL

**end**

---

The following lemma proves the correctness of Algorithm 2. The lemmas used in proof are Lemmas 2.2, 2.3 and 2.4.

► **Lemma 2.5.**  $\text{VERIFY-GUESS}(D, \hat{t}, \epsilon)$  in expectation makes  $\tilde{O}\left(\frac{m}{\hat{t}}\right)$  many NEIGHBOR queries to the graph  $G$  and behaves as follows:

- (i) If  $\hat{t} \geq \frac{200 \log n}{\epsilon^2} |\text{CUT}(G)|$ , then  $\text{VERIFY-GUESS}(D, \hat{t}, \epsilon)$  rejects  $\hat{t}$  with probability at least  $1 - \frac{1}{n^9}$ .
- (ii) If  $\hat{t} \leq |\text{CUT}(G)|$ , then  $\text{VERIFY-GUESS}(D, \hat{t}, \epsilon)$  accepts  $\hat{t}$  with probability at least  $1 - \frac{1}{n^9}$ . Moreover, in this case,  $\text{VERIFY-GUESS}(D, \hat{t}, \epsilon)$  reports an  $(1 \pm \epsilon)$ -approximation to  $\text{CUT}(G)$ .

**Proof.**  $\text{VERIFY-GUESS}(D, \hat{t}, \epsilon)$  calls  $\text{SAMPLE}(D, \epsilon)$  for  $\Gamma = 100 \log n$  times with  $p$  being set to  $\min \left\{ \frac{2000 \log n}{\epsilon^2 \hat{t}}, 1 \right\}$ . Recall from Lemma 2.4 that each call to  $\text{SAMPLE}(D, p)$  makes in expectation at most  $2pm$  many  $\text{NEIGHBOR}$  queries, and returns a random subgraph  $H(V, E_p)$ , where each edge in  $E(G)$  is included in  $E_p$  with probability  $p$ . So,  $\text{VERIFY-GUESS}(D, \hat{t}, \epsilon)$  makes in expectation  $\mathcal{O}(pm \log n) = \tilde{\mathcal{O}}(m/\hat{t})$  many  $\text{NEIGHBOR}$  queries and generates  $\Gamma$  many random subgraphs of  $G$ . The subgraphs are denoted by  $H_1(V, E_p^1), \dots, H_\Gamma(V, E_p^\Gamma)$ .

- (i) Let  $\hat{t} \geq \frac{2000 \log n}{\epsilon^2} |\text{CUT}(G)|$ . From Lemma 2.3, we have that  $H_i$  will be connected with probability at most  $\frac{1}{10}$ . Observe that in expectation, we get that at least  $\frac{9\Gamma}{10}$  many  $H_i$ 's will be disconnected. By Chernoff bound (see Lemma B.1 in Section B), the probability that at most  $\frac{\Gamma}{2}$  many  $H_i$ 's are disconnected is at most  $\frac{1}{n^{10}}$ . Therefore,  $\text{VERIFY-GUESS}(D, \hat{t}, \epsilon)$  rejects any  $\hat{t}$  satisfying  $\hat{t} \geq \frac{2000 \log n}{\epsilon^2} |\text{CUT}(G)|$  with probability at least  $1 - \frac{1}{n^9}$ .
- (ii) Let  $\hat{t} \leq |\text{CUT}(G)|$ . Using Lemma 2.2, we have that every cut of size  $k$  in  $G$  has size  $pk(1 \pm \epsilon)$  in  $H_i$  with probability at least  $1 - \frac{1}{n^{10}}$ . Therefore, with probability at least  $1 - \frac{1}{n^{10}}$ , for all  $i \in [\Gamma]$ , every cut of size  $k$  in  $G$  has size  $pk(1 \pm \epsilon)$  in  $H_i$ . This implies that if  $\hat{t} \leq |\text{CUT}(G)|$  then  $\text{VERIFY-GUESS}(D, \hat{t}, \epsilon)$  accepts any  $\hat{t}$  with probability at least  $1 - \frac{1}{n^9}$ . Moreover, for any  $H_i$ , observe that  $\frac{|\text{CUT}(H_i)|}{p}$  is a  $(1 \pm \epsilon)$ -approximation to  $|\text{CUT}(G)|$ . Hence, when  $\hat{t} \leq |\text{CUT}(G)|$ ,  $\text{VERIFY-GUESS}(D, \hat{t}, \epsilon)$  also returns a  $(1 \pm \epsilon)$  approximation to  $|\text{CUT}(G)|$  with probability  $1 - \frac{1}{n^9}$ .  $\blacktriangleleft$

$\text{ESTIMATOR}(\epsilon)$  (Algorithm 3) will estimate the size of the minimum cut in  $G$  using  $\text{DEGREE}$  and  $\text{NEIGHBOR}$  queries. The main subroutine used by the algorithm will be  $\text{VERIFY-GUESS}(D, \hat{t}, \epsilon)$ .

The following lemma shows that with high probability  $\text{ESTIMATOR}(\epsilon)$  correctly estimates the size of the minimum cut in the graph  $G$ , and it also bounds the expected number of queries used by the algorithm.

**► Lemma 2.6.**  *$\text{ESTIMATOR}(\epsilon)$  returns a  $(1 \pm \epsilon)$  approximation to  $|\text{CUT}(G)|$  with probability at least  $1 - \frac{1}{n^8}$  by making in expectation  $\min \left\{ m + n, \frac{m}{\epsilon} \right\} \text{poly}(\log n, \frac{1}{\epsilon})$  queries and each query is either a  $\text{DEGREE}$  or a  $\text{NEIGHBOR}$  query to the unknown graph  $G$ .*

**Proof.** Without loss of generality, assume that  $n$  is a power of 2. If  $m < n - 1$  or if there exists a  $i \in [n]$  such that  $d_i = 0$  then the graph  $G$  is disconnected. In this case the algorithm  $\text{ESTIMATOR}(\epsilon)$  makes  $n$   $\text{DEGREE}$  queries and returns the correct answer. Thus we assume that  $m \geq n - 1$ .

First, we prove the correctness and query complexity when the graph is connected, that is,  $t \geq 1$ . Note that  $\text{ESTIMATOR}(\epsilon)$  calls  $\text{VERIFY-GUESS}(D, \hat{t}, \epsilon)$  for different values of  $\hat{t}$  starting from  $\frac{n}{2}$ . Recall that  $\kappa = \frac{2000 \log n}{\epsilon^2}$ . For a particular  $\hat{t}$  with  $\hat{t} \geq \kappa t$ ,  $\text{VERIFY-GUESS}(D, \hat{t}, \epsilon)$  does not  $\text{REJECT}$   $\hat{t}$  with probability at most  $\frac{1}{n^9}$  by Lemma 2.5 (i). So, by the union bound, the probability that  $\text{VERIFY-GUESS}(D, \hat{t}, \epsilon)$  will either  $\text{ACCEPT}$  or  $\text{FAIL}$  for some  $\hat{t}$  with  $\hat{t} \geq \kappa t$ , is at most  $\frac{\log n}{n^9}$ . Hence, with probability at least  $1 - \frac{\log n}{n^9}$ , we can say that  $\text{VERIFY-GUESS}(D, \hat{t}, \epsilon)$  rejects all  $\hat{t}$  with  $\hat{t} \geq \kappa t$ .

Observe that, from Lemma 2.5 (ii), the first time  $\hat{t}$  satisfies the following inequality  $\frac{t}{2} < \hat{t} \leq t$ ,  $\text{VERIFY-GUESS}(D, \hat{t}, \epsilon)$  will accept  $\hat{t}$  with probability at least  $1 - \frac{1}{n^9}$ . Therefore, for the first time  $\text{VERIFY-GUESS}(D, \hat{t}, \epsilon)$  will either  $\text{ACCEPT}$  or  $\text{FAIL}$ , then  $\hat{t}$  satisfies the following inequality  $\frac{t}{2} < \hat{t} < \kappa t$  with probability at least  $1 - \frac{\log n + 1}{n^9}$ . Let  $\hat{t}_0$  denote the first time  $\text{VERIFY-GUESS}$  returns  $\text{ACCEPT}$  or  $\text{FAIL}$ . From the description of  $\text{ESTIMATOR}(\epsilon)$ , note that, we get  $\hat{t}_u$  by dividing  $\hat{t}_0$  by  $\kappa$ . Note that, with probability at least  $1 - \frac{1 + \log n}{n^9}$ , we have

---

**Algorithm 3** ESTIMATOR( $\epsilon$ ).

---

**Input:** DEGREE and NEIGHBOR query access to an unknown graph  $G$ , and a parameter  $\epsilon \in (0, 1)$ .

**Output:** Either returns a  $(1 \pm \epsilon)$ -approximation to  $t = |\text{CUT}(G)|$  or FAIL

Find the degrees of all the vertices in  $G$  by making  $n$  many DEGREE queries;  
 // Let  $D = \{d(1), \dots, d(n)\}$ , where  $d(i)$  denotes the degree of the  $i$ -th vertex in  $G$ ;  
 If  $\exists i \in [n]$  such that  $d(i) = 0$ , then return  $t = 0$  and QUIT. Otherwise, proceed as follows.

Find  $m = \frac{1}{2} \sum_{i=1}^n d(i)$ . If  $m < n - 1$ , return  $t = 0$  and QUIT. Otherwise, proceed as follows.

Set  $\kappa = \frac{2000 \log n}{\epsilon^2}$

Initialize  $\hat{t} = \frac{n}{2}$ .

**while** ( $\hat{t} \geq 1$ ) **do**

- Call VERIFY-GUESS( $D, \hat{t}, \epsilon$ ).
- if** (VERIFY-GUESS( $D, \hat{t}, \epsilon$ ) returns REJECT) **then**
  - | set  $\hat{t} = \frac{\hat{t}}{2}$  and continue.
- end**
- else**
  - // Note that in this case VERIFY-GUESS( $D, \hat{t}, \epsilon$ ) either returns FAIL or ACCEPT.
  - Set  $\hat{t}_u = \max \left\{ \frac{\hat{t}}{\kappa}, 1 \right\}$ .
  - Call VERIFY-GUESS( $D, \hat{t}_u, \epsilon$ ).
  - if** (VERIFY-GUESS( $D, \hat{t}_u, \epsilon$ ) returns FAIL or REJECT) **then**
    - | return FAIL as the output of ESTIMATOR( $\epsilon$ )
  - end**
  - else**
    - | Let  $\tilde{t}$  be the output of VERIFY-GUESS( $D, \hat{t}_u, \epsilon$ ).
    - | Return  $\tilde{t}$  as the output of ESTIMATOR( $\epsilon$ ).
  - end**

**end**

**Output:** Return that the graph  $G$  is disconnected.

---

$\hat{t}_u < t$ . We then call the procedure VERIFY-GUESS( $D, \hat{t}, \epsilon$ ) with  $\hat{t} = \hat{t}_u$ . By Lemma 2.5 (ii), VERIFY-GUESS( $D, \hat{t}_u, \epsilon$ ) will ACCEPT and report a  $(1 \pm \epsilon)$  approximation to  $t$  with probability at least  $1 - \frac{1}{n^9}$ .

We will now analyze the number of DEGREE and NEIGHBOR queries made by the algorithm. We make an initial  $n$  many queries to construct the set  $D$ . Then at the worst case, we call VERIFY-GUESS( $D, \hat{t}, \epsilon$ ) for  $\hat{t} = \frac{n}{2}, \dots, t'$  and  $\hat{t} = \frac{t'}{\kappa} \geq \frac{t}{2\kappa}$ , where  $\frac{t}{2} < t' < \kappa t$ . It is because VERIFY-GUESS( $D, \hat{t}, \epsilon$ ) accepts  $\hat{t}$  with probability  $1 - \frac{1}{n^9}$  when the first time  $\hat{t}$  satisfy the inequality  $\hat{t} \leq t$ . Hence, by Lemma 2.5 and the facts that  $n \leq \frac{m}{t}$  and  $\hat{t}_u \geq \frac{t}{2\kappa}$  with probability at least  $1 - \frac{\log n + 1}{n^9}$ , in expectation the total number of queries made by the algorithm is at most  $n + \log n \cdot \left(1 - \frac{\log n + 1}{n^9}\right) \cdot \tilde{\mathcal{O}}\left(\frac{2\kappa m}{t}\right) + \log n \cdot \left(\frac{\log n + 1}{n^9}\right) \cdot \tilde{\mathcal{O}}(m) = \tilde{\mathcal{O}}\left(\frac{m}{t}\right)$ .

Note that each query made by ESTIMATOR( $\epsilon$ ) is either a DEGREE or a NEIGHBOR query.

Now we analyze the case when  $t = 0$ . Observe that VERIFY-GUESS( $D, \hat{t}, \epsilon$ ) rejects all  $\hat{t} \geq 1$  with probability  $1 - \frac{\log n}{n^9}$ , and therefore, ESTIMATOR( $\epsilon$ ) will report  $t = 0$ . As we have called

VERIFY-GUESS( $D, \hat{t}, \epsilon$ ) for all  $\hat{t} = \frac{n}{2}, \dots, 1$ , the number of queries made by ESTIMATOR( $\epsilon$ ), in the case when  $t = 0$ , is  $\tilde{\mathcal{O}}(m) + n$ . Note that the additional term of  $n$  in the bound comes from the fact that to compute  $D$ , the algorithm needs to make  $n$  many DEGREE queries. ◀

### 3 Lower bounds

In this Section, we prove Theorems 1.2 and 1.3 using reductions from suitable problems in communication complexity. In Section 3.1, we discuss about two party communication complexity along with the problems that will be used in our reductions. We will discuss the proofs of Theorems 1.2 and 1.3 in Section 3.2.

#### 3.1 Communication Complexity

In two-party communication complexity there are two parties, Alice and Bob, that wish to compute a function  $\Pi : \{0, 1\}^N \times \{0, 1\}^N \rightarrow \{0, 1\} \cup \{0, 1\}^n$  \*\*. Alice is given  $\mathbf{x} \in \{0, 1\}^N$  and Bob is given  $\mathbf{y} \in \{0, 1\}^N$ . Let  $x_i$  ( $y_i$ ) denotes the  $i$ -th bit of  $\mathbf{x}$  ( $\mathbf{y}$ ). While the parties know the function  $\Pi$ , Alice does not know  $\mathbf{y}$ , and similarly Bob does not know  $\mathbf{x}$ . Thus they communicate bits following a pre-decided protocol  $\mathcal{P}$  in order to compute  $\Pi(\mathbf{x}, \mathbf{y})$ . We say a randomized protocol  $\mathcal{P}$  computes  $\Pi$  if for all  $(\mathbf{x}, \mathbf{y}) \in \{0, 1\}^N \times \{0, 1\}^N$  we have  $\mathbb{P}[\mathcal{P}(\mathbf{x}, \mathbf{y}) = \Pi(\mathbf{x}, \mathbf{y})] \geq 2/3$ . The model provides the parties access to common random string of arbitrary length. The cost of the protocol  $\mathcal{P}$  is the maximum number of bits communicated, where maximum is over all inputs  $(\mathbf{x}, \mathbf{y}) \in \{0, 1\}^N \times \{0, 1\}^N$ . The communication complexity of the function is the cost of the most efficient protocol computing  $\Pi$ . For more details on communication complexity see [26]. We now define two functions  $k$ -INTERSECTION and FIND- $k$ -INTERSECTION and discuss their communication complexity. Both these functions will be used in our reductions.

► **Definition 3.1** (FIND- $k$ -INTERSECTION). Let  $k, N \in \mathbb{N}$  such that  $k \leq N$ . Let  $S = \{(\mathbf{x}, \mathbf{y}) \in \{0, 1\}^N \times \{0, 1\}^N : \sum_{i=1}^N x_i y_i = k\}$ . The FIND- $k$ -INTERSECTION function on  $N$  bits is a partial function and is defined as  $\text{FIND-INT}_k^N : S \rightarrow \{0, 1\}^N$ , and is defined as  $\text{FIND-INT}_k^N(\mathbf{x}, \mathbf{y}) = \mathbf{z}$ , where  $z_i = x_i y_i$  for each  $i \in [N]$ .

Note that the objective is that at the end of the protocol Alice and Bob know  $\mathbf{z}$ .

► **Definition 3.2** ( $k$ -INTERSECTION). Let  $k, N \in \mathbb{N}$  such that  $k \leq N$ . Let  $S = \{(\mathbf{x}, \mathbf{y}) : \sum_{i=1}^N x_i y_i = k \text{ or } k - 1\}$ . The  $k$ -INTERSECTION function on  $N$  bits is a partial function denoted by  $\text{INT}_k^N : S \rightarrow \{0, 1\}$ , and is defined as follows:  $\text{INT}_k^N(\mathbf{x}, \mathbf{y}) = 1$  if  $\sum_{i=1}^N x_i y_i = k$  and 0, otherwise.

In communication complexity, the  $k$ -INTERSECTION function on  $N$  bits when  $k = 1$  is known as DISJOINTNESS function on  $N$ . The following lemmas follow easily from the communication complexity of DISJOINTNESS (see [26]).

► **Lemma 3.3.** *Let  $k, N \in \mathbb{N}$  such that  $k \leq cN$  for some constant  $c < 1$ . The randomized communication complexity of FIND- $k$ -INTERSECTION function on  $N$  bits is  $\Omega(N)$ .*

► **Lemma 3.4.** *Let  $k, N \in \mathbb{N}$  such that  $k \leq cN$  for some constant  $c < 1$ . The randomized communication complexity of  $k$ -INTERSECTION function on  $N$  bits ( $\text{INT}_k^N$ ) is  $\Omega(N)$ .*

---

\*\* The co-domain of  $\Pi$  looks odd, as the co-domain is  $\{0, 1\}$  usually. However, we need  $\{0, 1\} \cup \{0, 1\}^n$  to take care of all the problems in communication complexity we discuss in this paper.



### 3.2 Proofs of Theorems 1.2 and 1.3

The proofs of Theorems 1.2 and 1.3 are inspired from the lower bound proof of Eden and Rosenbaum [9] for estimating MINCUT<sup>††</sup>.

**Proof of Theorem 1.2.** We prove by giving a reduction from FIND- $t/2$ -INTERSECTION on  $N$  bits. Without loss of generality assume that  $t$  is even. Let  $\mathbf{x}$  and  $\mathbf{y}$  be the inputs of Alice and Bob. Note that  $\sum_{i=1}^N x_i y_i = t/2$ .

We first discuss a graph  $G_{(\mathbf{x}, \mathbf{y})}(V, E)$  that can be generated from  $(\mathbf{x}, \mathbf{y})$ , such that  $|V| = n$  and  $|E| = m \geq 2nt$ , and works as the “hard” instance for our proof. Note that  $G_{(\mathbf{x}, \mathbf{y})}$  should be such that no useful information about the MINCUT can be derived by knowing only one of  $\mathbf{x}$  and  $\mathbf{y}$ . Let  $s = t + \sqrt{t^2 + (m - nt)/2}$  and  $N = s^2$ . In particular,  $2t \leq s \leq 2t + 3\sqrt{m}$ . Also,  $s \geq \sqrt{m/2}$  and therefore  $s = \Theta(\sqrt{m})$ .

#### The graph $G_{(\mathbf{x}, \mathbf{y})}$ and its properties:

$G_{(\mathbf{x}, \mathbf{y})}$  has the following structure.

- $V = S_A \cup T_A \cup S_B \cup T_B \cup C$  such that  $|S_A| = |T_A| = |S_B| = |T_B| = s$  and  $|C| = n - 4s$ . Let  $S_A = \{s_i^A : i \in [s]\}$  and similarly  $T_A = \{t_i^A : i \in [s]\}$ ,  $S_B = \{s_i^B : i \in [s]\}$  and  $T_B = \{t_i^B : i \in [s]\}$ .
- Each vertex in  $C$  is connected to  $2t$  different vertices in  $S_A$ .
- For  $i, j \in [s]$ : if  $x_{ij} = y_{ij} = 1$ , then  $(s_i^A, t_j^B) \in E$  and  $(s_i^B, t_j^A) \in E$ ; otherwise,  $(s_i^A, t_j^A) \in E$  and  $(s_i^B, t_j^B) \in E$ .

► **Observation 3.5.**  $G_{(\mathbf{x}, \mathbf{y})}$  satisfies the following properties.

**Property-1:** The degree of every vertex in  $C$  is  $2t$ . For any  $v \notin C$ , the neighbors of  $v$  inside  $C$  are fixed irrespective of  $\mathbf{x}$  and  $\mathbf{y}$ ; and the number of neighbors outside  $C$  is  $s \geq 2t$ .

**Property-2:** There are  $t$  edges between the vertex sets  $(C \cup S_A \cup T_A)$  and  $(S_B \cup T_B)$ , and removing them  $G_{(\mathbf{x}, \mathbf{y})}$  becomes disconnected.

**Property-3:** Every pair of vertices  $(S_A \cup T_A \cup C)$  is connected by at least  $3t/2$  edge disjoint paths. Also, every pair of vertices in  $(S_B \cup T_B)$  is connected by at least  $3t/2$  edge disjoint paths.

**Property-4:** The set of  $t$  edges between the vertex sets  $(C \cup S_A \cup T_A)$  and  $(S_B \cup T_B)$  forms the unique global minimum cut of  $G(\mathbf{x}, \mathbf{y})$ ,

**Property-5:**  $x_{ij} = y_{ij} = 1$  if and only if  $(s_i^A, t_j^B)$  and  $(s_i^B, t_j^A)$  are the edges in the unique global minimum cut of  $G_{(\mathbf{x}, \mathbf{y})}$ .

**Proof.** Property-1 and Property-2 directly follow from the construction. Now, we will prove Property-3. We first show that every pair of vertices  $(S_A \cup T_A \cup C)$  is connected by at least  $3t/2$  edge disjoint paths by breaking the analysis into the following cases.

- (i) Consider  $s_i^A, s_j^A \in S_A$ , for  $i, j \in [s]$ . Under the promise that  $\sum_{i=1}^N x_i y_i = t/2$ ,  $s_i^A, s_j^A$  have at least  $s - t \geq 3t/2$  common neighbors in  $T_A$  and thus there are at least  $3t/2$  edge disjoint paths connecting them.
- (ii) Consider  $s_i^A \in S_A$  and  $t_j^A \in T_A$ , for  $i, j \in [s]$ . Let  $s_{j_1}^A, \dots, s_{j_{3t/2}}^A$  be  $3t/2$  distinct neighbors of  $t_j^A$  in  $S_A$ . Since,  $s_i^A$  has  $3t/2$  common neighbors with each  $s_{j_r}^A$ ,  $r \in [3t/2]$ , there is a matching of size  $3t/2$ . Denote this matching by  $(t_{j_r}^A, s_{j_r}^A)$ ,  $r \in [3t/2]$ . Thus  $(s_i^A, t_{j_r}^A), (t_{j_r}^A, s_{j_r}^A), (s_{j_r}^A, t_j^A)$ , for  $r \in [3t/2]$ , forms a set of edge disjoint paths of size  $3t/2$

<sup>††</sup> Note that Eden and Rosenbaum [9] stated the result in terms  $k$ -Edge Connectivity.

## 6:12 Query Complexity of Global Minimum Cut

from  $s_i^A$  to  $t_j^A$ , each of length 3. In case  $s_i^A$  is one of the neighbors of  $t_j^A$ , then one of the  $3t/2$  paths gets reduced to  $(s_i^A, t_j^A)$ , a length 1 path that is edge disjoint from the remaining paths.

- (iii) Consider  $u, v \in C$ . Let  $u_1, \dots, u_{2t} \in S_A$  and  $v_1, \dots, v_{2t} \in S_A$  be the neighbors of  $u$  and  $v$  respectively in  $S_A$ . If for some  $i, j \in [2t]$ ,  $u_i = v_j$  then  $(u, u_i), (u_i, v_j), (v_j, v)$  is a desired path. Thus, assume  $u_i \neq v_j$  for all  $i, j \in [2t]$ . For all  $i \in [2t]$ , since  $u_i$  and  $v_i$  have at least  $3t/2$  common neighbors in  $T_A$  we can find  $3t/2$  edge disjoint paths  $(u_i, t_i^A), (t_i^A, v_i)$ , where  $t_i^A \in T_A$ . Existence of  $3t/2$  edge disjoint paths from  $u \in C$  to  $v \in S_A$  can be proved as in (i). and from  $u \in C$  to  $v \in T_A$  can be proved as in (ii).

Similarly, we can show that every pair of vertices in  $(S_B \cup T_B)$  is connected by  $3t/2$  many edge disjoint paths.

Observe that Property-4 follows from Property-3, and Property-5 follows from the construction of  $G_{(\mathbf{x}, \mathbf{y})}$  and Property-4. ◀

Now, by contradiction assume that there exists an algorithm  $\mathcal{A}$  that makes  $o(m)$  queries to  $G_{(\mathbf{x}, \mathbf{y})}$  and finds all the edges of a global minimum cut with probability  $2/3$ . Now, we give a protocol  $\mathcal{P}$  for FIND- $t/2$ -INTERSECTION on  $N$  bits when the  $\mathbf{x}$  and  $\mathbf{y}$  are the inputs of Alice and Bob, respectively. Note that  $x, y \in \{0, 1\}^N$  such that  $\sum_{i=1}^N x_i y_i = t/2$ .

### Protocol $\mathcal{P}$ for FIND- $t/2$ -INTERSECTION

Alice and Bob run the query algorithm  $\mathcal{A}$  when the unknown graph is  $G_{(\mathbf{x}, \mathbf{y})}$ . Now we explain how they simulate the local queries and random edge query on  $G_{(\mathbf{x}, \mathbf{y})}$  by communication. We would like to note that each query can be answered deterministically.

**DEGREE query:** By Property-1, the degree of every vertex does not depend on the inputs of Alice and Bob, and therefore any degree query can be simulated without any communication.

**NEIGHBOR query:** For  $v \in C$ , the set of  $2t$  neighbors are fixed by the construction. So, any neighbor query involving any  $v \in C$  can be answered without any communication. For  $i \in [s]$  and  $s_i^A \in S_A$ , let  $N_C(s_i^A)$  be the set of fixed neighbors of  $s_i^A$  inside  $C$ . So, by Property-1,  $d(s_i^A) = |N_C(s_i^A)| + s$ <sup>††</sup>. The labels of the neighbors of  $s_i^A$  are such that the first  $|N_C(s_i^A)|$  many neighbors are inside  $C$ , and they are arranged in a fixed but arbitrary order. For  $j \in [s]$ , the  $(|N_C(v)| + j)$ -th neighbor of  $s_i^A$  is either  $t_j^B$  or  $s_j^A$  depending on whether  $x_{ij} = y_{ij} = 1$  or not, respectively. So, any neighbor query involving vertex in  $S_A$  can be answered by 2 bits of communication. Similar arguments also hold for the vertices in  $S_B \cup T_A \cup T_B$ .

**ADJACENCY query:** Observe that each adjacency query can be answered by at most 2 bits of communication, and it can be argued like the NEIGHBOR query.

**RANDOM EDGE query:** By Property-1, the degree of any vertex  $v \in V$  is independent of the inputs of Alice and Bob. Alice and Bob use shared randomness to sample a vertex in  $V$  proportional to its degree. Let  $r \in V$  be the sampled vertex. They again use shared randomness to sample an integer  $j$  in  $[d(v)]$  uniformly at random. Then they determine the  $j$ -th neighbor of  $r$  using NEIGHBOR query. Observe that this procedure simulates a RANDOM EDGE query by using at most 2 bits of communication.

Using the fact that  $G_{(\mathbf{x}, \mathbf{y})}$  satisfies Property-4 and 5, the output of algorithm  $\mathcal{A}$  determines the output of protocol  $\mathcal{P}$  for FIND- $t/2$ -INTERSECTION. As each query of  $\mathcal{A}$  can be simulated

---

<sup>††</sup>  $d(u)$  denotes the degree of the vertex  $u$  in  $G_{(\mathbf{x}, \mathbf{y})}$

by at most two bits of communication by the protocol  $\mathcal{P}$ , the number of bits communicated is  $o(m)$ . Recall that  $N = s^2$  and  $s = \Theta(\sqrt{m})$ . So, the number of bits communicated by Alice and Bob in  $\mathcal{P}$  is  $o(N)$ . This contradicts Theorem 3.3. ◀

**Proof of Theorem 1.3.** The proof of this theorem uses the same construction as the one used in the proof of Theorem 1.2. The “hard” communication problem to reduce from is  $t/2$ -INTERSECTION (see Definition 3.2) on  $N$  bits, where  $N = s^2$  and  $s = \Theta(\sqrt{m})$ . ◀

## 4 Conclusion

Our work first and foremost closes a gap in the query complexity of a fundamental problem of finding a minimum cut using local queries. The strength of our algorithm lies in its simplicity – it uses existing ingredients in a fashion suitable for the query framework. The crucial idea was to ensure that cuts are preserved in a sparsified graph in a query framework. We discuss the application of our approach to other cut problems in Appendix A.

---

## References

- 1 K. J. Ahn, S. Guha, and A. McGregor. Graph Sketches: Sparsification, Spanners, and Subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*, pages 5–14, 2012.
- 2 M. Aliakbarpour, A. S. Biswas, T. Gouleakis, J. Peebles, R. Rubinfeld, and A. Yodpinyanee. Sublinear-Time Algorithms for Counting Star Subgraphs via Edge Sampling. *Algorithmica*, 80(2):668–697, 2018.
- 3 S. Assadi, M. Kapralov, and S. Khanna. A Simple Sublinear-Time Algorithm for Counting Arbitrary Subgraphs via Edge Sampling. In *Proceedings of the 9th Innovations in Theoretical Computer Science Conference, ITCS*, pages 6:1–6:20, 2019.
- 4 L. Babai, P. Frankl, and J. Simon. Complexity classes in communication complexity theory (preliminary version). In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science, FOCS*, pages 337–347, 1986.
- 5 E. Blais, J. Brody, and K. Matulef. Property Testing Lower Bounds via Communication Complexity. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC*, pages 210–220, 2011.
- 6 D. P. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 1st edition, 2009.
- 7 T. Eden, A. Levi, D. Ron, and C. Seshadhri. Approximately Counting Triangles in Sublinear Time. *SIAM J. Comput.*, 46(5):1603–1646, 2017.
- 8 T. Eden, D. Ron, and C. Seshadhri. On Approximating the Number of  $k$ -Cliques in Sublinear Time. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 722–734, 2018.
- 9 T. Eden and W. Rosenbaum. Lower Bounds for Approximating Graph Parameters via Communication Complexity. In *Proceedings of the 21st International Conference on Approximation Algorithms for Combinatorial Optimization Problems, APPROX*, pages 11:1–11:18, 2018.
- 10 T. Eden and W. Rosenbaum. On Sampling Edges Almost Uniformly. In *Proceedings of the 1st Symposium on Simplicity in Algorithms, SOSA*, pages 7:1–7:9, 2018.
- 11 U. Feige. On Sums of Independent Random Variables with Unbounded Variance and Estimating the Average Degree in a Graph. *SIAM J. Comput.*, 35(4):964–984, 2006.
- 12 M. Ghaffari and B. Haeupler. Distributed algorithms for planar networks II: low-congestion shortcuts, mst, and min-cut. In *Proceedings of the 2016 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 202–219. SIAM, 2016.
- 13 M. Ghaffari and F. Kuhn. Distributed minimum cut approximation. In *Distributed Computing*, volume 8205 of *Lecture Notes in Computer Science*, pages 1–15, 2013.

- 14 M. Ghaffari and K. Nowicki. Massively parallel algorithms for minimum cut. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, pages 119–128. ACM, 2020.
- 15 O. Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.
- 16 O. Goldreich, S. Goldwasser, and D. Ron. Property Testing and its Connection to Learning and Approximation. *J. ACM*, 45(4):653–750, 1998.
- 17 O. Goldreich and D. Ron. Approximating Average Parameters of Graphs. *Random Structures & Algorithms*, 32(4):473–493, 2008.
- 18 M. Gonen, D. Ron, and Y. Shavitt. Counting Stars and Other Small Subgraphs in Sublinear-Time. *SIAM Journal on Discrete Mathematics*, 25(3):1365–1411, 2011.
- 19 A. Graur, T. Pollner, V. Ramaswamy, and S. M. Weinberg. New Query Lower Bounds for Submodular Function Minimization. In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference, ITCS*, volume 151, pages 64:1–64:16, 2020.
- 20 A. Hajnal, W. Maass, and G. Turán. On the communication complexity of graph properties. In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, STOC*, pages 186–191, 1988.
- 21 M. Kapralov, Y. T. Lee, C. Musco, C. Musco, and A. Sidford. Single pass spectral sparsification in dynamic streams. In *Proceedings of the 55th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 561–570, 2014.
- 22 D. R. Karger. Global Min-cuts in RNC, and Other Ramifications of a Simple Min-Cut Algorithm. In *Proceedings of the 4th Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, SODA*, pages 21–30, 1993.
- 23 D. R. Karger and C. Stein. An  $\tilde{O}(n^2)$  Algorithm for Minimum Cuts. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing, STOC*, pages 757–765, 1993.
- 24 T. Kaufman, M. Krivelevich, and D. Ron. Tight Bounds for Testing Bipartiteness in General Graphs. *SIAM J. Comput.*, 33(6):1441–1483, 2004.
- 25 K. Kawarabayashi and M. Thorup. Deterministic edge connectivity in near-linear time. *J. ACM*, 66(1):4:1–4:50, 2019.
- 26 E. Kushilevitz. Communication complexity. In *Advances in Computers*, volume 44, pages 331–360. Elsevier, 1997.
- 27 A. McGregor. Graph Stream Algorithms: A Survey. *SIGMOD Rec.*, 43(1):9–20, 2014.
- 28 S. Mukhopadhyay and D. Nanongkai. Weighted Min-Cut: Sequential, Cut-Query, and Streaming Algorithms. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 496–509, 2020.
- 29 A. Rubinfeld, T. Schramm, and S. M. Weinberg. Computing Exact Minimum Cuts Without Knowing the Graph. In *Proceedings of the 9th Innovations in Theoretical Computer Science Conference, ITCS*, pages 39:1–39:16, 2018.

## **A** Application of our approach to other cut problems

**Sublinear time algorithm for Global minimum cut.** For simplicity, the algorithm (Algorithm 3) presented for estimating global minimum cut is  $\tilde{O}(m)$ . But, our algorithm can be adapted to get a sublinear time algorithm (with time complexity  $\tilde{O}\left(\frac{m}{\hat{t}}\right)$ ) for estimating the size of the global minimum cut in the graph. We will sample  $\tilde{O}\left(\frac{m}{\hat{t}}\right)$  random edges with replacement from the graph  $G$  using  $\tilde{O}\left(\frac{m}{\hat{t}}\right)$  using local queries, where  $\hat{t}$  is the guess for the size of the global minimum, rather than sampling each edge with probability  $p = \tilde{O}\left(\frac{1}{\hat{t}}\right)$  as we have done in the Algorithm 2. Observe that, after finding the degrees of all the vertices, a random edge can be generated using  $\mathcal{O}(1)$  local queries. The rest of the algorithm and its analysis can be adapted directly. Therefore, we have the following result.

► **Theorem A.1** (Estimating Global minimum cut in sublinear time). *There exists an algorithm, with DEGREE and NEIGHBOR query access to an unknown graph  $G = (V, E)$ , that solves the minimum cut estimation problem with high probability. With high probability, the time complexity and the query complexity of the algorithm is  $\min \left\{ m + n, \frac{m}{t} \right\} \text{poly} \left( \log n, \frac{1}{\epsilon} \right)$ .*

**Global minimum  $r$ -way cut.** Global minimum  $r$ -cut, for a graph  $G = ([n], E)$ ,  $|V| = n$  and  $|E| = m$ , is a partition of the vertex set  $[n]$  into  $r$ -sets  $S_1, \dots, S_r$  such that the following is minimized:  $|\{i, j\} \in E : \exists k, \ell (k \neq \ell) \in [r], \text{ with } i \in S_k \text{ and } j \in S_\ell\}|$ .

Let  $\text{CUT}_r(G)$  denote the set of edges corresponding to a minimum  $r$ -cut, i.e., the edges that goes across different partitions, and by the size of minimum  $r$ -cut, we mean  $|\text{CUT}_r(G)|$ . The sampling and verification idea used in the proof of Theorem 1.1 can be extended directly, together with [22, Corollary 8.2], to get the following result.

► **Theorem A.2.** *There exists an algorithm, with DEGREE and NEIGHBOR query access to an unknown graph  $G = ([n], E)$ , that with high probability outputs a  $(1 \pm \epsilon)$ -approximation of the size of the minimum  $r$ -cut of  $G$ . The expected number of queries used by the algorithm is  $\min \left\{ m + n, \frac{m}{t_r} \right\} \text{poly} \left( r, \log n, \frac{1}{\epsilon} \right)$ , where  $t_r = |\text{CUT}_r(G)|$ .*

**Minimum cuts in simple multigraphs.** A graph with multiple edges between a pair of vertices in the graph but without any self loops are called *simple multigraphs*. If we have DEGREE and NEIGHBOR query access \* to simple multigraphs then we can directly get the following generalization of Theorem 1.1.

► **Theorem A.3** (Minimum cut estimation in simple multigraphs using local queries). *There exists an algorithm, with DEGREE and NEIGHBOR query access to an unknown simple multigraph  $G = (V, E)$ , that solves the minimum cut estimation problem with high probability. The expected number of queries used by the algorithm is  $\min \left\{ m + n, \frac{m}{t} \right\} \text{poly} \left( \log n, \frac{1}{\epsilon} \right)$ , where  $n$  is the number of vertices in the multigraph,  $m$  is the number of edges in the multigraph and  $t$  is the number of edges in a minimum cut.*

## B Probability Results

► **Lemma B.1** (See [6]). *Let  $X = \sum_{i \in [n]} X_i$  where  $X_i, i \in [n]$ , are independent random variables,  $X_i \in [0, 1]$  and  $\mathbb{E}[X]$  is the expected value of  $X$ . Then*

(i) For  $\epsilon > 0$

$$\Pr[|X - \mathbb{E}[X]| > \epsilon \mathbb{E}[X]] \leq \exp \left( -\frac{\epsilon^2}{3} \mathbb{E}[X] \right).$$

(ii) Suppose  $\mu_L \leq \mathbb{E}[X] \leq \mu_H$ , then for  $0 < \epsilon < 1$

(a)  $\Pr[X > (1 + \epsilon)\mu_H] \leq \exp \left( -\frac{\epsilon^2}{3} \mu_H \right)$ .

(b)  $\Pr[X < (1 - \epsilon)\mu_L] \leq \exp \left( -\frac{\epsilon^2}{2} \mu_L \right)$ .

---

\* For simple multigraphs, we will assume that the neighbors of a vertex are stored with multiplicities.



# A Constant-Factor Approximation for Weighted Bond Cover

Eun Jung Kim ✉

Université Paris-Dauphine, PSL University, CNRS, LAMSADE, 75016, Paris, France

Euiwoong Lee<sup>1</sup> ✉

University of Michigan, Ann Arbor, MI, USA

Dimitrios M. Thilikos ✉

LIRMM, Univ. Montpellier, CNRS, Montpellier, France

---

## Abstract

---

The WEIGHTED  $\mathcal{F}$ -VERTEX DELETION for a class  $\mathcal{F}$  of graphs asks, weighted graph  $G$ , for a minimum weight vertex set  $S$  such that  $G - S \in \mathcal{F}$ . The case when  $\mathcal{F}$  is minor-closed and excludes some graph as a minor has received particular attention but a constant-factor approximation remained elusive for WEIGHTED  $\mathcal{F}$ -VERTEX DELETION. Only three cases of minor-closed  $\mathcal{F}$  are known to admit constant-factor approximations, namely VERTEX COVER, FEEDBACK VERTEX SET and DIAMOND HITTING SET. We study the problem for the class  $\mathcal{F}$  of  $\theta_c$ -minor-free graphs, under the equivalent setting of the WEIGHTED  $c$ -BOND COVER problem, and present a constant-factor approximation algorithm using the primal-dual method. For this, we leverage a structure theorem implicit in [Joret et al., SIDMA'14] which states the following: any graph  $G$  containing a  $\theta_c$ -minor-model either contains a large two-terminal *protrusion*, or contains a constant-size  $\theta_c$ -minor-model, or a collection of pairwise disjoint *constant-sized* connected sets that can be contracted simultaneously to yield a dense graph. In the first case, we tame the graph by replacing the protrusion with a special-purpose weighted gadget. For the second and third case, we provide a weighting scheme which guarantees a local approximation ratio. Besides making an important step in the quest of (dis)proving a constant-factor approximation for WEIGHTED  $\mathcal{F}$ -VERTEX DELETION, our result may be useful as a template for algorithms for other minor-closed families.

**2012 ACM Subject Classification** Mathematics of computing → Approximation algorithms; Mathematics of computing → Combinatorial algorithms; Theory of computation → Approximation algorithms analysis; Theory of computation → Graph algorithms analysis

**Keywords and phrases** Constant-factor approximation algorithms, Primal-dual method, Bonds in graphs, Graph minors, Graph modification problems

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.7

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/abs/2105.00857>

**Funding** *Eun Jung Kim*: Supported by the ANR projects ASSK (ANR-18-CE40-0025-01) and ESIGMA (ANR-17-CE23-0010) from French National Research Agency.

*Dimitrios M. Thilikos*: Supported by the ANR projects DEMOGRAPH (ANR-16-CE40-0028), ESIGMA (ANR-17-CE23-0010), and the French-German Collaboration ANR/DFG Project UTMA (ANR-20-CE92-0027).

---

<sup>1</sup> Part of this work was done when the second author was a postdoc at New York University and supported by Simons Collaboration on Algorithms and Geometry.

## 1 Introduction and main ideas of our results

For a class  $\mathcal{F}$  of graphs, the problem WEIGHTED  $\mathcal{F}$ -VERTEX DELETION asks, given weighted graph  $G = (V, E, w)$ , for a vertex set  $S \subseteq V$  of minimum weight such that  $G - S$  belongs to the class  $\mathcal{F}$ . The WEIGHTED  $\mathcal{F}$ -VERTEX DELETION captures classic graph problems such as WEIGHTED VERTEX COVER and WEIGHTED FEEDBACK VERTEX SET, which corresponds to  $\mathcal{F}$  being the classes of edgeless and acyclic graphs, respectively. A vast literature is devoted to the study of (WEIGHTED)  $\mathcal{F}$ -VERTEX DELETION for various instantiations of  $\mathcal{F}$ , both in approximation algorithms and in parameterized complexity. Much of the work considers a class  $\mathcal{F}$  that is characterized by a set of forbidden (induced) subgraphs [41, 33, 14, 1, 28, 3, 4, 5, 6, 37] or that is minor-closed [22, 16, 18, 20, 31, 19, 30, 8, 2, 23, 21, 44, 11], thus characterized by a (finite) set of forbidden minors.

Lewis and Yannakakis [35] showed that  $\mathcal{F}$ -VERTEX DELETION, the unweighted version of WEIGHTED  $\mathcal{F}$ -VERTEX DELETION, is NP-hard whenever  $\mathcal{F}$  is nontrivial (there are infinitely many graphs in and outside of  $\mathcal{F}$ ) and hereditary (is closed under taking induced subgraphs). It was also long known that  $\mathcal{F}$ -VERTEX DELETION is APX-hard for every non-trivial hereditary class  $\mathcal{F}$  [39]. So, the natural question is for which class  $\mathcal{F}$ , (WEIGHTED)  $\mathcal{F}$ -VERTEX DELETION admits constant-factor approximation algorithms.

When  $\mathcal{F}$  is characterized by a finite set of forbidden induced subgraphs, a constant-factor approximation for WEIGHTED  $\mathcal{F}$ -VERTEX DELETION is readily derived with LP-rounding technique. Lund and Yannakakis [39] conjectured that for  $\mathcal{F}$  characterized by a set of minimal forbidden induced subgraphs, the finiteness of  $\mathcal{F}$  defines the borderline between approximability and inapproximability with constant ratio of  $\mathcal{F}$ -VERTEX DELETION. This conjecture was refuted due to the existence of 2-approximation for WEIGHTED FEEDBACK VERTEX SET [7, 12, 16]. Since then, a few more classes with an infinite set of forbidden induced subgraphs are known to allow constant-factor approximations for  $\mathcal{F}$ -VERTEX DELETION, such as block graphs [1], 3-leaf power graphs [5], interval graphs [14], ptolemaic graphs [6], and bounded treewidth graphs [20, 23]. That is, we are only in the nascent stage when it comes to charting the landscape of (WEIGHTED)  $\mathcal{F}$ -VERTEX DELETION as to constant-factor approximability. In the remainder of this section, we focus on the case where  $\mathcal{F}$  is a minor-closed class.

**Known results on (WEIGHTED)  $\mathcal{F}$ -VERTEX DELETION.** According to Robertson and Seymour theorem, every non-trivial minor-closed graph class  $\mathcal{F}$  is characterized by a finite set, called (*minor*) *obstruction set*, of minimal forbidden minors, called (*minor*) *obstructions* [43]. It is also well-known that  $\mathcal{F}$  has bounded treewidth if and only if one of the obstructions is planar [42]. Therefore, the  $\mathcal{F}$ -VERTEX DELETION for  $\mathcal{F}$  excluding at least one planar graph as a minor can be deemed a natural extension of FEEDBACK VERTEX SET. In this context, it is not surprising that  $\mathcal{F}$ -VERTEX DELETION, for minor-closed  $\mathcal{F}$ , attracted particular attention in parameterized complexity, where Feedback Vertex Set was considered the flagship problem serving as an igniter and a testbed for new techniques.

For every minor-closed  $\mathcal{F}$ , the class of yes-instances to the decision version of  $\mathcal{F}$ -VERTEX DELETION is minor-closed again (for every fixed size of a solution), thus there exists a finite obstruction set for the set of its yes-instances. With a minor-membership test algorithm [26], this implies that  $\mathcal{F}$ -VERTEX DELETION is fixed-parameter tractable. The caveat is, such a fixed-parameter algorithm is non-uniform and non-constructive, and the exponential term in the running time is gigantic. Much endeavour was made to reduce the parametric dependence of such algorithms for  $\mathcal{F}$ -VERTEX DELETION. The case when  $\mathcal{F}$  has bounded treewidth is



now understood well. The corresponding  $\mathcal{F}$ -VERTEX DELETION is known to be solvable in time  $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$  [20, 31] and the single-exponential dependency on  $k$  is asymptotically optimal under the *Exponential Time Hypothesis*<sup>2</sup> [31]. (See also [44] for recent parameterized algorithms for general minor-closed  $\mathcal{F}$ 's).

Turning to approximability, the (unweighted)  $\mathcal{F}$ -VERTEX DELETION can be approximated within a constant-factor when  $\mathcal{F}$  has bounded treewidth, say  $t$ , or equivalently when the obstruction set of  $\mathcal{F}$  contains some planar graph. The first general result in this direction was the randomized  $f(t)$ -approximation of Fomin et al. [20]. Gupta et al. [23] made a further progress with an  $\mathcal{O}(\log t)$ -approximation algorithm. Unfortunately, such approximation algorithms whose approximation ratio depends only on  $\mathcal{F}$  are not known when the input is *weighted*. A principal reason for this is that most of the techniques developed for the unweighted case do not extend to the weighted setting. In this direction, Agrawal et al. [2] presented a randomized  $\mathcal{O}(\log^{1.5} n)$ -approximation algorithm and a deterministic  $\mathcal{O}(\log^2 n)$ -approximation algorithm which run in time  $n^{\mathcal{O}(t)}$  when  $\mathcal{F}$  has treewidth at most  $t$ . It is reported in [2] that an  $\mathcal{O}(\log n \cdot \log \log n)$ -approximation can be deduced from the approximation algorithm of Bansal et al. [8] for the edge deletion variant of WEIGHTED  $\mathcal{F}$ -VERTEX DELETION. For the class  $\mathcal{F}$  of planar graphs, Kawarabayashi and Sidiropoulos [30] presented an algorithm for  $\mathcal{F}$ -VERTEX DELETION with polylogarithmic approximation ratio running in quasi-polynomial time. Beyond this work, no nontrivial approximation algorithm is known for  $\mathcal{F}$  of unbounded treewidth.

Regarding *constant-factor* approximability for WEIGHTED  $\mathcal{F}$ -VERTEX DELETION with minor-closed  $\mathcal{F}$ , only three results are known till now. For the WEIGHTED VERTEX COVER, it was observed early that a 2-approximation can be instantly derived from the half-integrality of LP [40]. The local-ratio algorithm by Bar-Yehuda and Even [10] was presumably the first primal-dual algorithm and laid the groundwork for subsequent development of the primal-dual method.<sup>3</sup> For the WEIGHTED FEEDBACK VERTEX SET, 2-approximation algorithms were proposed using the primal-dual method [7, 12, 16]. Furthermore, a constant-factor approximation algorithm was given for WEIGHTED DIAMOND HITTING SET by Fiorini, Joret, and Pietropaoli [18] in 2010. To the best of our knowledge, no progress is made on approximation with constant ratio for minor-closed  $\mathcal{F}$  since then.

For minor-closed  $\mathcal{F}$  with graphs of bounded treewidth, the known approximation algorithms for (WEIGHTED)  $\mathcal{F}$ -VERTEX DELETION take one of the following two avenues. First, the algorithms in [8, 2, 23] draw on the fact that a graph of constant treewidth has a constant-size separator which breaks down the graph into *smaller* pieces. The measure for smallness is an important design feature of these algorithms. Regardless of the design specification, however, it seems there is an inherent bottleneck to extend these algorithmic strategy to handle weights while achieving a constant approximation ratio; the above results either use an algorithm for the BALANCED SEPARATOR problem that does not admit a constant-factor approximation ratio, under the Small Set Expansion Hypothesis [38], or use a relationship between the size of the separator and the size of resulting pieces that do not hold for weighted graphs.

The second direction is the primal-dual method [10, 7, 12, 16, 18]. The constant-factor approximation of [20] for  $\mathcal{F}$ -VERTEX DELETION is also based on the same core observation of the primal-dual algorithm such as [12]. The 2-approximation for WEIGHTED FEEDBACK

<sup>2</sup> The ETH states that 3-SAT on  $n$  variables cannot be solved in time  $2^{o(n)}$ , see [27] for more details.

<sup>3</sup> In this paper, we consider local-ratio and primal-dual as the same algorithms design paradigm and use the word primal-dual throughout the paper even when the underlying LP is not explicitly given. We refer the reader to the classic survey of Bar-Yehuda et al. [9] for the equivalence.

VERTEX SET became available by introducing a new LP formulation which translates the property “ $G - X$  is a forest” in terms of the *sum of degree contribution* of  $X$ . The idea of expressing the sparsity condition of  $G - X$  in terms of the degree contribution of  $X$  again played the key role in [18] for WEIGHTED DIAMOND HITTING SET. However, the (extended) sparsity inequality of [18] is highly intricate as the LP constraint describes the precise structure of diamond-minor-free graphs (after *taming* the graph via some special protrusion replacer). Therefore, expressing the sparsity condition for other classes  $\mathcal{F}$  with tailor-made LP constraints is likely to be prohibitively convoluted. This implies that a radical simplification of the known algorithm for, say, WEIGHTED DIAMOND HITTING SET will be necessary if one intends to apply the primal-dual method for broader classes.

**Our result and the key ideas.** Let  $\theta_c$  be the graph on two vertices joined by  $c$  parallel edges. The central problem we study is the WEIGHTED  $\mathcal{F}$ -VERTEX DELETION where  $\mathcal{F}$  is the class of  $\theta_c$ -minor-free graphs: a weighted graph  $G = (V, E, w)$  is given as input, and the goal is to find a vertex set  $S$  of minimum weight such that  $G - S$  is  $\theta_c$ -minor-free. We call this particular problem the WEIGHTED  $c$ -BOND COVER problem, as we believe that this nomenclature is more adequate for reasons to be clear in Section 2. Our main result is the following.

► **Theorem 1.** *There is a constant-factor approximation algorithm for WEIGHTED  $c$ -BOND COVER running in uniformly<sup>4</sup> polynomial time.*

Let us briefly recall the classic 2-approximation algorithms for WEIGHTED FEEDBACK VERTEX SET [7, 12]. These algorithms repeatedly delineate a vertex subset  $S$  on which the induced subgraph contains an obstruction (a cycle), and “peel off” a weighted graph on  $S$  from the current weighted graph so that the weight of at least one vertex of the current graph drops to zero. The crux of this approach is to create a weighted graph to peel off (or *design a weighting scheme*) on which every (minimal) feasible solution is consistently an  $\alpha$ -approximate solution. We remark that peeling-off of a weighted graph on  $S$  can be viewed as increasing the dual variable (from zero) corresponding to  $S$  until some dual constraint becomes tight, as articulated in [16].

If one aims to capitalize on the power of the primal-dual method for other minor-closed classes and ultimately for arbitrary  $\mathcal{F}$  with graphs of bounded treewidth, more sophisticated weighting scheme is needed. As we already mentioned, this was successfully done by Fiorini, Joret and Pietropaoli [18] for WEIGHTED DIAMOND HITTING SET, where their primal-dual algorithm is based on an intricate LP formulation. Our primal-dual algorithm diverges from such tactics, and instead use the next structural theorem as a guide for the weighting scheme. Before we present it, we need to define some basic concepts.

Given two disjoint subsets  $X, Y$  of  $V(G)$ , the *edges crossing*  $X$  and  $Y$  is the set of edges with one endpoint in  $X$  and the other in  $Y$ . Notice that  $\theta_c$  is a minor of  $G$  iff  $G$  contains two disjoint connected sets  $X$  and  $Y$  crossed by  $c$  edges of  $G$ . We call the union  $M := X \cup Y$   $\theta_c$ -*model* in  $G$ .

Given a positive integer  $c$ , a  $c$ -*outgrowth* of a graph  $G$  is a triple  $\mathbf{K} = (K, u, v)$  where  $u, v$  are distinct vertices of  $G$ ,  $K$  is a component of  $G \setminus \{u, v\}$ ,  $N_G(V(K)) = \{u, v\}$ , and the graph, denoted by  $K^{(x,y)}$ , obtained from  $G[V(K) \cup \{u, v\}]$  if we remove all edges with endpoints  $u$  and  $v$  is  $\theta_c$ -minor free. The *size* of a  $c$ -outgrowth of  $G$  is the size of  $K$ . A *cluster collection*

<sup>4</sup> We use the term “uniformly polynomial” in order to indicate that a constructive algorithm exists that, for every  $c$ , runs in  $f(c) \cdot n^{O(1)}$  time for some constructible function  $f$ .

of a graph  $G$  is a non-empty collection  $\mathcal{C} = \{C_1, \dots, C_r\}$  of pairwise disjoint non-empty connected subsets of  $V(G)$ . In case  $\bigcup_{C \in \mathcal{C}} C = V(G)$  we say that  $\mathcal{C}$  is a *cluster partition* of  $G$ . The *capacity* of a cluster collection  $\mathcal{C}$  is the maximum number of vertices of a cluster in  $\mathcal{C}$ . We use the notation  $G/\mathcal{C}$  for the multigraph obtained from  $G[\bigcup_{C \in \mathcal{C}} C]$  by contracting<sup>5</sup> all edges in  $G[C_i]$  for each  $i \in \{1, \dots, r\}$ .

► **Theorem 2.** *There is a function  $f_1 : \mathbb{N}^2 \rightarrow \mathbb{N}$  such that, for every two positive integers  $c$  and  $t$ , there is a uniformly polynomial time algorithm that, given as input a graph  $G$ , outputs one of the following:*

1. a  $c$ -outgrowth of size at least  $c$ , or
2. a  $\theta_c$ -model  $M$  of  $G$  of size at most  $f_1(c, t)$ , or
3. a cluster collection  $\mathcal{C}$  of  $G$  of capacity at most  $f_1(c, t)$  such that  $\delta(G/\mathcal{C}) \geq t$ , or
4. a report that  $G$  is  $\theta_c$ -minor free.

(By  $\delta(G)$  we denote the minimum *edge-degree* of a vertex in  $G$ . The *edge-degree* of a vertex  $v$  of  $G$ , denoted by  $\text{edeg}_G(v)$ , is the number of edges that are incident to  $v$ .) A variant of Theorem 2 was originally proved by Joret et al. [29] without the capacity condition on a cluster collection in Case 3. It turns out that imposing the capacity condition of Case 3 is crucial for designing a weighting scheme.

At each iteration, our primal-dual algorithm invokes Theorem 2. Depending on the outcome, the algorithm either runs a *replacer* (defined in Section 2) and reduces the size of a  $c$ -outgrowth, or computes a suitable weighted graph which we call  $\alpha$ -*thin layer* (defined in Section 3), using a suitable weighting scheme, thus reducing the current weight. In both cases, we convert the current weighted graph  $G = (V, E, w)$  into a new weighted graph  $G' = (V', E', w')$  on a strictly smaller number of vertices so that an  $\alpha$ -approximate solution for  $G'$  implies an  $\alpha$ -approximate solution for  $G$  for some particular value of  $\alpha$ .

We stress that the replacer is compatible with any approximation ratio in the sense that the optimal weight of a solution is unchanged and every solution after the replacement can be transformed to a solution that is at least as good. When Theorem 2 reports a constant-sized  $\theta_c$ -model, it is easy to see that a uniformly weighted  $\alpha$ -thin layer suffices. The gist of Theorem 2 is in the third case, which promises a collection of pairwise disjoint *constant-sized* connected sets.

Let us first consider the simplest such case where all connected sets are singletons, namely when  $\delta(G) \geq t$ . It is not difficult to see that, if we consider  $t := 6c$  and under the *edge-degree-proportional* weight function, that is for every  $v \in V(G)$ ,  $w(v) := \text{edeg}_G(v)$ , any feasible solution to WEIGHTED  $c$ -BOND COVER is a 4-approximate solution.

In the general case where we have a collection of pairwise disjoint connected sets, each of size at most  $r$ , the critical observation (Lemma 3) is that if the contraction of these sets yields a graph of minimum edge-degree at least  $t := 8c$ , then a weighting scheme akin to the simple case also works. That is, any feasible solution to WEIGHTED  $c$ -BOND COVER is a  $4r$ -approximate solution. The overall primal-dual framework is summarized in Section 3.

## 2 Preliminary definitions and results

We use  $\mathbb{N}$  for the set of non-negative integers and  $\mathbb{R}_{\geq 0}$  for the set of non-negative reals. Given some  $r \in \mathbb{N}$ , we define  $[r] = \{1, \dots, r\}$ . Given some collection  $\mathcal{A}$  of objects on which the union operation can be defined, we define  $\bigcup \mathcal{A} = \bigcup_{A \in \mathcal{A}} A$ . All graphs we consider are multigraphs

<sup>5</sup> When considering edge contractions we sum up edge multiplicities of multiple edges that are created during the contraction. However, when a loop appears after a contraction, then we suppress it.

## 7:6 Approximation for Weighted Bond Cover

without loops. We denote a graph by  $G = (V, E)$  where  $V$  and  $E$  are its vertex and edge set respectively. A vertex-weighted graph is denoted by  $G = (V, E, w)$  where  $w : V(G) \rightarrow \mathbb{R}_{\geq 0}$  and we say that  $G$  is a  $w$ -weighted graph. We use  $V(G)$  and  $E(G)$  for the vertex set and the edge multiset of  $G$ . We also refer to  $|V(G)|$  as the *size* of  $G$ . If  $X \subseteq V(G)$ , we denote by  $G[X]$  the subgraph of  $G$  induced by  $X$  and by  $G - X$  the graph  $G[V(G) \setminus X]$ . We say that  $X$  is *connected in  $G$*  if  $G[X]$  is connected. A graph  $H$  is a *minor* of a graph  $G$  if  $H$  can be obtained from a subgraph of  $G$  after contracting edges. Given a graph  $H$ , we say that  $G$  is  *$H$ -minor free* if  $G$  does not contain  $H$  as a minor. We denote by  $N_G(v)$  the set of all neighbors of  $v$  in  $G$ .

**Covering bonds.** Let  $G$  be a graph. Given a bipartition  $\{V_1, V_2\}$  of  $V(G)$ , the set of edges crossing  $V_1$  and  $V_2$  is called the *cut* of  $\{V_1, V_2\}$  and an edge set is a *cut* if it is a cut of some vertex bipartition. A minimal non-empty cut is known as a *bond* in the literature. We remark that the bonds of  $G$  are precisely the circuits of the cographic matroid of  $G$ . Given a positive integer  $c$ , a  *$c$ -bond* of a graph  $G$  is any minimal cut of  $G$  of size at least  $c$ . The problem of finding the maximum  $c$  for which a graph  $G$  contains a  $c$ -bond has been examined both from the approximation [25, 15] and the parameterized point of view [17]. Given a set  $S \subseteq V(G)$ , we say that  $S$  is a  *$c$ -bond cover* of  $G$  if  $G - S$  is  $\theta_c$ -minor free. Notice that  $S$  is a  $c$ -bond cover iff  $G \setminus S$  does not contain a  $c$ -bond. Given a weighted graph  $G = (V, E, w)$  with  $w : V(G) \rightarrow \mathbb{R}_{\geq 0}$ , a *minimum weight  $c$ -bond cover* of  $G$  is a  $c$ -bond cover  $S$  where the weight of  $S$ , defined as  $w(S) := \sum_{v \in S} w(v)$ , is minimized.

It is easy to prove that, for every  $c \in \mathbb{N}$ , a graph  $G$  contains  $\theta_c$  as a minor iff it has a  $c$ -bond. This means that when  $\mathcal{F}$  is the class of  $\theta_c$ -minor-free graphs, then WEIGHTED  $\mathcal{F}$ -VERTEX DELETION can be restated as follows.

WEIGHTED  $c$ -BOND COVER

*Input:* a vertex weighted graph  $G = (V, E, w)$ .

*Solution:* a minimum weight  $c$ -bond cover of  $G$ .

**The weighting scheme.** Let  $G$  be a graph and let  $\mathcal{C}$  be a cluster partition of  $G$  of capacity at most  $r$ . Given a cluster  $C \in \mathcal{C}$  we denote by  $\text{ext}_C(C)$  (or simply  $\text{ext}(C)$ ) the set of edges with one endpoint in  $C$  and the other not in  $C$ .

Let now  $G$  be an instance of WEIGHTED  $c$ -BOND COVER for some positive integer  $c$ . We define the vertex weighting function  $w_C : V(G) \rightarrow \mathbb{R}_{\geq 0}$  so that if  $v \in C \in \mathcal{C}$ , then

$$w_C(v) = \frac{|\text{ext}(C)|}{|C|}. \quad (1)$$

When  $\mathcal{C}$  is clear from the context, we simply write  $w$  instead  $w_C$ . The main result of this section is that, with respect to the weight function  $w$  in Equation 1, every  $c$ -bond cover of  $G$  is a  $4r$ -approximation.

► **Lemma 3.** *Let  $c$  be a non negative integer,  $G$  be a graph,  $r$  be a positive integer,  $\mathcal{C}$  be a cluster partition of  $G$  of capacity at most  $r$  and such that  $\delta(G/\mathcal{C}) \geq 8c$ , and  $w : V(G) \rightarrow \mathbb{R}_{\geq 0}$  be a vertex weighting function as in Equation 1. Then for every  $c$ -bond cover  $X$  of  $G$ , it holds that  $\frac{1}{2r} \cdot |E(G/\mathcal{C})| \leq \sum_{v \in X} w(v) \leq 2 \cdot |E(G/\mathcal{C})|$ .*

**Proof.** For the upper bound, note that  $\sum_{v \in X} w(v) = \sum_{v \in X} \frac{|\text{ext}(C)|}{|C|} \leq \sum_{C \in \mathcal{C}} \sum_{v \in C} \frac{|\text{ext}(C)|}{|C|} = \sum_{C \in \mathcal{C}} |\text{ext}(C)| = \sum_{x \in V(G/\mathcal{C})} \text{edeg}_{G/\mathcal{C}}(x) = 2 \cdot |E(G/\mathcal{C})|$ .

For the lower bound, let  $X$  be a  $c$ -bond cover of  $G$  and let  $F = V(G) \setminus X$ ,  $\mathcal{C}_X = \{C \in \mathcal{C} \mid C \cap X \neq \emptyset\}$ , and  $\mathcal{C}_F = \mathcal{C} \setminus \mathcal{C}_X$ . Since  $\delta(G/\mathcal{C}) \geq 8c$ , we obtain that  $|E(G/\mathcal{C})|/2 \geq 2c \cdot |V(G/\mathcal{C})| = 2c \cdot |\mathcal{C}|$ . We claim that  $\sum_{C \in \mathcal{C}_X} |\text{ext}(C)| \geq |E(G/\mathcal{C})|/2$ . Indeed, if this is not the case then, by the fact that  $|E(G/\mathcal{C})| \leq |E(G[F]/\mathcal{C}_F)| + \sum_{C \in \mathcal{C}_X} |\text{ext}(C)|$ , we have that  $|E(G[F]/\mathcal{C}_F)| > |E(G/\mathcal{C})|/2 \geq 2c \cdot |\mathcal{C}| \geq 2c \cdot |\mathcal{C}_F|$  and this last inequality, gives that  $\theta_c$  is a minor of  $G/\mathcal{C}_F$  which is a minor of  $G[F]$ , a contradiction. Here we use the fact that for every  $\theta_c$ -minor free multigraph  $G$ , it holds that  $|E(G)| \leq 2c \cdot |V(G)|$  (following from the main combinatorial result of [36]). Therefore, since each set in  $\mathcal{C}_X$  contains at least one vertex of  $X$ , we obtain  $\sum_{v \in X} w(v) \geq \sum_{C \in \mathcal{C}_X} \frac{|\text{ext}(C)|}{|C|} \geq \frac{1}{r} \sum_{C \in \mathcal{C}_X} |\text{ext}(C)| \geq \frac{|E(G/\mathcal{C})|}{2r}$ , which proves the lower bound.  $\blacktriangleleft$

**Replacing outgrowths.** A  $c$ -outgrowth replacer (hereinafter *replacer*) is a uniformly polynomial-time algorithm which, given a weighted graph  $G = (V, E, w)$  and a  $c$ -outgrowth  $\mathbf{K} = (K, u, v)$  of size at least  $c$ , outputs a weighted graph  $G' = (V', E', w')$  with the following property.

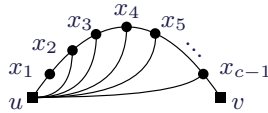
1.  $\mathbf{K}$  is replaced by another  $c$ -outgrowth  $\mathbf{K}' = (K', u, v)$  of size at most  $c - 1$ .
2.  $\text{opt}(G) = \text{opt}(G')$ .
3. Given a  $c$ -bond cover  $S' \subseteq V(G')$ , one can construct in polynomial time a  $c$ -bond cover  $S \subseteq V(G)$  such that  $w(S) \leq w(S')$ .

We now present our  $c$ -outgrowth replacer. Given a  $w$ -weighted graph  $G$ , we denote by  $\text{opt}(G)$  the weight of an optimal solution for WEIGHTED  $c$ -BOND COVER on  $G$ .

**► Lemma 4.** *For every positive  $c \in \mathbb{N}$ , there is a  $c$ -outgrowth replacer. In particular, an  $\alpha$ -approximate solution for  $G'$  implies an  $\alpha$ -approximate solution for  $G$ .*

**Proof.** For  $i \in \{0, \dots, c - 1\}$ , let  $K_i^{(u,v)}$  be the graph obtained from  $K^{(u,v)}$  by adding  $i$  edges connecting  $u$  and  $v$ . Obviously  $K_0^{(u,v)}$  equals  $K^{(u,v)}$ . Let also  $T_i \subseteq V(K)$  be a minimum weight set contained in  $V(K)$  such that  $K_i^{(u,v)} - T_i$  is  $\theta_c$ -minor-free, and  $w_i = w(T_i)$ . Note that  $T_i \subseteq V(K)$  implies that  $T_i$  contains neither  $u$  nor  $v$ . For example,  $T_{c-1}$  is a minimum (internal) vertex cut separating  $u$  and  $v$  in  $K^{(u,v)}$ , and  $w_{c-1} = w(T_{c-1})$  is finite since there is no edge between  $u$  and  $v$  in  $K^{(u,v)}$ . By definition, it holds that  $0 = w_0 \leq w_1 \leq \dots \leq w_{c-1} < \infty$ , and these values can be computed in uniformly polynomial time by using dynamic programming on  $\theta_c$ -minor free graphs (that is bounded treewidth graphs). We also remark that  $T_j$  is a  $c$ -bond cover of  $K_i^{(u,v)}$  for all  $i \leq j$ . We construct the  $c$ -outgrowth  $\mathbf{K}' = (K', u, v)$  so that  $K'^{(u,v)}$  is as follows (see Figure 1).

- $V(K'^{(u,v)}) = \{u, v, x_1, \dots, x_{c-1}\}$  where  $K' = \{x_1, \dots, x_{c-1}\}$ . For each  $1 \leq i \leq c - 1$ , the weight of  $x_i$  is  $w_i$ .
- There are edges  $(u, x_1), (x_1, x_2), \dots, (x_{c-2}, x_{c-1}), (x_{c-1}, v)$ . Additionally for each  $2 \leq i \leq c - 1$ , there is an edge  $(x_i, u)$ .



**■ Figure 1** The construction of the replacement  $c$ -outgrowth  $\mathbf{K}' = (K', u, v)$ .

We observe that for each  $i \in \{0, \dots, c - 1\}$ , the set  $\{x_i\}$  is the minimum weight  $c$ -bond cover of  $K_i^{(u,v)}$ . The next claims are handy (the proof is omitted in this extended abstract).

## 7:8 Approximation for Weighted Bond Cover

▷ **Claim 5.** Let  $(K, u, v)$  be a  $c$ -outgrowth in  $G$  and let  $M = (X, Y)$  be a minimal  $\theta_c$ -model in  $G$ . If  $M$  does not contain  $u$ , then we have  $(X \cup Y) \cap V(K) = \emptyset$ . Furthermore, if  $S$  is a minimal  $c$ -bond cover of  $G$  and if  $S$  contains  $u$  or  $v$ , say  $u$ , then  $S \cap V(K) = \emptyset$ .

▷ **Claim 6.** Let  $Z$  be a  $c$ -bond cover of  $G - V(K)$  and let  $\ell$  be the maximum integer<sup>6</sup> such that  $G - (K \cup Z)$  contains a  $\theta_\ell$ -model with  $u$  and  $v$  in different sets. Then  $Z' = Z \cup T_\ell$  is a  $c$ -bond cover of  $G$ .

We begin with proving the third condition of the replacer. Let  $G'$  be the graph where  $K^{(u,v)}$  is replaced by  $K'^{(u,v)}$ . It suffices to prove the second statement for an arbitrary minimal  $c$ -bond cover  $S' \subseteq V(G')$  of  $G'$ .

First, assume that  $S'$  contains  $u$  or  $v$ , say  $u$ . Claim 5 is applied to  $G'$  verbatim with  $G \leftarrow G'$ ,  $K \leftarrow K'$ ,  $K^{(u,v)} \leftarrow K'^{(u,v)}$ , and we deduce that  $S' \cap V(K') = \emptyset$ . Now we take  $S \leftarrow S'$ , and let us argue that  $S$  is a  $c$ -bond cover of  $G$ . Again Claim 5 implies that if  $G - S$  contains a  $\theta_c$ -model, then one can find one disjoint from  $V(K)$ . This is not possible because  $S = S'$  is a  $c$ -bond cover of  $G - K = G' - K'$ .

Secondly, let us assume that  $S' \cap \{u, v\} = \emptyset$ . Let  $\ell$  be the maximum integer such that  $G' - (K' \cup S')$  contains a  $\theta_\ell$ -model  $M = (X, Y)$  with  $u$  and  $v$  in different sets, say  $u \in X$  and  $v \in Y$ . Clearly  $\ell$  is strictly smaller than  $c$  because  $S'$  is a  $c$ -bond cover of  $G' - K'$ . Note that  $K_\ell'^{(u,v)}$  is obtained from  $G'[X \cup Y \cup V(K')]$  by contracting  $X$  and  $Y$ . Because  $S' \cap V(K')$  is a  $c$ -bond cover of  $G'[X \cup Y \cup V(K')]$ , it is also a  $c$ -bond cover of  $K_\ell'^{(u,v)}$ . Therefore we have  $w_\ell \leq w(S' \cap V(K'))$ .

Let  $S = (S' \setminus V(K')) \cup T_\ell$  be a vertex set of  $G$  and note that  $w(S) \leq w(S')$ . Now applying Claim 6 to  $G$  with  $Z \leftarrow S' \setminus V(K')$  (as a vertex set of  $G$ ), we conclude that  $S$  is a  $c$ -bond cover of  $G$ . This proves the third condition of the replacer, which also establishes  $\text{opt}(G) \leq \text{opt}(G')$  in the second condition of the replacer.

It remains to show  $\text{opt}(G) \geq \text{opt}(G')$ . Consider an optimal  $c$ -bond cover  $S$  of  $G$ , and let  $p$  be the maximum integer such that  $G - (K \cup S)$  contains a  $\theta_p$ -model  $M = (X, Y)$  with  $u$  and  $v$  in different sets. Again we apply Claim 6 with  $G \leftarrow G'$ ,  $Z \leftarrow S \setminus V(K)$  (as a vertex set of  $G'$ ),  $K \leftarrow K'$  and  $T_\ell \leftarrow \{x_p\}$ , and derive that  $(S \setminus V(K)) \cup \{x_p\}$  is a  $c$ -bond cover of  $G'$ . Lastly, observe that  $K_p^{(u,v)}$  is a minor of  $G[X \cup Y \cup V(K)]$ , and because  $S \cap V(K)$  is a  $c$ -bond cover of the latter, it is also a  $c$ -bond cover of the former. Therefore, we have  $w(S \cap V(K)) \geq w_p$ , from which we have  $\text{opt}(G) = w(S) \geq w(S \setminus V(K)) + w_p = w((S \setminus V(K)) \cup \{x_p\}) \geq \text{opt}(G')$ . ◀

### 3 The primal-dual approach

We begin the section by formalizing the notion of  $\alpha$ -thin layer. An  $\alpha$ -thin layer of a weighted graph  $G = (V, E, w)$  is a weighted graph  $H = (V, E, w^\alpha)$  such that the following holds.

- $w^\alpha(v) \leq w(v)$  for every  $v \in V$ ,
- $w^\alpha(v) = w(v)$  for some  $v \in V$ , and
- $w^\alpha(S) \geq (1/\alpha) \cdot w^\alpha(V)$  for any  $c$ -bond cover  $S \subseteq V$  of  $H$ .

We are now ready to prove our main approximation result.

► **Theorem 7.** *There is a uniformly polynomial-time algorithm which, given a positive integer  $c$  and a weighted graph  $G = (V, E, w)$ , computes a  $c$ -bond cover of weight at most  $\alpha \cdot \text{opt}(G)$  for some  $\alpha = \alpha(c)$ .*

<sup>6</sup> If  $u$  and  $v$  are not connected in  $G - (K \cup Z)$ , we let  $\ell = 0$ .

**Proof.** The algorithm initially sets  $G_1 = G$ , and iteratively constructs a sequence of weighted graphs  $G_i = (V_i, E_i, w_i)$  for  $i = 0, 1, \dots$ . At  $i$ -th iteration, we run the algorithm  $\mathcal{A}$  of Theorem 2 for  $t = 8c$ . If  $\mathcal{A}$  detects a  $c$ -outgrowth of size at least  $c$ , then we call the algorithm of Lemma 4, which is clearly a replacer. We run the replacer on  $G_i$  and set  $G_{i+1}$  to be the output of the replacer. If a  $\theta_c$ -model  $M$  of  $G_i$  of size at most  $f_1(c, t)$  is detected by  $\mathcal{A}$ , then let  $\epsilon := \min\{w_i(v) : v \in M\}$  and consider the weighted graph  $H_i = (V_i, E_i, w_i^o)$  with the weight function  $w$  as follows:

$$w_i^o(v) = \begin{cases} \epsilon & \text{if } v \in M \\ 0 & \text{otherwise.} \end{cases}$$

It is obvious that  $H_i$  is an  $\alpha$ -thin layer with  $\alpha = f_1(c, t)$ .

In the third case, note that the cluster collection  $\mathcal{C}$  forms a cluster partition of  $G_i[\mathbf{UC}]$ . Consider the weight function  $w : \mathbf{UC} \rightarrow \mathbb{R}_{\geq 0}$  as in Equation 1 of  $G_i[\mathbf{UC}]$ . Let  $\epsilon := \min\{w_i(v)/w(v) : v \in \mathbf{UC}\}$  and  $H_i = (V_i, E_i, w_i^o)$  be the weighted graph, where

$$w_i^o(v) = \begin{cases} \epsilon \cdot w(v) & \text{if } v \in \mathbf{UC} \\ 0 & \text{otherwise.} \end{cases}$$

Let us verify that  $H_i$  is an  $\alpha$ -thin layer of  $G_i$  for  $\alpha = 4r$ , where  $r = f_1(c, t)$ . It is straightforward to see that the first two requirements of  $\alpha$ -thin layer are met due to the choice of  $\epsilon$ . To check the last requirement, consider an arbitrary  $c$ -bond cover  $S \subseteq V_i$  of  $H_i$ . By Lemma 3, it holds that

$$\frac{\epsilon}{2r} \cdot |E(G_i[\mathbf{UC}]/\mathcal{C})| \leq \sum_{v \in S} w_i^o(v) \leq \sum_{v \in V_i} w_i^o(v) \leq 2\epsilon \cdot |E(G_i[\mathbf{UC}]/\mathcal{C})|,$$

and therefore,

$$\sum_{v \in S} w_i^o(v) \geq \frac{\epsilon}{2r} \cdot |E(G_i[\mathbf{UC}]/\mathcal{C})| \geq \frac{1}{4r} \cdot \sum_{v \in V_i} w_i^o(v).$$

In both the second and the third cases, we set  $G_{i+1}$  to be the weighted graph  $(V_i, E_i, w_i - w_i^o)$  after removing all vertices of weight zero.

Finally, if  $\mathcal{A}$  reports that  $G_i$  is  $\theta_c$ -minor free, then we terminate the iteration. Let  $G = G_1, G_2, \dots, G_\ell$  be the constructed sequence of weighted graph at the end, with  $G_\ell$  being a  $\theta_c$ -minor-free graph. Observe that our algorithm strictly decrease the number of vertices before the  $\ell$ -th iteration, and thus  $\ell \leq n$ .

To establish the main statement, it suffices show that there is a polynomial-time algorithm which produces an  $4r$ -approximate solution for  $G_i$  given an  $4r$ -approximate solution  $T_{i+1}$  for  $G_{i+1}$ , where  $r = f_1(c, t)$  and  $t = 8c$ . This trivially holds if the execution of  $\mathcal{A}$  at  $i$ -th iteration calls the replacer.

Suppose that  $i$ -th iteration produces an  $\alpha$ -thin layer  $H_i = (V_i, E_i, w_i^o)$ , and recall that every  $\alpha$ -thin layer produced in our algorithm satisfies  $\alpha \leq 4r$ . As  $T_{i+1}$  is an  $4r$ -approximate solution for  $G_{i+1}$ , we have

$$\text{opt}(G_{i+1}) \geq (1/4r) \cdot w_{i+1}(T_{i+1}), \quad (2)$$

▷ **Claim 8.**  $T_i := T_{i+1} \cup (V_i \setminus V_{i+1})$  is an  $4r$ -approximate solution for  $G_i$ .

**Proof.** Let  $D_i = V_i \setminus V_{i+1}$ , namely the vertices deleted from  $G_i$  to obtain  $G_{i+1}$ . It is obvious that  $T_{i+1} \cup D_i$  is a feasible solution for  $G_i$ , that is, a  $c$ -bond cover of  $G_i$  because  $G_{i+1} - T_{i+1} = G_i - (T_{i+1} \cup D_i)$  and  $T_{i+1}$  is a  $c$ -bond cover of  $G_{i+1}$ . Let  $Q \subseteq V_i$  be an optimal

## 7:10 Approximation for Weighted Bond Cover

solution for  $G_i$ . Then  $Q$  is a feasible solution for  $H_i$  and  $Q \cap V_{i+1}$  is a feasible solution for  $G_{i+1}$ , therefore

$$w_i^o(Q) \geq (1/4r) \cdot w_i^o(V_i) \text{ and} \quad (3)$$

$$w_{i+1}(Q \cap V_{i+1}) \geq \text{opt}(G_{i+1}), \quad (4)$$

where the inequality 3 is due to the third requirement of  $\alpha$ -thin layer. Furthermore, it holds that

$$w_i(v) = w_i^o(v) + w_{i+1}(v) \quad \text{for each } v \in V_{i+1} \text{ and} \quad (5)$$

$$w_i(v) = w_i^o(v) \quad \text{for each } v \in D_i. \quad (6)$$

Therefore,

$$\begin{aligned} w_i(Q) &= w_i^o(Q) + w_{i+1}(Q \cap V_{i+1}) && \because (5), (6) \\ &\geq (1/4r) \cdot w_i^o(V_i) + \text{opt}(G_{i+1}) && \because (3), (4) \\ &\geq (1/4r) \cdot w_i^o(T_{i+1} \cup D_i) + (1/4r) \cdot w_{i+1}(T_{i+1}) && \because (2) \\ &= (1/4r) \cdot (w_i^o(T_{i+1}) + w_{i+1}(T_{i+1})) + (1/4r) \cdot w_i^o(D_i) \\ &= (1/4r) \cdot w_i(T_{i+1} \cup D_i) && \because (5), (6) \end{aligned}$$

and the claim follows.  $\triangleleft$

We inductively obtain a  $4r$ -approximate solution for  $G_i$ , and finally for the graph  $G_1 = G$ . This finishes the proof.  $\blacktriangleleft$

## 4 Discussion

In this paper we construct a polynomial-time constant-factor approximation algorithm for the WEIGHTED  $\mathcal{F}$ -VERTEX DELETION problem in the case  $\mathcal{F}$  is the class of graphs not containing a  $c$ -bond or, alternatively, the  $\theta_c$ -minor free graphs. The constant-factor of our approximation algorithm is a (constructible) function of  $c$  and the running time is uniformly polynomial. Our results, in case  $c = 2$ , yield a constant-factor approximation for the WEIGHTED FEEDBACK VERTEX SET. Also, a constant-factor approximation for WEIGHTED DIAMOND HITTING SET can easily be derived for the case where  $c = 3$ . For this we apply our results on simple graphs and observe that each time a  $\theta_3$ -minor-model appears, this model, under the absence of multiple edges, should contain 4 vertices and therefore is a minor-model of the diamond  $K_4^-$  (that is  $K_4$  without an edge).

Certainly the general open question is whether WEIGHTED  $\mathcal{F}$ -VERTEX DELETION admits a constant-factor approximation for more general instantiations of the minor-closed class  $\mathcal{F}$ . In this direction, the challenge is to use our approach when the graphs in  $\mathcal{F}$  have bounded treewidth (or, equivalently, if the minor obstruction of  $\mathcal{F}$  contains some planar graph). For this, one needs to extend the structural result of Theorem 2 and, based on this to build a replacer as in Lemma 4.

Given an  $r \in \mathbb{N}$ , an  $r$ -*protrusion* of a graph  $G$  is a set  $X \subseteq V(G)$  such that  $G[X]$  has treewidth at most  $t$  and  $|\partial_G(X)| \leq t$ , where  $\partial_G(X)$  is the set of vertices of  $X$  that are incident to edges not in  $G[X]$ . We conjecture that a possible extension of Theorem 2 might be the following.

**► Conjecture 9.** *There are functions  $f_2 : \mathbb{N}^2 \rightarrow \mathbb{N}$  and  $f_3 : \mathbb{N}^3 \rightarrow \mathbb{N}$  such that, for every  $h$ -vertex planar graph  $H$  and every two positive integers  $t, p$ , there is a uniformly polynomial time algorithm that, given as input a graph  $G$ , outputs one of the following:*



1. an  $f_2(h, t)$ -protrusion  $X$  of size at least  $p$ , or
2. a minor-model of  $H$  of size at most  $f_3(h, t, p)$ , or
3. a cluster collection  $\mathcal{C}$  of  $G$  of capacity at most  $f_3(h, t, p)$  such that  $\delta(G/\mathcal{C}) \geq t$ , or
4. a report that  $G$  is  $H$ -minor free.

Given a proof of some suitable version of Conjecture 9 at hand, cases 1,2, and 3 above can be treated using the method proposed in this paper. In the first case, we need to find a *weighted protrusion replacer* that can replace, in the weighed graph  $G = (V, E, w)$ , the subgraph  $G[X]$  by another one (glued on the same boundary) and create a new weighted graph  $G' = (V', E', w')$  so that an optimal solution has the same weight in both instances. In our case, the role of a protrusion is played by the  $c$ -outgrowth, where  $X$  is the vertex set of  $K^{(u,v)}$  that has treewidth at most  $2c$  and  $|\partial_G(X)| \leq 2$ , i.e.,  $V(K^{(u,v)})$  is a  $2c$ -protrusion of  $G$ . In the case of  $\theta_c$ , the the replacer is given in Lemma 4. The existence of such a replacer in the general case is wide open, first because the boundary  $\partial_G(X)$  has bigger size (depending on  $h$  but perhaps also on  $t$ ) and second, and most important, because we now must deal with *weights* which does not permit us to use any protrusion replacement machinery such as the one used in [20, 19] unweighted version of the problem (based on the, so called, FII-property [13] for more details).

We believe that a possible way to prove Conjecture 9 is to use as departure the proof of the main combinatorial result in [45]. However, in our opinion, the most challenging step is to design a weighted protrusion replacer (or, on the negative side, to provide instantiations of  $H$  where such a replacer does not exist). As such a replacer needs to work on the presence of weights, we suggest that its design might use techniques related to mimicking networks technology [24, 34].

Finally, since our algorithm is based on the primal-dual framework and proceeds by constructing suitable weights for the second and third case where every feasible solution is  $\mathcal{O}(1)$ -approximate, one can ask whether it is possible to *bypass* the need for a replacer and construct suitable weights for the first case. Indeed, the previous approximation algorithms for WEIGHTED FEEDBACK VERTEX SET [7, 12, 16] designed suitable weights even for the case 1 where every *minimal* solution is  $\mathcal{O}(1)$ -approximate. (And used the additional “reverse delete” step at the end to ensure that the final solution remains minimal, for every weighted graph constructed.) In the full version of the paper [32], we show that such weights *cannot exist* for a simple planar graph  $H$ , which suggests that replacers are inherently needed for this class of algorithms for WEIGHTED  $\mathcal{F}$ -VERTEX DELETION.

---

## References

- 1 Akanksha Agrawal, Sudeshna Kolay, Daniel Lokshtanov, and Saket Saurabh. A faster FPT algorithm and a smaller kernel for block graph vertex deletion. In *LATIN 2016: theoretical informatics*, volume 9644 of *Lecture Notes in Comput. Sci.*, pages 1–13. Springer, Berlin, 2016. doi:10.1007/978-3-662-49529-2\_1.
- 2 Akanksha Agrawal, Daniel Lokshtanov, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. Polylogarithmic approximation algorithms for weighted- $\mathcal{F}$ -deletion problems. In *Approximation, randomization, and combinatorial optimization. Algorithms and techniques*, volume 116 of *LIPICs. Leibniz Int. Proc. Inform.*, pages Art. No. 1, 15. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2018.
- 3 Akanksha Agrawal, Daniel Lokshtanov, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. Feedback vertex set inspired kernel for chordal vertex deletion. *ACM Trans. Algorithms*, 15(1):Art. 11, 28, 2019. doi:10.1145/3284356.

- 4 Akanksha Agrawal, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. Interval vertex deletion admits a polynomial kernel. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1711–1730. SIAM, Philadelphia, PA, 2019. doi:10.1137/1.9781611975482.103.
- 5 Jungho Ahn, Eduard Eiben, O-joung Kwon, and Sang-il Oum. A polynomial kernel for 3-leaf power deletion. In *45th International Symposium on Mathematical Foundations of Computer Science*, volume 170 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 5:1–5:14. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2020.
- 6 Jungho Ahn, Eun Jung Kim, and Euiwoong Lee. Towards constant-factor approximation for chordal / distance-hereditary vertex deletion. 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 62:1–62:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ISAAC.2020.62.
- 7 Vineet Bafna, Piotr Berman, and Toshihiro Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM J. Discret. Math.*, 12(3):289–297, 1999. doi:10.1137/S0895480196305124.
- 8 Nikhil Bansal, Daniel Reichman, and Seeun William Umboh. LP-based robust algorithms for noisy minor-free and bounded treewidth graphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1964–1979. SIAM, Philadelphia, PA, 2017. doi:10.1137/1.9781611974782.128.
- 9 Reuven Bar-Yehuda, Keren Bendel, Ari Freund, and Dror Rawitz. Local ratio: A unified framework for approximation algorithms. in memoriam: Shimon even 1935-2004. *ACM Computing Surveys (CSUR)*, 36(4):422–463, 2004.
- 10 Reuven Bar-Yehuda and Simon Even. A local-ratio theorem for approximating the weighted vertex cover problem. In G. Ausiello and M. Lucertini, editors, *Analysis and Design of Algorithms for Combinatorial Problems*, volume 109 of *North-Holland Mathematics Studies*, pages 27–45. North-Holland, 1985. doi:10.1016/S0304-0208(08)73101-3.
- 11 Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos. A complexity dichotomy for hitting connected minors on bounded treewidth graphs: the chair and the banner draw the boundary. 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 951–970. SIAM, 2020. doi:10.1137/1.9781611975994.57.
- 12 Ann Becker and Dan Geiger. Optimization of pearl’s method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artificial Intelligence*, 83(1):167–188, 1996. doi:10.1016/0004-3702(95)00004-6.
- 13 Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninx, Saket Saurabh, and Dimitrios M. Thilikos. (meta) kernelization. *J. ACM*, 63(5):44:1–44:69, 2016. doi:10.1145/2973749.
- 14 Yixin Cao. Linear recognition of almost interval graphs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1096–1115. ACM, New York, 2016. doi:10.1137/1.9781611974331.ch77.
- 15 Robert D. Carr, Lisa Fleischer, Vitus J. Leung, and Cynthia A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. 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 106–115. ACM/SIAM, 2000. URL: <http://dl.acm.org/citation.cfm?id=338219.338241>.
- 16 Fabián A. Chudak, Michel X. Goemans, Dorit S. Hochbaum, and David P. Williamson. A primal-dual interpretation of two 2-approximation algorithms for the feedback vertex set problem in undirected graphs. *Oper. Res. Lett.*, 22(4-5):111–118, 1998. doi:10.1016/S0167-6377(98)00021-2.

- 17 Hiroshi Eto, Tesshu Hanaka, Yasuaki Kobayashi, and Yusuke Kobayashi. Parameterized algorithms for maximum cut with connectivity constraints. In Bart M. P. Jansen and Jan Arne Telle, editors, *14th International Symposium on Parameterized and Exact Computation, IPEC 2019, September 11-13, 2019, Munich, Germany*, volume 148 of *LIPICs*, pages 13:1–13:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.IPEC.2019.13.
- 18 Samuel Fiorini, Gwenaël Joret, and Ugo Pietropaoli. Hitting diamonds and growing cacti. In *Integer Programming and Combinatorial Optimization – IPCO 2010*, volume 6080 of *Lecture Notes in Comput. Sci.*, pages 191–204. Springer, Berlin, 2010. doi:10.1007/978-3-642-13036-6\_15.
- 19 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, Geevarghese Philip, and Saket Saurabh. Hitting forbidden minors: Approximation and kernelization. *SIAM J. Discret. Math.*, 30(1):383–410, 2016. doi:10.1137/140997889.
- 20 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar  $\mathcal{F}$ -Deletion: approximation, kernelization and optimal FPT algorithms. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science – FOCS 2012*, pages 470–479. IEEE Computer Soc., Los Alamitos, CA, 2012.
- 21 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Hitting topological minors is FPT. 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 1317–1326. ACM, 2020. doi:10.1145/3357713.3384318.
- 22 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Approximation schemes via width/weight trade-offs on minor-free graphs. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2299–2318. SIAM, 2020. doi:10.1137/1.9781611975994.141.
- 23 Anupam Gupta, Euiwoong Lee, Jason Li, Pasin Manurangsi, and Michał Włodarczyk. Losing treewidth by separating subsets. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1731–1749. SIAM, Philadelphia, PA, 2019. doi:10.1137/1.9781611975482.104.
- 24 Torben Hagerup, Jyrki Katajainen, Naomi Nishimura, and Prabhakar Ragde. Characterizing multiterminal flow networks and computing flows in networks of small treewidth. *J. Comput. Syst. Sci.*, 57(3):366–375, 1998. doi:10.1006/jcss.1998.1592.
- 25 David J. Haglin and Shankar M. Venkatesan. Approximation and intractability results for the maximum cut problem and its variants. *IEEE Trans. Computers*, 40(1):110–113, 1991. doi:10.1109/12.67327.
- 26 Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424–435, 2012. doi:10.1016/j.jctb.2011.07.004.
- 27 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.
- 28 Bart M. P. Jansen and Marcin Pilipczuk. Approximation and kernelization for chordal vertex deletion. *SIAM J. Discrete Math.*, 32(3):2258–2301, 2018. doi:10.1137/17M112035X.
- 29 Gwenaël Joret, Christophe Paul, Ignasi Sau, Saket Saurabh, and Stéphan Thomassé. Hitting and harvesting pumpkins. *SIAM J. Discret. Math.*, 28(3):1363–1390, 2014. doi:10.1137/120883736.
- 30 Ken-ichi Kawarabayashi and Anastasios Sidiropoulos. Polylogarithmic approximation for minimum planarization (almost). In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 779–788. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.77.
- 31 Eun Jung Kim, Alexander Langer, Christophe Paul, Felix Reidl, Peter Rossmanith, Ignasi Sau, and Somnath Sikdar. Linear kernels and single-exponential algorithms via protrusion decompositions. *ACM Trans. Algorithms*, 12(2):21:1–21:41, 2016. doi:10.1145/2797140.

- 32 Eun Jung Kim, Euiwoong Lee, and Dimitrios M. Thilikos. A constant-factor approximation for weighted bond cover, 2021. [arXiv:2105.00857](https://arxiv.org/abs/2105.00857).
- 33 Stefan Kratsch and Magnus Wahlström. Compression via matroids: a randomized polynomial kernel for odd cycle transversal. *ACM Trans. Algorithms*, 10(4):Art. 20, 15, 2014. doi:10.1145/2635810.
- 34 Robert Krauthgamer and Inbal Rika. Mimicking networks and succinct representations of terminal cuts. 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 1789–1799. SIAM, 2013. doi:10.1137/1.9781611973105.128.
- 35 John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *J. Comput. System Sci.*, 20(2):219–230, 1980. ACM-SIGACT Symposium on the Theory of Computing (San Diego, Calif., 1978).
- 36 Stratis Limnios, Christophe Paul, Joanny Perret, and Dimitrios M. Thilikos. Edge degeneracy: Algorithmic and structural results. *Theor. Comput. Sci.*, 839:164–175, 2020. doi:10.1016/j.tcs.2020.06.006.
- 37 Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan, Geevarghese Philip, and Saket Saurabh. A  $(2 + \epsilon)$ -factor approximation algorithm for split vertex deletion. 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 80:1–80:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ICALP.2020.80.
- 38 Anand Louis, Prasad Raghavendra, and Santosh Vempala. The complexity of approximating vertex expansion. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 360–369. IEEE, 2013.
- 39 Carsten Lund and Mihalis Yannakakis. The approximation of maximum subgraph problems. In Andrzej Lingas, Rolf G. Karlsson, and Svante Carlsson, editors, *Automata, Languages and Programming, 20th International Colloquium, ICALP93, Lund, Sweden, July 5-9, 1993, Proceedings*, volume 700 of *Lecture Notes in Computer Science*, pages 40–51. Springer, 1993. doi:10.1007/3-540-56939-1\_60.
- 40 George L. Nemhauser and Leslie E. Trotter Jr. Properties of vertex packing and independence system polyhedra. *Math. Program.*, 6(1):48–61, 1974. doi:10.1007/BF01580222.
- 41 Bruce Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004. doi:10.1016/j.orl.2003.10.009.
- 42 Neil Robertson and Paul D. Seymour. Graph minors. V. excluding a planar graph. *J. Comb. Theory, Ser. B*, 41(1):92–114, 1986. doi:10.1016/0095-8956(86)90030-4.
- 43 Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner’s conjecture. *J. Comb. Theory, Ser. B*, 92(2):325–357, 2004. doi:10.1016/j.jctb.2004.08.001.
- 44 Ignasi Sau, Giannos Stamoulis, and Dimitrios M. Thilikos. An fpt-algorithm for recognizing  $k$ -apices of minor-closed graph classes. 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 95:1–95:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ICALP.2020.95.
- 45 Wouter Cames van Batenburg, Tony Huynh, Gwenaél Joret, and Jean-Florent Raymond. A tight erdős-pósa function for planar minors. 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 1485–1500. SIAM, 2019. doi:10.1137/1.9781611975482.90.

# Truly Asymptotic Lower Bounds for Online Vector Bin Packing

János Balogh ✉

Institute of Informatics, University of Szeged, Hungary

Ilan Reuven Cohen ✉

Faculty of Engineering, Bar-Ilan University, Ramat-Gan, Israel

Leah Epstein ✉

Department of Mathematics, University of Haifa, Israel

Asaf Levin ✉

Faculty of Industrial Engineering and Management, Technion, Haifa, Israel

---

## Abstract

In this work, we consider online  $d$ -dimensional vector bin packing. It is known that no algorithm can have a competitive ratio of  $o(d/\log^2 d)$  in the absolute sense, although upper bounds for this problem have always been presented in the asymptotic sense. Since variants of bin packing are traditionally studied with respect to the asymptotic measure, and since the two measures are different, we focus on the asymptotic measure and prove new lower bounds of the asymptotic competitive ratio. The existing lower bounds prior to this work were known to be smaller than 3, even for very large  $d$ . Here, we significantly improved on the best known lower bounds of the asymptotic competitive ratio (and as a byproduct, on the absolute competitive ratio) for online vector packing of vectors with  $d \geq 3$  dimensions, for every dimension  $d$ . To obtain these results, we use several different constructions, one of which is an adaptive construction with a lower bound of  $\Omega(\sqrt{d})$ . Our main result is that the lower bound of  $\Omega(d/\log^2 d)$  on the competitive ratio holds also in the asymptotic sense. This result holds also against randomized algorithms, and requires a careful adaptation of constructions for online coloring, rather than simple black-box reductions.

**2012 ACM Subject Classification** Mathematics of computing → Approximation algorithms; Mathematics of computing

**Keywords and phrases** Bin packing, online algorithms, approximation algorithms, vector packing

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.8

**Category** APPROX

**Related Version** *Previous Version:* <https://arxiv.org/abs/2008.00811>

*Previous Version:* <https://arxiv.org/abs/2007.15709>

**Funding** The research of J. Balogh was supported by the project “Extending the activities of the HU-MATHS-IN Hungarian Industrial and Innovation Mathematical Service Network” EFOP-3.6.2-16-2017-00015, and the project “Integrated program for training new generation of scientists in the fields of computer science,” EFOP-3.6.3-VEKOP-16-2017-00002, supported by the European Union and co-funded by the European Social Fund. The research of A. Levin was partially supported by ISF - Israeli Science Foundation grant number 308/18.

**Acknowledgements** The results of this paper are based on the arxiv versions [9, 11]. Part of the work in [11] has been done while Ilan Reuven Cohen was a postdoctoral fellow at CWI Amsterdam and TU Eindhoven, he would like to thank Nikhil Bansal for discussions and suggestions related to fractional coloring and ideas from [27].



© János Balogh, Ilan Reuven Cohen, Leah Epstein, and Asaf Levin;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 8; pp. 8:1–8:18



Leibniz International Proceedings in Informatics

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

## 1 Introduction

We study the classical vector packing problem (VP) [21, 20, 2, 1, 3]. In VP with dimension  $d \geq 2$ , a set of items is given, where every item is a non-zero  $d$ -dimensional vector, whose components are rational numbers in  $[0, 1]$ . This set is to be partitioned into subsets called bins, such that the vector sum of every subset does not exceed 1 in any component. Here, we consider lower bounds on the worst-case performance guarantees of online algorithms for VP. Online algorithms obtain input items one by one, and pack each new item irrevocably before the next item is presented, into an empty (new), or non-empty bin. Such algorithms receive an input as a sequence, while offline algorithms receive an input as a set. An arbitrary optimal offline algorithm, which uses the minimum number of bins for packing the items of the input or instance, is denoted by  $OPT$ . For an input  $L$  and algorithm  $A$ , we let  $A(L)$  denote the number of bins that  $A$  uses to pack  $L$ , also termed the *cost of algorithm  $A$  on input  $L$* . For a randomized algorithm we denote the expected value of the cost of the algorithm  $A$  on input  $L$  as  $\mathbb{E}[A(L)]$ . We also let  $OPT(L)$  denote the number of bins that  $OPT$  uses for a given input  $L$ . The absolute competitive ratio of an algorithm  $A$  is defined as the supremum ratio over the following ratios. These are the ratios for all inputs  $L$ , where the ratio for  $L$  is the ratio between the number of its bins  $A(L)$  and  $OPT(L)$ , the number of the bins of  $OPT$ . The asymptotic competitive ratio is the limit of absolute competitive ratios  $R_K$  when  $K$  tends to infinity, and  $R_K$  takes into account only inputs for which  $OPT$  uses at least  $K$  bins. That is, the asymptotic competitive ratio of  $A$  is:

$$\lim_{K \rightarrow \infty} \sup_{OPT(L) \geq K} \frac{A(L)}{OPT(L)}.$$

For randomized online algorithms the definition for the asymptotic competitive ratio and for the absolute competitive ratio is the same, except that we consider  $\mathbb{E}[A(L)]$  instead of  $A(L)$ . In this paper, we focus mostly on the asymptotic competitive ratio, which is the most natural measure for bin packing algorithms, and we sometimes refer to it by the term *competitive ratio*. When we discuss the absolute competitive ratio, we use this last term explicitly.

Note that VP is defined as a distinct optimization problem for every fixed value of  $d$ . For each such value (which is a dimension), there might be an online algorithm that is the best possible with respect to the absolute competitive ratio (the algorithm whose absolute competitive ratio is minimized), and there might be an online algorithm that is the best possible with respect to the asymptotic competitive ratio. These algorithms may be different. Denote by  $ABS(d)$  the best possible absolute competitive ratio of a deterministic online algorithm for VP with dimension  $d$ , and let  $ASYM(d)$  denote the best possible asymptotic competitive ratio of a deterministic online algorithm for VP with dimension  $d$ . If there are values of  $d$  for which such best possible online algorithms do not exist, we define the corresponding values of  $ABS(d)$  and  $ASYM(d)$  as the infimums of the corresponding ratios, where the infimum is taken over all online algorithms for VP with  $d$  dimensions. Thus, these values are well-defined for all  $d$  even though we currently do not know their values. It is known that these values are mostly linear in  $d$  (see below).

Observe that both  $ABS(d)$  and  $ASYM(d)$  are monotone non-decreasing functions of  $d$ , as we can use the online algorithm with larger dimension in order to pack the lower dimension vectors by simulating a higher dimension input using additional components, which are always set to 0. Furthermore, by the definitions of the asymptotic competitive ratio and the absolute competitive ratio, we conclude that for every  $d$ , we have  $ASYM(d) \leq ABS(d)$ .

In this work, we improve on the known lower bounds of the asymptotic competitive ratio for all fixed values of  $d$  that are at least 3 for online VP. We focus mostly on deterministic

online algorithms, while our main result uses an oblivious adversary and holds also against randomized online algorithm. This also improves the known results for the absolute ratios, that is, we improve upon the state of the art of lower bounding  $\text{ASYM}(d)$  for all  $d \geq 3$ . Our main result is that the order of growth of the asymptotic competitive ratio is  $\Omega\left(\frac{d}{(\log d)^2}\right)$ . Recall that the term asymptotic here, does not refer to the asymptotic growth of  $d$ , but to the asymptotic definition of the competitive ratio, which is the most common measure for bin packing problems. To do that, we present four constructions leading to different lower bounds on  $\text{ASYM}(d)$ . For every specific value of  $d$ , one may use the construction that leads to the largest possible lower bound. In Section 2, we present our results for very large values of  $d$ . That is, we show that for  $d > 16$ , there is a lower bound of  $\frac{d-1}{8(\log_2 d)^3}$  on  $\text{ASYM}(d)$ , and for sufficiently large and fixed values of  $d$ , the lower bound determined for the asymptotic competitive ratio for online VP is  $\frac{d}{2^{11} \cdot (\log_2 d)^2}$ . The lower bounds presented in Section 2 are based on an oblivious adversary. An elaborate explanation for using values of  $d$  that are sufficiently large is also provided in Section 2.

Next, we present the improved lower bounds for relatively small values of  $d$  against deterministic online algorithms. In Section 3, we show a lower bound of at least  $\frac{\lfloor \sqrt{d-1} \rfloor + 2}{2}$  that applies for all fixed values of  $d \geq 2$ . For example, in the case  $d = 14$  we obtain a lower bound of 2.5 on the asymptotic competitive ratio. Then, in Section 4 we show a lower bound of  $\frac{9}{4} = 2.25$  on  $\text{ASYM}(3)$ . Finally, in Section 5, we introduce a lower bound of  $\frac{76}{29} \approx 2.62$  on  $\text{ASYM}(8)$ . The last three constructions, presented in Sections 3, 4, and 5, are based on a technique called adaptive constructions for packing problems, explained in detail below. We note that our lower bound of  $\frac{\lfloor \sqrt{d-1} \rfloor + 2}{2}$  is the largest lower bound that we establish for a large variety of values of  $d$ . This holds even for extremely large values of  $d$  like  $d = 2^{30}$ , for which the value of this lower bound is at least  $2^{14} > 16000$ . Comparing this result to the values of lower bounds presented in Section 2, we find that they are much smaller, and in fact the values resulting from the constructions of that section are not larger than 4972 and 583. Thus, for any reasonable value of  $d$ , the lower bounds obtained based on the adaptive construction method are much better than the ones in Section 2.

The lower bounds on  $\text{ABS}(d)$  established in Azar et al. [2] were not computed explicitly, in the sense that they are only stated as  $\Omega(d^{1-\varepsilon})$  for every  $\varepsilon > 0$ . These lower bounds hold only in the absolute sense, and their order of growth is  $\Omega\left(\frac{d}{(\log_2 d)^2}\right)$ . In order to compare their bounds to our lower bounds on  $\text{ASYM}(d)$ , we examined their proofs. Their lower bounds are basically the deterministic lower bounds on node coloring of graphs where the nodes arrive in an online fashion, thus they are using black-box reductions from the (deterministic) lower bounds of Halldórsson and Szegedy [22]. These black-box reductions are the main reason that lower bounds cannot be obtained on the asymptotic competitive ratio, but only on the absolute competitive ratio, which may be larger (and it is not the standard tool to analyze bin packing problems). Note that we improve the known absolute lower bound for a large number of values of  $d$ . For example, the deterministic lower bound resulting from the construction of [22] for  $d = 60$  is 3.75, while our results imply a lower bound above 4.8 (which holds even in the asymptotic case, and for a slightly lower dimension of 57).

In the work of [22], the part of randomized lower bounds consists of two constructions. In the first one, the value of  $d$  is relatively large, thus we can use Stirling's formula to approximate  $\lceil \log_2 d \rceil!$  with a constant multiplicative error, while the second one holds for all dimensions  $d$ . In Section 2, we use the ideas of Halldórsson and Szegedy [22] to obtain similar lower bounds on the asymptotic competitive ratio of VP. Unlike the work of [2], we do not apply black-box reduction from node coloring. The main motivation for that is that we need a lower bound with respect to a different coloring problem in which we are interested

in *fractional coloring*. Since this variant was not studied before, we cannot use an existing result for it, nor can we use a black-box reduction from it. Instead, we present the lower bound construction for VP. The transition to this fractional coloring is the main tool that we use in order to lift the construction of [2], so that it will allow us to prove a lower bound on the asymptotic competitive ratio for VP. We refer to [11] for a different approach that is based on a black-box reduction from fractional coloring to randomized coloring together with a black-box reduction from the randomized lower bound of [22]. This alternative approach of [11] leads to inferior lower bounds, therefore, we base our result on the approach of [9] instead.

Next, we survey the literature for the bounds of  $\text{ASYM}(d)$ . The special case of  $d = 1$  of VP is simply the bin packing problem (BP), which has been studied for half a century [23, 24]. The current best bounds on  $\text{ASYM}(1)$  (for the online version) are a lower bound of 1.54278 [7] and an upper bound of 1.57829 [5], while  $\text{ABS}(1) = \frac{5}{3}$  [8]. In [4, 6], the authors considered a generalization of BP, known as the Cardinality Constrained Bin Packing (CCBP). CCBP is defined as follows; each item has a scalar size in  $[0, 1]$  (one-dimensional), and in addition, the constraint that the sum of items sizes in each bin does not exceed 1, there is an integer parameter  $k \geq 2$ , such that no bin can have more than  $k$  items. Note that CCBP is a special case of VP with  $d \geq 2$ , by defining a vector for each item, where its first component is the item's size and its second component is  $\frac{1}{k}$  (the other components are zeros). In [6], the authors consider the case of VP with  $d = 2$ , and present its difference from CCBP. They demonstrated that  $\text{ASYM}(2) \geq 2.03731129$  with respect to VP in two dimensions, and thus for all  $d \geq 2$  it was established that  $\text{ASYM}(d) \geq 2.03731129$ , whereas for CCBP there is a 2-competitive algorithm. In [3] it is shown that when  $d$  grows to infinity there is a lower bound that tends to  $e$ , thus  $\lim_{d \rightarrow \infty} \text{ASYM}(d) \geq e$ . It is interesting to note that the lower bound in [3] applies even if all components of all vectors are small. That is, for every small  $\varepsilon > 0$ , where all components are smaller than  $\varepsilon$ , the lower bound of [3] holds, and it holds even when  $\varepsilon$  tends to 0. This matches the upper bound for the very specific case of VP (of small components) established by [1]. Prior to the publications of [3, 6], the lower bounds on  $\text{ASYM}(d)$  were weaker [20, 14, 13].

As for upper bounds on  $\text{ASYM}(d)$ , the best known result [21] is (still) that the First-Fit algorithm has an asymptotic competitive ratio of  $d + 0.7$  for VP in  $d$  dimensions. Prior to that work, there was a slightly weaker bound of  $d + 1$  established by Kou and Markowsky [26]. For the absolute competitive ratio, the resulting bound is also  $d + O(1)$  [21, 26] (an upper bound of  $O(d)$  follows from the simple property that for greedy algorithms, no two bins of the output have a sum of at most 1 in all components). With respect to the literature on VP regarding oblivious adversary, all known upper bounds are using deterministic algorithms, while the lower bound of [3] is the unique lower bound larger than 2 that holds also using oblivious adversary (by applying standard adaptation). The lower bound of [6] is stated only for deterministic algorithms, and the means for adjusting it for use of an oblivious adversary are unknown.

We stress that prior to our work it was unknown whether there is an online algorithm  $A$  for VP for any dimension  $d$  (or for sufficiently large dimensions) such that for every input  $L$ , its cost  $A(L)$  satisfies  $A(L) \leq 3 \cdot \text{OPT}(L) + d$  (where  $d$  is the dimension of the input). This is due to the fact that the known lower bound of  $e$  holds for any additive term, while the non-constant known lower bound uses only  $d$  items for dimension  $d$ .

Vast literature is available on the offline versions of all the problems mentioned above [16, 10, 12, 15, 18, 19, 25, 28]. In particular, similar lower bounds to those of [1] on the absolute approximation ratio for VP (under a certain standard complexity assumption, that



$\text{NP} \neq \text{ZPP}$ ) were observed for the offline scenario [16, 12], also by reductions from coloring problems. The lower bounds are for the absolute measures, since for every dimension  $d$ , the number of items is simply  $d$ . The other bin packing problems discussed here (BP and CCBP) admit polynomial time asymptotic approximation schemes [15, 18, 19, 25], but one can show that unless  $\text{P} = \text{NP}$ , such schemes cannot exist for the absolute cases, and the absolute approximation ratio is at least 1.5. This can serve as evidence that lower bounds for the absolute approximation ratio or the absolute competitive ratio are not sufficient for the analysis of the asymptotic approximation ratio or the asymptotic competitive ratio. In fact, it is true also for online bin packing problems that the absolute measure is different from the asymptotic one. For example, for BP, the best possible absolute competitive ratio is  $\frac{5}{3}$  [8], while the asymptotic competitive ratio is significantly smaller [5]. For CCBP, one example is the parameter  $k = 4$ , for which the best possible absolute competitive ratio is 2 [4, 6], while an improved asymptotic competitive ratio below 2 is known [17].

Omitted proofs will appear in the full version of this work.

## 2 The lower bound for large values of $d$

In this section we consider the cases of very large dimensions. We will prove a lower bound of  $\Omega(\frac{d}{\log^2 d})$  for sufficiently large dimensions  $d$ . Our lower bounds are not smaller than the weaker results of Azar et al. [2], which were established only for the absolute competitive ratio measure (as a function of the dimension, for large enough values of  $d$ ). Here, we consider the stronger measure of the asymptotic competitive ratio, and even prove these lower bounds for randomized algorithms.

The input's sequence presented by the adversary is constructed in  $d$  phases, where each phase consists of  $N$  identical items. The construction uses integer parameter  $\nu$  (which will be chosen later as a function of the number of dimensions), and maintains  $\nu$  classes and  $\mathcal{S}$ , a subset of the  $\nu$  class sets,  $\mathcal{S} \subseteq X = 2^{[\nu]} \setminus \emptyset$  where  $[\nu] = \{1, 2, \dots, \nu\}$ , let  $0 < \epsilon \leq \frac{1}{Nd}$ . The adversary examines the online algorithm's expected values, and associates a class and a set for each phase, which will determine the items' phase components. We denote  $c_j \in [\nu]$ , and  $S_j \in \mathcal{S}$  as the class and the set of the phase  $j$ , respectively.

Formally, at the beginning of phase  $j \in [d]$ , the adversary examines the algorithm's expected assignment and chooses a set  $S_j \in \mathcal{S}$ . The adversary presents  $N$  identical items, where each item component is defined as follows: the items have 1 as their  $j$ th component, all components of indexes larger than  $j$  are 0, while a component of index  $i < j$  is  $\epsilon$  if  $c_i \notin S_j$  and 0 otherwise. Note that the previous phases classes  $c_1, \dots, c_{j-1}$  have been already set. Finally, after examining the expected assignment of these vectors the adversary associates a class  $c_j \in S_j$  for the phase.

First, observe that the non zero components' values are either 1 or  $\epsilon \leq \frac{1}{Nd}$ . Since the number of items is  $Nd$ , a subset of the items cannot be assigned to the same bin, only if exists a component for which there is an item with a value of 1 and another item with a non-zero value. In addition, a construction of this form satisfies that any solution has a cost of at least  $N$  (and of at most  $Nd$ ), regardless of its specific details. Therefore, by letting  $N$  grow to infinity by proving a lower bound for these instances a lower bound on the asymptotic competitive ratio follows. Next, consider a bin  $B$  with a fixed realization of the random bits of the online algorithm (at some point during the lower bound construction). We associate the subset  $S(B) \subseteq [\nu]$  of classes with every such bin, where  $S(B)$  contains the classes of items that are packed into  $B$ , that is,  $\ell \in S(B)$  if there exists at least one phase  $j$  such that  $\ell = c_j$  and an item of phase  $j$  is packed into  $B$ . The set  $S(B)$ , named the associated set of

## 8:6 Lower Bounds for Online Vector Bin Packing

$B$  may be extended later. Intuitively, the adversary chooses a set  $S_j$  and a class  $c_j$  in each phase, in order to increase the (expected) cardinality of the associated set ( $|S(B)|$ ) of the online algorithm bins. Then, by using a corresponding potential function, we will obtain the lower bound of the expected number of bins of the online algorithm. Full details of the adversary's choices are presented later. First, we introduce several properties, which hold for any choice of adversary. The following claim characterizes the possible associated set that an algorithm may pack the items of phase  $j$ .

▷ **Claim 1.** Any algorithm may pack an item of phase  $j$  in bin  $B$ , if and only if it does not contain another item of phase  $j$ , and before the item is added it holds that  $S(B) \subseteq S_j$ .

*Proof.* Two items of the phase  $j$  cannot fit into a common bin, since there is 1 in the  $j$ 'th component of the items. The number of items is  $N \cdot d$ , hence a collection of items fit into a bin, if and only if, no item in the collection has an  $\varepsilon$  component when another item of the collection has a value of 1 at the same component.

Given a bin  $B$ , for any  $\ell \in S(B)$  there exists at least one value  $j'$ , such that an item of phase  $j'$  is assigned to  $B$ , and the class of phase  $j'$  is  $\ell$ . By the construction definition, the  $j'$  component of this item (and the bin  $B$ ) is 1, and if  $\ell \notin S_j$  the  $j'$ th component of items of phase  $j$  is  $\varepsilon$ . Hence, if item of phase  $j$  can be assigned to bin  $B$  then  $\ell \in S_j$ . On the other hand, if  $S(B) \subseteq S_j$  then for any component  $i < j$  of  $B$  which is 1, we have  $c_i \in S(B) \subseteq S_j$ , and by definition the  $j$ 'th phase vector value at component  $i$  is 0. ◁

We observe that there exists a solution which uses at most  $\nu \cdot N$  bins.

▷ **Claim 2.** For any choice of  $c_j, S_j$  there exists an offline solution which uses at most  $\nu \cdot N$  bins.

*Proof.* A solution that opens  $N$  bins for each class and assigns the phase's vector according to the class is a feasible assignment by Claim 1, since for a bin we use to pack in phase  $j$  we have  $S(B) \subseteq \{c_j\}$  and  $c_j \in S_j$  by the algorithm's definition. ◁

Next, we will present an important characterization of the set  $\mathcal{S}$  necessary for the analysis. The set  $\mathcal{S}$  in the construction will depend on a parameters pair  $(\alpha, \beta)$ , and will require that  $(\alpha, \beta)$  would be *satisfiable*.

► **Definition 3.** Given  $\nu$ ,  $(\alpha, \beta)$  is *satisfiable* if there exists a set of subsets  $\mathcal{S} \subseteq X$  such that  $|\mathcal{S}| \geq \alpha$ , and for every pair  $S, S' \in \mathcal{S}$  such that  $S \neq S'$  we have that their symmetric difference  $S \Delta S'$  satisfies  $|S \Delta S'| \geq \beta$ .

The next claim characterizes two satisfiable pairs, which will be used in our lower bound construction.

▷ **Claim 4.** For any  $\nu$ ,  $(\alpha, \beta) = (2^\nu - 1, 1)$  is satisfiable, and if  $\nu$  is sufficiently large then  $(\alpha, \beta) = (2^{\nu/4}, 0.3\nu)$  is satisfiable.

*Proof.* For the first part, consider  $\mathcal{S}$  to be the set  $X$ , which has  $2^\nu - 1$  elements, as required. Each pair of distinct elements represents non-equal subsets of  $[\nu]$ , so that they differ by at least one element. We fix the value of  $\beta$  to 1, in this case.

The second part was proven by [22], which demonstrated that if we pick a random sub-collection of subsets of  $[\nu]$  with  $2^{\lceil \nu/4 \rceil}$  subsets, each consisting of exactly  $\lceil \nu/2 \rceil$  elements of  $[\nu]$  (chosen independently and uniformly at random), then with some positive probability (for large enough value of  $\nu$ ) each pair of these selected subsets satisfies the condition on their symmetric difference. Using the probabilistic method, they were able to prove our claim (deterministically) for large enough values of  $\nu$  (that they have not specified). ◁

For a specific satisfiable pair  $(\alpha, \beta)$  and their corresponding set  $\mathcal{S}$ , and for any associated set  $S(B) \subseteq [\nu]$ , we denote a subset in  $S \in \mathcal{S}$  that *represents*  $B$ , (where  $S$  is represented by  $B$ ) if  $|S \Delta S(B)| \leq \frac{\beta}{5}$ . We will prove that any set is represented by at most a single set in  $\mathcal{S}$ .

▷ **Claim 5.** If for every pair  $S, S' \in \mathcal{S}$  such that  $S \neq S'$  we have  $|S \Delta S'| \geq \beta$  and  $\beta \geq 1$ , then any bin  $B$  is represented by at most one set  $S \in \mathcal{S}$ .

*Proof.* Assume for contradiction that a bin  $B$  is represented by two sets  $S_1, S_2 \in \mathcal{S}$ . Recall that  $|S_1 \Delta S_2| \geq \beta$  (by assumption), but  $|S(B) \Delta S_i| \leq \frac{\beta}{5}$  for  $i = 1, 2$  (by the definition of representation). First, consider the elements of  $S_1 \setminus S_2$ . Some of these elements belong to  $S(B)$ , while other do not. Observe that  $(S_1 \setminus S_2) \cap S(B) \subseteq S(B) \Delta S_2$  so  $|(S_1 \setminus S_2) \cap S(B)| \leq \frac{\beta}{5}$ . Since  $(S_1 \setminus S_2) \setminus S(B) \subseteq S(B) \Delta S_1$ , we conclude that  $|(S_1 \setminus S_2) \setminus S(B)| \leq \frac{\beta}{5}$ . Therefore,  $|S_1 \setminus S_2| \leq \frac{2\beta}{5}$ . Similarly (by changing the roles of  $S_1$  and  $S_2$ ) we conclude that  $|S_2 \setminus S_1| \leq \frac{2\beta}{5}$ . Thus,  $|S_1 \Delta S_2| \leq \frac{4\beta}{5}$ , contradicting our assumption about  $\mathcal{S}$ . ◁

We are ready to prove our main lemma, which achieves a lower bound on the expected number of bins opened. We will demonstrate that the adversary may choose a set and a class that will increase the cardinality of the associated sets of the algorithm's bins in every step, which we will capture by using a potential function. Our potential function is the expected value of the sum of  $|S(B)|$  over all bins  $B$  of the algorithm. That is,  $\Phi = \mathbb{E}[\sum_B |S(B)|]$ . Observe that the expected value of the cost of the online algorithm is at most  $\Phi$  and not smaller than  $\frac{\Phi}{\nu}$ , since for any  $B$  it holds that  $|S(B)| \leq \nu$ . Let  $\gamma = \lceil \frac{\beta}{5} \rceil$ , this parameter is chosen to ensure that each associated set of a bin is represented by at most one set  $S \in \mathcal{S}$ .

► **Lemma 6.** For a satisfiable pair  $(\alpha, \beta)$  there exists an adversary strategy, such that the expected number of bins opened by the online algorithm is at least  $\min\{\alpha \cdot N/2, d \cdot \frac{\gamma}{\nu} \cdot \frac{N}{2}\}$ .

*Proof.* We will show that if the expected number of bins before phase  $j$  is less than  $\alpha \cdot N/2$ , then there exists  $S_j, c_j$ , which will increase the potential  $\Phi$  by at least  $\frac{\gamma}{\nu} \cdot \frac{N}{2}$ . The lemma holds since we can perform  $d$  phases, and since the number of bins opened is at least  $\frac{\Phi}{\nu}$ .

Prior to phase  $j$ , we let  $n(S)$  for a set  $S \in \mathcal{S}$  be the expected value of the number of bins (of the online algorithm) that it represents. After the assignment of the  $j$ 'th phase items, we denote by  $\tilde{n}^j(S)$  for a set  $S \in 2^{[\nu]}$  the expected number of bins associated with  $S$  and a  $j$ 'th phase item assigned into it. Note that the adversary may use  $n(S), \tilde{n}^j(S)$  when determining  $S_j, c_j$ , respectively, and that these values do not depend on the realization of the random bits used by the algorithm.

**Determining  $S_j$ .** We obtained  $\sum_{S \in \mathcal{S}} n(S) \leq \alpha \cdot \frac{N}{2}$  through the assumption that the expected number of bins is less than  $\alpha \cdot N/2$ , and since every bin of the algorithm is represented by at most one set. The adversary sets  $S_j \in \mathcal{S}$  such that  $n(S_j)$  is below  $\frac{N}{2}$ , and the existence of  $S_j$  can be guaranteed by the pigeonhole principle.

**Determining  $c_j$ .** The algorithm assigns each item of phase  $j$  to a distinct bin; by linearity of expectation we have,

$$N = \sum_S \tilde{n}^j(S) = \sum_{|S \Delta S_j| < \gamma} \tilde{n}^j(S) + \sum_{|S \Delta S_j| \geq \gamma} \tilde{n}^j(S) \leq \frac{N}{2} + \sum_{|S \Delta S_j| \geq \gamma} \tilde{n}^j(S),$$

where the first inequality is obtained by our choice of  $S_j$ . Therefore, we have

$$N/2 \leq \sum_{|S \Delta S_j| \geq \gamma} \tilde{n}^j(S) \leq \sum_{\ell \in S_j} \frac{\sum_{S: \ell \notin S} \tilde{n}^j(S)}{\gamma},$$

where the second inequality follows, since  $\tilde{n}^j(S) > 0$  only if  $S \subseteq S_j$  by Claim 1, so the same  $S$  will appear at least  $\gamma$  times in the inner summation on the right hand side. The adversary set  $c_j \in S_j$  such that  $\sum_{S:c_j \notin S} \tilde{n}^j(S)$  is above  $\frac{\gamma}{\nu} \cdot \frac{N}{2}$ , the existence of  $c_j$  can be guaranteed by the pigeonhole principle since  $|S_j| \leq \nu$ . Note that the increase in the potential function is exactly  $\sum_{S:c_j \notin S} \tilde{n}^j(S)$ , as required.  $\blacktriangleleft$

## 2.1 Assembling the pieces together

Recall that by Claim 2 the cost of the optimal solution is at most  $\nu \cdot N$ . By Lemma 6 any online algorithm will use at least  $\min\{\alpha \cdot N/2, d \cdot \frac{\gamma}{\nu^2} \cdot \frac{N}{2}\}$  number of bins. This value is maximized if  $d \cdot \frac{\gamma}{\nu^2} \cdot \frac{N}{2} \approx \frac{\alpha N}{2}$ . Thus, we need a method for selecting  $\nu$ , if the dimension  $d$  is given. We will use a value  $\nu$  for which the corresponding pair  $(\alpha, \beta)$  satisfies  $d \geq \frac{\nu^2 \alpha}{\gamma}$  (where  $\gamma$  is determined by  $\beta$ ), and consequently the resulting lower bound would be  $\frac{\alpha}{2\nu}$ .

First, consider the case where  $d$  is relatively small and we use  $(\alpha, \beta) = (2^\nu - 1, 1)$  and thus  $\gamma = 1$ , and  $\nu$  is an integer such that  $\nu^2 \cdot (2^\nu - 1) \leq d$ . It is sufficient to require that  $\nu^2 \cdot 2^\nu \leq d$ , that is satisfied by letting  $\nu = \lfloor \log_2 d - 2 \log_2 \log_2 d \rfloor$ , as for this choice  $\nu^2 2^\nu \leq (\log_2 d)^2 \cdot \frac{d}{(\log_2 d)^2} = d$ . The resulting lower bound is not smaller than

$$\frac{\alpha}{2\nu} \geq \frac{2^{\log_2 d - 2 \log_2 \log_2 d} - 1}{4 \cdot (\log_2 d - 2 \log_2 \log_2 d)} \geq \frac{d - 1}{8(\log_2 d)^3},$$

where the last inequality holds for  $d > 16$ , as for these values of  $d$  we have that  $8 \log_2 \log_2 d < 4 \log_2 d$ .

Next, consider the case where the dimension is higher, and we could use  $(\alpha, \beta) = (2^{\nu/4}, 0.3\nu)$ , and thus  $\gamma = 0.06\nu$ . We pick  $\nu$  as the largest integer such that  $\frac{\alpha \nu^2}{\gamma} \leq d$  that is,  $\alpha \nu = \nu 2^{\nu/4} \leq 0.06 \cdot d$ . Letting  $\nu' = \frac{\nu}{4}$  we will require  $\nu' \cdot 2^{\nu'} \leq 0.015d$ . This condition is satisfied, e.g. for  $\nu' = \lfloor \log_2 d - \log_2 \log_2 d - 7 \rfloor$ , as for this choice of  $\nu'$  we have

$$\nu' \cdot 2^{\nu'} \leq (\log_2 d) \cdot 2^{\log_2 d - \log_2 \log_2 d - 7} = (\log_2 d) \cdot \frac{d}{\log_2 d \cdot 2^7} = \frac{d}{2^7} \leq 0.015d$$

and  $\nu'$  is an integer and thus also  $\nu = 4\nu'$  is integer. The resulting lower bound is

$$\frac{\alpha}{2\nu} = \frac{2^{\nu'}}{8\nu'} \geq \frac{2^{\log_2 d - \log_2 \log_2 d - 8}}{8 \cdot (\log_2 d - \log_2 \log_2 d - 7)} \geq \frac{\frac{d}{\log_2 d}}{2^{11} \cdot \log_2 d} = \frac{d}{2^{11} \cdot (\log_2 d)^2},$$

that holds for large enough values of  $d$ .

Thus, we conclude the construction of this section by the following theorem.

► **Theorem 7.** *For every fixed dimension  $d > 16$ , there is a lower bound of*

$$\frac{d - 1}{8(\log_2 d)^3},$$

*on the asymptotic competitive ratio of online randomized algorithms for VP. For sufficiently large and fixed values of  $d$  the lower bound on the asymptotic competitive ratio is*

$$\frac{d}{2^{11} \cdot (\log_2 d)^2}.$$

### 3 A lower bound for medium sized dimensions

Let  $\alpha, \beta \geq 2$  be the two positive integers such that  $d \geq 2 + \alpha(\beta - 2)$ . Next we show a lower bound of  $\frac{\alpha \cdot \beta}{\alpha + \beta - 2}$  for the corresponding special case. Note that choosing the values  $\alpha = \beta$  results in a lower bound of  $\Omega(\sqrt{d})$  so for very large dimension this result is inferior to the general lower bound we considered earlier. However, the hidden constants in the  $\Omega$  notation are smaller for the current construction leading to better lower bounds for medium sized dimensions.

Let  $N > d$  be a large integer such that  $\frac{N}{\alpha}$  is an integer. Our sequence of items may have up to  $N^3$  items, and we let  $\varepsilon < \frac{1}{N^3}$ . We will use the adaptive construction method to generate a sequence of scalar values where  $a_i$  is the value associated with the  $i$ th item, such that all values are smaller than  $\varepsilon$ , and furthermore the following condition holds. If an item is large, then its value is at least  $N^3$  times larger than the value of a small item. The logical condition that we will use to define small and large items is that a  $d$ -dimensional item is large if it is packed into an empty bin and otherwise it is small. We stress the property that every item of the construction will have a one-dimensional associated value, and we will explain how this value is used in the definition of the  $d$ -dimensional item.

During the adaptive construction, after packing the current item, there will be a value  $\mu < \frac{1}{N} < \frac{1}{2}$ . The value of  $\mu$  may decrease (but cannot increase) after assigning an item. The value  $\mu$  will satisfy the property that the value of every large item that appeared up to (and including) the current iteration is strictly larger than  $\mu$  while for any subset of items  $S$  where  $S$  contains only small items that appear in the instance (both during the prefix and later on) or large items that appear later on in the input sequence, the total value of  $S$  is strictly smaller than  $\mu$ . This is obtained by letting  $\mu$  be the current upper bound on values of items that can still be either small or large at termination, and reducing the length of the interval of possible values in the adaptive construction by a multiplicative factor of  $N^3$  after each item. This is done by using the scheme of [6], and setting the predefined constant mentioned earlier ( $k$ ) to  $N^3$ , this ensure that all such subsets  $S$ , whose number of elements will be less than  $N^3$  (as this will be a valid upper bound on the number of items in the entire construction), will satisfy the requirement. Note that, in the construction all the successive items to a large item must be at least  $N^3$  smaller than this large item since those items could be eventually small.

Our construction will have  $\beta$  phases, and it will be useful to denote by  $\mu_i$  the current value of  $\mu$  at the end of phase  $i$  (i.e., after packing the last item of phase  $i$  and modifying the current interval according to the rules of the adaptive construction). Furthermore, phase  $i$  uses the value  $\mu_{i-1}$ .

The first and last phase have special properties while the intermediate phases ( $\beta - 2$  phases) are all similar. Each phase lasts until the first point in time in which the algorithm has opened  $N$  new bins during this phase. Thus, we will ensure that the total cost of the algorithm is  $N \cdot \beta$ . We also maintain an integer value  $\pi$  denoting the *current component* that is being dealt with, and it has a special role in the construction. We will show later that  $\pi$  will always be an index of a component (i.e.,  $\pi \leq d$ ), even though it is increased frequently. The value  $\pi$  is an index of a component such that items have a very big (and close to 1) component of this index. By increasing  $\pi$ , we change (and increase) the position of this very big component in the construction. This value is initially set as  $\pi = 2$  (while the very first component has a special role during the first phase), and it increases gradually, each time by 1, and it never decreases, as items are being presented. We will see that the value of  $\pi$  never exceeds  $d$ , and during the presentation of items of intermediate phases it will hold that  $\pi \leq d - 1$ .

### The first phase

We construct a sequence of items, where the first component of every vector is  $\frac{1}{N}$  while every other component equals to the value of the current item. Note that this phase ends after at most  $N^2$  items, since any phase ends after the algorithm used  $N$  new bins, every new bin can contain at most  $N$  items of this phase, and there are no bins of previous phases which can be used.

### The $\beta - 2$ intermediate phases

Every such phase will contain at most  $\alpha \cdot N$  items. We keep a counter  $j$  of the index of the phase, where  $j$  is initialized to 2. At the end of phase  $j - 1 \geq 1$  we set  $\mu_{j-1}$  as we described above and we start presenting new items of the  $j$ th phase.

Each item of these  $\beta - 2$  intermediate phases will consist of the following components. All components with indexes smaller than  $\pi$  are equal to 0, the current component with index  $\pi$  is set to  $1 - \mu_{j-1} > \frac{1}{2}$ , and all other components (of larger indexes) are equal to the value of the current item (of the adaptive construction). Recall that a new phase starts whenever the number of new bins during the current phase is  $N$ , and just before starting a new phase we also increase  $\pi$  by 1 (for  $j = 2$ ,  $\pi$  is not increased but it is initialized). However, there are other events where we decide to increase the value of  $\pi$  by 1. These additional events are stated as follows. Whenever the number of large items (according to the adaptive construction) that were packed while the value of  $\pi$  is its current value, is  $\frac{N}{\alpha}$ , we increase the value of  $\pi$  by 1. This is done since the number of large items whose  $\pi$ th component is very big is the maximum possible number. We will show that an increase in the value of  $\pi$  will happen after at most  $N$  consecutive items for which we used the same value of  $\pi$ . This happens either due to the latter rule or due to the end of the phase (since  $\pi$  is always increased due to that event). Before presenting the vectors of the last phase, we prove the main correctness claims regarding the intermediate phases that allow our construction to have the required structure and allow us to prove the claimed lower bound on the asymptotic competitive ratio.

► **Lemma 8.** *The items of phase  $j$  (where  $2 \leq j \leq \beta - 1$ ) cannot be packed into bins that were opened in an earlier phase, that is, bins used first for an item of an earlier phase. Additionally, the value of  $\pi$  remains constant without being increased for at most  $N$  items.*

► **Lemma 9.** *During an intermediate phase, the value of  $\pi$  is increased at most  $\alpha$  times by 1 (including the increase due to the end of the phase).*

We consider the value of  $\pi$  at the beginning of the last phase. By the last lemma, we conclude that just before the moment when phase  $\beta - 1$  ends (the last intermediate phase), we have  $\pi \leq 2 + (\beta - 2) \cdot \alpha + (\beta - 1) \leq d - 1$  and  $\pi$  is increased to a value of at most  $d$  once that phase ends. This final value (of at most  $d$ ) for  $\pi$  was not used as the value of  $\pi$  in the definition of items of any phase (after the very last time that  $\pi$  was increased, no items are defined so it was not used to define an item).

### The last phase

In the last phase we present exactly  $N$  identical items that are defined as follows. In component  $d$  they are equal to  $1 - \mu_{\beta-1}$  and all other components are 0. Observe that every bin that the algorithm has opened in one of the earlier phases has one large item whose  $d$ th component is larger than  $\mu_{\beta-1}$ , and thus the algorithm needs to open  $N$  new bins for these  $N$  items of the last phase.

### Proving the resulting lower bound

Since there are  $\beta$  phases and the algorithm is forced to open  $N$  new bins in every phase, we conclude that the cost of the algorithm is exactly  $N \cdot \beta$ . Since  $N$  could be an arbitrary large integer, in order to prove the lower bound on the asymptotic competitive ratio of the algorithm, it suffices to show that the optimal offline cost is at most  $N \cdot (1 + \frac{\beta-2}{\alpha}) + 1$ . In order to present this proof, we will consider all items of the last phase as small items. We present an offline solution of cost at most  $N \cdot (1 + \frac{\beta-2}{\alpha}) + 1$ . This offline solution will pack all large items into  $N \cdot (\frac{\beta-2}{\alpha}) + 1$  bins, and all small items into a disjoint set of  $N$  bins, and in total we will use at most  $N \cdot (1 + \frac{\beta-2}{\alpha}) + 1$  bins. First, we consider the large items.

► **Lemma 10.** *There is an offline solution that packs all large items into at most  $N \cdot (\frac{\beta-2}{\alpha}) + 1$  bins.*

Next, we consider packing of the small items into  $N$  bins.

► **Lemma 11.** *There is an offline solution that packs all small items into  $N$  bins.*

Thus, we conclude the following result.

► **Theorem 12.** *If  $d \geq \alpha(\beta - 2) + 2$ , then there is no online algorithm for VP whose asymptotic competitive ratio is smaller than  $\frac{\beta}{1 + \frac{\beta-2}{\alpha}} = \frac{\alpha \cdot \beta}{\alpha + \beta - 2}$ .*

For large values of  $d$ , we can use  $\alpha = \beta = \lfloor \sqrt{d-1} \rfloor + 1 \geq \lceil \sqrt{d-1} \rceil$  for which  $\alpha(\beta-2)+2 \leq (\sqrt{d-1} + 1) \cdot (\sqrt{d-1} - 1) + 2 = d - 1 - 1 + 2 = d$  and this lower bound on the asymptotic competitive ratio is  $\frac{\alpha \cdot \beta}{\alpha + \beta - 2} = \frac{\alpha^2}{2\alpha - 2} = \frac{\alpha + 1}{2} + \frac{1}{2(\alpha - 1)} > \frac{\lfloor \sqrt{d-1} \rfloor + 1}{2} + 1 \geq \frac{\sqrt{d-1} + 1}{2} > \frac{\sqrt{d}}{2}$ . However, for small dimensions we could do better. For example for  $d = 98$ , we could pick  $\alpha = 12, \beta = 10$  and the lower bound on the asymptotic competitive ratio is  $\frac{120}{20} = 6$  whereas using  $\alpha = \beta = 10$  the lower bound is  $\frac{100}{18} \approx 5.555$ .

For small dimensions, namely  $d = 6, 7, 9, 10$  and  $11$ , the next estimation can be used. Letting  $\beta = 3$  and  $\alpha = d - 2$ , the lower bound is greater or equal to  $\frac{3\alpha}{\alpha + 1} = \frac{3}{1 + \frac{1}{\alpha}} = \frac{3}{1 + \frac{1}{d-2}}$ . It gives a lower bound of 2.4, 2.5, 2.625, 2.666, and 2.7 for the cases of  $d = 6, 7, 9, 10, 11$ , respectively. We mention several other small values of  $d$ . For  $d = 12$ , we can use  $\alpha = 5$  and  $\beta = 4$  to obtain a lower bound of  $\frac{20}{7} \approx 2.857$ . For  $d = 14$ , we can use  $\alpha = 6$  and  $\beta = 4$  to obtain a lower bound of 3. For  $d = 16$ , we can use  $\alpha = 7$  and  $\beta = 4$  to obtain a lower bound of  $\frac{28}{9} \approx 3.111$ . Improved bounds for the cases  $d = 3, 4, 5, 8$  are presented in the next sections.

## 4 The case $d = 3$

Recall that the known lower bound on the asymptotic competitive ratio for  $d = 2$  is just slightly above 2, and this was the best known constant lower bound for any small value of  $d$  till now. We prove here a lower bound of 2.25 for the case  $d = 3$ , and explain how it can be slightly improved.

We will use an adaptive construction as explained earlier. The construction is based on that of [6].

Let  $K > 1000$  be a large integer, and let  $\varepsilon > 0$  be a small constant (in particular,  $\varepsilon < \frac{1}{K} < 0.001$ ). The input consists of three parts and we describe the parts one by one.

### The first part of the input

Using an adaptive construction of values, we define a sequence of values in  $(0, \varepsilon)$  such that any large value is strictly larger by a multiplicative factor larger than  $10K$  from any small value. The binary condition is that the item is packed into an empty bin by the online algorithm.

## 8:12 Lower Bounds for Online Vector Bin Packing

Thus, an item packed into an empty bin is large, and otherwise the item is small. The number of items will be  $2 \cdot K \cdot N$  for a large integer  $N > 0$ . Letting  $a_1, a_2, \dots, a_{2KN}$  be the sequence of values, the vector for item  $i$  is defined as follows. The first component is  $\frac{1}{K}$ , and each one of the two other components is equal to  $a_i$ . Let  $\gamma$  be a threshold such that if the  $i$ th constructed value is small, it holds that  $0 < a_i < \frac{\gamma}{10K}$  and otherwise  $\gamma < a_i < \varepsilon$ . The input up to this point is denoted by  $I_0$ .

► **Lemma 13.** *The optimal cost for packing  $I_0$  is  $2N$ .*

Let  $X$  denote the number of bins used by the algorithm for the first part of the input and by definition this is also the number of large items. Let  $\mu = \frac{\gamma}{10}$ . Thus, every  $K$  small values have total value below  $\mu$ .

### The second part of the input

For the value of  $\mu$  that is based on the action of the algorithm, we define the next part of the input. There are  $N$  items of each one of the two types:  $(0, 1 - 4 \cdot \mu, \mu)$  and  $(0, \mu, 1 - 4 \cdot \mu)$ . So, in total there are  $2N$  items of these types. The input at this time is called  $I_1$ , i.e.  $I_1$  is the input consisting of the first two parts of the input together.

### Two offline packings of $I_1$

We define two offline packings, for which the first part of the input is packed in a fixed manner. For each possibility of the third part of the input, we will use one of those offline packings that we present here. Consider the first part of the input (the items of  $I_0$ ), and separate small items from large items. Large items are packed such that every bin has  $K$  of them (where one such bin may have a smaller number of these items). The large items require  $\lceil \frac{X}{K} \rceil$  bins. These are feasible bins because none of the components of the sum of any bin is above 1, since no component of any item of the first part is above  $\frac{1}{K}$ . Small items are also packed  $K$  in each bin, and there are at most  $2N$  such bins since the total number of items for the first part is  $2KN$ . The total number of small items is  $2KN - X$ . The first component of the bin has load 1, but the other components have loads below  $\mu$ . Every such bin can also receive one item of each type of the second part. In one packing, we partition the items of the second part into pairs where every pair consists of items of different types. In this offline packing each pair is packed together, these pairs are first packed into bins with  $K$  small items of the first part, and if there are any unpacked items of the second part, they are packed into new bins (also in pairs). In the second packing, every bin gets just one such item of the second part. For both offline packings, the numbers of bins do not exceed  $\frac{X}{K} + 1 + 2N$ .

### The third part of the input

The third part of the input may contain two alternative sets of items. In the first case, leading to the input  $I_{21}$ , there are  $N$  items of the type  $(0, 1 - \mu, 1 - \mu)$ . Every such item is packed into a different bin by any algorithm. In the offline packing, these items are packed first into bins with (at most)  $K$  small items of the first part (but without large items of the first part and without any items of the second part) and then into new bins. The online algorithm cannot combine such items with any item into the same bin, as we will see.

In the second case, leading to the input  $I_{22}$ , there are  $N$  items of each of the types:  $(0, 3\mu, 1 - 2\mu)$ ,  $(0, 1 - 2\mu, 3\mu)$ . No pair of such items can be packed into one bin, but it can



join a bin with (at most)  $K$  small items of the first part and one item of the second part (of the suitable type) but without large items of the first part. It also cannot be packed with an item of the other type of the second part.

This concludes the description of the input construction. Next, we turn our attention to proving the resulting lower bound on the asymptotic competitive ratio for the case  $d = 3$ .

## Proving the resulting lower bound

Here, we prove the following result.

► **Theorem 14.** *There is no online algorithm for the case  $d = 3$  whose asymptotic competitive ratio is smaller than  $\frac{9}{4}$ .*

A very slight improvement over the lower bound which we proved above in Theorem 14 can be obtained as follows, similarly to the known construction for  $d = 2$  [6]. An alternative second part of the input will contain items whose first component is not zero but some multiple of  $\frac{1}{K}$ . The second component will be slightly larger than  $\frac{1}{3}$ , where there will be large items and small items (small items are those that are packed by the algorithm into a bin that cannot receive another item), and all these items have second components larger than  $\frac{1}{3}$ . There may be a third part of the input (in this alternative input), similarly to the construction of [6]. We omit the details as the idea is similar and the improvement is very small. We note that this value of 2.25 is a valid lower bound for the cases  $d \geq 4$  as well. For the case  $d = 5$  we could get the same lower bound by another method in Section 3 as well, where we proved significantly larger values for larger dimensions.

## 5 The case $d = 8$

We consider this special case as well, in order to demonstrate that the asymptotic competitive ratio grows relatively fast with the dimension. We picked the value of  $d = 8$  as for this dimension we are able to exhibit new properties of instances leading to improved lower bounds. Once again the lower bound construction consists of three parts.

### The first part of the input

The first part of the construction is identical to that of the case  $d = 3$ , including the property that the values of  $K$  and  $\varepsilon$  are the same, with the only change that the components equal to  $a_i$  are not just the second and third components, but all components with indexes  $2, 3, \dots, 7$  are equal to  $a_i$ . The first component is still  $\frac{1}{K}$ , while the 8th component is equal to zero. The values  $\gamma$  and  $\mu$  are defined as in the first part of the construction for the case  $d = 3$ .

### The second part of the input

The second part of the input consists of  $6N$  items consisting of six groups each of which has  $N$  vectors, where every  $N$  vectors of a common group are identical. All these vectors have 8th components equal to  $\frac{1}{3}$  and first components equal to zero. The other components are equal to either 0 or to  $1 - 3\mu$ , where every item has exactly one component equal to  $1 - 3\mu$ , and we will call it the large component of the item. We will have the items of group  $j$  (for  $j = 1, 2, \dots, 6$ ) having component  $j + 1$  equal to  $1 - 3\mu$  while all other components (excluding the 8th component) are zero.

### Analyzing the packing of the algorithm at the end of the second part

Before describing the third part of the input, we introduce some notation and properties of the packing of the algorithm at the end of the second part of the input. The items of the second part are defined so that no bin can have more than three such items by the constraint on the 8th component, and all (at most three) items of one bin have distinct large components since  $1 - 3\mu > \frac{1}{2}$ . We will distinguish the cases where a bin contains three, two, or just one item of the second part of the input, introducing notation for their corresponding bin numbers.

For  $j_1, j_2, j_3 \in \{2, 3, 4, 5, 6, 7\}$ , where  $j_1 < j_2 < j_3$ , let  $X_{j_1, j_2, j_3}$  be the number of bins with (exactly) three items of the second part of the input, whose large components are  $j_1, j_2$ , and  $j_3$ . There are 20 such variables. For  $j_4, j_5 \in \{2, 3, 4, 5, 6, 7\}$ , where  $j_4 < j_5$ , let  $Y_{j_4, j_5}$  be the number of bins with (exactly) two items of the second part of the input, whose large components are  $j_4$  and  $j_5$ . There are 15 such variables. For  $j_6 \in \{2, 3, 4, 5, 6, 7\}$ , let  $Z_{j_6}$  be the number of bins with (exactly) one item of the second part of the input, whose large component is  $j_6$ . There are six such variables. Since every bin opened by the algorithm for the first part of the input has a sum of components of items above  $\gamma = 10\mu$  in components  $2, 3, \dots, 7$ , all these bins of the algorithm are new.

The number of bins opened by the algorithm for the first part of the input is denoted by  $Q$  and it satisfies

$$Q \geq 2N .$$

The sum of variables of the form  $X_{j_1, j_2, j_3}$  is denoted by  $X$ , the sum of variables of the form  $Y_{j_4, j_5}$  is denoted by  $Y$ , and the sum of variables of the form  $Z_{j_6}$  is denoted by  $Z$ . That is,  $X = \sum_{j_1, j_2, j_3} X_{j_1, j_2, j_3}$ ,  $Y = \sum_{j_4, j_5} Y_{j_4, j_5}$ , and  $Z = \sum_{j_6} Z_{j_6}$ . By counting the number of items of the second part, we have

$$3X + 2Y + Z = 6N .$$

### The third part of the input

The third part has one of ten possible sets of items, of similar structures. These items have six non-zero components, which are components  $2, 3, \dots, 7$ . Every item has three components whose values are  $2\mu$ , and three components whose values are  $1 - \mu$ . No two such items can be packed into the same bin since the sum of such a component for two items is either  $2\mu + 1 - \mu > 1$  or  $2(1 - \mu) > 1$ . For a triple  $\{2, j_7, j_8\}$  where  $j_7, j_8 \in \{3, 4, 5, 6, 7\}$  are fixed component indexes, and  $j_7 < j_8$ , the input consists of  $N$  items whose components  $2, j_7, j_8$  are equal to  $1 - \mu$  and the components  $\{2, 3, 4, 5, 6, 7\} \setminus \{2, j_7, j_8\}$  are equal to  $2\mu$ , and  $N$  items whose components  $2, j_7, j_8$  are equal to  $2\mu$  and the components  $\{2, 3, 4, 5, 6, 7\} \setminus \{2, j_7, j_8\}$  are equal to  $1 - \mu$ . This completes the construction of the input. Next, we prove the resulting lower bound.

### Proving the resulting lower bound

Recall the decision variables  $X, Y, Z, Q$  whose values are determined by the algorithm but they satisfies the conditions  $Q \geq 2N$  and  $3X + 2Y + Z = 6N$  established above. We first upper bound the optimal offline cost after the third part of the input and then present a lower bound on the maximum cost of the algorithm on these 10 inputs that can be constructed in the third part of the input. As in the proof for  $d = 3$  we can assume  $\frac{Q}{K} \leq 6$ .

► **Lemma 15.** *For each of the ten inputs that can be created at the end of the third part, there is an offline solution whose cost is at most  $2N + 7$ .*

The algorithm can combine into a common bin some items of the third part with items of the second part but it cannot use bins that were used for items of the first part for packing items of the second or third parts. We describe only bins without any items of the first part because any bin of the algorithm containing an item of the first part cannot receive any additional items. We discuss such bins with two or three items of the second part (since bins with just one item can always receive additional items). For a set  $\{2, j_7, j_8\}$ , there may be bins with two items of the second part, where one of the items has a large component in the set  $\{2, j_7, j_8\}$  and the other one has a large component in the set  $\{2, 3, 4, 5, 6, 7\} \setminus \{2, j_7, j_8\}$ . In addition, there may be bins with three items of the second part, where the set of large components is none of the sets  $\{2, j_7, j_8\}$  and  $\{2, 3, 4, 5, 6, 7\} \setminus \{2, j_7, j_8\}$  (comparing them as sets and not as ordered tuples).

By Lemma 15, for large values of  $N$  we find for the asymptotic competitive ratio  $R$  that

$$Q + X + Y + Z \leq 2RN .$$

We introduce two new variables  $Y', X'$  where  $Y'$  is the number of bins of the algorithm with two items of the second part that cannot receive an item of the third part, and  $X'$  is the number of bins of the algorithm with three items of the second part that cannot receive an item of the third part (for the choice of third part, that is, we fix the third part temporarily). Then, by considering this input using the fact that the third part of the input requires packing items into at least  $2N$  bins and we cannot use  $Q + Y' + X'$  of the bins which were opened for the first or second parts, we conclude that

$$Q + Y' + X' + 2N \leq 2RN .$$

We take the sum of the last inequality for all ten options of  $j_7, j_8$ , where the right hand side is  $20RN$ , and the multiplier of  $N$  on the left hand side is  $20N$ . Since we consider all options for the third part of the input, the values  $X'$  and  $Y'$  can have different values, and more precisely, each one has up to ten different values.

We count the multiplier of each variable as follows. Variables of the form  $X_{j_1, j_2, j_3}$  are included in all variables  $X'$  except for the option where  $j_1, j_2, j_3$  are components of equal values, (the algorithm chose exactly the same subset as the one chosen for the third part of the input) which is just one case of the third part. Thus, the multiplier of  $X_{j_1, j_2, j_3}$  is 9. Variables of the form  $Y_{j_4, j_5}$  are included in all variables  $Y'$  except for cases where  $j_4$  and  $j_5$  are components of equal values, which is the case if  $\{j_4, j_5\} \subseteq \{2, j_7, j_8\}$  or  $\{j_4, j_5\} \subseteq \{2, 3, 4, 5, 6, 7\} \setminus \{2, j_7, j_8\}$ . Out of the 15 variables, there are nine such options that are included (in the sense that the bins cannot be used for items of the third part of the input) and six that are not included. Thus, every variable is included in six of the ten partitions, and in this sum of constraints every variable  $Y_{j_4, j_5}$  has a multiplier of 6. We get

$$10Q + 6Y + 9X + 20N \leq 20RN . \tag{1}$$

Using  $3X + 2Y + Z = 6N$  and  $Q + X + Y + Z \leq 2RN$ , we find by subtraction that  $2X + Y - Q \geq 6N - 2RN$  or alternatively

$$9X + 4.5Y - 4.5Q \geq 27N - 9RN .$$

By subtracting the last inequality from (1), we have  $1.5Y + 14.5Q + 47N \leq 29RN$ . Since  $Y \geq 0$  and  $Q \geq 2N$  hold, we establish that  $76N \leq 29RN$  and therefore  $R \geq \frac{76}{29} \approx 2.620689655$  as we summarize in the following theorem.

► **Theorem 16.** *There is no online algorithm for the case  $d = 8$  whose asymptotic competitive ratio is smaller than  $\frac{76}{29} \approx 2.620689655$ .*

---

### References

- 1 Y. Azar, I. R. Cohen, A. Fiat, and A. Roytman. Packing small vectors. In *Proc. of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA'16)*, pages 1511–1525, 2016.
- 2 Y. Azar, I. R. Cohen, S. Kamara, and F. B. Shepherd. Tight bounds for online vector bin packing. In *Proc. of the 45th ACM Symposium on Theory of Computing (STOC'13)*, pages 961–970, 2013.
- 3 Y. Azar, I. R. Cohen, and A. Roytman. Online lower bounds via duality. In *Proc. of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA'17)*, pages 1038–1050, 2017.
- 4 L. Babel, B. Chen, H. Kellerer, and V. Kotov. Algorithms for on-line bin-packing problems with cardinality constraints. *Discrete Applied Mathematics*, 143(1-3):238–251, 2004.
- 5 J. Balogh, J. Békési, Gy. Dósa, L. Epstein, and A. Levin. A new and improved algorithm for online bin packing. In *Proc. of the 26th European Symposium on Algorithms (ESA'18)*, pages 5:1–5:14, 2018.
- 6 J. Balogh, J. Békési, Gy. Dósa, L. Epstein, and A. Levin. Online bin packing with cardinality constraints resolved. *Journal of Computer and System Sciences*, 112:34–49, 2020.
- 7 J. Balogh, J. Békési, Gy. Dósa, L. Epstein, and A. Levin. A new lower bound for classic online bin packing. *Algorithmica*, 83(7):2047–2062, 2021.
- 8 J. Balogh, J. Békési, Gy. Dósa, J. Sgall, and R. van Stee. The optimal absolute ratio for online bin packing. *Journal of Computer and System Sciences*, 102:1–17, 2019.
- 9 J. Balogh, L. Epstein, and A. Levin. Truly asymptotic lower bounds for online vector bin packing. *CoRR*, abs/2008.00811, 2020. [arXiv:2008.00811](https://arxiv.org/abs/2008.00811).
- 10 N. Bansal, A. Caprara, and M. Sviridenko. A new approximation method for set covering problems, with applications to multidimensional bin packing. *SIAM Journal on Computing*, 39(4):1256–1278, 2009.
- 11 N. Bansal and I. R. Cohen. An asymptotic lower bound for online vector bin packing. *CoRR*, abs/2007.15709, 2020. [arXiv:2007.15709](https://arxiv.org/abs/2007.15709).
- 12 N. Bansal, M. Eliás, and A. Khan. Improved approximation for vector bin packing. In *Proc. of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA'16)*, pages 1561–1579, 2016.
- 13 D. Blitz. Lower bounds on the asymptotic worst-case ratios of on-line bin packing algorithms. Technical Report 114682, University of Rotterdam, 1996. M.Sc. thesis.
- 14 D. Blitz, A. van Vliet, and G. J. Woeginger. Lower bounds on the asymptotic worst-case ratio of online bin packing algorithms. Unpublished manuscript, 1996.
- 15 A. Caprara, H. Kellerer, and U. Pferschy. Approximation schemes for ordered vector packing problems. *Naval Research Logistics*, 92:58–69, 2003.
- 16 C. Chekuri and S. Khanna. On multidimensional packing problems. *SIAM Journal on Computing*, 33(4):837–851, 2004.
- 17 L. Epstein. Online bin packing with cardinality constraints. *SIAM Journal on Discrete Mathematics*, 20(4):1015–1030, 2006.
- 18 L. Epstein and A. 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.
- 19 W. Fernandez de la Vega and G. S. Lueker. Bin packing can be solved within  $1 + \varepsilon$  in linear time. *Combinatorica*, 1(4):349–355, 1981.
- 20 G. Galambos, H. Kellerer, and G. J. Woeginger. A lower bound for online vector packing algorithms. *Acta Cybernetica*, 10:23–34, 1994.
- 21 M. R. Garey, R. L. Graham, and D. S. Johnson. Resource constrained scheduling as generalized bin packing. *Journal of Combinatorial Theory Series A*, 21(3):257–298, 1976.

- 22 M. M. Halldórsson and M. Szegedy. Lower bounds for on-line graph coloring. *Theoretical Computer Science*, 130(1):163–174, 1994.
- 23 D. S. Johnson. Fast algorithms for bin packing. *Journal of Computer and System Sciences*, 8:272–314, 1974.
- 24 D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, 3:256–278, 1974.
- 25 N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS'82)*, pages 312–320, 1982.
- 26 L. T. Kou and G. Markowsky. Multidimensional bin packing algorithms. *IBM Journal of Research and Development*, 21(5):443–448, 1977. doi:10.1147/rd.215.0443.
- 27 L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete mathematics*, 13(4):383–390, 1975.
- 28 S. Sandeep. Almost optimal inapproximability of multidimensional packing problems. *CoRR*, abs/2101.02854, 2021. arXiv:2101.02854.

## **A** Adaptive constructions for deterministic algorithms

In addition to the lower bound for a general  $d$ , we present lower bounds that achieve better guarantees for relatively small values of  $d$  versus deterministic algorithms. The lower bounds are based on a method that produces a sequence of values with several important properties, and are produced by an adversary according to the algorithm’s behavior. Specifically, we define inputs using a method presented in the past [6]. In this method, a binary condition on the assignment of every item is defined (e.g., whether the item is assigned into a new bin), and it is used by the adversary (who presents the input) for the definition of the properties of the following item. More precisely, the adversary keeps an active interval of (scalar) values (contained in  $(0, 1)$ ), and it modifies the interval after the assignment of every input item by the algorithm. These values are not necessarily the actual sizes of the input items, even in the one-dimensional case, although item sizes are based on them in a simple pre-specified way. This means that the generated value is not necessarily the size of the new input item, nor will it always be equal to a component of its vector. Nevertheless, this generated value is used in the definition of the new item, for example, it may be subtracted from some fixed value, or added to a fixed size.

The number of required items is decided in advance, or (in some cases) an upper bound on the number of required items is provided in advance in cases where the exact number of required items is revealed later on. This number is used to decide upon the initial interval of the values, as well. The initial interval is also based on the required sizes and properties of the values. The initial interval is always defined such that the smallest size is strictly positive and the largest size is sufficiently small.

Input items are presented one by one. After the assignment of an item by the algorithm, the validity of the condition is tested for that item. During the process of input construction, it is ensured (via a process resembling binary search, or geometric binary search) that values corresponding to items satisfying the binary condition are larger by a pre-specified (constant) multiplicative factor than the value of any item not satisfying the binary condition. In this way, the process determines two regions, as explained below. We will call the resulting ranges of values *large* and *small*, where the two ranges are disjoint. Items with large values, i.e., from the large range, are called large, and items with small values, i.e., from the small range, are called small.

## 8:18 Lower Bounds for Online Vector Bin Packing

Note that when an item is presented, its size is defined without any prior knowledge of the assignment, so it is still unknown at that moment whether the binary condition holds for this item. Thus, its value is defined without any knowledge of its dimensions. That knowledge is gained based on the action of the algorithm once the item is packed. Based on the packing, if its value is required to be large, future values will be much smaller, and if its value is required to be small, future values will be much larger.

The construction allows us to define positive values smaller than a given value  $\varepsilon > 0$ , such that for a pre-defined (constant) multiplicative factor  $k$ , any large value is more than  $k$  times larger than any small value. Thus, there is a value  $\gamma < \varepsilon$  such that every small value is smaller than  $\frac{\gamma}{k}$  and every large value is larger than  $\gamma$ . If items are one dimensional, and their sizes are simply these values, it would imply that, for example, an item of size  $1 - \gamma$  can be packed with  $k$  small items, but cannot be packed with one large item into the same bin. Note that in this case large items are also quite small, although not as small as the small items. It is possible to define items differently in one dimension, i.e., not only in the way that their sizes are equal to the values. One option is to use the values as complements of sizes (to 1). Another option is to use an additive term, for example, items can have sizes of  $\frac{1}{3}$  plus the defined value. In this case, one can define a value of  $\varepsilon$  such that  $\frac{1}{3} + \varepsilon < \frac{1}{2}$ , for example. For items that are vectors, one can define a part of the components to be determined based on the corresponding value. For example, it is possible in the case  $d = 5$  that two components will be equal to the value, while three other components are equal to zero.

# Fine-Grained Completeness for Optimization in P

**Karl Bringmann**

Saarland University, Saarland Informatics Campus, Saarbrücken, Germany

Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

**Alejandro Cassis**

Saarland University, Saarland Informatics Campus, Saarbrücken, Germany

Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

**Nick Fischer**

Saarland University, Saarland Informatics Campus, Saarbrücken, Germany

Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

**Marvin Künnemann**

Institute for Theoretical Studies, ETH Zürich, Switzerland

---

## Abstract

---

We initiate the study of fine-grained completeness theorems for exact and approximate optimization in the polynomial-time regime.

Inspired by the first completeness results for decision problems in P (Gao, Impagliazzo, Kolokolova, Williams, TALG 2019) as well as the classic class MAXSNP and MAXSNP-completeness for NP optimization problems (Papadimitriou, Yannakakis, JCSS 1991), we define polynomial-time analogues MAXSP and MINSP, which contain a number of natural optimization problems in P, including Maximum Inner Product, general forms of nearest neighbor search and optimization variants of the  $k$ -XOR problem. Specifically, we define MAXSP as the class of problems definable as  $\max_{x_1, \dots, x_k} \#\{(y_1, \dots, y_\ell) : \phi(x_1, \dots, x_k, y_1, \dots, y_\ell)\}$ , where  $\phi$  is a quantifier-free first-order property over a given relational structure (with MINSP defined analogously). On  $m$ -sized structures, we can solve each such problem in time  $O(m^{k+\ell-1})$ . Our results are:

- We determine (a sparse variant of) the Maximum/Minimum Inner Product problem as complete under *deterministic* fine-grained reductions: A strongly subquadratic algorithm for Maximum/Minimum Inner Product would beat the baseline running time of  $O(m^{k+\ell-1})$  for all problems in MAXSP/MINSP by a polynomial factor.
- This completeness transfers to approximation: Maximum/Minimum Inner Product is also complete in the sense that a strongly subquadratic  $c$ -approximation would give a  $(c + \varepsilon)$ -approximation for all MAXSP/MINSP problems in time  $O(m^{k+\ell-1-\delta})$ , where  $\varepsilon > 0$  can be chosen arbitrarily small. Combining our completeness with (Chen, Williams, SODA 2019), we obtain the perhaps surprising consequence that refuting the OV Hypothesis is *equivalent* to giving a  $O(1)$ -approximation for all MINSP problems in faster-than- $O(m^{k+\ell-1})$  time.
- By fine-tuning our reductions, we obtain mild algorithmic improvements for solving and approximating all problems in MAXSP and MINSP, using the fastest known algorithms for Maximum/Minimum Inner Product.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Problems, reductions and completeness

**Keywords and phrases** Fine-grained Complexity & Algorithm Design, Completeness, Hardness of Approximation in P, Dimensionality Reductions

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.9

**Category** APPROX

**Related Version** *Full Version:* <https://arxiv.org/abs/2107.01721>

**Funding** *Karl Bringmann, Alejandro Cassis, Nick Fischer:* This work is part of the project TIPEA that has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No. 850979). *Marvin Künnemann:* Research supported by Dr. Max Rössler, by the Walter Haefner Foundation, and by the ETH Zürich Foundation. Part of this research was performed while the author was employed at MPI Informatics.



© Karl Bringmann, Alejandro Cassis, Nick Fischer, and Marvin Künnemann;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 9; pp. 9:1–9:22



Leibniz International Proceedings in Informatics

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

## 1 Introduction

For decades, increasingly strong hardness of approximation techniques have been developed to pinpoint the best approximation guarantees achievable in polynomial time. Among the early successes of the field, we find the MAXSNP completeness theorems by Papadimitriou and Yannakakis [24], giving the first strong evidence against PTASes for MAX-SAT and related problems. Such completeness theorems constitute valuable tools in complexity theory: Generally speaking, proving a problem  $A$  to be complete for a class  $\mathcal{C}$  shows that  $A$  is *the* representing problem for  $\mathcal{C}$ . The precise notion of completeness is typically chosen such that a certain algorithm for  $A$  would yield unexpected algorithms for the whole class  $\mathcal{C}$  – thus establishing that  $A$  is unlikely to admit such an algorithm. However, a completeness result may also open up algorithmic uses. Namely, since any problem in  $\mathcal{C}$  can be reduced to its complete problem  $A$ , we may find (possibly mildly) improved algorithms for all problems in  $\mathcal{C}$  by making algorithmic progress on the single problem  $A$ .

Given this usefulness, it may be surprising that there are currently no completeness results for studying optimization barriers *within the polynomial-time regime*, e.g., for approximability in strongly subquadratic time (in fact, even for studying decision problems, completeness results are an exception rather than the norm, see [29] for a recent survey of the field). Thus, this work sets out to initiate the quest for completeness results for optimization in P, which corresponds to studying the (in-)approximability of problems on large data sets.

### Previous Completeness Results in P

The essentially only known completeness result in fine-grained complexity theory in P is a recent result by Gao, Impagliazzo, Kolokolova, and Williams [18]: The orthogonal vectors problem (OV)<sup>1</sup> is established as complete problem for the class of model-checking first-order properties<sup>2</sup> under fine-grained reductions<sup>3</sup>. From this completeness, they derive in particular:

- **Hardness:** If there are  $\gamma, \delta > 0$  such that OV with moderate dimension  $d = n^\gamma$  can be solved in time  $O(n^{2-\delta})$ , then there is some  $\delta' > 0$  such that all  $(k+1)$ -quantifier first-order properties can be model-checked in time  $O(m^{k-\delta'})$  for  $k \geq 2$ . The negation of this statement's premise is known as the moderate-dimensional OV Hypothesis; the consequence would be very surprising, as model-checking first-order properties is a very general class of problems for which no  $O(m^{k-\delta})$ -time algorithm is known. This result can be seen as support for the moderate-dimensional OV Hypothesis.
- **Algorithms:** Using a stronger notion than fine-grained reductions, Gao et al. also prove that mildly subquadratic algorithms for OV have algorithmic consequences for model-checking first-order properties. Specifically, by combining their reductions with the fastest known algorithm for OV [4, 12], they obtain an  $m^k/2^{\Omega(\sqrt{\log m})}$ -time algorithm for model-checking any  $(k+1)$ -quantifier first-order property.

<sup>1</sup> Given two sets of  $n$  vectors in  $\{0, 1\}^d$ , determine whether there exists a pair of vectors, one of each set, that are orthogonal.

<sup>2</sup> Let  $\phi$  be a first-order property (in prenex normal form) over a relational structure of size  $m$ . Given the structure, determine whether  $\phi$  holds. See Section 2 for details.

<sup>3</sup> For a formal definition of fine-grained reductions, see [11, 18]. For this paper, the reader may think of the following slightly simpler notion: A fine-grained reduction from a problem  $P_1$  with presumed time complexity  $T_1$  to a problem  $P_2$  with presumed time complexity  $T_2$  is an algorithm  $A$  for  $P_1$  that has oracle access to  $P_2$  and whenever we use an  $O(T_2(n)^{1-\delta})$  algorithm for the calls to the  $P_2$ -oracle (for some  $\delta > 0$ ), there is a  $\delta' > 0$  such that  $A$  runs in time  $O(T_1(n)^{1-\delta'})$ .



No comparable fine-grained completeness results are known for polynomial-time optimization problems, raising the question: Can we give completeness theorems also for a general class of optimization problems in P, both for exact and approximate computation?

### Hardness of Approximation in P

Studying the fine-grained approximability of polynomial-time optimization problems (hardness of approximation in P), is a recent and influential trend: After a breakthrough result by Abboud, Rubinfeld, and Williams [3] establishing the *Distributed PCP in P* framework, a number of works gave strong conditional lower bounds, including results for nearest neighbor search [28] or a tight characterization of the approximability of maximum inner product [13, 15]. Further results include work on approximating graph problems [25, 6, 10, 22], the Fréchet distance [7], LCS [1, 2], monochromatic inner product [23], earth mover distance [26], as well as equivalences for fine-grained approximation in P [15, 14, 10]. Related work studies the inapproximability of parameterized problems, ruling out certain approximation guarantees within running time  $f(k)n^{g(k)}$  under parameter  $k$  (such as FPT time  $f(k)$  poly( $n$ ), or  $n^{o(k)}$ ), see [17] for a recent survey.<sup>4</sup>

### An Optimization Class: Polynomial-Time Analogues of MaxSNP

We define a natural and interesting class of polynomial-time optimization problems, inspired by the approach of Gao et al. [18] as well as the classic class MAXSNP introduced by Papadimitriou and Yannakakis [24] to study the approximability of NP optimization problems.

The definition of MAXSNP is motivated by Fagin’s theorem (see, e.g., [20, 19]), which characterizes NP as the family of problems expressible as  $\exists S \forall \bar{y} \exists \bar{z} \phi(\bar{y}, \bar{z}, G, S)$  where  $G$  is a given relational structure,  $\exists S$  ranges over a relational structure  $S$  and  $\forall \bar{y} \exists \bar{z} \phi(\bar{y}, \bar{z}, G, S)$  is a  $\forall^* \exists^*$ -quantified first-order property. A subclass of this is SNP, which consists of those problems expressible without the  $\exists \bar{z}$ -part. Its natural optimization variant is MAXSNP, defined as the set of problems expressible as  $\max_S \#\{\bar{y} : \phi(\bar{y}, G, S)\}$ . Notably, this class of problems contains central optimization problems (MAX-3-SAT, MAX-CUT, etc.), all of which admit a constant-factor approximation in polynomial time. Using a notion of MAXSNP-completeness, Papadimitriou and Yannakakis identified several problems (including MAX-3-SAT and MAX-CUT) as hardest-to-approximate in this class, giving a justification for the lack of a PTAS for these problems.<sup>5</sup>

To study the same type of questions in the polynomial-time regime, the perhaps most natural approach is to restrict the syntax defining MAXSNP problems such that it solely contains polynomial-time problems. Specifically, we replace  $\max_S$  by a maximization over a bounded number of  $k$  variables  $x_1, \dots, x_k$  and restrict the counting operator to tuples  $\bar{y} = (y_1, \dots, y_\ell)$  of bounded length  $\ell$ . The resulting formula  $\max_{x_1, \dots, x_k} \#\{(y_1, \dots, y_\ell) : \phi(x_1, \dots, x_k, y_1, \dots, y_\ell)\}$  can be easily seen (see the full version of the paper [8, Appendix A]) to be solvable in time  $O(m^{k+\ell-1})$ , where  $m$  denotes the problem size. We define  $\text{MAXSP}_{k,\ell}$  to denote the class of these optimization problems and let  $\text{MAXSP} = \bigcup_{k \geq 2, \ell \geq 1} \text{MAXSP}_{k,\ell}$ . Note that here, “SP” stands for “strict P” in analogy to the name “strict NP” of SNP. We refer to Section 2 for more details.

<sup>4</sup> Note that these parameterized inapproximability results do not necessarily apply to the case of a fixed parameter  $k$ , which would correspond to our setting. See [22] for an interesting exception.

<sup>5</sup> A stronger justification was later given by the PCP theorem, establishing inapproximability even under  $P \neq NP$ . In general, these two approaches (approximation-preserving completeness theorems as well as proving inapproximability under established assumptions on exact computation) can result in incomparable hardness of approximation results.

We obtain an analogous minimization class MINSP by replacing max by min everywhere. These classes include interesting problems:

- Vector-definable problems: Let  $\Sigma = \{0, \dots, c\}$  be a fixed alphabet and  $f : \Sigma^k \rightarrow \{0, 1\}$  be an arbitrary Boolean function. Then we can express the following problem: Given sets  $X_1, \dots, X_k$  in  $\Sigma^d$  of vectors, maximize (or minimize)  $\sum_{i=1}^d f(x_1[i], \dots, x_k[i])$  over all  $x_1 \in X_1, \dots, x_k \in X_k$ . Each such problem is definable in MAXSP $_{k,1}$ /MINSP $_{k,1}$ , e.g.:
  - Maximum Inner Product (MAXIP): Given sets  $X_1, X_2 \subseteq \{0, 1\}^d$ , maximize the inner product  $x_1 \cdot x_2$  over  $x_1 \in X_1, x_2 \in X_2$ . To see that this problem is in MAXSP $_{2,1}$ , consider the formula  $\max_{x_1 \in X_1, x_2 \in X_2} \#\{y \in Y : E(x_1, y) \wedge E(x_2, y)\}$ , where  $E(x, y)$  indicates that the  $y$ -th coordinate of  $x$  is equal to 1.
  - Consider minimization with  $k = 2$  and view  $f : \Sigma^2 \rightarrow \{0, 1\}$  as classifying pairs of characters as *similar* (0) or *dissimilar* (1). This expresses the following problem that generalizes the nearest-neighbor problem over the Hamming metric: Given a set of length- $d$  strings over  $\Sigma$ , determine the most similar pair of strings by minimizing the number of dissimilar characters.
  - View  $\Sigma$  as the finite field  $\mathbb{F}_q$  and let  $f(z_1, \dots, z_k) = 1$  iff  $\sum_{i=1}^k z_i \equiv 0 \pmod{q}$ . This gives optimization variants of the  $k$ -XOR problem [21, 16], generalized to arbitrary finite fields.
- Beyond vector-definable problems, in MAXSP $_{2,\ell-2}$  we can express the graph problem of computing, over all edges  $e$ , the maximum number of length- $\ell$  circuits containing  $e$ :

$$\max_{x_1, x_2} \#\{(y_1, \dots, y_{\ell-2}) : E(x_1, x_2) \wedge E(x_2, y_1) \wedge \dots \wedge E(y_{\ell-3}, y_{\ell-2}) \wedge E(y_{\ell-2}, x_1)\}.$$

In fact, MAXSP also contains generalizations of this problem to other pattern graphs than length- $\ell$  circuits (e.g., length- $\ell$  cycles or  $\ell$ -cliques), even arbitrary fixed patterns in *hypergraphs*.

We let  $m$  denote the size of the relational structure, that is, the number of tuples in an explicit representation of all relations. For vector-definable examples, the input can be represented as a relational structure of size  $m = O(nd \log |\Sigma|)$ , which is the natural input size. Note, however, that the relational structure also allows us to succinctly encode *sparse* vectors in very large dimension (such as  $d = \Theta(n)$ ), which is why we often refer to MAXSP and MINSP as describing a *sparse setting*. It is easy to see that each MAXSP or MINSP formula  $\psi$  can be solved in time  $O(m^{k+\ell-1})$  (see [8, Appendix A] in the full version); note that for a fixed  $\psi, k$  and  $\ell$  always denote the number of maximization/minimization and counting variables, respectively. Can we obtain completeness results with respect to improvements over this baseline running time?

### (Sparse) Maximum Inner Product

Our results prove the Maximum Inner Product problem (MAXIP) as representative for the class MAXSP. We will formally introduce two important variants of this problem.

► **Problem 1 (MAXIP).** *Given two sets of  $n$  vectors  $X_1, X_2 \subseteq \{0, 1\}^d$ , the task is to compute the maximum inner product  $\langle x_1, x_2 \rangle = \sum_j x_1[j] \cdot x_2[j]$  for  $x_1 \in X_1, x_2 \in X_2$ .*

When  $d = n^\gamma$  for some (small)  $\gamma > 0$ , we speak of the *moderate-dimensional* MAXIP problem. In this paper, we also use MAXIP in another context, depending on the input format. To make the distinction explicit, let us formally introduce the *Sparse* Maximum Inner Product problem (SPARSE MAXIP):

► **Problem 2 (SPARSE MAXIP).** *Given two sets of  $n$  vectors  $X_1, X_2 \subseteq \{0, 1\}^d$ , sparsely represented as a list of pairs  $(x_i, j)$  which represent the one-coordinates  $x_i[j] = 1$ , the task is to compute the maximum inner product  $\langle x_1, x_2 \rangle$  for  $x_1 \in X_1, x_2 \in X_2$ .*

For moderate-dimensional MAXIP we measure the complexity in  $n$  and for SPARSE MAXIP we measure the complexity in  $m$ , the total number of one-coordinates. We note that SPARSE MAXIP is also special in our setting as this problem can be seen as a member of  $\text{MAXSP}_{2,1}$ . Indeed, SPARSE MAXIP is the same problem as maximizing the formula

$$\psi = \max_{x_1 \in X_1, x_2 \in X_2} \#\{y \in [d] : E(x_1, y) \wedge E(x_2, y)\},$$

where  $E(x_i, y)$  indicates that the  $y$ -th coordinate of  $x_i$  is equal to 1. We also define the (Sparse) Minimum Inner Product problems (MINIP, SPARSE MINIP) as the analogous problems with the task to minimize  $\langle x_1, x_2 \rangle$ .

## 1.1 Our Results

Our first main result is a completeness theorem for exact optimization, establishing Maximum Inner Product as complete for MAXSP (and Minimum Inner Product for MINSIP).

► **Theorem 3** (SPARSE MAXIP is MAXSP-complete). *SPARSE MAXIP is complete for the class MAXSP under fine-grained reductions: If there is some  $\delta > 0$  such that SPARSE MAXIP can be solved in time  $O(m^{2-\delta})$ , then for every  $\text{MAXSP}_{k,\ell}$  formula  $\psi$ , there is some  $\delta' > 0$  such that  $\psi$  can be solved in time  $O(m^{k+\ell-1-\delta'})$ .*

*The analogous statement holds for minimization, if we replace SPARSE MAXIP and MAXSP by SPARSE MINIP and MINSIP, respectively.*

Turning to the approximability of MAXSP and MINSIP, we show how to obtain a fine-grained completeness that even preserves approximation factors (up to an arbitrarily small blow-up). Here and throughout the paper, we say that an algorithm gives a  $c$ -approximation for a maximization problem if it outputs a value in the interval  $[c^{-1} \cdot \text{OPT}, \text{OPT}]$ , where OPT is the optimal value. For minimization, the algorithm computes a value in the interval  $[\text{OPT}, c \cdot \text{OPT}]$ .

► **Theorem 4** (SPARSE MAXIP is MAXSP-complete, (almost) approximation preserving). *Let  $c \geq 1$  and  $\varepsilon > 0$ . If there is some  $\delta > 0$  such that SPARSE MAXIP can be  $c$ -approximated in time  $O(m^{2-\delta})$ , then for every  $\text{MAXSP}_{k,\ell}$  formula  $\psi$ , there is some  $\delta' > 0$  such that  $\psi$  can be  $(c + \varepsilon)$ -approximated in time  $O(m^{k+\ell-1-\delta'})$ .*

*The analogous statement for minimization holds for SPARSE MINIP and MINSIP.*

As a key technical step to obtain Theorems 3 and 4, we prove a *universe reduction* for MAXSP/MINSIP formulas (detailed in Sections 3 and 4.3). Along the way, this universe reduction establishes the following fine-grained equivalence between the sparse and moderate-dimensional settings of MAXIP/MINIP.

► **Theorem 5** (Equivalence between MAXIP and SPARSE MAXIP).

- *There are some  $\gamma, \delta > 0$  such that MAXIP with dimension  $d = n^\gamma$  can be solved in time  $O(n^{2-\delta})$  if and only if there is some  $\delta' > 0$  such that SPARSE MAXIP can be solved in time  $O(m^{2-\delta'})$ .*
- *Let  $c > 1$  and  $\varepsilon > 0$ . If there are some  $\gamma, \delta > 0$  such that MAXIP with dimension  $d = n^\gamma$  can be  $c$ -approximated in time  $O(n^{2-\delta})$  then there is some  $\delta' > 0$  such that SPARSE MAXIP can be  $(c + \varepsilon)$ -approximated in time  $O(m^{2-\delta'})$ . Conversely, if there is some  $\delta > 0$  such that SPARSE MAXIP can be  $c$ -approximated in time  $O(m^{2-\delta})$  then there are some  $\gamma, \delta' > 0$  such that MAXIP with dimension  $d = n^\gamma$  can be  $c$ -approximated in time  $O(n^{2-\delta'})$ .*

*The analogous statements for minimization hold for MINIP.*

We prove Theorems 3, 4 and 5 in Section 4.1.

### Consequences for Hardness of Approximation

As a consequence of the above completeness results and dimension reduction, we obtain the following statements.

- Since Maximum Inner Product and Minimum Inner Product are subquadratic equivalent in moderate dimensions [15, Theorem 1.6], we obtain from Theorems 3 and 5 that a strongly subquadratic algorithm solving moderate-dimensional Maximum Inner Product *exactly* would give a polynomial-factor improvement over the  $O(m^{k+\ell-1})$  running time for all MAXSP and MINSP formulas. This adds an additional surprising consequence of fast Maximum Inner Product algorithms, besides refuting the Orthogonal Vectors Hypothesis.
- There is a  $O(1)$ -approximation beating the quadratic baseline for moderate-dimensional Maximum Inner Product *if and only if* there is a  $O(1)$ -approximation beating the  $O(m^{k+\ell-1})$  time baseline for all MAXSP formulas. To obtain this result combine the fine-grained equivalence of  $O(1)$ -approximation of moderate-dimensional MAXIP and SPARSE MAXIP (Theorem 5) with the completeness of SPARSE MAXIP (Theorem 4). This adds an additional consequence of fast Maximum Inner Product approximation, besides refuting SETH [3, 13].
- In the minimization world, we obtain a tight connection between approximating MINSP formulas and OV: The (moderate-dimensional) OV hypothesis is *equivalent* to the non-existence of a  $O(1)$ -approximation for all MINSP formulas in time  $O(m^{k+\ell-1})$ . To obtain this result, combine the equivalence of moderate-dimensional OV Hypothesis and non-existence of a  $O(1)$ -approximation for moderate-dimensional MINIP [15, Theorem 1.5] with the equivalence of  $O(1)$ -approximation algorithms for moderate-dimensional MINIP and MINSP (Theorem 4 and Theorem 5). Interestingly, this can be seen as additional support for the Orthogonal Vectors Hypothesis.

### Algorithms: Lower-Order Improvements

Since Maximum Inner Product has received significant interest for improved algorithms (see particularly [13, 15]), we turn to the question whether our completeness result also yields lower-order algorithmic improvements for all problems in the class. Indeed, by combining the best known Maximum/Minimum Inner Product algorithms with our reductions, we obtain the following general results for MAXSP and MINSP. We give the proofs for both theorems in Section 4.1.

► **Theorem 6** (Lower-Order Improvement for Exact MAXSP and MINSP). *We can exactly optimize any  $\text{MAXSP}_{k,\ell}$  and  $\text{MINSP}_{k,\ell}$  formula in randomized time  $m^{k+\ell-1}/\log^{\Omega(1)} m$ .*

Interestingly, for constant-factor approximations, a complete shave of logarithmic factors is possible.

► **Theorem 7** (Lower-Order Improvement for Approximate MAXSP and MINSP). *For every constant  $c > 1$ , we can  $c$ -approximate every  $\text{MAXSP}_{k,\ell}$  and  $\text{MINSP}_{k,\ell}$  formula in time  $m^{k+\ell-1}/2^{\Omega(\sqrt{\log m})}$ . For  $\text{MAXSP}_{k,\ell}$  the algorithm is deterministic; for  $\text{MINSP}_{k,\ell}$  it uses randomization.*

## 2 Preliminaries

For an integer  $k \geq 1$ , we set  $[k] = \{1, \dots, k\}$ . Moreover, we write  $\tilde{O}(T) = T \log^{O(1)} T$ .

### First-Order Model-Checking

A *relational structure*  $(X, R_1, \dots, R_r)$  consists of  $n$  objects  $X$  and relations  $R_j \subseteq X^{a_j}$  (of arbitrary arities  $a_j$ ) between these objects. A *first-order formula* is a quantified formula of the form

$$\psi = (Q_1 x_1) \dots (Q_k x_k) \phi(x_1, \dots, x_k),$$

where  $Q_i \in \{\exists, \forall\}$  and  $\phi$  is a Boolean formula over the predicates  $R_j(x_{i_1}, \dots, x_{i_{a_j}})$ . Given a relational structure, the *model-checking problem* (or *query evaluation problem*) is to check whether  $\psi$  holds on the given structure, that is, for  $x_1, \dots, x_k$  ranging over  $X$  and by instantiating the predicates  $R_j(x_{i_1}, \dots, x_{i_{a_j}})$  in  $\phi$  according to the structure,  $\psi$  is valid.

Following previous work in this line of research [18, 9], we assume that the input is represented sparsely – that is, we assume that the relational structure is written down as an exhaustive enumeration of all records in all relations; let  $m$  denote the total number of such entries. This convention is reasonable as this data format is common in the context of database theory and also for the representation of graphs (where it is called the *adjacency list* representation). By ignoring objects not occurring in any relation, we may always assume that  $n \leq O(m)$ .

It is often convenient to assume that each variable  $x_i$  ranges over a separate set  $X_i$ . We can make this assumption without loss generality, by introducing some additional unary predicates.

### MaxSP $_{k,\ell}$ and MinSP $_{k,\ell}$

In analogy to first-order properties with quantifier structure  $\exists^k \forall^\ell$  (with maximization instead of  $\exists$  and counting instead of  $\forall$ ), we now define a class of optimization problems: Let MAXSP $_{k,\ell}$  be the class containing all formulas of the form

$$\psi = \max_{x_1, \dots, x_k, y_1, \dots, y_\ell} \# \phi(x_1, \dots, x_k, y_1, \dots, y_\ell), \quad (1)$$

where, as before,  $\phi$  is a Boolean formula over some predicates of arbitrary arities. We similarly define MINSP $_{k,\ell}$  with “min” in place of “max”. Occasionally, we write OPTSP $_{k,\ell}$  to refer to both of these classes simultaneously, and we write “opt” as a placeholder for either “max” or “min”. In analogy to the model-checking problem for first-order properties, we associate to each formula  $\psi \in \text{OPTSP}_{k,\ell}$  an algorithmic problem:

► **Definition 8** (MAX( $\psi$ ) and MIN( $\psi$ )). *Let  $\psi \in \text{MAXSP}_{k,\ell}$  be as in (1). Given a relational structure on objects  $X$ , the MAX( $\psi$ ) problem is to compute*

$$\text{OPT} = \max_{x_1, \dots, x_k \in X} \#_{y_1, \dots, y_\ell \in X} \phi(x_1, \dots, x_k, y_1, \dots, y_\ell).$$

*We similarly define MIN( $\psi$ ) for  $\psi \in \text{MINSP}_{k,\ell}$ . Occasionally, for  $\psi \in \text{OPTSP}_{k,\ell}$ , we write OPT( $\psi$ ) to refer to both problems simultaneously.*

As before, we usually assume (without loss of generality) that each variable ranges over a separate set:  $x_i \in X_i$ ,  $y_i \in Y_i$ . In particular, as claimed before we can express the SPARSE MAXIP formula

$$\psi = \max_{x_1 \in X_1, x_2 \in X_2} \#\{y \in [d] : E(x_1, y) \wedge E(x_2, y)\}$$

in a way which is consistent with Definition 8 by introducing three unary predicates for  $X_1$ ,  $X_2$  and  $[d]$ . For convenience, we introduce some further notation: For objects  $x_1 \in X_1, \dots, x_k \in X_k$ , we denote by  $\text{Val}(x_1, \dots, x_k) = \#_{y_1, \dots, y_\ell} \phi(x_1, \dots, x_k, y_1, \dots, y_\ell)$  the *value* of  $(x_1, \dots, x_k)$ .

Definition 8 introduces  $\text{MAX}(\psi)$  and  $\text{MIN}(\psi)$  as *exact* optimization problems (i.e., OPT is required to be computed exactly). We say that an algorithm computes a *c-approximation* for  $\text{MAX}(\psi)$  if it computes any value in the interval  $[c^{-1} \cdot \text{OPT}, \text{OPT}]$ . Similarly, a *c-approximation* for  $\text{MIN}(\psi)$  computes any value in  $[\text{OPT}, c \cdot \text{OPT}]$ .

The problem  $\text{OPT}(\psi)$  can be solved in time  $O(m^{k+\ell-1})$  for all  $\text{OPTSP}_{k,\ell}$  formulas  $\psi$ , by a straightforward extension of the model-checking baseline algorithm; see the full version of our paper [8, Appendix A] for details. As this is clearly optimal for  $k + \ell = 2$ , we will often implicitly assume that  $k + \ell \geq 3$  in the following.

As we show in the full version of the paper [8, Appendix B], we can *exactly* solve  $\text{OPTSP}_{k,\ell}$  in time  $O(m^{k+\ell-3/2})$  when  $\ell \geq 2$ . Thus, in the remaining sections we will be working with the *hardest* case  $\ell = 1$ . For convenience we write  $\text{MAXSP}_k := \text{MAXSP}_{k,1}$ , and similarly for  $\text{MINSP}_k$  and  $\text{OPTSP}_k$ . Since for a fixed formula  $\psi \in \text{OPTSP}$ ,  $k$  and  $\ell$  are constants,  $f(k, \ell)$ -factors are hidden in the  $O$ -notation throughout the paper.

### 3 Technical Overview

In this section we give an overview of the main technical ideas used to give our completeness result (Theorem 3). Let  $\psi$  be a  $\text{MAXSP}_{k,\ell}$  formula. We will outline the reduction from  $\text{MAX}(\psi)$  to  $\text{SPARSE MAXIP}$ . Since for  $\ell \geq 2$  we can solve  $\text{MAX}(\psi)$  in time  $O(m^{k+\ell-3/2})$  (see the full version of our paper [8, Appendix B]) we focus on the case of  $\ell = 1$ . The reduction consists of two phases. In the first phase (Appendix A), we reduce  $\psi$  to an intermediate problem called the *Hybrid Problem* which captures the core hardness, but is more restricted. For now, the reader can think of the Hybrid Problem as a vector-definable problem (as introduced in the introduction)  $\max_{x_1 \in X_1, \dots, x_k \in X_k} \sum_{i=1}^d f(x_1[i], \dots, x_k[i])$  with  $X_1, \dots, X_k \subseteq \{0, 1\}^d$ ; we define it formally in Section 4.2. Since a Hybrid Problem is more restricted than the general problem  $\text{MAX}(\psi)$ , the first phase consists of the following 4 steps in which we progressively restrict the shape of  $\psi$ :

1. Remove all *hyperedges*, that is,  $\psi$  no longer contains predicates of arity  $\geq 3$  so an instance of  $\text{MAX}(\psi)$  can be thought of as a graph with parallel (or alternatively, colored) edges.
2. Remove all edges between vertices  $x_i$  and  $x_j$  that we maximize over. We will call these *cross edges*. After this step the only remaining edges are between vertices  $x_i$  and the counting variable  $y$ .
3. Remove all *parallel edges* (or alternatively, colored edges), that is, we combine parallel edges into simple edges.
4. Remove unary predicates, finally turning the  $\text{MAX}(\psi)$  instances into graphs. At this point it becomes simple to rewrite  $\text{MAX}(\psi)$  as a Hybrid Problem.

The second phase of the reduction is to reduce the Hybrid Problem to a  $\text{SPARSE MAXIP}$  instance (Section 4.3). The general idea of this step seems straightforward: For simplicity again let us focus on a vector-definable problem  $\max_{x_1 \in X_1, \dots, x_k \in X_k} \sum_{i=1}^d f(x_1[i], \dots, x_k[i])$  with  $X_1, \dots, X_k \subseteq \{0, 1\}^d$ . We can precisely “cover” each  $f(x_1[i], \dots, x_k[i])$  by at most  $2^k$  summands expressing

$$\sum_{\substack{\alpha_1, \dots, \alpha_k \in \{0,1\} \\ f(\alpha_1, \dots, \alpha_k) = 1}} [(x_1[i], \dots, x_k[i]) = (\alpha_1, \dots, \alpha_k)],$$

where the outer  $[\cdot]$  denotes the Iverson brackets. Observe that each such summand is equivalent to the MAXIP function, *up to complementing some  $x_j[i]$ 's* (i.e. each summand can be expressed as MAXIP by setting  $x_j[i] := 1 - x_j[i]$  whenever  $\alpha_j = 0$ ). The issue, however, is that complementing  $x_j[i]$ 's means complementing a binary relation of size  $O(m)$  (between  $n$  vectors and  $d$  coordinates). Since complementing a sparse relation generally produces a dense relation (here: of size  $\Omega(nd)$ ), this will produce a prohibitively large problem size for the SPARSE MAXIP formulation if  $d$  is large.

The natural approach to overcome this issue is to reduce the dimension of the Hybrid Problem, so that we can afford the complementation step. One challenge in this is that MAXSP formula might have its optimal objective value anywhere in  $\{0, \dots, m^\ell\}$ , but reducing the dimension from  $d \leq m^\ell$  to, say,  $d = m^\gamma$  also reduces the range of possible objective values to  $\{0, \dots, m^\gamma\}$ . It appears counter-intuitive that such a “compression” of objective values should be possible while allowing us to reconstruct the optimum value *exactly*. Perhaps surprisingly, we are able to achieve this by a simple deterministic dimension reduction.

The idea of our dimension reduction is as follows. For concreteness, focus on the SPARSE MAXIP problem. Starting from a SPARSE MAXIP instance  $X_1, X_2 \subseteq \{0, 1\}^d$ , we construct a hash function  $h : \{0, 1\}^d \mapsto \{0, 1\}^{d'}$  with  $d' \ll d$ , which maps every one-entry to  $t$  coordinates in  $[d']$ . More precisely, for every coordinate  $i \in [d]$ , we deterministically choose an auxiliary vector  $w_i \in \{0, 1\}^{d'}$  with exactly  $t$  one-entries for some parameter  $t$ . Then, the hash function is defined as  $h(x) = \bigvee_{i:x[i]=1} w_i$  (here the OR is applied coordinate-wise).

We say that there is a collision between two vectors  $x_1, x_2$  if there are distinct  $i, j \in [d]$  such that  $x_1[i] = x_2[j] = 1$  and the auxiliary vectors  $w_i$  and  $w_j$  share a common one-entry. Ideally, every pair of vectors  $x_1 \in X_1, x_2 \in X_2$  is hashed *perfectly*, meaning that no collision takes place. In that case, it holds that  $\langle h(x_1), h(x_2) \rangle = t \cdot \langle x_1, x_2 \rangle$  and thus also  $\text{OPT}' = t \cdot \text{OPT}$ , where  $\text{OPT}$  and  $\text{OPT}'$  are the objective values of the original and the hashed instance, respectively. However, in reality we cannot expect the hashing to be perfect. Note that nevertheless the difference  $|\langle h(x_1), h(x_2) \rangle - t \cdot \langle x_1, x_2 \rangle|$  is at most the number of collisions between  $x_1$  and  $x_2$ .

We will construct  $h$  in such a way that for all pairs  $x_1, x_2$ , the number of collisions is small, say at most  $C$ . Then by setting  $t > 2C$ , we ensure that  $|t \cdot \text{OPT} - \text{OPT}'| < t/2$  so we can recover  $t \cdot \text{OPT}$  by computing  $\text{OPT}'$  and rounding to the closest multiple of  $t$ . In particular, the optimal pair of vectors in the hashed instance correspond to the pair with maximum inner product in the original instance. Note that we crucially use the fact that MAXIP is expressive enough to *compute the value* of the inner product, which allows us to get rid of the small additive error introduced by the hashing (after rounding).

In Section 4.3 we show that the desired hash function exists and is in fact deterministic: Pick any  $t$  primes  $p_1, \dots, p_t$  of size  $\Theta(t \log t)$  and let  $d' = p_1 + \dots + p_t$ . We identify  $[d']$  with  $\{(i, p_j) : 1 \leq j \leq t, 0 \leq i < p_j\}$  and assign the auxiliary vector  $w_i$  to have one-entries exactly at all coordinates  $(i \bmod p_j, p_j)$ ,  $1 \leq j \leq t$ . A simple calculation shows that with this construction the number of collisions between  $x_1$  and  $x_2$  is at most  $\|x_1\|_1 \cdot \|x_2\|_1 \cdot \log d$ , see Lemma 15. With some additional tricks, we can control this quantity.

Our analysis allows us to even maintain  $c$ -approximate solutions, albeit with an arbitrarily small blow-up due to the small error introduced by rounding. Finding a fully approximation-preserving reduction remains a challenge for future work. Additionally, we need to take great care that our reductions are efficient enough to even transfer  $\log^{0.1} n$ -improvements, to obtain our speed-up for exact optimization (Theorem 6).

### Comparison to Gao et al.'s Work

Our reduction is similar to the work of Gao, Impagliazzo, Kolokolova and Williams [18], showing that the sparse version of Orthogonal Vectors is complete for model checking first-order properties. Here we discuss the key differences.

This first phase of our reduction follows the same structure as in Gao et al., but we simplify the proof significantly: One major difference is that they define a more complicated version of the Hybrid Problem including cross predicates [18, Section 5.2]. Borrowing ideas from [9], we remove the cross predicates at an earlier stage of the reduction (Step 2), which simplifies the remaining Steps 3 and 4. The absence of cross predicates also simplifies the baseline algorithm (see [8, Appendix A] in the full version). More generally, by splitting the reduction into a chain of four steps we cleanly separate the main technical ideas used in the first phase; see Appendix A for more details. In the same spirit we simplify Gao et al.'s improved algorithm [18, Section 9.2] for all problems with more than 1 counting quantifier avoiding their case distinction of 9 different cases by using a simple basis to represent all Boolean functions  $\phi : \{0, 1\}^3 \rightarrow \{0, 1\}$ ; see [8, Appendix B] in the full version.

In the second phase of the reduction, their work faces the same main challenge as ours. Specifically, reducing their Hybrid Problem to OV naively requires complementing a sparse binary relation, possibly resulting in a large dense complement. They solve this issue by designing a similar dimension reduction as ours using a Bloom filter. Naturally their dimension reduction is randomized, but they also provide a derandomization. However, note that there is a crucial difference: They reduce to OV which is a *decision* problem, while we reduce to the *optimization* problem MAXIP. For this reason, the dimension reductions differ in nature: On the one hand, we exploit that MAXIP is more expressive than OV – namely that MAXIP can handle a small number of errors if we round the result, while for OV any introduced error would result in vectors that are not orthogonal anymore. On the other hand, by reducing to OV, Gao et al. do not have to worry about “compressing” the range of possible optimal values, or making the reduction approximation-preserving. For these reasons, their dimension reduction would be unsuitable in our work, and ours would be unsuitable in their work.

## 4 The Reduction

In this section we give the proofs of our main results. The following lemma captures our reduction in all generality. Let  $k$ -MAXIP denote the generalization of the MAXIP problem with the objective to compute  $\max_{x_1 \in X_1, \dots, x_k \in X_k} \langle x_1, \dots, x_k \rangle$ , where  $\langle x_1, \dots, x_k \rangle = \sum_y x_1[y] \cdot \dots \cdot x_k[y]$ . We define  $k$ -MINIP analogously.

► **Lemma 9.** *Let  $s(n) \leq n^{1/6}$  be a nondecreasing function and let  $c \geq 1$  be constant. Assume that  $k$ -MAXIP in dimension  $d = \tilde{O}(s(n)^4 \log^2 n)$  can be  $c$ -approximated in time  $O(n^k/s(n))$ , and let  $\psi$  be an arbitrary  $\text{MAXSP}_k$  formula.*

- *If  $c = 1$  (i.e., we are in the case of exact computation), then  $\text{MAX}(\psi)$  can be exactly solved in time  $O(m^k/s(\sqrt[k+1]{m}))$ .*
- *If  $c > 1$ , then  $\text{MAX}(\psi)$  can be  $(c + \varepsilon)$ -approximated in time  $O(m^k/s(\sqrt[k+1]{m}))$ , for any constant  $\varepsilon > 0$ .*

*The analogous statement holds for  $k$ -MINIP and  $\text{MINSP}_k$ .*

The outline for this section is as follows. First we show how to derive the completeness result (Theorems 3 and 4) and the lower-order improvements (Theorems 6 and 7) from Lemma 9 in Section 4.1. Then we present the proof of Lemma 9, which is carried out in



two phases as explained in the technical overview. In Section 4.2 we formally introduce the intermediate problem called the *Hybrid Problem*. In Section 4.3 we give a fine-grained reduction from the Hybrid Problem to Maximum or Minimum Inner Product (Lemma 14). Finally, in Appendix A we reduce any  $\text{OPT}_{k,\ell}$  formula to the Hybrid Problem (Lemma 17), thus finishing the proof of Lemma 9. We will pay particularly close attention to the exact savings  $s$  in every step.

## 4.1 Consequences

First we derive the completeness Theorems 3 and 4 from Lemma 9.

**Proof of Theorems 3 and 4.** Let  $c \geq 1$  denote the approximation ratio (that is,  $c = 1$  for Theorem 3 and  $c \geq 1$  for Theorem 4). Assuming that SPARSE MAXIP can be  $c$ -approximated in time  $O(m^{2-\delta})$  for some  $\delta > 0$ , we obtain an algorithm for  $c$ -approximating MAXIP in dimension  $d = n^{4\delta/9}$  in time  $O((nd)^{2-\delta}) = O(n^{2-\delta}d^2) = O(n^{2-\delta/9})$ . We also obtain an algorithm for  $c$ -approximating  $k$ -MAXIP in the same dimension in time  $O(n^{k-\delta/9})$  (brute-force all options for the first  $k-2$  vectors, then use the 2-MAXIP algorithm). We can now plug this improved algorithm into our reduction: Setting  $s(n) = n^{\delta/9}/\text{polylog}(n)$  we have that  $k$ -MAXIP in dimension  $d = \tilde{O}(s(n)^4 \log^2 n)$  can be  $c$ -approximated in time  $O(n^k/s(n))$ . Thus, if  $c = 1$  we obtain by Lemma 9 that  $\text{OPT}(\psi)$  can be exactly solved in time  $O(m^k/s(n)^{\frac{1}{k+1}}) = O(m^{k-\beta})$  for  $\beta = \frac{\delta}{9(k+1)} > 0$ . If  $c > 1$ , we obtain that  $\text{OPT}(\psi)$  can be  $(c + \varepsilon)$ -approximated in the same running time, for an arbitrarily small constant  $\varepsilon > 0$ . ◀

Next, we prove Theorem 5.

**Proof of Theorem 5.** The reductions from SPARSE MAXIP to MAXIP and from SPARSE MINIP to MINIP for both the exact and approximate settings are a direct consequence of Lemma 9.

For the other direction, assume there exists some  $\delta > 0$  such that SPARSE MAXIP can be  $c$ -approximated in time  $O(m^{2-\delta})$ . Set  $\gamma := \delta/2$  and observe that any MAXIP instance with  $d = n^\gamma$  yields a SPARSE MAXIP instance of size  $m = O(nd) = O(n^{1+\gamma})$ . Since we can solve this instance in time  $O(m^{2-\delta}) = O(n^{(1+\gamma)(2-\delta)}) = O(n^{(1+\delta/2)(2-\delta)}) = O(n^{2-\delta^2/2})$ , we obtain a  $O(n^{2-\delta'})$ -algorithm for MAXIP with  $d = n^\gamma$  and  $\delta' = \delta^2/2$ . Note that this works for both the exact ( $c = 1$ ) and approximate ( $c > 1$ ) settings. The proof for the minimization case is analogous. ◀

To prove Theorems 6 and 7, we make use of the following state-of-the-art algorithms for MAXIP and MINIP, established in three previous papers [5, 13, 15].

► **Theorem 10** (Improved Algorithms for MAXIP and MINIP [5, 13, 15]).

- $k$ -MAXIP and  $k$ -MINIP in dimension  $d = O(\log^{2.9} n)$  can be exactly solved in randomized time  $O(n^k/\log^{100} n)$  [5].
- For any constant  $c > 1$ ,  $k$ -MAXIP in dimension  $d = 2^{O(\sqrt{\log n})}$  can be  $c$ -approximated in deterministic time  $n^k/2^{\Omega(\sqrt{\log n})}$  [13, Theorem 1.5].
- For any constant  $c > 1$ ,  $k$ -MINIP in dimension  $d = 2^{O(\sqrt{\log n})}$  can be  $c$ -approximated in randomized time  $n^k/2^{\Omega(\sqrt{\log n})}$  [15, Theorem 1.7].

**Proof of Theorems 6 and 7.** To prove Theorem 6, we plug in the first algorithm from Theorem 10 into Lemma 9 and choose  $s(n) = \log^{0.1} n$ . We obtain an exact  $\text{OPTSP}_k$  algorithm in time  $m^k/\log^{\Omega(1)} m$ .

For Theorem 7, we plug the second and third algorithms from Theorem 10 into Lemma 9 and choose  $s(n) = 2^{O(\sqrt{\log n})}$ . We get a  $c$ -approximation for  $\text{OPTSP}_k$  in time  $m^k/2^{\Omega(\sqrt{\log m})}$ , for any constant  $c > 1$ . ◀

Note that only one of these algorithms is deterministic; other known deterministic algorithms are not efficient enough for our reduction<sup>6</sup>.

## 4.2 The Hybrid Problem

We start with another problem definition.

► **Definition 11** (Basic Problem). *Given set families  $\mathcal{S}_1, \dots, \mathcal{S}_k$  over a universe  $U$ , the Basic Maximization Problem of type  $\tau \in \{0, 1\}^k$  is to compute*

$$\text{OPT} = \max_{S_1 \in \mathcal{S}_1, \dots, S_k \in \mathcal{S}_k} \left| \left( \bigcap_{i:\tau[i]=1} S_i \right) \setminus \left( \bigcup_{i:\tau[i]=0} S_i \right) \right|.$$

For example, the Basic Problem of type  $\tau = 11$  is to maximize the common intersection of two sets  $S_1$  and  $S_2$ , the Basic Problem of type  $\tau = 10$  is to maximize the number of elements in  $S_1$  not contained in  $S_2$  and the Basic Problem of type  $\tau = 00$  is to maximize the number of universe elements contained in neither  $S_1$  nor  $S_2$ .

Note that every Basic Problem can be seen as an  $\text{OPTSP}_k$  formula: We introduce objects for all sets  $S_i$  and all universe elements  $u$ , and connect  $S_i$  to  $u$  via an edge  $E(S_i, u)$  if and only if  $u \in S_i$ . Consistent with this analogy, we define  $n$  as the total number of sets  $S_i$  and  $m$  as the total cardinality of all sets  $S_i$  and, as before, study the Basic Problem with respect to the sparsity  $m$ .

► **Definition 12** (Hybrid Problem). *Given set families  $\mathcal{S}_1, \dots, \mathcal{S}_k$  over a universe  $U$ , which is partitioned into  $2^k$  parts  $U = \bigcup_{\tau \in \{0,1\}^k} U_\tau$ , the Hybrid Maximization Problem is to compute*

$$\text{OPT} = \max_{S_1 \in \mathcal{S}_1, \dots, S_k \in \mathcal{S}_k} \sum_{\tau \in \{0,1\}^k} \left| U_\tau \cap \left( \bigcap_{i:\tau[i]=1} S_i \right) \setminus \left( \bigcup_{i:\tau[i]=0} S_i \right) \right|.$$

We similarly define Basic Minimization Problems and define  $\epsilon$ -approximations of Basic Problems in the obvious way. For any  $S_1, \dots, S_k$  and  $\tau \in \{0, 1\}^k$  we denote by  $\text{Val}_\tau(S_1, \dots, S_k)$  the *value* of the Basic Problem constraint of type  $\tau$ :

$$\text{Val}_\tau(S_1, \dots, S_k) := \left| U_\tau \cap \left( \bigcap_{i:\tau[i]=1} S_i \right) \setminus \left( \bigcup_{i:\tau[i]=0} S_i \right) \right|.$$

And we use  $\text{Val}(S_1, \dots, S_k) := \sum_{\tau} \text{Val}_\tau(S_1, \dots, S_k)$  to denote the total value of the sets  $S_1, \dots, S_k$  in a Hybrid Problem instance.

Intuitively, the Hybrid Problem simultaneously optimizes Basic Problem constraints of different types. If we could afford to complement (parts of) the sets  $S_i$ , then there is a straightforward reduction from the Hybrid Problem to a Basic Problem of arbitrary type  $\tau$ : For each constraint of type  $\tau' \neq \tau$ , we simply complement all sets  $S_i$  with  $\tau'[i] \neq \tau[i]$  (more precisely, construct sets  $S'_i$  such that  $U_{\tau'} \cap S'_i = U_{\tau'} \setminus S_i$ ) and reinterpret the  $\tau'$ -constraint as type  $\tau$ . In summary:

<sup>6</sup> Focus on exact  $\text{MAXSP}_k$  for illustration: To obtain the same savings as in Theorem 6, we would need a deterministic algorithm for  $\text{MAXIP}$  in dimension  $d = O(\log^{2.9} n)$  running in time  $O(n^2 / \log^{100} n)$ . However, for this speed-up the current best algorithm [5] requires  $d = O(\log^{1.9} n)$ , so one needs to either improve the algorithm or improve our dimension reduction (Lemma 9) to dimension  $d = \text{poly}(s(n)) \log n$ , say.

► **Observation 13.** *In time  $O(n|U|)$ , any Hybrid Problem instance can be converted into an equivalent Basic Problem instance of arbitrary type  $\tau$ . The sparsity of the constructed instance is up to  $n|U|$ .*

However, being in the sparse setup we cannot tolerate the blow-up in the sparsity. Therefore, in order to efficiently apply Observation 13, we first have to control the universe size  $|U|$ .

### 4.3 Universe Reduction

The goal of this section is to reduce the Hybrid Problem to  $k$ -MAXIP. We give a reduction which closely preserves the savings  $s(n)$  achieved by exact or approximate  $k$ -MAXIP algorithms (losing only polynomial factors in  $s(n)$ ). As a drawback, the reduction slightly worsens the approximation factor, turning a  $c$ -approximation into a  $(c + \varepsilon)$ -approximation.

► **Lemma 14.** *Let  $s(n) \leq n^{1/6}$  be a nondecreasing function and assume that  $k$ -MAXIP in dimension  $d = \tilde{O}(s(n)^4 \log^2 n)$  can be  $c$ -approximated in time  $O(n^k/s(n))$ .*

- *If  $c = 1$  (i.e., we are in the case of exact computation), then the Hybrid Problem can be exactly solved in time  $O(m^k/s(m))$ .*
- *If  $c > 1$ , then the Hybrid Problem can be  $(c + \varepsilon)$ -approximated in time  $O(m^k/s(m))$ , for any constant  $\varepsilon > 0$ .*

*The analogous statement holds for  $k$ -MINIP and  $\text{MINSP}_k$ .*

On a high level, we prove Lemma 14 by first using a deterministic construction to reduce the universe size, and then reducing further to  $k$ -MAXIP as in Observation 13. The following lemma provides our universe reduction in the form of a hash-like function  $h$ .

► **Lemma 15.** *Let  $U$  be a universe and let  $t$  be a parameter. There exists a universe  $U'$  of size at most  $4t^2 \log t$  and a function  $h$  mapping elements in  $U$  to size- $t$  subsets of  $U'$ , such that the following properties hold. By abuse of notation, we write  $h(S) = \bigcup_{u \in S} h(u)$  for sets  $S \subseteq U$ .*

1. (Hashing.) *For all sets  $S \subseteq U$ , it holds that  $|h(S)| \geq t|S| - |S|^2 \log |U|$ .*
2. (Efficiency.) *Evaluating  $h(u)$  takes time  $\tilde{O}(t)$ .*

**Proof.** We start with the construction of  $h$ . By the Prime Number Theorem, there exist  $t$  primes  $p_1, \dots, p_t$  in the interval  $[2t \log t, 4t \log t]$  (for large enough  $t$ , see [27, Corollary 3] for the quantitative version). Let  $U' = \{(i, j) : 1 \leq i \leq t, 0 \leq j < p_i\}$ , then  $|U'| \leq 4t^2 \log t$ . We identify  $U$  with  $[|U|]$  in an arbitrary way and define  $h(u) = \{(i, u \bmod p_i) : 1 \leq i \leq t\}$  for  $u \in U$ .

In order to prove the first property, let us define the *collision number* of two distinct elements  $u, u' \in U$  as  $|h(u) \cap h(u')|$ . It is easy to see that the collision number of any such pair is at most  $\log |U|$ : For any prime  $p_i$ , we have that  $u \bmod p_i = u' \bmod p_i$  if and only if  $p_i$  divides  $u - u'$ . Since  $u - u'$  has absolute value at most  $U$ , there can be at most  $\log |U|$  distinct prime factors  $p_i$  of  $u - u'$ . It follows that  $t|S| - |h(S)| \leq \sum_{u, u' \in S} |h(u) \cap h(u')| \leq |S|^2 \log |U|$ .

Finally, the function can be efficiently evaluated: Computing the primes  $p_1, \dots, p_t$  takes time  $O(t \log t \log \log t)$  using Eratosthenes' sieve, for example. After this precomputation, evaluating  $h(u)$  in time  $O(t)$  is straightforward. ◀

► **Lemma 16 (Universe Reduction).** *Let  $\mathcal{S}_1, \dots, \mathcal{S}_k$  over the universe  $U = \bigcup_{\tau} U_{\tau}$  be a Hybrid Problem instance of maximum set size  $s = \max_{S_i \in \mathcal{S}_i} |S_i|$ , and let  $t$  be a parameter. In time  $\tilde{O}(mt)$  we can compute a number  $\Delta \geq 0$  and a new Hybrid Problem instance  $\mathcal{S}'_1, \dots, \mathcal{S}'_k$  over a small universe  $U' = \bigcup_{\tau} U'_{\tau}$  of size  $|U'| = O(t^2 \log t)$  such that:*

1. The sets  $S_i \in \mathcal{S}_i$  and the sets  $S'_i \in \mathcal{S}'_i$  stand in one-to-one correspondence.
2. For all  $S_1 \in \mathcal{S}_1, \dots, S_k \in \mathcal{S}_k$ , it holds that:

$$|t \cdot \text{Val}(S_1, \dots, S_k) - \text{Val}(S'_1, \dots, S'_k) - \Delta| = O(s^2 \log |U|).$$

**Proof.** We first describe how to construct the new instance. The first goal is to design individual universe reductions for all subuniverses  $U_\tau$ , that is, we construct new universes  $U'_\tau$  and functions  $h_\tau$  mapping  $U_\tau$  to size- $t$  subsets of  $U_\tau$ . We distinguish two cases:

- If  $|U_\tau| \leq 4t \log t$ , then we simply take  $U'_\tau$  as  $t$  copies of  $U_\tau$  and let  $h_\tau$  be the function which maps any element to its  $t$  copies in  $U_\tau$ . It holds that  $|U'_\tau| = t \cdot |U_\tau| \leq 4t^2 \log t$ .
- If  $|U_\tau| > 4t \log t$ , then we apply Lemma 15 with parameter  $t$  to obtain  $U'_\tau$  and  $h_\tau$ . The lemma guarantees that  $|U'_\tau| \leq 4t^2 \log t$ .

Next, we assemble these individual reductions into one. Set  $U' = \bigcup_\tau U'_\tau$ , where we treat the sets  $U'_\tau$  as disjoint. Since in both of the previous two cases we have  $|U_\tau| = O(t^2 \log t)$  it follows that  $|U| = \sum_\tau |U_\tau| = O(t^2 \log t)$ . Let  $h$  be the function which is piece-wise defined by the  $h_\tau$ 's, that is,  $h$  returns  $h_\tau(u)$  on input  $u \in U_\tau$ . Recall the notation  $h(S) = \bigcup_{u \in S} h(u)$ . The new Hybrid Problem instance is constructed by hashing every set  $S_i \in \mathcal{S}_i$  into the smaller universe, that is, we set  $S'_i := h(S_i) \in \mathcal{S}'_i$ . Property 1 is immediate from this construction, and the computation takes time  $\tilde{O}(mt)$ .

It remains to prove Property 2. For the remainder of the proof fix some sets  $S_1, \dots, S_k$  and let  $S = S_1 \cup \dots \cup S_k$  (clearly,  $S$  has size  $O(s)$ ). We start with the (unrealistic) assumption that  $S$  is hashed *perfectly*, that is,  $|h(S)| = t|S|$ . In this case we claim that:

- $t \cdot \text{Val}_\tau(S_1, \dots, S_k) = \text{Val}_\tau(h(S_1), \dots, h(S_k))$  for all  $\tau \neq 0^k$ ,
  - $t \cdot \text{Val}_\tau(S_1, \dots, S_k) = \text{Val}_\tau(h(S_1), \dots, h(S_k)) + \Delta$  for  $\tau = 0^k$ , where  $\Delta := t \cdot |U_{0^k}| - |U'_{0^k}|$ .
- Indeed, if  $S$  is hashed perfectly then we exactly scale the number of satisfying elements by a factor of  $t$  for every type  $\tau \neq 0^k$ . This holds because a satisfying assignment for  $\tau \neq 0^k$  corresponds to some element of the universe  $u \in U_\tau$  for which  $u \in S_i$  for all  $i$ 's such that  $\tau[i] = 1$ . The perfect hashing implies that the element  $u$  in these sets  $S_i$  gets mapped to  $t$  different elements in the new universe, and since there are no collisions these form  $t$  satisfying assignments in the hashed instance. The type  $\tau = 0^k$  is exceptional because each satisfying assignment does not correspond to any  $u \in U_{0^k}$ . Instead, the hashing scales the number of *falsifying* elements of type  $0^k$ ,  $|U_{0^k} \cap S|$ . The number of *satisfying* elements of type  $0^k$ ,  $|U_{0^k} \setminus S|$ , is preserved up to an additive error of exactly  $\Delta$ .

We will now remove the unrealistic assumption that  $h$  is hashed perfectly. The strategy is to define another function  $h^*$  obtained from  $h$  by artificially making the hashing with  $S$  perfect. To that end, we list the elements in  $S$  in an arbitrary order  $s_1, \dots, s_{|S|}$ , and start with the assignment  $h^*(s_j) = h(s_j)$ . As long as there exist indices  $i < j$  such that  $h^*(s_i)$  and  $h^*(s_j)$  share a common element  $z$ , we reassign  $h^*(s_j) := h^*(s_j) \setminus \{z\} \cup \{z'\}$  for some unused universe element  $z' \in U'$ . The function  $h^*$  obtained in this way also maps elements of  $U$  to size- $t$  subsets of  $U'$  and hashes  $S$  perfectly. Let  $Z$  be the set of all pairs of elements  $z$  and  $z'$  that occurred in the process; since there are exactly  $t|S| - |h(S)|$  iterations we have  $|Z| \leq 2t|S| - 2|h(S)|$  and by Lemma 15 it follows that  $|Z| = O(s^2 \log |U|)$ . By the definition of  $h^*$ , it is clear that  $|\text{Val}(h(S_1), \dots, h(S_k)) - \text{Val}(h^*(S_1), \dots, h^*(S_k))| \leq |Z|$ . Therefore, by the previous paragraph (applied with  $h^*$ ) and by an application of the triangle inequality, we obtain:

- $|t \cdot \text{Val}_\tau(S_1, \dots, S_k) - \text{Val}_\tau(h(S_1), \dots, h(S_k))| = O(s^2 \log |U|)$  for all  $\tau \neq 0^k$ ,
- $|t \cdot \text{Val}_\tau(S_1, \dots, S_k) - \text{Val}_\tau(h(S_1), \dots, h(S_k)) - \Delta| = O(s^2 \log |U|)$  for  $\tau = 0^k$ .

The claimed Property 2 is now immediate by summing over all types  $\tau$  and by another application of the triangle inequality.

Finally, it remains to prove that  $\Delta \geq 0$ . There are two cases depending on how the set  $U'_{0^k}$  was constructed: In the first case of the construction we have  $t \cdot |U_{0^k}| = |U'_{0^k}|$  and thus  $\Delta = 0$ . In the second case we have  $|U'_\tau| \leq 4t^2 \log t < t \cdot |U_\tau|$  and thus  $\Delta = t \cdot |U_{0^k}| - |U'_{0^k}| > 0$ . ◀

Having established the universe reduction, we can finally prove Lemma 14.

**Proof of Lemma 14.** The algorithm consists of three steps, which are implemented in the same way for all combinations of maximization versus minimization and exact versus approximate computation.

1. (*Eliminating heavy sets.*) We say that a set  $S_i \in \mathcal{S}_i$  is *heavy* if  $|S_i| > s(m)$ , and *light* otherwise. Our first goal is to eliminate all heavy sets. Since the total cardinality of all sets  $S_i$  is bounded by  $m$ , there can be at most  $O(m/s(m))$  heavy sets. Therefore, we can brute-force over every such set  $S_i$  and solve the remaining Hybrid Problem on  $k - 1$  set families using the baseline algorithm in time  $O(m^{k-1})$ . Afterwards, we can safely remove all heavy sets. Overall, this step takes time  $O(m^k/s(m))$ .
2. (*Reduction to  $k$ -MAXIP or  $k$ -MINIP.*) In the remaining instance we have that  $|S_i| \leq s(m)$  for all sets  $S_i$ . Therefore, we can apply the universe reduction from Lemma 16 (with some parameter  $t$  to be specified in the next step) to obtain an instance  $\mathcal{S}'_1, \dots, \mathcal{S}'_k$  over a smaller universe  $U' = \bigcup_\tau U'_\tau$  of size  $O(t^2 \log t)$ , and an offset  $\Delta \geq 0$ .  
The Hybrid Maximization Problem instance  $\mathcal{S}'_1, \dots, \mathcal{S}'_k$  reduces to  $k$ -MAXIP in the natural way: Recall that  $k$ -MAXIP is the same as the Basic Problem of type  $\tau = 1^k$ . Hence, we can apply Observation 13 to reduce to an instance of  $k$ -MAXIP with  $n = O(m)$  vectors in dimension  $O(t^2 \log t)$  in time  $O(n|U'|) = O(nt^2 \log t)$ . An analogous reduction works for Hybrid Minimization Problems and  $k$ -MINIP.
3. (*Recovering the optimal value.*) Solve (or approximate) the constructed  $k$ -MAXIP instance and let  $\text{ALG}'$  denote the output. Then compute  $\text{ALG} := (\text{ALG}' + \Delta)/t$  and return  $\text{ALG}$  rounded to an integer. The precise way of rounding depends on maximization versus minimization and exact versus approximate, see the following analysis.

Let  $\varepsilon > 0$  be a constant which we will specify later, and set  $t = Cs(m)^2 \log m$  for some sufficiently large constant  $C = C(\varepsilon)$ . Then by Property 2 of Lemma 16 we have

$$\left| \text{Val}(S_1, \dots, S_k) - \frac{\text{Val}(S'_1, \dots, S'_k) + \Delta}{t} \right| = O\left(\frac{s(m)^2 \log m}{t}\right) < \varepsilon.$$

In particular, it holds that

$$\left| \text{OPT} - \frac{\text{OPT}' + \Delta}{t} \right| < \varepsilon, \tag{2}$$

where  $\text{OPT}$  and  $\text{OPT}'$  are the optimal values of the original and the reduced instance, respectively. As the new universe has size  $O(t^2 \log t) = \tilde{O}(s(m)^4 \log m^2)$  as claimed, we can indeed use the efficient  $O(m^k/s(m))$ -time  $k$ -MAXIP or  $k$ -MINIP algorithm in the third step. The total running time is as stated: Recall that  $s(m) \leq m^{1/6}$  and thus all previous steps run in time  $O(m^k/s(m))$ . It remains to argue about the guarantees of the reduction; we need to consider three cases:

- (*Exact maximization or minimization:  $c = 1$ .)* It suffices to set  $\varepsilon < \frac{1}{2}$ . Since we can exactly compute  $\text{ALG}' = \text{OPT}'$ , by rounding  $\text{ALG} = (\text{ALG}' + \Delta)/t$  to the nearest integer, we obtain the only integer in the interval  $((\text{OPT}' + \Delta)/t - \frac{1}{2}, (\text{OPT}' + \Delta)/t + \frac{1}{2})$ , and thus we output  $\text{OPT}$ .

- (Approximate maximization:  $c > 1$ .) We have  $c^{-1}\text{OPT}' \leq \text{ALG}' \leq \text{OPT}'$  and therefore

$$\begin{aligned}\text{ALG} &= \frac{\text{ALG}' + \Delta}{t} \leq \frac{\text{OPT}' + \Delta}{t} \stackrel{(2)}{\leq} \text{OPT} + \varepsilon, \\ \text{ALG} &= \frac{\text{ALG}' + \Delta}{t} \geq \frac{c^{-1}(\text{OPT}' + \Delta)}{t} \stackrel{(2)}{\geq} c^{-1}(\text{OPT} - \varepsilon) \geq c^{-1}\text{OPT} - \varepsilon,\end{aligned}$$

where in the first inequality of the second line we used both  $\text{ALG}' \geq c^{-1}\text{OPT}'$  and  $c > 1$ . From these bounds we derive that the algorithm should return  $\lceil \text{ALG} - \varepsilon \rceil$ . Indeed, as  $\lceil \text{ALG} - \varepsilon \rceil \leq \text{OPT}$  this is always a feasible solution. Moreover, the solution is  $\frac{c}{1-2\varepsilon}$ -approximate: If  $\text{OPT} = 0$ , then  $\lceil \text{ALG} - \varepsilon \rceil = 0$  (if we set  $\varepsilon < \frac{1}{2}$ ). If  $\text{OPT} \geq 1$ , then  $\lceil \text{ALG} - \varepsilon \rceil \geq \frac{1}{c}\text{OPT} - 2\varepsilon \geq \frac{1-2\varepsilon}{c}\text{OPT}$ . Setting  $\varepsilon$  small enough yields approximation ratio  $c + \varepsilon'$ , for any  $\varepsilon' > 0$ .

- (Approximate minimization:  $c > 1$ .) We have  $\text{OPT}' \leq \text{ALG}' \leq c \cdot \text{OPT}'$  and therefore

$$\begin{aligned}\text{ALG} &= \frac{\text{ALG}' + \Delta}{t} \leq \frac{c \cdot \text{OPT}' + \Delta}{t} \leq \frac{c \cdot (\text{OPT} + \Delta)}{t} \stackrel{(2)}{\leq} c \cdot \text{OPT} + c \cdot \varepsilon, \\ \text{ALG} &= \frac{\text{ALG}' + \Delta}{t} \geq \frac{\text{OPT}' + \Delta}{t} \stackrel{(2)}{\geq} \text{OPT} - \varepsilon.\end{aligned}$$

In this case the algorithm should return  $\lfloor \text{ALG} + \varepsilon \rfloor$ . This solution is always feasible as  $\text{OPT} \leq \lfloor \text{ALG} + \varepsilon \rfloor$ . Moreover, the solution is  $c(1+2\varepsilon)$ -approximate: If  $\text{OPT} = 0$ , then  $\lfloor \text{ALG} + \varepsilon \rfloor = 0$  (if we set  $\varepsilon < \frac{1}{2}$ ). If  $\text{OPT} \geq 1$ , then  $\lfloor \text{ALG} + \varepsilon \rfloor \leq c \cdot \text{OPT} + (c+1)\varepsilon \leq c(1+2\varepsilon)\text{OPT}$ . We may again set  $\varepsilon$  small enough to obtain approximation ratio  $c + \varepsilon'$ , for any  $\varepsilon' > 0$ . ◀

---

## References

- 1 Amir Abboud and Arturs Backurs. Towards hardness of approximation for polynomial time problems. In *Proceedings of the 8th Conference on Innovations in Theoretical Computer Science*, volume 67 of *ITCS '17*, pages 11:1–11:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.
- 2 Amir Abboud and Aviad Rubinfeld. Fast and deterministic constant factor approximation algorithms for LCS imply new circuit lower bounds. In *Proceedings of the 9th Conference on Innovations in Theoretical Computer Science*, volume 94 of *ITCS '18*, pages 35:1–35:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
- 3 Amir Abboud, Aviad Rubinfeld, and Ryan Williams. Distributed PCP theorems for hardness of approximation in P. In *Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer Science*, FOCS '17, pages 25–36. IEEE Computer Society, 2017.
- 4 Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 218–230. SIAM, 2015.
- 5 Josh Alman, Timothy Chan, and Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In *Proceedings of the 57th IEEE Annual Symposium on Foundations of Computer Science*, FOCS '16, pages 467–476. IEEE Computer Society, 2016.
- 6 Arturs Backurs, Liam Roditty, Gilad Segal, Virginia Vassilevska Williams, and Nicole Wein. Towards tight approximation bounds for graph diameter and eccentricities. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing*, STOC '18, pages 267–280. ACM, 2018.
- 7 Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly sub-quadratic algorithms unless SETH fails. In *Proceedings of the 55th IEEE Annual Symposium on Foundations of Computer Science*, FOCS '15, pages 661–670. IEEE Computer Society, 2014.

- 8 Karl Bringmann, Alejandro Cassis, Nick Fischer, and Marvin Künnemann. Fine-grained completeness for optimization in P. *CoRR*, abs/2107.01721, 2021. URL: <http://arxiv.org/abs/2107.01721>.
- 9 Karl Bringmann, Nick Fischer, and Marvin Künnemann. A fine-grained analogue of Schaefer’s theorem in P: Dichotomy of  $\exists^k\forall$ -quantified first-order graph properties. In *Proceedings of the 34th Computational Complexity Conference*, volume 137 of *CCC ’19*, pages 31:1–31:27. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.
- 10 Karl Bringmann, Marvin Künnemann, and Karol Wegrzycki. Approximating APSP without scaling: Equivalence of approximate min-plus and exact min-max. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing*, STOC ’19, pages 943–954. ACM, 2019.
- 11 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 *Proceedings of the 7th ACM Conference on Innovations in Theoretical Computer Science*, ITCS ’16, pages 261–270. ACM, 2016.
- 12 Timothy M. Chan and Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov-Smolensky. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA ’16, pages 1246–1255. SIAM, 2016.
- 13 Lijie Chen. On the hardness of approximate and exact (bichromatic) maximum inner product. *Theory of Computing*, 16(4):1–50, 2020. doi:10.4086/toc.2020.v016a004.
- 14 Lijie Chen, Shafi Goldwasser, Kaifeng Lyu, Guy N. Rothblum, and Aviad Rubinfeld. Fine-grained complexity meets IP = PSPACE. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA ’19, pages 1–20. SIAM, 2019.
- 15 Lijie Chen and Ryan Williams. An equivalence class for orthogonal vectors. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA ’19, pages 21–40. SIAM, 2019.
- 16 Martin Dietzfelbinger, Philipp Schlag, and Stefan Walzer. A subquadratic algorithm for 3XOR. In *Proceedings of the 43rd International Symposium on Mathematical Foundations of Computer Science*, volume 117 of *MFCS ’18*, pages 59:1–59:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
- 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):146, 2020. doi:10.3390/a13060146.
- 18 Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and Ryan Williams. Completeness for first-order properties on sparse structures with algorithmic applications. *ACM Trans. Algorithms*, 15(2):23:1–23:35, 2019.
- 19 Erich Grädel, Phokion G. Kolaitis, Leonid Libkin, Maarten Marx, Joel Spencer, Moshe Y. Vardi, Yde Venema, and Scott Weinstein. *Finite Model Theory and Its Applications*. Springer Berlin Heidelberg, 2007.
- 20 Neil Immerman. *Descriptive Complexity*. Springer New York, 1999.
- 21 Zahra Jafargholi and Emanuele Viola. 3SUM, 3XOR, triangles. *Algorithmica*, 74(1):326–343, 2016.
- 22 C. S. Karthik, Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. *J. ACM*, 66(5):33:1–33:38, 2019.
- 23 Karthik C. S. and Pasin Manurangsi. On closest pair in euclidean metric: Monochromatic is as hard as bichromatic. In *Proceedings of the 10th Conference on Innovations in Theoretical Computer Science*, volume 124 of *ITCS ’19*, pages 17:1–17:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.
- 24 Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43(3):425–440, 1991.
- 25 Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, STOC ’13, pages 515–524. ACM, 2013.

- 26 Dhruv Rohatgi. Conditional hardness of earth mover distance. In Dimitris Achlioptas and László A. Végh, editors, *Proceedings of Approximation, Randomization, and Combinatorial Optimization (APPROX/RANDOM'19)*, volume 145 of *LIPIcs*, pages 12:1–12:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.12.
- 27 J. Barkley Rosser and Lowell Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois J. Math.*, 6(1):64–94, March 1962. doi:10.1215/ijm/1255631807.
- 28 Aviad Rubinfeld. Hardness of approximate nearest neighbor search. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing, STOC '18*, pages 1260–1268. ACM, 2018.
- 29 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians, ICM '18*, pages 3447–3487, 2018.

## A Reducing $\text{OptSP}_k$ Formulas to the Hybrid Problem

In this section we give the first phase of the reduction, where we reduce  $\text{OPT}(\psi)$  to the Hybrid Problem. The main lemma is the following. As before, let  $s(m) \leq m^{1/6}$  be a nondecreasing function and let  $c \geq 1$  be constant.

► **Lemma 17.** *Let  $k \geq 2$ . If the Hybrid Problem can be  $c$ -approximated in time  $O(m^k/s(m))$ , then  $\text{OPT}(\psi)$  can be  $c$ -approximated in time  $O(m^k/s(\sqrt[k+1]{m}))$ , for any  $\text{OPTSP}_k$  formula  $\psi$ .*

Recall that we only have to deal with  $\text{OPTSP}_k = \text{OPTSP}_{k,1}$  formulas, as any  $\text{OPTSP}_{k,\ell}$  problem with  $\ell > 1$  directly admits an improved algorithm; see [8, Appendix B] in the full version. As explained in Section 3, we prove Lemma 17 by progressively simplifying  $\text{OPT}(\psi)$  in four steps:

1. Remove all *hyperedges*, that is,  $\psi$  no longer contains predicates of arity  $\geq 3$  so an instance of  $\text{OPT}(\psi)$  can be thought of as a (colored) graph.
2. Remove all *cross edges*, that is, edges between vertices  $x_i$  and  $x_j$  that we maximize over.
3. Remove all *parallel edges* (or alternatively, colored edges), that is, we combine parallel edges into simple edges.
4. Remove unary predicates, finally turning the  $\text{OPT}(\psi)$  instances into graphs. At this point it becomes simple to rewrite  $\text{OPT}(\psi)$  as a Hybrid Problem.

### Step 1: Removing Hyperedges

As a first step, we eliminate all *hyperpredicates*, that is, predicates of arity  $\geq 3$ . Formally, we prove the following lemma.

► **Lemma 18.** *Suppose that, for any  $\text{OPTSP}_k$  formula  $\psi$  not containing hyperpredicates,  $\text{OPT}(\psi)$  can be  $c$ -approximated in time  $O(m^k/s(m))$ . Then  $\text{OPT}(\psi)$  can be  $c$ -approximated in time  $O(m^k/s(m))$  for any  $\text{OPTSP}_k$  formula  $\psi$ .*

The proof is quite similar to [18, Section 7]. We start with a technical lemma:

► **Lemma 19.** *Let*

$$\psi = \text{opt}_{x_1, \dots, x_k} \# \left( E(x_i, x_j) \wedge \phi(x_1, \dots, x_k, y) \right),$$

*for some  $i, j \in [k], i \neq j$  and arbitrary  $\phi$ . Then  $\text{OPT}(\psi)$  can be solved exactly in time  $O(m^{k-1/2})$ .*



**Proof.** Let us begin with the simplest case  $k = 2$ . For a vertex  $x$  in the given instance, let  $\deg(x)$  denote the total number of records containing  $x$  over all relations. We distinguish the following three cases:

**Case 1:**  $\deg(x_1) \geq \sqrt{m}$ . We explicitly list all vertices  $x_1$  with  $\deg(x_1) \geq \sqrt{m}$ ; note that there can be at most  $O(\sqrt{m})$  such elements since the sparsity of the  $\text{MAX}(\psi)$  instance is bounded by  $O(m)$ . The remaining  $\text{MAXSP}_1$  formula can be solved in time  $O(m)$  using the baseline algorithm. In total, this step takes time  $O(\sqrt{m}m) = O(m^{3/2})$ .

**Case 2:**  $\deg(x_2) \geq \sqrt{m}$ . By exchanging the roles of  $x_1$  and  $x_2$ , we deal with this case in the same way as case 1.

**Case 3:**  $\deg(x_1) < \sqrt{m}$  and  $\deg(x_2) < \sqrt{m}$ . Assuming that the previous two cases were executed, we can assume that  $\deg(x_1) < \sqrt{m}$  and  $\deg(x_2) < \sqrt{m}$  for all remaining objects  $x_1, x_2$ . We exploit that any non-zero solution  $(x_1, x_2)$  of  $\text{MAX}(\psi)$  satisfies  $E(x_1, x_2)$ : It suffices to maximize over all  $O(m)$  edges  $E(x_1, x_2)$ , counting the number of  $y$ 's satisfying  $\phi$ . Since  $\deg(x_1) < \sqrt{m}$  and  $\deg(x_2) < \sqrt{m}$ , we can enumerate and test all objects  $y$  which are connected to either  $x_1$  or  $x_2$  by some relation in time  $O(\sqrt{m})$ . What remains are objects  $y$  not connected to either  $x_1$  or  $x_2$  by any relation. To account for these missing objects, we can substitute *false* for all non-unary predicates in  $\phi$ ; what remains is a Boolean function over unary predicates over  $y$ . We can precompute the number of  $y$ 's satisfying that function in linear time, so again the total time is  $O(m + m\sqrt{m}) = O(m^{3/2})$ .

It remains to lift this proof to the general case  $k > 2$ . We brute-force over all  $x$ -variables except for  $x_i$  and  $x_j$ . This amounts for a factor  $O(m^{k-2})$  in the running time. What remains is a  $\text{MAXSP}_2$  formula in the shape as before which can be solved exactly in time  $O(m^{3/2})$  by the previous case analysis. In total this takes time  $O(m^{k-2}m^{3/2}) = O(m^{k-1/2})$ . The proof works in exactly the same way for minimization problems. ◀

**Proof of Lemma 18.** Let  $\psi = \max_{x_1, \dots, x_k} \#_y \phi(x_1, \dots, x_k, y)$  be a  $\text{MAXSP}_k$  formula possibly containing some hyperpredicates. We introduce a new binary relation  $N(x_i, x_j)$  defined as follows: For any  $x_i, x_j \in V$  it holds that  $N(x_i, x_j) = \text{true}$  if and only if  $x_i$  and  $x_j$  are connected by some (hyper-)edge. Observe that any (hyper-)edge contributes to at most a constant number of records  $N(x_i, x_j)$ , so we can construct  $N$  in time  $O(m)$  and the sparsity blows up only by a constant factor. We can now rewrite  $\psi$  via

$$\psi_0 = \max_{x_1, \dots, x_k} \#_y \left( \left( \bigwedge_{i \neq j} \bar{N}(x_i, x_j) \right) \wedge \phi_0(x_1, \dots, x_k, y) \right)$$

and

$$\psi_{i,j} = \max_{x_1, \dots, x_k} \#_y \left( N(x_i, x_j) \wedge \phi(x_1, \dots, x_k, y) \right),$$

where  $\phi_0$  is obtained from  $\phi$  by replacing all occurrences of hyperpredicates by *false*. It follows that we can express

$$\text{OPT} = \max\{\text{OPT}_0, \max_{i \neq j} \text{OPT}_{i,j}\},$$

where  $\text{OPT}_0$  is the optimal value of  $\psi_0$ , and  $\text{OPT}_{i,j}$  is the optimal value of  $\psi_{i,j}$ . Observe that  $\psi_0$  is a  $\text{MAXSP}_k$  formula not involving any hyperpredicates, so we can by assumption  $c$ -approximate  $\text{OPT}_0$  in time  $T(m)$ . Moreover, the formulas  $\psi_{i,j}$  are precisely in the shape to apply Lemma 19, so we can compute  $\text{OPT}_{i,j}$  exactly in time  $O(m^{k-1/2})$ . ◀

**Step 2: Removing Cross Edges**

Next, the goal is to remove all binary predicates  $E(x_i, x_j)$  between two  $x$ -variables. Let us call these predicates  $E(x_i, x_j)$  *cross predicates* and the associated entries  $(x_i, x_j)$  *cross edges*.

► **Lemma 20.** *Suppose that, for any  $\text{OPTSP}_k$  formula  $\psi$  not containing hyperpredicates and cross predicates,  $\text{OPT}(\psi)$  can be  $c$ -approximated in time  $O(m^k/s(m))$ . Then  $\text{OPT}(\psi)$  can be  $c$ -approximated in time  $O(m^k/s({}^{k+1}\sqrt{m}))$  for any  $\text{OPTSP}_k$  formula  $\psi$  not containing hyperpredicates.*

**Proof.** Let  $\psi = \max_{x_1, \dots, x_k} \#_y \phi(x_1, \dots, x_k, y)$  and let  $E_1, \dots, E_r$  denote the cross predicates in the given instance. We define

$$\psi_0 := \max_{x_1, \dots, x_k} \#_y \left( \left( \bigwedge_{\ell, i, j} \bar{E}_\ell(x_i, x_j) \right) \wedge \phi_0(x_1, \dots, x_k, y) \right)$$

and

$$\psi_{\ell, i, j} := \max_{x_1, \dots, x_k} \#_y \left( E_\ell(x_i, x_j) \wedge \phi(x_1, \dots, x_k, y) \right),$$

where  $\ell \in [r]$  and  $i \neq j \in [k]$  and  $\phi_0$  is the propositional formula obtained from  $\phi$  by substituting all predicates  $E_\ell(x_i, x_j)$  by *false*. It is easy to verify that

$$\text{OPT} = \max\{\text{OPT}_0, \max_{\ell, i, j} \text{OPT}_{\ell, i, j}\},$$

where  $\text{OPT}_0$  and  $\text{OPT}_{\ell, i, j}$  are the optimal values of  $\text{MAX}(\psi_0)$  and  $\text{MAX}(\psi_{\ell, i, j})$ , respectively. Using Lemma 19, we can compute  $\text{OPT}_{\ell, i, j}$  exactly in time  $O(m^{k-1/2})$  for all  $\ell, i, j$ . It remains to efficiently solve  $\text{MAX}(\psi_0)$  to compute  $\text{OPT}_0$ .

As described before, we can always assume that each variable ranges over a separate set:  $x_i \in X_i, y \in Y$ . We call a vertex  $x_i$  *heavy* if it has degree at least  ${}^{k+1}\sqrt{m}$ , and *light* otherwise. The first step is to eliminate all heavy vertices; there can exist at most  $O(m/{}^{k+1}\sqrt{m})$  many such vertices  $x_i$ . Fixing  $x_i$ , we can solve the remaining problem in time  $O(m^{k-1})$  using the baseline algorithm. We keep track of the optimal solution detected in this way. This precomputation step takes time  $O(m^k/{}^{k+1}\sqrt{m})$  and afterwards we can safely remove all heavy vertices.

Next, partition each set  $X_i$  into several groups  $X_{i,1}, \dots, X_{i,g}$  such that the total degree of all vertices in a group is  $O({}^{k+1}\sqrt{m})$ , and the number of groups is  $g = O(m/{}^{k+1}\sqrt{m})$ . This is implemented by greedily inserting vertices into  $X_{i,j}$  until its total degree exceeds  ${}^{k+1}\sqrt{m}$ . As each vertex inserted in that way is light, we can overshoot by at most  ${}^{k+1}\sqrt{m}$ .

Let  $\psi_1 := \max_{x_1, \dots, x_k} \#_y \phi_0(x_1, \dots, x_k, y)$ ; note that  $\psi_1$  equals  $\psi_0$  except that it disregards the cross predicates. Therefore, by assumption we can  $c$ -approximate  $\text{MAX}(\psi_1)$  in time  $O(m^k/s(m))$ . The algorithm continues as follows:

1. For all combinations  $(j_1, \dots, j_k) \in [g]^k$ , compute a  $c$ -approximation of  $\text{MAX}(\psi_1)$  on the input  $X_{1,j_1}, \dots, X_{k,j_k}$ . We keep track of the  $\binom{k}{2}mn^{k-2} + 1$  combinations with largest values (breaking ties arbitrarily).
2. For any of the top-most  $\binom{k}{2}mn^{k-2} + 1$  combinations  $(j_1, \dots, j_k)$ , solve  $\text{MAX}(\psi_0)$  exactly on  $X_{1,j_1}, \dots, X_{k,j_k}$  using the baseline algorithm. Return the best solution detected in this step or the precomputation phase.

We begin with the correctness of the algorithm. First, the value of any solution  $(x_1, \dots, x_k)$  in  $\text{MAX}(\psi_0)$  is at least as large as its value in  $\text{MAX}(\psi_1)$ . In particular, the optimal solution of  $\text{MAX}(\psi_0)$  has value at least  $\text{OPT}_0$  in  $\text{MAX}(\psi_1)$ . We next establish an upper bound on

the number *false positives*, that is, tuples  $(x_1, \dots, x_k)$  of different value in  $\text{MAX}(\psi_0)$  than in  $\text{MAX}(\psi_1)$ . Observe that any such false positive contains at least one edge  $(x_i, x_j)$  and since there are at most  $m$  edges, at most  $\binom{k}{2}$  choices of  $i, j$  and at most  $n^{k-2}$  choices for the remaining vertices  $x_\ell$ ,  $\ell \neq i, j$ , we can indeed bound the number of false positives by  $\binom{k}{2}mn^{k-2}$ . Thus, if we witness the top-most  $\binom{k}{2}mn^{k-2} + 1$  solutions of  $\text{MAX}(\psi_1)$  in step 1, among these there exists at least one solution of value  $\geq \text{OPT}_0/c$  in  $\text{MAX}(\psi_0)$ .

Finally, let us bound the running time of the above algorithm. Recall that removing heavy vertices accounts for  $O(m^k / \sqrt[k+1]{m})$  time. In step 1, the  $\text{MAX}(\psi_1)$  algorithm is applied  $g^k$  times on instances of size  $O(\sqrt[k+1]{m})$ , which takes time  $O((m/\sqrt[k+1]{m})^k \cdot (\sqrt[k+1]{m})^k / s(\sqrt[k+1]{m})) = O(m^k / s(\sqrt[k+1]{m}))$ . Step 2 runs the baseline algorithm  $mn^{k-2} = O(m^{k-1})$  times on instances of size  $O(\sqrt[k+1]{m})$ , which takes time  $O(m^{k-1}(\sqrt[k+1]{m})^k) = O(m^k / \sqrt[k+1]{m})$ . Thus, the total running time is  $O(m^k / \sqrt[k+1]{m} + m^k / s(\sqrt[k+1]{m}))$ . As  $s(m) \leq m$ , this is as claimed. The proof for the maximization variant is complete and there are only minor adaptations necessary for minimization.  $\blacktriangleleft$

### Step 3: Removing Parallel Edges

After applying the previous steps we can assume that  $\psi$  is an  $\text{OPTSP}_k$  formula not containing hyperedges or cross edges. Let  $E_1, \dots, E_r$  be the binary relations featured in  $\psi$ . We say that  $\psi$  does not have *parallel edges* if  $r = 1$ . In an instance of  $\text{OPT}(\psi)$  with parallel edges, any pair of vertices  $(x_i, y)$  may be connected by up to  $r$  parallel edges, or equivalently by an edge of  $2^r$  possible *colors*. We adopt the second viewpoint for this step: Let  $\chi(x_i, y) := (E_1(x_i, y), \dots, E_r(x_i, y)) \in \{0, 1\}^r$  be the *color* of the edge  $(x_i, y)$  and let  $\chi(x_1, \dots, x_k, y) := (\chi(x_1, y), \dots, \chi(x_k, y)) \in (\{0, 1\}^r)^k$  be the color of the tuple  $(x_1, \dots, x_k, y)$ .

► **Lemma 21.** *Suppose that, for any  $\text{OPTSP}_k$  formula  $\psi$  not containing hyperedges, cross edges and parallel edges,  $\text{OPT}(\psi)$  can be  $c$ -approximated in time  $O(m^k/s(m))$ . Then  $\text{OPT}(\psi)$  can be  $c$ -approximated in time  $O(m^k/s(m))$  for any  $\text{OPTSP}_k$  formula  $\psi$  not containing hyperedges and cross edges.*

**Proof.** Let  $E_1, \dots, E_r$  denote the binary relations featured in the given instance; our goal is to construct a new instance with only a single edge predicate  $E$ . We leave the vertex sets  $X_i$  unchanged and construct  $Y' = \{y_\alpha : y \in Y, \alpha \in (\{0, 1\}^r)^k\}$ , i.e., each vertex  $y \in Y$  is copied  $2^{rk} = O(1)$  times and each copy  $y_\alpha$  is indexed by a  $k$ -tuple of colors  $\alpha = (\alpha_1, \dots, \alpha_k) \in (\{0, 1\}^r)^k$ . For every  $\alpha$  we also introduce a new unary predicate  $C_\alpha$  and assign  $C_\alpha(y_{\alpha'})$  if and only if  $\alpha = \alpha'$ .

Now let  $i \in [k]$  and let  $x_i \in X_i$  and  $y \in Y$  be arbitrary vertices in the original instance. We assign the edges in the constructed instance as follows. If  $\chi(x_i, y) = 0 = (0, \dots, 0)$ , then  $x_i$  and  $y$  are not connected and we do not introduce new edges. So suppose that  $\chi(x_i, y) \neq 0$ . Then we add  $2 \cdot 2^{r(k-1)}$  edges

- $E(x_i, y_\beta)$ , for all  $\beta \in (\{0, 1\}^r)^k$  with  $\beta_i = \chi(x_i, y)$ , and
- $E(x_i, y_\gamma)$ , for all  $\gamma \in (\{0, 1\}^r)^k$  with  $\gamma_i = 0$ .

Clearly the sparsity of the new instance is bounded by  $2 \cdot 2^{r(k-1)}m = O(m)$  plus the contribution of the new unary predicates which is also  $O(m)$ .

Now let  $\psi = \text{opt}_{x_1, \dots, x_k} \#_y \phi(x_1, \dots, x_k)$ . To define an equivalent  $\text{MAXSP}_k$  formula  $\psi'$ , for any  $\alpha \in (\{0, 1\}^r)^k$  let  $\phi_\alpha$  denote the formula obtained from  $\phi$  by substituting  $E_j(x_i, y)$  by *true* if  $\alpha_{i,j} = 1$  and by *false* otherwise. We define  $\psi' = \max_{x_1, \dots, x_k} \#_y \phi'(x_1, \dots, x_k, y)$ ,

where  $\phi'(x_1, \dots, x_k, y)$  is

$$\bigvee_{\alpha \in \{0,1\}^r} \left( \underbrace{C_\alpha(y)}_{(i)} \wedge \underbrace{\left( \bigwedge_{i \in [k]} (E(x_i, y) \iff \alpha_i \neq 0) \right)}_{(ii)} \wedge \underbrace{\phi_\alpha(x_1, \dots, x_k, y)}_{(iii)} \right).$$

As desired, the constructed instance contains only a single binary predicate and no cross or hyperedges. It remains to argue that the value of any tuple  $(x_1, \dots, x_k)$  is not changed by the reduction. Indeed, for all  $y \in Y$  we prove the following two conditions and thereby the claim.

- $\phi'(x_1, \dots, x_k, y_\alpha) = \phi(x_1, \dots, x_k, y)$  for  $\alpha = \chi(x_1, \dots, x_k, y)$ ,
- $\phi'(x_1, \dots, x_k, y_\alpha) = \text{false}$  for all  $\alpha \neq \chi(x_1, \dots, x_k, y)$ .

The first bullet is simple to verify: In the evaluation of  $\phi'(x_1, \dots, x_k, y_\alpha)$  we only have to focus on the  $\alpha$ -disjunct by the constraint (i). The constraint (ii) is satisfied by our construction of  $E$  and therefore only (iii) is decisive:  $\phi'(x_1, \dots, x_k, y_\alpha) = \phi_\alpha(x_1, \dots, x_k, y_\alpha) = \phi(x_1, \dots, x_k, y)$ . Next, focus on the second bullet. For  $\alpha \neq \chi(x_1, \dots, x_k, y)$  there exists some index  $i$  such that  $\alpha_i \neq \chi(x_i, y)$ . By (i), we again only need to consider the  $\alpha$ -disjunct. We now prove that  $E(x_i, y) \iff \alpha_i = 0$  which falsifies (ii) and shows  $\phi'(x_1, \dots, x_k, y_\alpha) = \text{false}$ . On the one hand, if  $\alpha_i \neq 0$  then there is no edge  $E(x_i, y_\alpha)$ , since  $0 \neq \alpha_i \neq \chi(x_i, y)$ . On the other hand, if  $\alpha_i = 0$  then we added an edge  $E(x_i, y_\alpha)$ . ◀

#### Step 4: Removing Unary Predicates

As the final simplification, we eliminate unary predicates and show that the resulting problem can be reduced to the Hybrid Problem.

**Proof of Lemma 17.** By applying the reductions in Lemmas 18, 20 and 21, it suffices to show that any  $\text{OPTSP}_k$  property  $\psi$  not containing hyperpredicates, cross edge predicates and parallel edge predicates can be reduced to the Hybrid Problem. The shape of  $\psi$  is significantly restricted and contains only the following three types of relations: Unary predicates on  $X_1, \dots, X_k$ , unary predicates on  $Y$  and binary predicates of the form  $E(x_i, y)$  for  $i \in [k]$ .

We can assume that there are no unary predicates on  $X_1, \dots, X_k$  as follows: By enumerating all possible assignments of these unary predicates, and by restricting the sets  $X_1, \dots, X_k$  to those vertices matching the current assignment, we create a constant number of instances each without unary predicates on  $X_1, \dots, X_k$ .

This leaves only unary predicates on  $Y$  and the edge predicates  $E(x_i, y)$ . Let  $\psi = \text{opt}_{x_1, \dots, x_k} \#_y \phi(x_1, \dots, x_k, y)$ . Another way to view this problem is associate a Boolean function  $\phi_y : \{0, 1\}^k \rightarrow \{0, 1\}$  to every vertex  $y \in Y$ , which takes as input  $E(x_i, y)$  and does no longer depend on the unary predicates of  $y$ . In that way, we can rewrite the objective as

$$\text{opt}_{x_1, \dots, x_k} \#_y \phi_y(E(x_1, y), \dots, E(x_k, y)).$$

Our goal is now to reinterpret this problem as an instance of the Hybrid Problem. As the universe, we assign

$$U = \left\{ (y, \tau) : y \in Y, \tau \in \{0, 1\}^k \text{ is a satisfying assignment of } \phi_y \right\},$$

along with the partition  $U = \bigcup_{\tau \in \{0,1\}^k} U_\tau$ ,  $U_\tau = U \cap (Y \times \{\tau\})$ . For every vertex  $x_i \in X_i$ , we construct a set  $S_i \in \mathcal{S}_i$  as  $S_i = \{(y, \tau) : E(x_i, y)\} \cap U$ . It is easy to check that the value of every solution is preserved in this way:  $\text{Val}(S_1, \dots, S_k) = \text{Val}(x_1, \dots, x_k)$ . The overhead of this rewriting step is  $O(m)$  and thus negligible in the running time bound. ◀

# An Estimator for Matching Size in Low Arboricity Graphs with Two Applications

Hossein Jowhari   

Department of Computer Science and Statistics, Faculty of Mathematics, K. N. Toosi University of Technology, Tehran, Iran

---

## Abstract

In this paper, we present a new degree-based estimator for the size of maximum matching in bounded arboricity graphs. When the arboricity of the graph is bounded by  $\alpha$ , the estimator gives a  $\alpha + 2$  factor approximation of the matching size. For planar graphs, we show the estimator does better and returns a 3.5 approximation of the matching size. Using this estimator, we get new results for approximating the matching size of planar graphs in the streaming and distributed models of computation. In particular, in the vertex-arrival streams, we get a randomized  $O(\frac{\sqrt{n}}{\varepsilon^2} \log n)$  space algorithm for approximating the matching size of a planar graph on  $n$  vertices within  $(3.5 + \varepsilon)$  factor. Similarly, we get a simultaneous protocol in the vertex-partition model for approximating the matching size within  $(3.5 + \varepsilon)$  factor using  $O(\frac{n^{2/3}}{\varepsilon^2} \log n)$  communication from each player. In comparison with the previous estimators, the estimator in this paper does not need to know the arboricity of the input graph and improves the approximation factor in the case of planar graphs.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Streaming, sublinear and near linear time algorithms

**Keywords and phrases** Data Stream Algorithms, Maximum Matching, Planar Graphs

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.10

**Category** APPROX

**Related Version** *Previous Version:* <https://arxiv.org/pdf/2011.11706.pdf>

**Funding** *Hossein Jowhari:* This work is supported by the Iranian Institute for Research in Fundamental Sciences (IPM), Project Number 98050014.

## 1 Introduction

A matching in a graph  $G = (V, E)$  is a subset of edges  $M \subseteq E$  where no two edges in  $M$  share an endpoint. A maximum matching of  $G$  has the maximum number of edges among all possible matchings. Let  $m(G)$  denote the *matching size* of  $G$ , in other words the size of a maximum matching in  $G$ . In this paper, we present algorithms for approximating  $m(G)$  in the sublinear models of computation. In particular, our results are for the vertex-arrival stream model (also known as the adjacency list streams). In the vertex-arrival model, the input stream is an arbitrary ordering of vertex set  $V$ . Additionally, followed by each  $u \in V$  in the stream, the algorithm also gets the list of the neighbors of  $u$ . This is in contrast with the *edge-arrival* version where the input stream is an arbitrary ordering of the edge set  $E$ .

The problem of estimating  $m(G)$  or computing an approximate maximum matching of  $G$  in the data stream model has been studied in several works [14, 13, 10, 6, 17]. Here we focus on algorithms for graphs with bounded arboricity. A graph  $G = (V, E)$  has arboricity bounded by  $\alpha$  if the edge set  $E$  can be partitioned into at most  $\alpha$  forests. A well-known fact (known as the Nash-William theorem [19]) states that a graph has arboricity  $\alpha$ , if and only if every induced subgraph on  $t$  vertices has at most  $\alpha(t - 1)$  number of edges. Graphs with low arboricity cover a wide range of graphs such as graphs with constant degree, planar graphs, and graphs with small tree-widths. In particular planar graphs have arboricity bounded by 3.



© Hossein Jowhari;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 10; pp. 10:1–10:13



Leibniz International Proceedings in Informatics

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

A simple reduction from counting distinct elements implies that the exact computation of  $m(G)$  in the data stream model has  $\Omega(n)$  space complexity even for trees and randomized algorithms (see [1] for the lower bound on distinct elements problem.) This negative result has inspired a growing interest in finding estimators for  $m(G)$  that take sublinear space to compute. Specially when the input graph  $G$  has low arboricity, it has been shown by Esfandiari *et al.* [9], and the subsequent works [16, 7, 18, 4], that it is possible to compute  $m(G)$  approximately in  $o(n)$  space by only checking the degree information and the immediate local neighborhood of the vertices (and the edges). This line of research has led to the invent of several *degree-based* estimators for the matching size. In this paper, we design another degree-based estimator for  $m(G)$  in low arboricity graphs that has certain advantages in comparison with the previous works and leads to new algorithmic results. Before describing our estimator we briefly review the previous ideas.

## 1.1 Previous Works

In the following we assume  $G$  has arboricity bounded by  $\alpha$ . Also, unless explicitly stated, all the algorithms mentioned below are randomized.

### Shallow edges, high degree vertices

Esfandiari *et al.* [9] were first to observe that one can approximately characterize the matching size of low arboricity graphs based on the degree information of the vertices and the local neighborhood of the edges. Let  $H$  denote the set of vertices with degree more than  $h = 2\alpha + 3$  and let  $F$  denote the set of edges with both endpoints having degree at most  $h$ . Esfandiari *et al.* have shown that  $m(G) \leq |H| + |F| \leq (5\alpha + 9)m(G)$ . Based on this estimator, the authors in [9] have designed a  $\tilde{O}(\varepsilon^{-2}\alpha n^{2/3})$  space algorithm for approximating  $m(G)$  within  $5\alpha + 9 + \varepsilon$  factor in the edge-arrival model.

### Fractional matchings

By establishing an interesting connection with fractional matchings and the Edmonds Polytope theorem, McGregor and Vorotnikova [16] have shown that the following quantity approximates  $m(G)$  within  $(\alpha + 2)$  factor.

$$(\alpha + 1) \sum_{(u,v) \in E} \min\left\{\frac{1}{\deg(u)}, \frac{1}{\deg(v)}, \frac{1}{\alpha + 1}\right\}.$$

Based on this estimator, the authors in [16] have given a  $\tilde{O}(\varepsilon^{-2}n^{2/3})$  space streaming algorithm (in the edge-arrival model) that approximate  $m(G)$  within  $\alpha + 2 + \varepsilon$  factor. Also in the same work, another degree-based estimator is given that returns a  $\frac{(\alpha+2)^2}{2}$  factor approximation of  $m(G)$ . A notable property of this estimator is that it can be implemented *deterministically* in the vertex-arrival model using only  $O(\log n)$  bits of space.

### $\alpha$ -Last edges

Cormode *et al.* [7] (later revised by McGregor and Vorotnikova [18]) have designed an improved estimator for  $m(G)$  that also depends on a given ordering of the edges. Given a stream of edges  $S = e_1, \dots, e_m$ , let  $E_\alpha(S)$  denote a subset of edges where  $(u, v) \in E_\alpha(S)$  iff the vertices  $u$  and  $v$  both appear at most  $\alpha$  times in  $S$  after the edge  $(u, v)$ . It is shown that  $m(G) \leq |E_\alpha(S)| \leq (\alpha + 2)m(G)$ . Moreover a  $O(\frac{1}{\varepsilon^2} \log^2 n)$  space streaming algorithm is shown that approximates  $|E_\alpha(S)|$  within  $1 + \varepsilon$  factor in the edge-arrival model.

## 1.2 The estimator in this paper

The new estimator is purely based on the degree of the vertices in the graph without any need to have a pre-knowledge of  $\alpha$ . To estimate the matching size, we count the number of what we call *locally superior* vertices in the graph. Consider the following definition.

► **Definition 1.** In graph  $G = (V, E)$ , we call  $u \in V$  a *locally superior vertex* if  $u$  has a neighbor  $v$  such that  $\deg(u) \geq \deg(v)$ . We let  $\ell(G)$  denote the number of locally superior vertices in  $G$ .

We show when the arboricity of  $G$  is bounded by  $\alpha$ ,  $\ell(G)$  approximates  $m(G)$  within  $(\alpha + 2)$  factor (Lemma 2.) This repeats the same bound obtained by the estimators in [16] and [7], however for planar graphs, we prove the approximation factor is at most 3.5 which beats the previous bounds (Lemma 5). This result is the main technical contribution of the paper. As an evidence on the improved approximation quality, consider the 4-regular planar graph on 9 vertices. Both of the estimators in [16] and [7], report 18 as the estimation for  $m(G)$  while the exact answer is 4. It follows their approximation factor is at least 4.5.

As a first application of Lemma 5, we obtain a randomized  $O(\frac{\sqrt{n}}{\varepsilon^2} \log n)$  space streaming algorithm for approximating  $m(G)$  within  $(3.5 + \varepsilon)$  factor in the vertex-arrival model. In terms of approximation factor, this improves over existing sub-linear algorithms [16, 18].

As another application of our estimator, we get a sublinear simultaneous protocol in the *vertex-partition* model for approximating  $m(G)$  when  $G$  is planar. In this model, the vertex set  $V$  is partitioned into  $t$  subsets  $V_1, \dots, V_t$  where each subset is given to a player. Additionally the  $i$ -th player knows the edges on  $V_i$ . The players do not communicate with each other. They only send one message to a *referee* whom at the end computes an approximation of the matching size. (The referee does not get any part of the input.) We assume the referee and the players have a shared source of randomness. Within this setting, we design a protocol that approximates  $m(G)$  within  $3.5 + \varepsilon$  factor using  $O(\frac{n^{2/3}}{\varepsilon^2} \log n)$  communication from each player. Note that for  $t > 3$  and  $t = o(n^{1/3})$ , this result is non-trivial. The best previous result, implicit in the works of [5, 16] computes a  $5 + \varepsilon$  factor approximation using  $\tilde{O}(n^{4/5})$  communication from each player. Also we should mention that, using the estimator in [16], one can get a deterministic simultaneous protocol where each player sends only  $O(\log n)$  bits to the referee. However this protocol computes a 12.5 factor approximation of  $m(G)$ .

## 1.3 Related Works

Kapralov *et al.* [13] have given an estimator for  $m(G)$  when  $G$  is a general graph. Their estimator gives an approximation of  $m(G)$  by looking at the degree information of the vertices in a series of nested subgraphs of  $G$ . The main challenge is implementing the estimator in sublinear space. Based on this estimator, Kapralov *et al.* has shown one can obtain a  $\text{poly}(\log n)$  approximation of  $m(G)$  in  $\text{poly}(\log n)$  space assuming the input edge stream is randomly ordered. There are subsequent works with improved results and analysis [14]. As far as we know, there is no similar result for arbitrarily ordered streams.

There is a large body of works that addresses the problem of finding a matching of near optimal size using  $O(n \cdot \text{poly}(\log n))$  space. This falls within the category of semi-streaming model. See the recent works [12, 3] for the latest results on this.

Maximum matching has also been studied within the context of distributed local algorithms [15] and massively parallel computations [2]. The main objective of these works is to find a large matching of near optimal size in a distributed manner using small number of communication rounds. See the works [8, 11] for related results on graphs with bounded arboricity.

## 2 Graph properties

In the following proofs, we let  $M \subseteq E$  denote a maximum matching in graph  $G$ . When the underlying graph is clear from the context, for the vertex set  $S$ , we use  $N(S)$  to denote the neighbors of the vertices in  $S$  excluding  $S$  itself. For vertex  $u$ , we simply use  $N(u)$  to denote the neighbors of  $u$ . The vertex  $v$  is a neighbor of the edge  $(x, y)$  if  $v$  is adjacent with  $x$  or  $y$ . When  $x$  is paired with  $y$  in the matching  $M$ , abusing the notation, we define  $M(x) = y$ .

► **Lemma 2.** *Let  $G = (V, E)$  be a graph with arboricity  $\alpha$ . We have*

$$m(G) \leq \ell(G) \leq (\alpha + 2)m(G).$$

**Proof.** The left hand side of the inequality is easy to show. For every edge in  $E$ , at least one of the endpoints is locally superior. Since edges in  $M$  are disjoint, at least  $|M|$  number of endpoints must be locally superior. This proves  $m(G) \leq \ell(G)$ .

To show the right hand side, we use a charging argument. Let  $L$  denote the locally superior vertices in  $G$ . Our goal is to show an upper bound on  $|L|$  in terms of  $|M|$  and  $\alpha$ . Let  $X \subseteq L$  be the set of locally superior vertices that are NOT endpoints of a matching edge. The challenge is to prove an upper on  $|X|$ .

The vertices in  $X$  do not contribute to the maximum matching. However all the vertices in  $N(X)$  must be endpoints of matching edges (otherwise  $M$  would not be maximal.) For the same reason, there cannot be an edge between the vertices in  $X$ . To prove an upper bound on  $|X|$ , in the first step, we assign a subset of vertices in  $X$  to edges in  $M$  in a way any target edge gets at most  $\alpha - 1$  locally superior vertices. We do the assignments in the following way.

### The Assignment Procedure

If we find a  $y \in N(X)$  with at most  $\alpha - 1$  neighbors in  $X$ , we assign all the neighbors of  $y$  in  $X$  to the matching edge  $(y, M(y))$ . We repeat this process, every time picking a vertex in  $N(X)$  with less than  $\alpha$  neighbors in  $X$  and do the assignment that we just described, until we cannot find such a vertex in  $N(X)$ . Note that when we assign a locally superior vertex  $x$ , we remove the edges on  $x$  before continuing the procedure.

Here we emphasize the fact that if  $y$  has a neighbor  $x \in X$ , then  $M(y)$  cannot have neighbors in  $X \setminus \{x\}$  (otherwise it would create an augmenting path and contradict with the optimality of  $M$ .)

Let  $X_1 \subseteq X$  be the assigned locally superior vertices and  $M_1 \subseteq M$  be the used matching edges in the assignment procedure. We have

$$|X_1| \leq (\alpha - 1)|M_1|. \tag{1}$$

Let  $X_2 = X \setminus X_1$  be the unassigned vertices in  $X$ . Now we try to prove an upper bound on  $|X_2|$ . For this, we need to make a few observations.

► **Observation 3.** *Let  $Y_2 = N(X_2)$ . The pair  $y$  and  $M(y)$  cannot be both in  $Y_2$ .*

**Proof.** Suppose  $y$  and  $M(y)$  are both in  $N(X_2)$ . Let  $B$  and  $C$  be the neighbors of  $y$  and  $M(y)$  in  $X_2$  respectively. If  $|B \cup C| > 1$ , then one can find an augmenting path of length 3 (with respect to  $M$ .) A contradiction.

On the other hand, if  $|B \cup C| = 1$ , then  $y$  and  $M(y)$  have only a shared neighbor  $x \in X_2$  which means the edge  $e = (y, M(y))$  should have been used by the assignment procedure and as result  $x \in X_1$ . Another contradiction. ◀



► **Observation 4.** *Every vertex  $x \in X_2$  has degree at least  $\alpha + 1$ .*

**Proof.** Consider  $x \in X_2$ . Suppose, for the sake of contradiction,  $\deg(x)$  is  $k$  where  $k \leq \alpha$ . Since  $x$  is a locally superior vertex, there must be a  $y \in N(x)$  with degree at most  $k$  in  $G$ . We know that  $y$  is an endpoint of a matching edge. In the assignments procedure, whenever we used an edge  $e \in M$  all the neighbors of its endpoints (in  $X$ ) were assigned. Since  $x$  is not assigned yet, it means the edge  $(y, M(y))$  has not been used. Consequently  $y$  must have at least  $\alpha$  neighbors in  $X_2$ . Counting the edge  $(y, M(y))$ , we should have  $\deg(y) \geq \alpha + 1$ . A contradiction. ◀

Let  $G' = (X_2 \cup Y_2, E')$  be a bipartite graph where  $E'$  is the set of edges between  $X_2$  and  $Y_2$ . From Observation 4, we have

$$(\alpha + 1)|X_2| \leq |E'|. \quad (2)$$

Since  $G'$  is a subgraph of  $G$ , its arboricity is bounded by  $\alpha$ . As result,

$$|E'| \leq \alpha(|X_2| + |Y_2|). \quad (3)$$

Recall that  $Y_2$  are endpoints of matching edges. Let  $M_2$  be those matching edges. Observation 3 implies that  $|Y_2| = |M_2|$ . As result, combining (2) and (3), we get the following.

$$|X_2| \leq \alpha|Y_2| = \alpha|M_2|. \quad (4)$$

To prove an upper bound on  $|L|$ , we also need to count the locally superior vertices that are endpoints of matching edges. Let  $Z = L \setminus X$ . We have  $|Z| \leq 2|M|$ . Summing up, we get

$$\begin{aligned} |L| &= |X_1| + |X_2| + |Z| \\ &\leq (\alpha - 1)|M_1| + \alpha|M_2| + 2|M| \\ &= \alpha(|M_1| + |M_2|) + 2|M| - |M_1| \\ &\leq (\alpha + 2)|M| - |M_1| \\ &\leq (\alpha + 2)|M| \end{aligned}$$

This proves the lemma. ◀

► **Lemma 5.** *Let  $G = (V, E)$  be a planar graph. We have  $\ell(G) \leq 3.5m(G)$ .*

**Proof.** For planar graphs, similar to what we did in the proof of Lemma 2, we first try to assign some of the vertices in  $X$  to the matching edges using a simple assignment procedure. (Recall that  $X$  is the set of vertices in  $L$  that are not endpoints of edges in  $M$ .)

### The Assignment Procedure

Let  $Y_1 = \emptyset$ . If we find a  $y \in N(X)$  with only 1 neighbor  $x \in X$ , we assign  $x$  to the matching edge  $(y, M(y))$ . Also we add  $y$  to  $Y_1$ . We continue the procedure until we cannot find such a vertex in  $N(X)$ . Note that when we assign a locally superior vertex  $x$ , we remove the edges on  $x$ .

Let  $X_1 \subseteq X$  be the assigned locally superior vertices and  $M_1 \subseteq M$  be the used matching edges in the assignment procedure. Note that  $|Y_1| = |M_1|$ . We have

$$|X_1| \leq |M_1|. \quad (5)$$

## 10:6 An Estimator for Matching Size in Low Arboricity Graphs with Two Applications

Let  $X_2 = X \setminus X_1$ . Using a similar argument that we used for proving Observation 4, we can show every vertex in  $X_2$  has degree at least 3. Also letting  $Y_2 = N(X_2)$ , we observe that  $y \in Y_2$  and  $M(y)$  cannot be both in  $Y_2$  as we noticed in the Observation 3. Let  $M_2 \subseteq M$  be the matching edges with one endpoint in  $Y_2$ . We have  $|Y_2| = |M_2|$ .

Now consider the bipartite graph  $G' = (X_2 \cup Y_2, E')$  where  $E'$  is the set of edges between  $X_2$  and  $Y_2$ . Every planar bipartite graph with  $n$  vertices has at most  $2n - 4$  edges<sup>1</sup>. Since  $G'$  is a bipartite planar graph, it follows,

$$3|X_2| \leq |E'| < 2(|X_2| + |Y_2|) = 2(|X_2| + |M_2|). \quad (6)$$

This shows  $|X_2| < 2|M_2|$ . Letting  $Z = L \setminus X$  and  $M_3 = M \setminus (M_1 \cup M_2)$ , we get

$$|L| = |X_1| + |X_2| + |Z| \leq |M_1| + 2|M_2| + 2|M| \leq 3|M| + |M_2| - |M_3|. \quad (7)$$

This already proves  $|L|$  is bounded by  $4|M|$ . To prove the bound claimed in the lemma, we also show that  $|L| \leq 3|M| + |M_1| + |M_3|$ . Combined with the inequality (7), this proves the lemma.

Let  $Y = Y_1 \cup Y_2$ . Note that  $Y$  are one side of the matching edges in  $M_1 \cup M_2$ . Let  $Y' = \{M(y) \mid y \in Y\}$ . We use a special subset of  $Y'$ , named  $Y''$  which is defined as follows. We let  $Y''$  denote the locally superior vertices in  $Y'$  that have degree 2 or they are adjacent with both endpoints of an edge in  $M_3$ . We make the following observation regarding the vertices in  $Y''$ .

► **Observation 6.** *We can assign each vertex  $y' \in Y''$  to a distinct  $e \in Y_1 \cup M_3$  where  $e$  has no neighbor in  $Y' \setminus \{y'\}$ .*

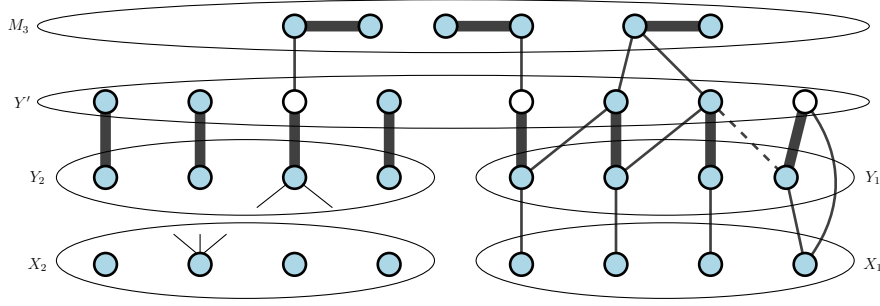
**Proof.** Consider  $y' \in Y''$ . If  $y'$  is adjacent with both endpoints of an edge  $e = (z, z') \in M_3$ , we assign  $y'$  to  $e$  (when there are multiple edges with this condition we pick one of them arbitrarily.) Note that  $z$  and  $z'$  cannot have neighbors in  $Y'$  other than  $y'$  because otherwise it would create an augmenting path.

Now suppose  $y'$  has degree 2. Since  $y'$  is a locally superior vertex, it must have a neighbor  $z$  of degree at most 2. The neighbor  $z$  cannot be in  $Y_2 \cup X_2$  because the vertices in  $Y_2 \cup X_2$  have degree at least 3. We distinguish between two cases.

- $M(y') \in Y_2$ . In this case,  $z$  cannot be in  $Y_1$  either because the vertices in  $Y_1$  are already of degree 2 without  $y'$ . Also  $z \notin X_1$  because otherwise it would create an augmenting path. The only possibility is that  $z$  is an endpoint of a matching edge in  $M_3$ . We assign  $y'$  to the matching edge  $(z, z') \in M_3$ . Note that  $z'$  cannot have a neighbor in  $Y' \setminus \{y'\}$  because it would create an augmenting path.
- $M(y') \in Y_1$ . Here  $z$  could be in  $X_1$ . If this is the case, then  $M(y')$  cannot have a neighbor in  $Y' \setminus \{y'\}$  because it would create an augmenting path. In this case, we assign  $y'$  to  $M(y')$ . If  $z = M(y')$ , then again we assign  $y'$  to  $M(y')$ . The only remaining possibility is that  $z$  an endpoint of a matching edge in  $M_3$  which we handle it similar to the previous case. ◀

Now, assume we assign the vertices in  $Y''$  to the elements in  $Y_1 \cup M_3$  according to the above observation. Let  $Y'_1 \subseteq Y_1$  and  $M'_3 \subseteq M_3$  be the vertices and edges that were used in the assignment. Let  $Y'''$  be the remaining locally superior vertices in  $Y'$ . Namely,  $Y''' = (L \cap Y') \setminus Y''$ . Before making the final point, we observe that only one endpoint of the

<sup>1</sup> For a short proof of this, combine the Euler's formula  $|V| - |E| + |F| = 2$  with the inequality  $2|E| \geq 4|F|$  caused by each face having at least 4 sides (since there are no odd cycles) and we get  $|E| \leq 2|V| - 4$ .



■ **Figure 1** A demonstration of the construction in the proof of lemmas 2 and 5. Thick edges represent matching edges. The unfilled vertices belong to the set  $Y''$ .

edges in  $M_3$  are adjacent with vertices in  $Y'''$ . Let  $Y_3$  be the endpoint of edges in  $M_3 \setminus M'_3$  that have neighbors in  $Y'''$ . Consider the bipartite graph  $G''(V'', E'')$  where

$$V'' = (X_2 \cup Y''') \cup (Y_2 \cup (Y_1 \setminus Y'_1) \cup Y_3)$$

and  $E''$  is the set of edges between  $X_2$  and  $Y_2$ , and the edges between  $Y'''$  and  $Y_2 \cup (Y_1 \setminus Y'_1) \cup Y_3$ .

Relying on the facts that  $G''$  is a planar bipartite graph,  $Y'''$  is composed of vertices with degree at least 3, and the edges on  $Y'''$  are all in  $E''$ , we have

$$3|X_2| + 3|Y'''| \leq |E''| \leq 2(|X_2| + |Y_2| + |Y_1 \setminus Y'_1| + |Y'''| + |Y_3|).$$

It follows,

$$\begin{aligned} |X_2| + |Y'''| &\leq 2(|Y_2| + |Y_1 \setminus Y'_1| + |Y_3|) \\ &\leq 2(|M_2| + |M_1| - |Y'_1| + |M_3| - |M'_3|) \\ &= 2(|M| - |Y'_1| - |M'_3|) \end{aligned}$$

Since  $|Y''| = |Y'_1| + |M'_3|$ , we get

$$|X_2| + |Y'''| \leq 2|M| - 2|Y''| \quad (8)$$

Let  $Z_1$ ,  $Z_2$  and  $Z_3$  denote the locally superior vertices that are endpoints of matching edges in  $M_1$ ,  $M_2$  and  $M_3$  respectively. From the definition of  $Y''$  and  $Y'''$ , we have

$$|Z_1| + |Z_2| \leq |M_1| + |M_2| + |Y''| + |Y'''| \quad (9)$$

From (8) and (9), we get

$$\begin{aligned} |L| &= |X_1| + |X_2| + |Z_1| + |Z_2| + |Z_3| \\ &\leq |M_1| + |X_2| + (|M_1| + |M_2| + |Y''| + |Y'''|) + 2|M_3| \\ &= 2|M_1| + (|X_2| + |Y'''|) + |M_2| + |Y''| + 2|M_3| \\ &\leq 2|M_1| + |M_2| + 2|M| - |Y''| + 2|M_3| \\ &= 3|M| + |M_1| + |M_3| - |Y''| \\ &\leq 3|M| + |M_1| + |M_3| \end{aligned}$$

This finishes the proof of the lemma. ◀

### 3 Algorithms

We first present a high-level sampling-based estimator for  $\ell(G)$ . Then we show how this estimator can be implemented in the streaming and distributed settings using small space and communication. For our streaming result, we use a combination of the estimator for  $\ell(G)$  and the greedy maximal matching algorithm. For the simultaneous protocol, we use the estimator for  $\ell(G)$  in combination with the edge-sampling primitive in [5] and an estimator in [16].

The high-level estimator (described in Algorithm 1) samples a subset of vertices  $S \subseteq V$  and computes the locally superior vertices in  $S$ . The quantity  $\ell(G)$  is estimated from the scaled ratio of the locally superior vertices in the sample set.

■ **Algorithm 1** The high-level description of the estimator for  $\ell(G)$ .

---

Run the following estimator  $r = \lceil \frac{8}{\varepsilon^2} \rceil$  number of times in parallel. In the end, report the average of the outcomes.

1. Sample  $s$  vertices (uniformly at random) from  $V$  without replacement.
  2. Let  $S$  be the set of sampled vertices.
  3. Compute  $S'$  where  $S'$  is the set of locally superior vertices in  $S$ .
  4. Return  $\frac{n}{s}|S'|$  as an estimation for  $\ell(G)$ .
- 

► **Lemma 7.** *Assuming  $s \geq \frac{n}{\ell(G)}$ , the high-level estimator in Algorithm 1 returns a  $1 + \varepsilon$  factor approximation of  $\ell(G)$  with probability at least  $7/8$ .*

**Proof.** Fix a parallel repetition of the algorithm and let  $X$  denote the outcome of the associated estimator. Assuming an arbitrary ordering on the locally superior vertices, let  $X_i$  denote the random variable associated with  $i$ -th locally superior vertex. We define  $X_i = 1$  if the  $i$ -th locally superior vertex has been sampled, otherwise  $X_i = 0$ . We have  $X = \frac{n}{s} \sum_{i=1}^{\ell(G)} X_i$ . Since  $Pr(X_i = 1) = \frac{s}{n}$ , we get  $E[X] = \ell(G)$ . Further we have

$$\begin{aligned} E[X^2] &= \frac{n^2}{s^2} E\left[\sum_{i,j} X_i X_j\right] = \frac{n^2}{s^2} \left[ \sum_i E[X_i^2] + \sum_{i \neq j} E[X_i X_j] \right] \\ &= \frac{n^2}{s^2} \left[ \frac{s}{n} \ell(G) + \binom{\ell(G)}{2} \frac{s(s-1)}{n(n-1)} \right] \\ &= \frac{n}{s} \ell(G) + \binom{\ell(G)}{2} \frac{n(s-1)}{s(n-1)} \\ &< \frac{n}{s} \ell(G) + \ell^2(G) \end{aligned}$$

Consequently,  $Var[X] = E[X^2] - E^2[X] < \frac{n}{s} \ell(G)$ .

Let  $Y$  be the average of the outcomes of  $r$  parallel and independent repetitions of the basic estimator. We have  $E[Y] = \ell(G)$  and  $Var[Y] < \frac{n}{sr} \ell(G)$ . Using the Chebyshev's inequality,

$$Pr(|Y - E[Y]| \geq \varepsilon E[Y]) \leq \frac{Var[Y]}{\varepsilon^2 E^2[X]} < \frac{n/s}{r \varepsilon^2 \ell(G)}.$$

Setting  $r = \frac{8}{\varepsilon^2}$  and  $s \geq \frac{n}{\ell(G)}$ , the above probability will be less than  $1/8$ . ◀

### 3.1 The streaming algorithm

We first note that we can implement the high-level estimator of Algorithm 1 in the vertex-arrival stream model using  $O(\frac{s}{\epsilon^2} \log n)$  space. Consider a single repetition of the estimator. The sampled set  $S$  is selected in the beginning of the algorithm (before the stream.) This can be done using a reservoir sampling strategy [20] in  $O(|S| \log n)$  space. To decide if  $u \in S$  is locally superior or not, we just need to store  $\deg(u)$  and the minimum degree of the neighbors that are visited so far. Note that when processing a vertex  $v \in V$  and its neighbors, we know if  $v$  is a neighbor of  $u$  or not. Consequently, checking if  $u$  is a locally superior or not takes  $O(\log n)$  bits of space. Therefore the whole space needed to implement a single repetition is  $O(s \log n)$  bits.

The streaming algorithm runs two threads in parallel. In one thread it runs the streaming implementation of Algorithm 1 after setting  $s = \lceil \sqrt{n} \rceil$ . In the other thread, it runs a greedy algorithm to find a maximal matching in the input graph. We stop the greedy algorithm whenever the size of the discovered matching  $F$  exceeds  $\sqrt{n}$ . In the end, if  $|F| < \sqrt{n}$ , we output  $|F|$  as an approximation for  $m(G)$ , otherwise we report the outcome of the first thread.

Note that if  $|F| < \sqrt{n}$ ,  $F$  is a maximal matching in  $G$ . Hence  $|F| \geq \frac{1}{2}m(G)$ . Assume  $|F| \geq \sqrt{n}$ . In this case the algorithm outputs the result of first thread. In this case, by Lemma 2, we know  $\ell(G) \geq \sqrt{n}$ . Consequently, by Lemma 7, the first thread returns a  $1 + O(\epsilon)$  approximation of  $\ell(G)$  and hence it returns a  $3.5 + O(\epsilon)$  approximation of  $m(G)$ . Since the greedy algorithm takes at most  $O(\sqrt{n})$  space, the space complexity of the algorithm is dominated by the space usage of the first thread. We get the following result.

► **Theorem 8.** *Let  $G$  be a planar graph. There is a randomized streaming algorithm in the vertex-arrival model that returns a  $3.5 + \epsilon$  factor approximation of  $m(G)$  using  $O(\frac{\sqrt{n}}{\epsilon^2})$  space.*

### 3.2 A simultaneous communication protocol

In this section we describe a communication protocol for approximating  $m(G)$  in the vertex-partition model. Recall that in this model the vertex set  $V$  is partitioned into  $t$  subsets  $V_1, \dots, V_t$  where the subset  $V_i$  is given to the  $i$ -th player. Additionally the  $i$ -th player knows the edges on the vertices in  $V_i$ . The players do not communicate with each other. They only send one message to a referee whom at the end computes an approximation of the matching size. Also we emphasize the assumption that the referee and the players have a shared source of randomness.

To describe the simultaneous protocol, we consider two cases separately: (a) when the matching size is low; to be precise, when it is smaller than some fixed value  $k = n^{1/3}$ , and (b) when the matching size is high, *i.e.* at least  $\Omega(k)$ . For each case, we describe a separate solution. The overall protocol will be the parallel run of these two solutions along with a sub-protocol to distinguish between the cases.

#### Graphs with large matching size

In the case when matching size is large, similar to what was done in the streaming model, we run an implementation of Algorithm 1 in the given simultaneous model. To see how this is implemented, in the simultaneous model all the players (including the referee) know the sampled set  $S$ . This results from access to the shared randomness. For each  $u \in S$ , the players send the minimum degree of the neighbors of  $u$  in his input to the referee. The player that owns  $u$ , also sends  $\deg(u)$  to the referee. Having received this information, the referee can decide if  $u$  is a locally superior vertex or not. As result, we can implement Algorithm 1 in the simultaneous model using a protocol with  $O(\frac{s}{\epsilon^2} \log n)$  message size.

## 10:10 An Estimator for Matching Size in Low Arboricity Graphs with Two Applications

### Graphs with small matching size

In the case where the matching size is small, we use the edge-sampling method of [5]. Here we review their basic sampling primitive in its general form. Given a graph  $G(V, E)$ , let  $c : V \rightarrow [b]$  be a totally random function that assigns each vertex in  $V$  a random number (color) in  $[b] = \{1, \dots, b\}$ . The set  $\text{Sample}_{b,d,1}$  is a random subset of  $E$  picked in the following way. Given a subset  $K \subseteq [b]$  of size  $d \in \{1, 2\}$ , let  $E_K$  be the edges of  $G$  where the color of their endpoints matches  $K$ . For example when  $K = \{3, 4\}$ , the set  $E_{\{3,4\}}$  contains all edges  $(u, v)$  such that  $\{c(u), c(v)\} = \{3, 4\}$ . For all  $K \subseteq [b]$  of size  $d$ , the set  $\text{Sample}_{b,d,1}$  picks a random edge from  $E_K$ . Finally, the random set  $\text{Sample}_{b,d,r}$  is the union of  $r$  independent instances of  $\text{Sample}_{b,d,1}$ . We have the following lemma from [5] (see Theorems 4 in the reference.)

► **Lemma 9.** *Let  $G = (V, E)$  be a graph. Assuming  $m(G) \leq k$ , with probability  $1 - 1/\text{poly}(k)$ , the random set  $\text{Sample}_{100k,2,O(\log k)}$  contains a matching of size  $m(G)$ .*

Note that, in the simultaneous vertex-partition model, the referee can obtain an instance of  $\text{Sample}_{b,d,1}$  via a protocol with  $O(b^d \log n)$  message size. To see this, using the shared randomness, the players pick the random function  $c : V \rightarrow [b]$ . Let  $E^{(i)}$  be the subset of edges owned by the  $i$ -th player. We have  $E = \bigcup_{i=1}^t E^{(i)}$ . To pick a random edge from  $E_K$  for a given  $K \subseteq [b]$ , the  $i$ -th player randomly picks an edge  $e \in E_K \cap E^{(i)}$  and sends it along with  $|E_K \cap E^{(i)}|$  to the referee. After receiving this information from all the players, the referee can generate a random element of  $E_K$ . Since there are  $O(b^d)$  different  $d$ -subsets of  $[b]$ , the size of the message from a player to the referee is bounded by  $O(b^d \log n)$  bits. Consequently, the referee can produce a rightful instance of  $\text{Sample}_{b,d,r}$  using  $O(rb^d \log n)$  communication from each player.

### How to distinguish between the cases?

For this task, we use a degree-based estimator by McGregor and Vorotnikova [16] described in the following lemma.

► **Lemma 10.** *Let  $G = (V, E)$  be a planar graph. Let  $A'(G) = \sum_{u \in V} \min\{\deg(u)/2, 4 - \deg(u)/2\}$ . We have*

$$m(G) \leq A'(G) \leq 12.5 m(G).$$

It is easy to see that, in the simultaneous vertex-partition model, we can implement this estimator with  $O(\log n)$  bits communication from each player.

### The final protocol

Let  $k = \lceil n^{1/3} \rceil$ . We run the following threads in parallel.

1. A protocol that implements the high-level estimator (Algorithm 1) with  $s = \lceil 12.5n/k \rceil$  as its input parameter according to the discussions above. Let  $z_1$  be the output of this protocol.
2. A protocol to compute an instance of  $\text{Sample}_{b,d,r}$  for  $b = 100k$  and  $d = 2$  and  $r = O(\log k)$ . Let  $z_2$  be the size of maximum matching in the sampled set.
3. A protocol to compute  $A'(G)$ . Let  $z_3$  be the output of this thread.

In the end, if  $z_3 \geq \frac{k}{12.5}$ , the referee outputs  $z_1$  as an approximation for  $m(G)$ , otherwise the referee reports  $z_2$  as the final answer.

► **Theorem 11.** *Let  $G$  be a planar graph on  $n$  vertices. The above simultaneous protocol, with probability  $3/4$ , returns a  $3.5 + O(\varepsilon)$  approximation of  $m(G)$  where each player sends  $O(\frac{n^{2/3}}{\varepsilon^2})$  bits to the referee.*

**Proof.** First we note that by choosing the constants large enough, we can assume the thread (2) errs with probability at most  $1/8$ . If  $z_3 \geq \frac{k}{12.5}$ , then we know  $m(G) \geq \frac{k}{12.5}$ . This follows from Lemma 10. Consequently by Lemma 2, we have  $\ell(G) \geq \frac{k}{12.5}$ . Therefore from Lemma 7, we have  $|z_1 - \ell(G)| \leq \varepsilon \ell(G)$  with probability at least  $7/8$ . It follows from Lemma 5 that  $(1 - \varepsilon)m(G) \leq z_1 \leq (3.5 + 3.5\varepsilon)m(G)$ .

On the other hand, if  $z_3 < \frac{k}{12.5}$ , by Lemma 10 we know that  $m(G)$  must be less than  $k$ . Having this, from Lemma 9, with probability at least  $7/8$ , we get  $z_2 = m(G)$ . In this case the protocol computes the exact matching size of the graph.

The message size of each player is dominated by the cost of the first thread which is  $O(n^{2/3}\varepsilon^{-2} \log n)$ . The total error probability is bounded by  $1/4$ . This finishes the proof. ◀

## 4 Conclusion

In this paper we presented a degree-based estimator for the size of maximum matching in planar graphs. We showed our estimator gives a 3.5 factor approximation of the matching size. This improves the approximation factor of the previous degree-based estimators. We do not have tight examples for our analysis. In fact, we conjecture that  $\ell(G)$  approximates  $m(G)$  within 3 factor when  $G$  is planar.

Using our estimator, we obtained an improved sublinear space algorithm for estimating the matching size in the vertex-arrival streams. We also showed a more efficient simultaneous protocol for estimating the matching size in planar graphs. Unfortunately, the new estimator, in spite of its simplicity, does not immediately lead to one-pass sublinear algorithm in the edge-arrival model. To decide if a vertex is locally superior, we need to know its neighbors and learn their degrees which becomes burdensome in one pass. However, given an extra pass over the stream the same space bound and approximation factor is achievable for the edge-arrival streams as well. It would be interesting to do this without the extra pass.

---

## References

- 1 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999. doi:10.1006/jcss.1997.1545.
- 2 Soheil Behnezhad, MohammadTaghi Hajiaghayi, and David G. Harris. Exponentially faster massively parallel maximal matching. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1637–1649. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00096.
- 3 Aaron Bernstein. Improved bounds for matching in random-order streams. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPICs*, pages 12:1–12:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ICALP.2020.12.
- 4 Marc Bury, Elena Grigorescu, Andrew McGregor, Morteza Monemizadeh, Chris Schwegelshohn, Sofya Vorotnikova, and Samson Zhou. Structural results on matching estimation with applications to streaming. *Algorithmica*, 81(1):367–392, 2019. doi:10.1007/s00453-018-0449-y.
- 5 Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with

- applications to finding matchings and related problems in dynamic graph streams. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1326–1344. SIAM, 2016. doi:10.1137/1.9781611974331.ch92.
- 6 Rajesh Hemant Chitnis, Graham Cormode, Mohammad Taghi Hajiaghayi, and Morteza Monemizadeh. Parameterized streaming: Maximal matching and vertex cover. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1234–1251. SIAM, 2015. doi:10.1137/1.9781611973730.82.
  - 7 Graham Cormode, Hossein Jowhari, Morteza Monemizadeh, and S. Muthukrishnan. The sparse awakens: Streaming algorithms for matching size estimation in sparse graphs. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, pages 29:1–29:15, 2017. doi:10.4230/LIPIcs.ESA.2017.29.
  - 8 Andrzej Czygrinow, Michal Hanckowiak, and Edyta Szymanska. Fast distributed approximation algorithm for the maximum matching problem in bounded arboricity graphs. In Yingfei Dong, Ding-Zhu Du, and Oscar H. Ibarra, editors, *Algorithms and Computation, 20th International Symposium, ISAAC 2009, Honolulu, Hawaii, USA, December 16-18, 2009. Proceedings*, volume 5878 of *Lecture Notes in Computer Science*, pages 668–678. Springer, 2009. doi:10.1007/978-3-642-10631-6\_68.
  - 9 Hossein Esfandiari, Mohammad Taghi Hajiaghayi, Vahid Liaghat, Morteza Monemizadeh, and Krzysztof Onak. Streaming algorithms for estimating the matching size in planar graphs and beyond. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1217–1233. SIAM, 2015. doi:10.1137/1.9781611973730.81.
  - 10 Hossein Esfandiari, Mohammad Taghi Hajiaghayi, and Morteza Monemizadeh. Finding large matchings in semi-streaming. In Carlotta Domeniconi, Francesco Gullo, Francesco Bonchi, Josep Domingo-Ferrer, Ricardo Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu, editors, *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain*, pages 608–614. IEEE Computer Society, 2016. doi:10.1109/ICDMW.2016.0092.
  - 11 Mohsen Ghaffari, Christoph Grunau, and Ce Jin. Improved MPC algorithms for mis, matching, and coloring on trees and beyond. In Hagit Attiya, editor, *34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference*, volume 179 of *LIPIcs*, pages 34:1–34:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.DISC.2020.34.
  - 12 Mohsen Ghaffari and David Wajc. Simplified and space-optimal semi-streaming  $(2+\epsilon)$ -approximate matching. In Jeremy T. Fineman and Michael Mitzenmacher, editors, *2nd Symposium on Simplicity in Algorithms, SOSA 2019, January 8-9, 2019, San Diego, CA, USA*, volume 69 of *OASICS*, pages 13:1–13:8. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/OASICS.SOSA.2019.13.
  - 13 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 734–751. SIAM, 2014. doi:10.1137/1.9781611973402.55.
  - 14 Michael Kapralov, Slobodan Mitrovic, Ashkan Norouzi-Fard, and Jakab Tardos. Space efficient approximation to maximum matching size from uniform edge samples. 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 1753–1772. SIAM, 2020. doi:10.1137/1.9781611975994.107.
  - 15 Zvi Lotker, Boaz Patt-Shamir, and Seth Pettie. Improved distributed approximate matching. *J. ACM*, 62(5):38:1–38:17, 2015. doi:10.1145/2786753.



- 16 A. McGregor and S. Vorotnikova. Planar matching in streams revisited. In *Proceedings of the 19th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, 2016.
- 17 Andrew McGregor. Finding graph matchings in data streams. In Chandra Chekuri, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, editors, *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th International Workshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, volume 3624 of *Lecture Notes in Computer Science*, pages 170–181. Springer, 2005. doi:10.1007/11538462\_15.
- 18 Andrew McGregor and Sofya Vorotnikova. A simple, space-efficient, streaming algorithm for matchings in low arboricity graphs. In *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA*, pages 14:1–14:4, 2018. doi:10.4230/OASIcs.SOSA.2018.14.
- 19 C. Nash-Williams. Decomposition of finite graphs into forests. *J. London Math. Soc.*, 39(12), 1964.
- 20 Jeffrey Scott Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985. doi:10.1145/3147.3165.



# An Optimal Algorithm for Triangle Counting in the Stream

Rajesh Jayaram ✉

Carnegie Mellon University, Pittsburgh, PA, USA

John Kallaugher ✉

The University of Texas at Austin, Austin, TX, USA

---

## Abstract

We present a new algorithm for approximating the number of triangles in a graph  $G$  whose edges arrive as an arbitrary order stream. If  $m$  is the number of edges in  $G$ ,  $T$  the number of triangles,  $\Delta_E$  the maximum number of triangles which share a single edge, and  $\Delta_V$  the maximum number of triangles which share a single vertex, then our algorithm requires space:

$$\tilde{O}\left(\frac{m}{T} \cdot (\Delta_E + \sqrt{\Delta_V})\right)$$

Taken with the  $\Omega\left(\frac{m\Delta_E}{T}\right)$  lower bound of Braverman, Ostrovsky, and Vilenchik (ICALP 2013), and the  $\Omega\left(\frac{m\sqrt{\Delta_V}}{T}\right)$  lower bound of Kallaugher and Price (SODA 2017), our algorithm is optimal up to log factors, resolving the complexity of a classic problem in graph streaming.

**2012 ACM Subject Classification** Theory of computation → Sketching and sampling

**Keywords and phrases** Triangle Counting, Streaming, Graph Algorithms, Sampling, Sketching

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.11

**Category** APPROX

**Related Version** *Full Version:* <https://arxiv.org/abs/2105.01785>

**Funding** *Rajesh Jayaram:* Rajesh Jayaram would like to acknowledge partial support from the Office of Naval Research (ONR) under grant Number N00014-18-1-2562, and the National Science Foundation (NSF) under Grant Number CCF-1815840.

*John Kallaugher:* John Kallaugher would like to acknowledge support from the National Science Foundation (NSF) under Grant Number CCF-1751040 (CAREER).

## 1 Introduction

Triangle counting is a fundamental problem in the study of graph algorithms, and one of the best studied in the field of graph streams. It arises in the analysis of social networks [5], web graphs [11], and spam detection [3], among other applications. From a theoretical perspective, it is of particular interest as the simplest subgraph counting problem that cannot be solved by considering only *local* information about individual vertices. In other words, counting triangles requires one to aggregate information between pairs of *non-incident* edges.

In this paper, we present an optimal algorithm for counting triangles in the *graph streaming* setting, settling a long line of work on this problem.



© Rajesh Jayaram and John Kallaugher;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 11; pp. 11:1–11:11



Leibniz International Proceedings in Informatics

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

## 1.1 Graph Streaming

In the (insertion-only) graph streaming setting, a graph  $G = (V, E)$  is received as a stream of edges  $(\sigma_t)_{t=1}^m$  from its edge set  $E$  in an arbitrary order, and an algorithm is required to output the answer to some problem at the end of the stream, using as little space as possible<sup>1</sup>. Variants on this model include turnstile streaming (in which edges may be deleted as well as inserted), and models that restrict what kind of state the algorithm may maintain.

## 1.2 Triangle Counting in Graph Streams

The theoretical study of graph streaming was initiated by [2], who studied the problem of triangle counting – the problem of estimating the number of three-cliques in a graph. They demonstrated that, in general, sublinear space algorithms cannot exist for this problem; namely, in the worst case any algorithm for triangle counting in a stream must use  $\Omega(n^2)$  bits of space. On the other hand, they also showed that, if one parameterizes in terms of the number of triangles  $T$ , one can often beat this pessimistic lower bound. In particular, they gave an algorithm that uses  $\tilde{O}\left(\frac{mn}{T}\right)^3$  space to count triangles in a graph with  $m$  edges,  $n$  vertices, and  $T$  triangles, based on streaming algorithms for approximating frequency moments.<sup>2</sup> Of course, it is unreasonable to assume that an algorithm knows the number of triangles  $T$  in advance, as this would make counting superfluous. Instead, it will suffice to have constant factor bounds on the parameters in question.<sup>3</sup>

Several years later, the upper bound for this problem was improved to  $\tilde{O}\left(\frac{mn}{T}\right)$  by [7], while [14] gave a (non-comparable) algorithm that samples edges and stores neighborhoods of their endpoints in order to find triangles, achieving  $\tilde{O}\left(\frac{md^2}{T}\right)$  space in graphs with maximum degree  $d$ . Both algorithms were later subsumed by the  $\tilde{O}\left(\frac{md}{T}\right)$  space algorithm of [23].

## 1.3 Additional Graph Parameters for Triangle Counting

Despite the large strides made by the aforementioned algorithms, none of them can achieve sublinear space, even for graphs guaranteed to have as many as  $\Omega(m)$  triangles, without bounding parameters of the graph other than  $m$  and  $T$ . This feature was shown to be necessary by [6], who constructed a family of graphs with either 0 or  $\Omega(m)$  triangles such that distinguishing between the two requires  $\Omega(m)$  space. However, this “hard instance” is an unusual graph – every triangle in it shares a single edge. This motivated the introduction of a new graph parameter  $\Delta_E$ , defined as the maximum number of triangles which share a single edge in  $G$ . When one parameterizes in terms of  $\Delta_E$ , the lower bound of [6] becomes  $\Omega\left(\frac{m\Delta_E}{T}\right)$ . As it happens, the maximum degree of graphs in this family is also  $\Delta_E$ , so in particular this proves [23] to be optimal among algorithms parametrized by only  $m$ ,  $d$ , and  $T$ .

<sup>1</sup> Other properties, such as update time, are also of interest, but space has been the primary object of study in the theory of streaming.

<sup>2</sup> Here we assume the desired approximation is a multiplicative  $(1 \pm \varepsilon)$  with success probability  $\delta$  for some positive constants  $\varepsilon, \delta$ . For most algorithms mentioned here, including our own, the dependence on non-constant  $\varepsilon, \delta$  will go as  $\varepsilon^{-2} \log \delta^{-1}$ . We use  $\tilde{O}(\cdot)$  to suppress logarithmic or polylogarithmic factors in the argument.

<sup>3</sup> One might hope to use these parameters adaptively, giving an algorithm that uses more space the smaller  $T$  is without needing a lower bound at the start. However, this is in general impossible, as a graph stream with few triangles and a graph stream with many triangles may be indistinguishable until the last few updates.

The first algorithm to directly take advantage of the new parameter  $\Delta_E$  was given by [24]. Their algorithm is simple: keep each edge in the stream independently with probability  $p$ , count the number of triangles  $T'$  in the resulting graph, and output  $T'p^{-3}$ . They show that setting  $p = O(\frac{1}{T^{1/3}} + \frac{\Delta_E}{T})$  suffices for an accurate count, and thereby achieve  $\tilde{O}(m(\frac{1}{T^{1/3}} + \frac{\Delta_E}{T}))$  space.

This algorithm has another important feature: it is a *non-adaptive sampling* algorithm – whether it keeps an edge it sees does not depend on the contents of the stream before the edge arrives. This means it can naturally handle *turnstile streams*, streams in which edges may be deleted as well as inserted. In fact, through the use of sketches for  $\ell_0$  sampling (see e.g. [10]) such algorithms may be converted into *linear sketches*, which are algorithms that store only a linear function of their input (when considered as a vector in  $\{0, 1\}^{\binom{V}{2}}$ ).

An improved non-adaptive sampling algorithm was given in [22], which used the technique of coloring vertices with one of  $k$  colors, and keeping all monochromatic edges. This improved the space usage of the algorithm to  $\tilde{O}(m(\frac{1}{\sqrt{T}} + \frac{\Delta_E}{T}))$ . In [17], it was shown (in combination with the existing lower bound of [6]) that this is optimal, even for insertion-only algorithms – for every  $T$  up to  $\Omega(m)$ , a family of graphs exist with  $\Delta_E \leq 1$  and either 0 or  $T$  triangles, such that  $\Omega(\frac{m}{\sqrt{T}})$  space is required to distinguish the two.

However, as with the lower bound of [6], the hard instance from [17] is a rather strange graph: this time every triangle shares a single *vertex*. Also similarly to the lower bound of [6], the bound from [17] weakens as the maximum number of triangles sharing a single vertex, a parameter denoted by  $\Delta_V$ , is restricted. In this case, when parameterized by  $\Delta_V$ , the lower bound becomes  $\Omega(\frac{m\sqrt{\Delta_V}}{T})$ . This was accompanied in [17] by an algorithm that achieves  $\tilde{O}(m(\frac{1}{T^{2/3}} + \frac{\sqrt{\Delta_V}}{T} + \frac{\Delta_E}{T}))$  space, improving on [22] for graphs with  $\Delta_V = o(T)$ .

Subsequently, it was shown in [15] that any linear sketching algorithm for counting triangles requires  $\Omega(\frac{m}{T^{2/3}})$  space, even if every triangle is disjoint from every other and therefore  $\Delta_E = \Delta_V \leq 1$ , and so the [17] algorithm is optimal among linear sketches. By the turnstile streaming-linear sketching equivalence of [20], this suggests that [17] is also optimal among turnstile streaming algorithms.<sup>4</sup>

However, this leaves open the question of how hard triangle counting is for algorithms that are *not* required to handle deletions (i.e., the standard “insertion-only” model). We resolve this question (up to a log factor, as with previous optimality results), by giving an optimal algorithm for triangle counting in insertion-only streams.

## 1.4 Our Algorithm

We give a new algorithm for counting triangles in insertion-only graph streams. For every  $\varepsilon, \delta \in (0, 1)$ , there is an algorithm for insertion-only graph streams that approximates the number of triangles in a graph  $G$  to  $\varepsilon T$  accuracy with probability  $1 - \delta$ , using

$$O\left(\frac{m}{T} \left(\Delta_E + \sqrt{\Delta_V}\right) \log n \frac{\log \frac{1}{\delta}}{\varepsilon^2}\right)$$

bits of space, where  $m$  is the number of edges in  $G$ ,  $T$  the number of triangles,  $\Delta_E$  the maximum number of triangles which share a single edge, and  $\Delta_V$  the maximum number of triangles which share a single vertex.

<sup>4</sup> However, the [20] equivalence depends on rather stringent conditions that a turnstile algorithm must satisfy. In [18], it was shown that relaxing these conditions allows turnstile streaming algorithms for triangle counting that are closer to the result of [14].

## 11:4 An Optimal Algorithm for Triangle Counting in the Stream

■ **Table 1** Best known upper and lower bounds for triangle counting for insertion-only and linear sketching algorithms.  $m$  is the number of edges,  $T$  the number of triangles,  $d$  the maximum degree, and  $\Delta_E, \Delta_V$  are the maximum number of triangles sharing an edge or a vertex respectively. Note that linear sketching upper bounds imply insertion-only upper bounds, while lower bounds are the opposite.

Paper	Space	Model
[23]	$\tilde{O}\left(\frac{md}{T}\right)$	Insertion-only
[6]	$\Omega\left(\frac{m\Delta_E}{T}\right)$	Insertion-only
[17]	$\Omega\left(\frac{m\sqrt{\Delta_V}}{T}\right)$	Insertion-only
[22]	$\tilde{O}\left(m\left(\frac{1}{\sqrt{T}} + \frac{\Delta_E}{T}\right)\right)$	Linear Sketching
[17]	$\tilde{O}\left(m\left(\frac{1}{T^{2/3}} + \frac{\sqrt{\Delta_V}}{T} + \frac{\Delta_E}{T}\right)\right)$	Linear Sketching
[15]	$\Omega\left(\frac{m}{T^{2/3}}\right)$	Linear Sketching
This work	$\tilde{O}\left(\frac{m}{T}(\sqrt{\Delta_V} + \Delta_E)\right)$	Insertion-only

This matches, up to a log factor (and for constant  $\varepsilon, \delta$ ), the lower bounds of [6] and [17]. It subsumes both the algorithm of [17] and the  $\tilde{O}\left(\frac{md}{T}\right)$  algorithm of [23], as in any graph with max degree  $d$ , we have  $\Delta_E \leq d$  and  $\Delta_V \leq \binom{d}{2}$ . This closes the line of work discussed above on the complexity of triangle counting in insertion-only streams.

### 1.5 Other Related Work

In the *multi-pass* streaming setting, an algorithm is allowed to pass over the input stream more than once. [9] shows multipass algorithms take  $\tilde{O}\left(m/\sqrt{T}\right)$  space for arbitrary graphs, giving an algorithm for two passes and a lower bound for a constant number of passes. [19] shows a three pass streaming algorithm using  $O(\sqrt{m} + m^{3/2}/T)$  space. [4] gave a  $O(m^{3/2}/T)$  four pass algorithm.

In the *adjacency-list* model, in which each vertex's list of neighbors is received as a block (and so in particular every edge is seen twice), [21] gave a  $O\left(m/\sqrt{T}\right)$  space one-pass algorithm, while [16] gave  $O(m/T^{2/3})$  space 2-pass algorithm, as well as tight (but conditional on open communication complexity conjectures) lower bounds for both.

The problem has also been studied in the query model, in which case rather than space the concern is minimizing time or query count. While this is a very different setting, similar concerns around mitigating the impact of “heavy” vertices or edges arise. [12] considered triangle counting in this setting, which was extended by [13] to general cliques and [1] to arbitrary constant-size subgraphs.

## 2 Overview of the Algorithm

At a high-level, many triangle counting algorithms in the literature adhere to the following template: **(1)** design a sampling scheme to sample triangles, **(2)** count the number of triangles which survive after this sampling process, **(3)** rescale the number of empirically sampled triangles by the expected fraction of surviving triangles to obtain an unbiased estimator for  $T$ .

As an example, one could sample each edge uniformly with probability  $q$  (this is the approach taken in [24]). Since for a triangle to survive all three of its edges must be sampled, the expected number of triangles that survive is  $Tq^3$ . Thus, rescaling the number of empirically sampled triangles by  $1/q^3$  yields an unbiased estimator. How large must  $q$  be to make this estimator accurate? In order to sample even a single triangle we need  $Tq^3 \geq 1$ , so clearly  $q$  must be at least  $1/T^{1/3}$ . Moreover, if  $\Delta_E$  is the largest number of triangles that share an edge, there might be as few as  $T/\Delta_E$  “heavy” edges such that sampling a triangle requires sampling at least one of them, and so  $q$  must be at least  $\Delta_E/T$ . It turns out that, up to constant factors, this is also sufficient, and so the space needed by this algorithm is  $\tilde{O}\left(m\left(\frac{1}{T^{1/3}} + \frac{\Delta_E}{T}\right)\right)$  bits.

The starting point for our algorithm is the following simple observation, which can be seen as an optimization to the sampling algorithm above. Given three edges  $uv, vw, wu \in E$  arriving in a stream in that order, once the first two edges  $uv, vw$  have been sampled and stored, upon seeing the “completing” edge  $wu$ , we will know that the triangle  $uvw$  exists in  $G$ , and may count it immediately – we get the closing edge of each triangle “for free”. Now for a single triangle to be sampled, we only need to sample the first two edges, and so the probability of finding any given triangle improves to  $q^2$ , allowing a space complexity of  $\tilde{O}\left(m\left(\frac{1}{\sqrt{T}} + \frac{m\Delta_E}{T}\right)\right)$ . However, when  $\Delta_V = o(T)$ , this is still weaker than allowed by the  $\Omega\left(\frac{m}{T}(\sqrt{\Delta_V} + \Delta_E)\right)$  lower bound that results from combining the results of [6, 17].

While the aforementioned algorithm is sub-optimal in general, notice that it does match the lower bounds in the extreme case when  $\Delta_V = T$ , and all triangles share a single vertex. On the other hand, when  $\Delta_V$  is smaller, there are more “fully disjoint” triangles in the graph. Consequentially, we can afford to subsample by *vertices*, as now dropping a single vertex cannot lose too large a fraction of our triangles. We may sample vertices uniformly with some probability  $p$ , and deterministically store all edges adjacent to at least one sampled vertex, again counting a triangle whenever we observe an edge  $wu$  closing a sampled pair  $uv, vw$ . Each such triangle will be counted iff the “first” vertex  $v$  of the triangle is sampled, and these may be divided among as few as  $T/\Delta_V$  “heavy” vertices, so  $p$  must be at least  $\Delta_V/T$ . This again turns out to be sufficient, for a space usage of  $\tilde{O}\left(\frac{m\Delta_V}{T}\right)$  (note that any pair of edges sharing an edge also share a vertex, so  $\Delta_E \leq \Delta_V$ , and thus this does not violate the known lower bounds). While this is an improvement on the aforementioned adaptive edge-sampling scheme for small  $\Delta_V$ , it becomes worse once  $\Delta_V > \sqrt{T}$ .

The crucial insight behind our algorithm is to merge the two aforementioned algorithms with a careful choice of parameterization. Specifically, we sample both edges *and* vertices, before counting triangles that we see closing our sampled wedges. Specifically, we sample vertices  $v \in V$  in the graph with probability  $p \in (0, 1]$ , and then “activate” each edge  $e \in E$  with probability  $q \in (0, 1]$ . When an edge  $uv \in E$  arrives in the stream, we store it iff  $uv$  is active *and* at least one of the vertices  $u$  or  $v$  was sampled. We denote by  $S$  the set of all edges stored by the algorithm. Finally, when a closing edge  $wu$  arrives that completes a triangle with edges  $uv, vw$  that were previously added to  $S$ , we check if the vertex  $v$  at the center of the wedge  $uv, vw$  was sampled, and if so we deterministically increment a counter  $\mathbf{C}$ .

Now observe that, for any given triangle  $uvw$ , the probability that  $uvw$  causes  $\mathbf{C}$  to be incremented is exactly  $pq^2$ . Thus, if we output the quantity  $\mathbf{C}/(pq^2)$  at the end of the stream, we obtain an unbiased estimator for the number of triangles in  $G$ .

Notice that when  $p = 1$  our algorithm reduces to the simpler edge-sampling algorithm stated above. At the other extreme, when  $q = 1$  our algorithm reduces to the vertex sampling algorithm. Intuitively, our choice of the parameters  $p$  and  $q$  are subject to the same constraints faced by the aforementioned edge- and vertex-sampling algorithms. Firstly,  $p$

## 11:6 An Optimal Algorithm for Triangle Counting in the Stream

must be at least  $\Delta_V/T$ , otherwise the algorithm could miss a “heavy” vertex. Furthermore, the product  $pq$  must be at least  $\Delta_E/T$ , to avoid missing “heavy” edges, and  $pq^2$  must be at least  $1/T$  to find any triangles at all. Putting these bounds together, it follows that  $q$  must be at least  $\max\left\{\frac{\Delta_E}{\Delta_V}, \frac{1}{\sqrt{\Delta_V}}\right\}$ .

As with all the algorithms discussed so far, this turns out to also be sufficient – we demonstrate that by fixing the sampling parameters<sup>5</sup>

$$p = \frac{\Delta_V}{T}, \quad q \geq \max\left\{\frac{\Delta_E}{\Delta_V}, \frac{1}{\sqrt{\Delta_V}}\right\}$$

we obtain an algorithm using space  $O\left(\frac{m}{T}(\Delta_E + \sqrt{\Delta_V}) \log n\right)$  which yields an  $O(T^2)$  variance estimator. We may therefore obtain a  $(1 \pm \varepsilon)$  multiplicative estimate with probability  $1 - \delta$  by using  $O\left(\frac{1}{\varepsilon^2} \log \frac{1}{\delta}\right)$  copies of this algorithm.

Consequently one obtains an algorithm matching, up to a log factor, the lower bounds of [6, 17], with optimal space usage in terms of  $m, T, \Delta_E, \Delta_V$ .

### 3 The Triangle Counting Algorithm

Let  $G = (V, E)$  be a graph on  $n$  vertices, received as a stream of undirected edges, adversarially ordered. Let  $m$  be the number of edges in the stream. We write the stream as  $\sigma = (\sigma_i)_{i=1}^m$ , with each  $\sigma_i \in E$ . We use  $T$  to refer to the number of triangles in  $G$ ,  $\Delta_E$  to refer to the maximum number of them sharing a single edge, and  $\Delta_V$  the maximum number sharing a single vertex.

► **Remark 1.** As with all streaming triangle counting algorithms, our algorithm will need to be parametrized by statistics of the graph that cannot be known exactly without trivializing the problem – in our case  $T$ ,  $\Delta_E$ , and  $\Delta_V$ . However, it will not be necessary to know these exactly – an upper bound on  $\Delta_E$ ,  $\Delta_V$  and a lower bound on  $T$  will be sufficient. If these bounds are tight up to a constant, the complexity of our algorithm will be unchanged, otherwise replace the parameters  $T$ ,  $\Delta_E$ ,  $\Delta_V$  with the respective upper and lower bounds.

#### 3.1 Description of the Algorithm

We begin by choosing two hash functions  $\mathbf{f} : V \rightarrow \{0, 1\}$  and  $\mathbf{g} : E \rightarrow \{0, 1\}$ , which will serve as our “vertex sampling” and “edge sampling” functions, respectively. We choose  $\mathbf{f}$  to be pair-wise independent.  $\mathbf{g}$  will only be evaluated at most once for each edge, and so we may choose it to be fully independent. We pick the two functions  $\mathbf{f}, \mathbf{g}$  such that

$$\mathbb{E}[\mathbf{f}(v)] = p$$

for each  $v \in V$  and

$$\mathbb{E}[\mathbf{g}(e)] = q$$

for each  $e \in E$ , where  $p, q$  are parameters to be set later. Such a hash function  $\mathbf{f}$  can be generated by taking a two-wise independent function  $\mathbf{h} : V \rightarrow [M]$ , where  $M = \text{poly}(n)$  is a sufficiently large multiple of  $1/p$ , and setting  $\mathbf{f}(v) = 1$  whenever  $\mathbf{h}(v) \leq pM$  (one can construct  $\mathbf{g}$  similarly using a four-wise independent hash function). Such functions can be generated and stored in at most  $O(\log n)$  bits of space [8].

<sup>5</sup> As mentioned earlier,  $\Delta_E \leq \Delta_V$ , while  $\Delta_V \leq T$  holds trivially. Thus  $p, q$  are valid probabilities.



The algorithm will be simple: sample vertices with probability  $p$ , sample incident edges with probability  $q$ . The formal description is given below in Algorithm 1.

■ **Algorithm 1** Triangle Counting Algorithm.

---

```

1: procedure TRIANGLECOUNTING( $p, q$ )
2:    $S \leftarrow \emptyset$ 
3:    $\bar{\mathbf{T}} \leftarrow 0$ 
4:   for each update  $uv$  do
5:     for  $u \in V$  do
6:       if  $\mathbf{f}(u) > 0 \wedge uv, uw \in S$  then
7:          $\bar{\mathbf{T}} += 1/pq^2$ 
8:       end if
9:     end for
10:    if  $\mathbf{g}(uv)(\mathbf{f}(u) + \mathbf{f}(v)) > 0$  then
11:       $S \leftarrow S \cup \{uv\}$ 
12:    end if
13:  end for
14:  return  $\bar{\mathbf{T}}$ .
15: end procedure

```

---

### 3.2 Analysis of the Algorithm

► **Lemma 2.** *This algorithm uses  $O(mpq \log n)$  bits of space.*

**Proof.** Besides an  $O(\log n)$  sized counter and the hash function  $\mathbf{f}$  ( $\mathbf{g}$  is never evaluated more than once for an edge and thus does not need to be stored), the algorithm maintains a set of edges. Each edge will be kept with probability at most  $2pq$  and takes  $O(\log n)$  space to store, so the result follows. ◀

We will write  $T_{uvw}$  for the variable that is 1 if  $uvw$  is a triangle in  $G$  with its edges arriving in the order  $(uv, uw, vw)$ , and 0 otherwise, and so

$$T = \sum_{(u,v,w) \in V^3} T_{uvw}.$$

We will write  $\bar{\mathbf{T}}_{uvw}$  for the random variable that is  $1/pq^2$  if  $T_{uvw} = 1$  and  $\mathbf{f}(u)\mathbf{g}(uv)\mathbf{g}(uw) = 1$ , and 0 otherwise. We will therefore have

$$\bar{\mathbf{T}} = \sum_{(u,v,w) \in V^3} \bar{\mathbf{T}}_{uvw}.$$

► **Lemma 3.**

$$\mathbb{E}[\bar{\mathbf{T}}] = T.$$

## 11:8 An Optimal Algorithm for Triangle Counting in the Stream

**Proof.** For any  $(u, v, w)$ ,  $\mathbf{f}(u)\mathbf{g}(uv)\mathbf{g}(uw) = 1$  with probability  $pq^2$ , so  $\mathbb{E}[\overline{\mathbf{T}}_{uvw}] = T_{uvw}$ . Therefore,

$$\begin{aligned}\mathbb{E}[\overline{\mathbf{T}}] &= \sum_{(u,v,w) \in V^3} \mathbb{E}[\overline{\mathbf{T}}_{uvw}] \\ &= \sum_{(u,v,w) \in V^3} T_{uvw} \\ &= T\end{aligned}$$

► **Lemma 4.**

$$\text{Var}(\overline{\mathbf{T}}) \leq T/pq^2 + T\Delta_E/pq + T\Delta_V/p.$$

**Proof.** Consider any (ordered) pair of triples  $(u, v, w), (x, y, z) \in V^3$  such that  $T_{uvw}T_{xyz} = 1$ . If  $(u, v, w) = (x, y, z)$ ,  $\overline{\mathbf{T}}_{uvw}\overline{\mathbf{T}}_{xyz} = 1/p^2q^4$  with probability  $pq^2$  and 0 otherwise, so

$$\mathbb{E}[\overline{\mathbf{T}}_{uvw}\overline{\mathbf{T}}_{xyz}] = \mathbb{E}[\overline{\mathbf{T}}_{uvw}^2] = 1/pq^2.$$

At most  $T$  such pairs of triples can exist.

Now, if  $|\{uv, uw\} \cap \{xy, xz\}| = 1$ , then  $u = x$  and so  $\overline{\mathbf{T}}_{uvw}\overline{\mathbf{T}}_{xyz} = 1/p^2q^4$  iff  $\mathbf{f}(u) = 1$  and  $\mathbf{g}(e) = 1$  for all  $e$  in the size-3 set  $\{uv, uw, xy, xz\}$ , which happens with probability  $pq^3$ , and so

$$\mathbb{E}[\overline{\mathbf{T}}_{uvw}\overline{\mathbf{T}}_{xyz}] = 1/pq.$$

Each triangle has at most  $\Delta_E$  other triangles it shares an edge with, so there are at most  $T\Delta_E$  such pairs.

If  $\{uv, uw\} \cap \{xy, xz\} = \emptyset$  but  $u = x$ , then  $\overline{\mathbf{T}}_{uvw}\overline{\mathbf{T}}_{xyz} = 1/p^2q^4$  iff  $\mathbf{f}(u) = 1$  and  $\mathbf{g}(e) = 1$  for all  $e$  in the size-4 set  $\{uv, uw, xy, xz\}$ , which happens with probability  $pq^4$ , and so

$$\mathbb{E}[\overline{\mathbf{T}}_{uvw}\overline{\mathbf{T}}_{xyz}] = 1/p.$$

Each triangle has at most  $\Delta_V$  other triangles it shares a vertex with, so there are at most  $T\Delta_V$  such pairs.

Finally, if  $\{u, v, w\} \cap \{x, y, z\} = \emptyset$ , then  $\overline{\mathbf{T}}_{uvw}\overline{\mathbf{T}}_{xyz} = 1/p^2q^4$  iff  $\mathbf{f}(u) = 1$ ,  $\mathbf{f}(x) = 1$ , and  $\mathbf{g}(e) = 1$  for all  $e$  in the size-4 set  $\{uv, uw, xy, xz\}$ , which happens with probability  $p^2q^4$ , and so

$$\mathbb{E}[\overline{\mathbf{T}}_{uvw}\overline{\mathbf{T}}_{xyz}] = 1.$$

At most  $T^2$  such pairs can exist. Therefore,

$$\begin{aligned}\mathbb{E}[\overline{\mathbf{T}}^2] &= \sum_{(u,v,w) \in V^3} \sum_{(x,y,z) \in V^3} \mathbb{E}[\overline{\mathbf{T}}_{uvw}\overline{\mathbf{T}}_{xyz}] \\ &= \sum_{(u,v,w) \in V^3} \mathbb{E}[\overline{\mathbf{T}}_{uvw}^2] + \sum_{(u,v,w) \in V^3} \left( \sum_{\substack{(x,y,z) \in V^3 \\ |\{uv, uw\} \cap \{xy, xz\}|=1}} \mathbb{E}[\overline{\mathbf{T}}_{uvw}\overline{\mathbf{T}}_{xyz}] + \right. \\ &\quad \left. \sum_{\substack{(x,y,z) \in V^3 \\ \{uv, uw\} \cap \{xy, xz\} = \emptyset \\ u=x}} \mathbb{E}[\overline{\mathbf{T}}_{uvw}\overline{\mathbf{T}}_{xyz}] + \sum_{\substack{(x,y,z) \in V^3 \\ \{u, v, w\} \cap \{x, y, z\} = \emptyset}} \mathbb{E}[\overline{\mathbf{T}}_{uvw}\overline{\mathbf{T}}_{xyz}] \right) \\ &\leq T/pq^2 + T\Delta_E/pq + T\Delta_V/p + T^2\end{aligned}$$

by adding the previously established bounds for all four kinds of pair. The lemma then follows from the fact that  $\text{Var}(\mathbf{T}) = \mathbb{E}[\mathbf{T}^2] - \mathbb{E}[\mathbf{T}]^2 = \mathbb{E}[\mathbf{T}^2] - T^2$ . ◀

We may now prove Theorem 1.4. For every  $\varepsilon, \delta \in (0, 1)$ , there is an algorithm for insertion-only graph streams that approximates the number of triangles in a graph  $G$  to  $\varepsilon T$  accuracy with probability  $1 - \delta$ , using

$$O\left(\frac{m}{T} (\Delta_E + \sqrt{\Delta_V}) \log n \frac{\log \frac{1}{\delta}}{\varepsilon^2}\right)$$

bits of space, where  $m$  is the number of edges in  $G$ ,  $T$  the number of triangles,  $\Delta_E$  the maximum number of triangles which share a single edge, and  $\Delta_V$  the maximum number of triangles which share a single vertex.

**Proof.** We may assume  $\Delta_V$  (more specifically, the upper bound we have on it) is at least 1, as otherwise we already know  $G$  to be triangle-free. By Lemmas 3 and 4, we can set  $p = \Delta_V/T$ ,  $q = \max\{\Delta_E/\Delta_V, 1/\sqrt{\Delta_V}\}$  and run Algorithm 1 to obtain an estimator with expectation  $T$  and variance at most  $3T^2$ . (These will give valid probabilities, as  $\Delta_V \leq T$  by definition, and  $\Delta_E$  is at least  $\Delta_V$ , as any pair of triangles sharing an edge also share a vertex.) By Lemma 2, this will take  $O\left(\frac{m}{T} (\Delta_E + \sqrt{\Delta_V}) \log n\right)$  space.

Repeating this  $36/\varepsilon^2$  times and taking the mean will give an estimator with expectation  $T$  and variance at most  $\varepsilon T^2/2$ . We can then repeat *this*  $O(\log \frac{1}{\delta})$  times and take the median to get an estimator that will be within  $\varepsilon T$  of  $T$  with probability  $1 - \delta$ . ◀

## 4 Conclusion

We resolve the complexity of triangle counting in the insertion-only streaming model, in terms of the well-studied natural graph parameters  $m, T, \Delta_E, \Delta_V$ . The results of [15] resolved this problem for the *linear sketching* model, and a result of [20] states that, under certain conditions, turnstile streaming algorithms are equivalent to linear sketches, suggesting that the algorithm of [17] is optimal for turnstile streams as well. However, [18] showed that an insertion-only algorithm of [14] can be converted into a turnstile streaming algorithm provided that, for instance, the length of the stream is reasonably constrained (with the number of insertions and deletions no more than  $O(1)$  times the final size of the graph). It remains open whether this algorithm can be converted into a turnstile algorithm under such constraints, or whether the bounded-stream turnstile complexity of triangle counting is somewhere between insertion-only and linear sketching.

Another natural question is about the choice of parameters – the algorithm of [22] is optimal in terms of  $m, T$ , and  $\Delta_E$ , but not when the parameter  $\Delta_V$  is considered. Are there natural extensions of the parametrization that allow for better results? The results of [17] include a proof of instance-optimality for a restricted subclass of non-adaptive sampling algorithms, but for more general algorithms it is clear that there are at least *unnatural* extensions of the parametrization that help. For instance, if all the edges of a graph are guaranteed to belong to high-degree vertices, but all the triangles belong to low-degree vertices, a simple filtering strategy allows an improvement.

In particular, the lower bound instances of [6, 17] are both sparse graphs, and so cannot be constructed if  $n$  is constrained to be small relative to  $m$  or  $T$ . For the most dense graphs (with  $\Theta(n^2)$  edges and  $\Theta(n^3)$  triangles) our algorithm and the algorithm of [17] are already trivially optimal up to log factors, since they use only  $\text{polylog}(n)$  bits. However, the complexity landscape for more general dense graphs remains open.

## References

- 1 Sepehr Assadi, Michael Kapralov, and Sanjeev Khanna. A simple sublinear-time algorithm for counting arbitrary subgraphs via edge sampling. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPICs*, pages 6:1–6:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.
- 2 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 '02*, pages 623–632, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics. URL: <http://dl.acm.org/citation.cfm?id=545381.545464>.
- 3 Luca Becchetti, Paolo Boldi, Carlos Castillo, and Aristides Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '08*, pages 16–24, New York, NY, USA, 2008. ACM. doi:10.1145/1401890.1401898.
- 4 Suman K. Bera and Amit Chakrabarti. Towards Tighter Space Bounds for Counting Triangles and Other Substructures in Graph Streams. In *34th Symposium on Theoretical Aspects of Computer Science (STACS 2017)*, volume 66 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- 5 Jonathan W. Berry, Bruce Hendrickson, Randall A. LaViolette, and Cynthia A. Phillips. Tolerating the community detection resolution limit with edge weighting. *Phys. Rev. E*, 83:056119, May 2011. doi:10.1103/PhysRevE.83.056119.
- 6 Vladimir Braverman, Rafail Ostrovsky, and Dan Vilenchik. How hard is counting triangles in the streaming model? In *Automata, Languages, and Programming*, pages 244–254. Springer, 2013.
- 7 Luciana S Buriol, Gereon Frahling, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Christian Sohler. Counting triangles in data streams. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 253–262. ACM, 2006.
- 8 J Lawrence Carter and Mark N Wegman. Universal classes of hash functions. *Journal of computer and system sciences*, 18(2):143–154, 1979.
- 9 Graham Cormode and Hossein Jowhari. A second look at counting triangles in graph streams. *Theoretical Computer Science*, 552:44–51, 2014.
- 10 Graham Cormode and Hossein Jowhari.  $L_p$  samplers and their applications: A survey. *ACM Comput. Surv.*, 52(1), 2019. doi:10.1145/3297715.
- 11 Jean-Pierre Eckmann and Elisha Moses. Curvature of co-links uncovers hidden thematic layers in the world wide web. *Proceedings of the National Academy of Sciences*, 99(9):5825–5829, 2002. doi:10.1073/pnas.032093399.
- 12 Talya Eden, Amit Levi, Dana Ron, and C. Seshadhri. Approximately counting triangles in sublinear time. In *Proceedings of the 56th FOCS*, pages 614–633. IEEE, 2015.
- 13 Talya Eden, Dana Ron, and C. Seshadhri. On approximating the number of  $k$ -cliques in sublinear time. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*, page 722–734, New York, NY, USA, 2018. Association for Computing Machinery.
- 14 Hossein Jowhari and Mohammad Ghodsi. New streaming algorithms for counting triangles in graphs. In *Computing and Combinatorics*, pages 710–716. Springer, 2005.
- 15 John Kallaugher, Michael K10.1145/3188745.3188810oapralov, and Eric Price. The sketching complexity of graph and hypergraph counting. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 556–567. IEEE, 2018.

- 16 John Kallaugher, Andrew McGregor, Eric Price, and Sofya Vorotnikova. The complexity of counting cycles in the adjacency list streaming model. In Dan Suciu, Sebastian Skritek, and Christoph Koch, editors, *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 119–133. ACM, 2019. doi:10.1145/3294052.3319706.
- 17 John Kallaugher and Eric Price. A hybrid sampling scheme for triangle counting. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1778–1797. SIAM, 2017.
- 18 John Kallaugher and Eric Price. Separations and equivalences between turnstile streaming and linear sketching. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1223–1236. ACM, 2020. doi:10.1145/3357713.3384278.
- 19 Mihail N Kolountzakis, Gary L Miller, Richard Peng, and Charalampos E Tsourakakis. Efficient triangle counting in large graphs via degree-based vertex partitioning. *Internet Mathematics*, 8(1-2):161–185, 2012.
- 20 Yi Li, Huy L. Nguyễn, and David P. Woodruff. Turnstile streaming algorithms might as well be linear sketches. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 174–183, 2014. doi:10.1145/2591796.2591812.
- 21 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 '16*, pages 401–411, New York, NY, USA, 2016. ACM. doi:10.1145/2902251.2902283.
- 22 Rasmus Pagh and Charalampos E Tsourakakis. Colorful triangle counting and a mapreduce implementation. *Information Processing Letters*, 112(7):277–281, 2012.
- 23 A. Pavan, Kanat Tangwongsan, Srikanta Tirthapura, and Kun-Lung Wu. Counting and sampling triangles from a graph stream. *Proc. VLDB Endow.*, 6(14):1870–1881, 2013. doi:10.14778/2556549.2556569.
- 24 Charalampos E Tsourakakis, U Kang, Gary L Miller, and Christos Faloutsos. Doulion: counting triangles in massive graphs with a coin. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 837–846. ACM, 2009.



# Matching Drivers to Riders: A Two-Stage Robust Approach

Omar El Housni ✉

School of Operations Research and Information Engineering, Cornell Tech, New York, NY, USA

Vineet Goyal ✉

Industrial Engineering and Operations Research, Columbia University, New York, NY, USA

Oussama Hanguir ✉

Industrial Engineering and Operations Research, Columbia University, New York, NY, USA

Clifford Stein ✉

Industrial Engineering and Operations Research, Columbia University, New York, NY, USA

---

## Abstract

Matching demand (riders) to supply (drivers) efficiently is a fundamental problem for ride-hailing platforms who need to match the riders (almost) as soon as the request arrives with only partial knowledge about future ride requests. A myopic approach that computes an optimal matching for current requests ignoring future uncertainty can be highly sub-optimal. In this paper, we consider a two-stage robust optimization framework for this matching problem where future demand uncertainty is modeled using a set of demand scenarios (specified explicitly or implicitly). The goal is to match the current request to drivers (in the first stage) so that the cost of first stage matching and the worst-case cost over all scenarios for the second stage matching is minimized. We show that this two-stage robust matching is NP-hard under both explicit and implicit models of uncertainty. We present constant approximation algorithms for both models of uncertainty under different settings and show they improve significantly over standard greedy approaches.

**2012 ACM Subject Classification** Theory of computation → Approximation algorithms analysis

**Keywords and phrases** matching, robust optimization, approximation algorithms

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.12

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/abs/2011.03624>

**Funding** *Clifford Stein*: Research partly supported by NSF Grants CCF-1714818 and CCF-1822809.

## 1 Introduction

Matching demand (riders) with supply (drivers) is a fundamental problem for ride-hailing platforms such as Uber, Lyft and DiDi. These platforms need to continually make efficient matching decisions with only partial knowledge of future ride requests. A common approach in practice is batched matching: instead of matching each request sequentially as it arrives, aggregate the requests for a short amount of time (typically one to two minutes) and match the aggregated requests to available drivers in one batch [42, 33, 44]. However, computing this batch matching myopically without considering future requests can lead to a highly sub-optimal outcome for some subsequent drivers and riders.

Motivated by this shortcoming, and by the possibility of using historical data to hedge against future uncertainty, we study a two-stage framework for matching problems where the future demand uncertainty is modeled as a set of scenarios that are specified explicitly or implicitly. The goal is to compute a matching between the available drivers and the first batch of riders such that the total worst-case cost of first stage and second stage matching



© Omar El Housni, Vineet Goyal, Oussama Hanguir, and Clifford Stein;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 12; pp. 12:1–12:22



Leibniz International Proceedings in Informatics

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

is minimized. More specifically, we consider an adversarial model of uncertainty where the adversary observes the first stage matching of our algorithms and presents a worst-case scenario from the list of specified scenarios in the second stage. We focus on the case where the first stage cost is the average weight of the first stage matching, and the second stage cost is the highest edge weight in the second stage matching. This is motivated by the goal of computing a low-cost first stage matching while also minimizing the worst case waiting time for any rider in any second stage. All the results of this paper hold when the first stage cost is the highest edge weight of the first stage matching. We also study several other metrics in the full version. We consider two common models to describe the uncertainty in the second stage: an *explicit* list of all possible scenarios and an *implicit* description of the scenarios using a cardinality constraint. Two-stage robust optimization is a popular model for hedging against uncertainty [8, 19]. Several combinatorial optimization problems have been studied in this model, including Set Cover, Capacity Planning [7, 11] and Facility Location [22]. While online matching is a classical problem in graph theory, two-stage matching problems with uncertainty, have not been studied extensively. We present related work in Section 1.2.

### 1.1 Our Contributions

**Problem definition.** We consider the following *Two-stage Robust Matching Problem*. We are given a set of drivers  $D$ , a set of first stage riders  $R_1$ , a universe of potential second stage riders  $R_2$  and a set of second stage scenarios  $\mathcal{S} \subseteq \mathcal{P}(R_2)$ <sup>1</sup>. We are given a metric distance  $d$  on  $V = R_1 \cup R_2 \cup D$ . The goal is to find a subset of drivers  $D_1 \subseteq D$  ( $|D_1| = |R_1|$ ) to match all the first stage riders  $R_1$  such that the sum of cost of first stage matching and worst-case cost of second stage matching (between  $D \setminus D_1$  and the riders in the second stage scenario) is minimized. More specifically,

$$\min_{D_1 \subseteq D} \left\{ cost_1(D_1, R_1) + \max_{S \in \mathcal{S}} cost_2(D \setminus D_1, S) \right\}.$$

The first-stage decision is denoted  $D_1$  and its cost is  $cost_1(D_1, R_1)$ . Similarly, the second stage cost for scenario  $S$  is denoted  $cost_2(D \setminus D_1, S)$ , and  $\max\{cost_2(D \setminus D_1, S) \mid S \in \mathcal{S}\}$  is the worst-case cost over all possible scenarios. Let  $|R_1| = m$ ,  $|R_2| = n$ . We denote the objective function for a feasible solution  $D_1$  by

$$f(D_1) = cost_1(D_1, R_1) + \max_{S \in \mathcal{S}} cost_2(D \setminus D_1, S).$$

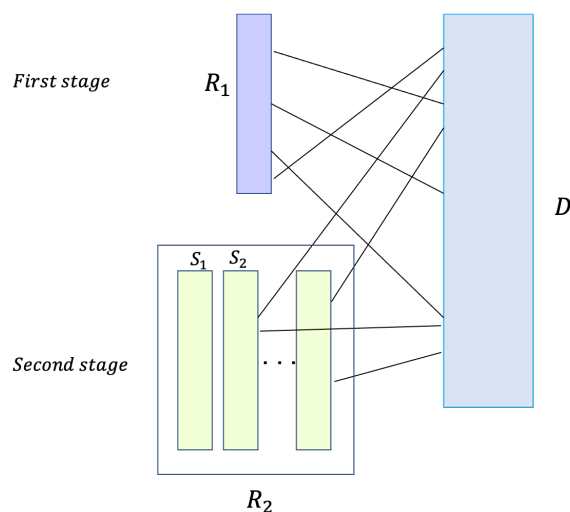
We assume that there are sufficiently many drivers to satisfy both first and second stage demand. Given an optimal first-stage solution  $D_1^*$ , we denote

$$\begin{aligned} OPT_1 &= cost_1(D_1^*, R_1), & OPT_2 &= \max\{cost_2(D \setminus D_1^*, S) \mid S \in \mathcal{S}\}, \\ OPT &= OPT_1 + OPT_2. \end{aligned}$$

We consider the setting where the first stage cost is the average weight of the matching between  $D_1$  and  $R_1$ , and the second stage cost is the bottleneck matching cost between  $D \setminus D_1$  and  $S$ . The bottleneck matching is the matching that minimizes the longest edge in a maximum cardinality matching between  $D \setminus D_1$  and  $S$ . We refer to this variant as the *Two-Stage Robust Matching Bottleneck Problem (TSRMB)*. Formally, let  $M_1$  be the minimum weight perfect matching between  $R_1$  and  $D_1$ , and given a scenario  $S$ , let  $M_2^S$  be

<sup>1</sup>  $\mathcal{P}(R_2)$  is the power set of  $R_2$ , the set of all subsets of  $R_2$ .





■ **Figure 1** Bipartite graph of drivers and riders in our two-stage matching problem.

the bottleneck matching between the scenario  $S$  and the available drivers  $D \setminus D_1$ , then the cost functions for the TSRMB are:

$$\text{cost}_1(D_1, R_1) = \frac{1}{m} \sum_{(i,j) \in M_1} d(i, j), \quad \text{and} \quad \text{cost}_2(D \setminus D_1, S) = \max_{(i,j) \in M_2^S} d(i, j).$$

The difference between the first and second stage metrics is motivated by the fact that the platform has access to the current requests and can exactly compute the cost of the matching. On the other hand, to ensure the robustness of the solution, we require all second stage assignments to have low waiting times by accounting for the maximum wait time in every scenario. We choose the first stage cost to be the average matching weight instead of the total weight for homogeneity reasons, so that first and second stage costs have comparable magnitudes. The bottleneck objective, i.e., finding a subgraph of a certain kind that minimizes the maximum edge cost in the subgraph, has been considered extensively in the literature [21, 16, 17]. While the main body of this paper will focus on studying TSRMB, we note that all our results hold when the first (resp. second) stage cost is equal to the highest edge weight in the first (resp. second) stage matching. In the full version, we study other variants of cost metrics, including a stochastic variant of TSRMB, and the case where both first and second stage costs are simply the total matching weights.

**Hardness.** We show that TSRMB is NP-hard even for two scenarios and NP-hard to approximate within a factor better than 2 for three scenarios. We also show that even when the scenarios are singletons, the problem is NP-hard to approximate within a factor better than 2. Given these hardness results, we focus on approximation algorithms for the TSRMB problem. A natural candidate is the greedy approach that minimizes only the first stage cost without considering the uncertainty in the second stage. However, we show that this myopic approach can be bad as  $\Omega(m) \cdot OPT$  (See Figure 2.)

**Approximations algorithms.** We consider both explicit and implicit models of uncertainty. For the case of explicit model with two scenarios, we give a constant factor approximation algorithm for TSRMB (Theorem 4). We further generalize the ideas of this algorithm to a

■ **Table 1** Summary of our results, where surplus  $\ell = |D| - |R_1| - k$ .

Uncertainty	Approx	Hardness
Explicit (2 scenarios)	5	NP-Hard
Explicit ( $p$ scenarios)	$O(p^{1.59})$	2
Implicit (surplus $\ell = 0$ )	3	-
Implicit ( $\ell < k$ and $k \leq \sqrt{n/2}$ )	17	2

constant approximation for any fixed number of scenarios (Theorem 6). Our approximation does not depend on the number of first stage riders or the size of scenarios but depends on the number of scenarios. The main idea is to reduce the problem with multiple scenarios to an instance with a single *representative scenario* while losing only a small factor. We then solve the single scenario instance (in polynomial time) to get an approximation for our original problem. The challenge in constructing the representative scenario is to find the right trade-off between capturing the demand of all second stage riders and keeping the cost of this scenario close to the optimal cost of the original instance.

For the implicit model of uncertainty, we consider the setting where we are given a universe of second stage riders  $R_2$  and an integer  $k$ , and any subset of size less than  $k$  can be a scenario. Therefore,  $\mathcal{S} = \{S \subset R_2 \text{ s.t. } |S| \leq k\}$ . The scenarios can be exponentially many in  $k$ , which makes even the evaluation of the cost of a feasible solution challenging and not necessarily achievable in polynomial time. Our analysis depends on the imbalance between supply and demand. In fact, when the number of drivers is very large compared to riders, the problem is less interesting in practice. However, it becomes interesting when the supply and demand are comparable. In this case, drivers might need to be shared between different scenarios. This leads us to define the notion of surplus  $\ell = |D| - |R_1| - k$ , which is the maximum number of drivers that we can afford not to use in a solution. As a warm-up, we first show that if the surplus is equal to zero (all the drivers are used), using any scenario as a representative scenario gives a 3-approximation. The problem becomes significantly more challenging even with a small surplus. We show that under a reasonable assumption on the size of scenarios, there is a constant approximation in the regime when the surplus  $\ell$  is smaller than the demand  $k$  (Theorem 9). Our algorithm is based on finding a clustering of drivers and riders that yields a simplified instance of TSRMB which can be solved within a constant factor. We show that we can cluster the riders into a ball (riders close to each others) and a set of *outliers* (riders far from each others) and apply ideas from the explicit scenario analysis. Finally, since the number of scenarios can be exponential, we construct a set of a polynomial number of proxy scenarios on which we evaluate any feasible solution within a constant approximation. Table 1 summarizes our results. Due to space constraints, we defer some of the proofs to the appendix.

## 1.2 Related Work

*Online bipartite matching.* Finding a maximum cardinality bipartite matching has received a considerable amount of attention over the years. Online matching was first studied by Karp *et al.* [27] in the adversarial model. Since then, many online variants have been studied [37]. This includes AdWords [4, 5, 38], vertex-weighted [1, 6], edge-weighted [20, 31], stochastic matching [12, 35, 39, 13], random vertex arrival [18, 26, 34, 23], and batch arrivals [32, 14, 44]. In the *online bipartite metric matching* variant, servers and clients correspond to points from a metric space, and the objective is to find the minimum weight maximum cardinality

matching. Khullet et al. [29] and Kalyanasundaram and Pruhs [24] provided deterministic algorithms in the adversarial model. In the random arrival model, Meyerson, et al. [40] and Bansal et al. [2] provided poly-logarithmic competitive algorithms. Recently, Raghvendra [41] presented a  $O(\log n)$ -competitive algorithm.

*Two-stage stochastic combinatorial optimization.* Within two-stage stochastic optimization, matching has been studied under various models. Kong and Schaefer [30] and Escoffier et al. [9] studied the stochastic two-stage maximum matching problem. Katriel et al. [28] studied the two-stage stochastic minimum weight maximum matching. Feng and Niazadeh [14] study  $K$ -stage variants of vertex weighted bipartite b-matching and AdWords problems, where online vertices arrive in  $K$  batches. More recently, Feng et al. [15] initiate the study and present online competitive algorithms for vertex-weighted two-stage stochastic matching as well as two-stage joint matching and pricing.

*Two-stage robust combinatorial optimization.* Within two-stage robust optimization, matchings have not been studied extensively. Matuschke et al. proposed a two-stage robust model for minimum weight matching with recourse [36]. Our model for TSRMB is different in three main aspects: i) We use a general class of uncertainty sets to describe the second stage scenarios while in [36] the only information given is the number of second stage vertices. ii) We do not allow any recourse and our first stage matching is irrevocable. iii) Our second stage cost is the bottleneck weight instead of the total weight.

## 2 Preliminaries

### 2.1 NP-hardness

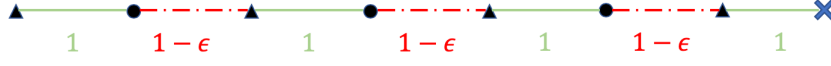
We show that TSRMB is NP-hard under both the implicit and explicit models. In the explicit model, it is NP-hard even for two scenarios and NP-hard to approximate within a factor better than 2 even for three scenarios.

In the explicit model with a polynomial number of scenarios, it is clear that the problem is in NP. However, in the implicit model, the problem can be described with a polynomial size input, but it is not clear that we can compute the total cost in polynomial time since there could be exponentially many scenarios. We show that it is NP-hard to approximate TSRMB in the implicit model within a factor better than 2 even when  $k = 1$ . The proof is presented in Appendix A.

► **Theorem 1.** *In the explicit model of uncertainty, TSRMB is NP-hard even with two scenarios. Furthermore, when the number of scenarios is  $\geq 3$ , there is no  $(2-\epsilon)$ -approximation algorithm for any fixed  $\epsilon > 0$ , unless  $P = NP$ . In the implicit model of uncertainty, even when  $k = 1$ , there is no  $(2-\epsilon)$ -approximation algorithm for TSRMB for any fixed  $\epsilon > 0$ , unless  $P = NP$ .*

### 2.2 Greedy Approach

A natural greedy approach is to choose the optimal matching for the first stage riders  $R_1$  without considering the second stage uncertainty. It can lead to a solution with a total cost that scales linearly with  $m$  (cardinality of  $R_1$ ) while  $OPT$  is a constant, even with one scenario. Consider the line example in Figure 2. We have  $m$  first stage riders and  $m + 1$  drivers alternating on a line with distances 1 and  $1 - \epsilon$ . There is one second stage rider at the right endpoint of the line. The greedy matching minimizes the first stage cost and incurs a total cost of  $(2 - \epsilon)(m + 1)$ , while the optimal cost is equal to 2. Therefore any attempt to have a good approximation needs to consider the second stage riders.



■ **Figure 2** Riders in first stage are depicted as black dots and drivers as black triangles. The second stage rider is depicted as a blue cross.

► **Lemma 2.** *The cost of the Greedy algorithm can be  $\Omega(m) \cdot OPT$ .*

### 2.3 Single Scenario

The *deterministic* version of the TSRMB problem, i.e., when there is only a single scenario in the second stage, can be solved exactly in polynomial time. This is a simple preliminary result which we need for the general case. Denote  $S$  a single second stage scenario. The instance  $(R_1, S, D)$  of TSRMB is then simply given by

$$\min_{D_1 \subset D} \left\{ cost_1(D_1, R_1) + cost_2(D \setminus D_1, S) \right\}.$$

Since the second stage problem is a bottleneck problem [21], the value of the optimal second stage cost  $w$  is one of the edge weights between  $D$  and  $S$ . We iterate over all possible values of  $w$  (at most  $|S| \cdot |D|$  values), delete all edges between  $R_2$  and  $D$  with weights strictly higher than  $w$  and set the weight of the remaining edges between  $S$  and  $D$  to zero. This reduces the problem to finding a minimum weight maximum cardinality matching. We can also use binary search to iterate over the edge weights. We present the details of this algorithm below and refer to it as *TSRMB-1-Scenario* in the rest of this paper.

We define the bottleneck graph of  $w$  to be  $BOTTLENECKG(w) = (R_1 \cup S \cup D, E_1 \cup E_2)$  where  $E_2 = \{(i, j) \in D \times S, d(i, j) \leq w\}$  and  $E_1 = \{(i, j) \in D \times R_1\}$ . Furthermore, we assume that there are  $q$  edges  $\{e_1, \dots, e_q\}$  between  $S$  and  $D$  with weights  $w_1 \leq w_2 \leq \dots \leq w_q$ .

■ **Algorithm 1**  $TSRMB-1-Scenario(R_1, S, D)$ .

**Input:** First stage riders  $R_1$ , scenario  $S$  and drivers  $D$ .

**Output:** First stage decision  $D_1$ .

```

1: for  $i \in \{1, \dots, q\}$  do
2:    $G_i := BOTTLENECKG(w_i)$ .
3:   Set all weights between  $D$  and  $S$  in  $G_i$  to be 0.
4:    $M_i :=$  minimum weight maximum cardinality matching on  $G_i$ .
5:   if  $R_1 \cup S$  is not completely matched in  $M_i$  then
6:     output certificate of failure.
7:   else
8:      $D_1^i :=$  first stage drivers in  $M_i$ .
9:   end if
10: end for
11: return  $D_1 = \arg \min_{D_1^i: 1 \leq i \leq q} \left\{ cost_1(D_1^i, R_1) + cost_2(D \setminus D_1^i, S) \right\}$ .
```

Note that the arg min in the last step of Algorithm 1 is only taken over values of  $i$  for which there was no certificate of failure.

► **Lemma 3.** *TSRMB-1-Scenario gives an optimal solution for the single scenario case.*

**Proof of Lemma 3.** Let  $OPT_1$  and  $OPT_2$  be the first and second stage cost of an optimal solution, and  $i \in \{1, \dots, q\}$  such that  $w_i = OPT_2$ . In this case,  $G_i$  contains all the edges of this optimal solution. By setting all the edges in  $E_2$  to 0, we are able to compute a minimum weight maximum cardinality matching between  $R_1 \cup S$  and  $D$  that matches both  $R_1$  and  $S$  and minimizes the weight of the edges matching  $R_1$ . The first stage cost of this matching is less than  $OPT_1$ , the second stage cost is clearly less than  $OPT_2$  because we only allowed edges with weight less than  $OPT_2$  in  $G_i$ . ◀

We also observe that we can use binary search in Algorithm 1 to iterate over the edge weights. For an iteration  $i$ , a failure to find a minimum weight maximum cardinality matching on  $G_i$  that matches both  $R_1$  and  $S$  implies that we need to try an edge weight higher than  $w_i$ . On the other hand, if  $M_i$  matches  $R_1$  and  $S$  such that  $D_1^i$  gives a smaller total cost, then the optimal bottleneck value is lower than  $w_i$ .

### 3 Explicit Scenarios

#### 3.1 Two scenarios

Our main contribution in this section is a constant approximation algorithm for TSRMB with two scenarios. Our analysis shows that we can reduce the problem to an instance with a single representative scenario by losing a small factor. We then use TSRMB-1-Scenario to solve the single representative scenario case.

Consider two scenarios  $\mathcal{S} = \{S_1, S_2\}$ . First, we can assume without loss of generality that we know the exact value of  $OPT_2$  which corresponds to one of the edges connecting second stage riders  $R_2$  to drivers  $D$  (we can iterate over all the weights of second stage edges). We construct a representative scenario that serves as a proxy for  $S_1$  and  $S_2$  as follows. In the second stage, if a pair of riders  $i \in S_1$  and  $j \in S_2$  is served by the same driver in the optimal solution, then they should be close to each other. Therefore, we can consider a single representative rider for each such pair. While it is not easy to guess all such pairs, we can approximately compute the representative riders by solving a maximum matching on  $S_1 \cup S_2$  with edges less than  $2OPT_2$ . More formally, let  $G_I$  be the induced bipartite subgraph of  $G$  on  $S_1 \cup S_2$  containing only edges between  $S_1$  and  $S_2$  with weight less than or equal to  $2OPT_2$ . We compute a maximum cardinality matching  $M$  between  $S_1$  and  $S_2$  in  $G_I$ , and construct a representative scenario containing  $S_1$  as well as the unmatched riders of  $S_2$ . We solve the single scenario problem on this representative scenario and return its optimal first stage solution. We show in Theorem 4 that this solution leads to a 5-approximation.

■ **Algorithm 2** Two explicit scenarios.

---

**Input:** First stage riders  $R_1$ , two scenarios  $S_1$  and  $S_2$ , drivers  $D$  and value of  $OPT_2$ .

**Output:** First stage decision  $D_1$ .

- 1: Let  $G_I$  be the induced subgraph of  $G$  on  $S_1 \cup S_2$  with only the edges between  $S_1$  and  $S_2$  of weights less than  $2OPT_2$ .
  - 2: Set  $M :=$  maximum cardinality matching between  $S_1$  and  $S_2$  in  $G_I$ .
  - 3: Set  $S_2^{Match} := \{r \in S_2 \mid \exists s \in S_1 \text{ s.t. } (s, r) \in M\}$  and  $S_2^{Unmatch} = S_2 \setminus S_2^{Match}$ .
  - 4: **return**  $D_1 :=$  TSRMB-1-Scenario( $R_1, S_1 \cup S_2^{Unmatch}, D$ ).
- 

▶ **Theorem 4.** *Algorithm 2 yields a solution with total cost less than  $OPT_1 + 5OPT_2$  for TSRMB with 2 scenarios.*

The proof of Theorem 4 relies on the following structural lemma where we show that the set  $D_1$  returned by Algorithm 2 yields a total cost at most  $(OPT_1 + 3OPT_2)$  when evaluated only on the single representative scenario  $S_1 \cup S_2^{Unmatch}$ .

► **Lemma 5.** *Let  $D_1$  be the set of first stage drivers returned by Algorithm 2. Then  $cost_1(D_1, R_1) + cost_2(D \setminus D_1, S_1 \cup S_2^{Unmatch}) \leq OPT_1 + 3OPT_2$ .*

**Proof.** It is sufficient to show the existence of a matching  $M_a$  between  $R_1 \cup S_1 \cup S_2^{Unmatch}$  and  $D$  with a total cost less than  $OPT_1 + 3OPT_2$ . This would imply that the optimal solution  $D_1$  of  $\text{TSRMB-1-Scenario}(R_1, S_1 \cup S_2^{Unmatch}, D)$  has a total cost less than  $OPT_1 + 3OPT_2$  and concludes the proof. We show the existence of  $M_a$  by construction.

**Step 1.** We first match  $R_1$  with their mates in the optimal solution of TSRMB. Hence, the first stage cost of our constructed matching  $M_a$  is  $OPT_1$ .

**Step 2.** Now, we focus on  $S_2^{Unmatch}$ . Let  $S_2^{Unmatch} = S_{12} \cup S_{22}$  be a partition of  $S_2^{Unmatch}$  where  $S_{12}$  contains riders with a distance less than  $2OPT_2$  from  $S_1$  and  $S_{22}$  contains riders with a distance strictly bigger than  $2OPT_2$  from  $S_1$ , where the distance from a set is the minimum distance to any element of the set. A rider in  $S_{22}$  cannot share any driver with a rider from  $S_1$  in the optimal solution of TSRMB, because otherwise, the distance between these riders will be less than  $2OPT_2$  by using the triangle inequality. Therefore we can match  $S_{22}$  to their mates in the optimal solution and add them to  $M_a$ , without using the optimal drivers of  $S_1$ . We pay less than  $OPT_2$  for matching  $S_{22}$ .

**Step 3.** We still need to simultaneously match riders in  $S_1$  and  $S_{12}$  to finish the construction of  $M_a$ . Notice that some riders in  $S_{12}$  might share their optimal drivers with riders in  $S_1$ . We can assume without loss of generality that all riders in  $S_{12}$  share their optimal drivers with  $S_1$  (otherwise we can match them to their optimal drivers without affecting  $S_1$ ). Denote  $S_{12} = \{r_1, \dots, r_q\}$  and  $S_1 = \{s_1, \dots, s_k\}$ . For each  $i \in [q]$  let's say  $s_i \in S_1$  is the rider that shares its optimal driver with  $r_i$ . We show that  $q \leq |M|$ . In fact, every rider in  $S_{12}$  shares its optimal driver with a different rider in  $S_1$ , and is therefore within a distance  $2OPT_2$  from  $S_1$  by the triangle inequality. But since  $S_{12}$  is not covered by the maximum cardinality matching  $M$ , this implies by the maximality of  $M$  that there are  $q$  other riders from  $S_2^{Match}$  that are covered by  $M$ . Hence  $q \leq |M|$ . Finally, let  $\{t_1, \dots, t_q\} \subset S_2^{Match}$  be the mates of  $\{s_1, \dots, s_q\}$  in  $M$ , i.e.,  $(s_i, t_i) \in M$  for all  $i \in [q]$ . Recall that  $d(s_i, t_i) \leq 2OPT_2$  for all  $i \in [q]$ . In what follows, we describe how to match  $S_{12}$  and  $S_1$ :

- (i) For  $i \in [q]$ , we match  $r_i$  to its optimal driver and  $s_i$  to the optimal driver of  $t_i$ . This is possible because the optimal driver of  $t_i$  cannot be the same as the optimal driver of  $r_i$  since both  $r_i$  and  $t_i$  are part of the same scenario  $S_2$ . Therefore, we pay a cost  $OPT_2$  for the riders  $r_i$  and a cost  $3OPT_2$  (follows from the triangle inequality) for the riders  $s_i$  where  $i \in [q]$ .
- (ii) We still need to match  $\{s_{q+1}, \dots, s_k\}$ . Consider a rider  $s_j$  with  $j \in \{q+1, \dots, k\}$ . If the optimal driver of  $s_j$  is not shared with any  $t_i \in \{t_1, \dots, t_q\}$ , then this optimal driver is still available and can be matched to  $s_j$  with a cost less than  $OPT_2$ . If the optimal driver of  $s_j$  is shared with some  $t_i \in \{t_1, \dots, t_q\}$ , then  $s_j$  is also covered by  $M$ . Otherwise  $M$  can be augmented by deleting  $(s_i, t_i)$  and adding  $(r_i, s_i)$  and  $(s_j, t_i)$ . Therefore  $s_j$  is covered by  $M$  and has a mate  $\tilde{t}_j \in S_2^{Match} \setminus \{t_1, \dots, t_q\}$ . Furthermore, the driver assigned to  $\tilde{t}_j$  is still available. We can then match  $s_j$  to the optimal driver of  $\tilde{t}_j$ . Similarly if the optimal driver of some  $s_{j'} \in \{s_{q+1}, \dots, s_k\} \setminus \{s_j\}$  is shared with  $\tilde{t}_j$ , then  $s_{j'}$  is covered by  $M$ . Otherwise  $(r_i, s_i, t_i, s_j, \tilde{t}_j, s_{j'})$  is an augmenting path in  $M$ . Therefore  $s_{j'}$  has a mate in  $M$  and we can match  $s_{j'}$  to the optimal driver of its

mate. We keep extending these augmenting paths until all the riders in  $\{s_{q+1}, \dots, s_k\}$  are matched. Furthermore, the augmenting paths  $(r_i, s_i, t_i, s_j, \tilde{t}_j, s_{j'} \dots)$  starting from two different riders  $r_i \in S_{12}$  are vertex disjoint. This ensures that every driver is used at most once. Again, by the triangle inequality, the edges that match  $\{s_{q+1}, \dots, s_k\}$  in our solution have weights less than  $3OPT_2$ .

Putting it all together, we have constructed a matching  $M_a$  where the first stage cost is exactly  $OPT_1$  and the second-stage cost is less than  $3OPT_2$  since the edges used for matching  $S_1 \cup S_2^{Unmatch}$  in  $M_a$  have a weight less than  $3OPT_2$ . Therefore, the total cost of  $M_a$  is less than  $OPT_1 + 3OPT_2$ . ◀

**Proof of Theorem 4.** Let  $D_1$  be the drivers returned by Algorithm 2. Lemma 5 implies

$$cost_1(D_1, R_1) + cost_2(D \setminus D_1, S_1) \leq OPT_1 + 3OPT_2 \quad (1)$$

and

$$cost_1(D_1, R_1) + cost_2(D \setminus D_1, S_2^{Unmatch}) \leq OPT_1 + 3OPT_2.$$

We have  $S_2 = S_2^{Match} \cup S_2^{Unmatch}$ . If the scenario  $S_2$  is realized, we use the drivers that were assigned to  $S_1$  in the matching constructed in Lemma 5 to match  $S_2^{Match}$ . This is possible with edges of weights less than  $cost_2(D \setminus D_1, S_1) + 2OPT_2$  because  $S_2^{Match}$  is matched to  $S_1$  with edges of weight less than  $2OPT_2$ . Hence,

$$cost_2(D \setminus D_1, S_2) \leq \max \{ cost_2(D \setminus D_1, S_2^{Unmatch}), cost_2(D \setminus D_1, S_1) + 2OPT_2 \},$$

and therefore

$$cost_1(D_1, R_1) + cost_2(D \setminus D_1, S_2) \leq OPT_1 + 5OPT_2. \quad (2)$$

From (1) and (2),  $cost_1(D_1, R_1) + \max_{S \in \{S_1, S_2\}} cost_2(D \setminus D_1, S) \leq OPT_1 + 5OPT_2$ . ◀

■ **Algorithm 3**  $p$  explicit scenarios.

**Input:** First-stage riders  $R_1$ , scenarios  $\{S_1, S_2, \dots, S_p\}$ , drivers  $D$  and value of  $OPT_2$ .

**Output:** First stage decision  $D_1$ .

- 1: Initialize  $\hat{S}_j := S_j$  for  $j = 1, \dots, p$ .
- 2: **for**  $i = 1, \dots, \log_2 p$  **do**
- 3:     **for**  $j = 1, 2, \dots, \frac{p}{2^i}$  **do**
- 4:          $\sigma(j) = j + \frac{p}{2^i}$
- 5:          $M_j :=$  maximum cardinality matching between  $\hat{S}_j$  and  $\hat{S}_{\sigma(j)}$  with edges of weight less than  $2 \cdot 3^{i-1} \cdot OPT_2$ .
- 6:          $\hat{S}_{\sigma(j)}^{Match} := \{r \in \hat{S}_{\sigma(j)} \mid \exists s \in \hat{S}_j \text{ s.t. } (s, r) \in M_j\}$ .
- 7:          $\hat{S}_{\sigma(j)}^{Unmatch} := \hat{S}_{\sigma(j)} \setminus \hat{S}_{\sigma(j)}^{Match}$
- 8:          $\hat{S}_j = \hat{S}_j \cup \hat{S}_{\sigma(j)}^{Unmatch}$ .
- 9:     **end for**
- 10: **end for**
- 11: **return**  $D_1 := \text{TSRMB-1-Scenario}(R_1, \hat{S}_1, D)$ .

### 3.2 Constant number of scenarios

We now consider the case of explicit list of  $p$  scenarios, i.e.,  $\mathcal{S} = \{S_1, S_2, \dots, S_p\}$ . Building upon the ideas from Algorithm 2, we present a  $O(p^{1.59})$ -approximation in this case. The idea is to construct the representative scenario recursively by processing pairs of “scenarios” at each step. Hence, we need  $O(\log_2 p)$  iterations to reduce the problem to an instance of a single scenario. At each iteration, we show that we only lose a multiplicative factor of 3 so that the final approximation ratio is  $O(3^{\log_2 p}) = O(p^{1.59})$ . We present details in Algorithm 3.

The approximation guarantee of our algorithm grows sub-quadratically with  $p$  and it is an interesting question if there exists an approximation that does not depend on the number of scenarios.

► **Theorem 6.** *Algorithm 3 yields a solution with total cost of  $O(p^{1.59}) \cdot OPT$  for TSRMB with an explicit list of  $p$  scenarios.*

**Proof of Theorem 6.** The algorithm reduces the number of considered “scenarios” by half in every iteration, until only one scenario remains. In iteration  $i$ , we have  $\frac{p}{2^{i-1}}$  scenarios that we aggregate in  $\frac{p}{2^i}$  pairs, namely  $(\hat{S}_j, \hat{S}_{\sigma(j)})$  for  $j \in \{1, 2, \dots, \frac{p}{2^i}\}$ . For each pair, we construct a single representative scenario which plays the role of the new  $\hat{S}_j$  at the start of the next iteration  $i + 1$ .

▷ **Claim.** There exists a first stage decision  $D_1^*$ , such that at every iteration  $i \in \{1, \dots, \log_2 p\}$ , we have for all  $j \in \{1, 2, \dots, \frac{p}{2^i}\}$ :

- (i)  $R_1$  can be matched to  $D_1^*$  with a first stage cost of  $OPT_1$ .
- (ii)  $\hat{S}_j \cup \hat{S}_{\sigma(j)}^{Unmatch}$  can be matched to  $D \setminus D_1^*$  with a second stage cost less than  $3^i \cdot OPT_2$ .
- (iii) There exists a matching between  $\hat{S}_{\sigma(j)}^{Match}$  and  $\hat{S}_j$  with edge weights less than  $2 \cdot 3^{i-1} \cdot OPT_2$ .

Proof of the claim. Statement (iii) follows from the definition of  $\hat{S}_{\sigma(j)}^{Match}$  in Algorithm 3. Let's show (i) and (ii) by induction over  $i$ .

- **Initialization:** for  $i = 1$ , let's take any two scenarios  $\hat{S}_j = S_j$  and  $\hat{S}_{\sigma(j)} = S_{\sigma(j)}$ . We know that these two scenarios can be matched to drivers of the optimal solution in the original problem with a cost less than  $OPT_2$ . In the proof of Lemma 5, we show that if we use the optimal first stage decision  $D_1^*$  of the original problem, then we can match  $\hat{S}_j$  and  $\hat{S}_{\sigma(j)}^{Unmatch}$  simultaneously to  $D \setminus D_1^*$  with a cost less than  $3OPT_2$ .
- **Maintenance.** Assume the claim is true for all values less than  $i \leq \log_2 p - 1$ . We show it is true for  $i + 1$ . Since the claim is true for iteration  $i$ , we know that at the start of iteration  $i + 1$ , for  $j \in \{1, \dots, \frac{p}{2^i}\}$ ,  $\hat{S}_j$  can be matched to  $D \setminus D_1^*$  with a cost less than  $3^i \cdot OPT_2$ . We can therefore consider a new TSRMB problem with  $\frac{p}{2^i}$  scenarios, where using  $D_1^*$  as a first stage decision ensures a second stage optimal value less than  $\widehat{OPT}_2 = 3^i \cdot OPT_2$ . By the proof of Lemma 5, and by using  $D_1^*$  as a first stage decision in this problem, we ensure that for  $j \in \{1, \dots, \frac{p}{2^{i+1}}\}$ ,  $\hat{S}_j$  and  $\hat{S}_{\sigma(j)}^{Unmatch}$  can be simultaneously matched to  $D \setminus D_1^*$  with a cost less than  $3\widehat{OPT}_2 = 3^{i+1} \cdot OPT_2$ . ◁

Our claim implies that in the last iteration  $i = \log_2 p$ :

- $R_1$  can be matched to  $D_1^*$  with a first stage cost of  $OPT_1$ .
- $\hat{S}_1$  can be matched to  $D \setminus D_1^*$  with a second stage cost less than  $3^{\log_2 p} \cdot OPT_2$ .

Computing the single scenario solution for  $\hat{S}_1$  will therefore yield a first stage decision  $D_1$  that gives a total cost less than  $OPT_1 + 3^{\log_2 p} \cdot OPT_2$  when the second stage is evaluated on the scenario  $\hat{S}_1$ . We now bound the cost of  $D_1$  on the original scenarios  $\{S_1, \dots, S_p\}$ . Consider a scenario  $S \in \{S_1, \dots, S_p\}$ . The riders in  $S \cap \hat{S}_1$  can be matched to some drivers



in  $D \setminus D_1$  with a cost less than  $OPT_1 + 3^{\log_2 p} \cdot OPT_2$ . As for other riders of  $S \setminus \hat{S}_1$ , they are not part of  $\hat{S}_1$  because they have been matched and deleted at some iteration  $i < \log_2 p$ . Consider riders  $r$  in  $S \setminus \hat{S}_1$  that were matched and deleted from a representative scenario at some iteration, then by statement (iii) in our claim, each  $r$  can be connected to a different rider in  $\hat{S}_1 \setminus (\hat{S}_1 \cap S)$  within a path of length at most

$$\sum_{t=1}^{\log_2 p} 2 \cdot 3^{t-1} \cdot OPT_2 = (3^{\log_2 p} - 1) \cdot OPT_2.$$

We know that  $R_1$  and  $\hat{S}_1$  can be matched respectively to  $D_1$  and  $D \setminus D_1$  with a total cost less than  $OPT_1 + 3^{\log_2 p} \cdot OPT_2$ . Therefore, we can match  $R_1$  and  $S$  respectively to  $D_1$  and  $D \setminus D_1$  with a total cost less than

$$OPT_1 + 3^{\log_2 p} \cdot OPT_2 + (3^{\log_2 p} - 1) \cdot OPT_2 = O(3^{\log_2 p}) \cdot OPT \simeq O(p^{1.59}) \cdot OPT.$$

Therefore, the worst-case total cost of the solution returned by Algorithm 3 is  $O(p^{1.59}) \cdot OPT$ . ◀

## 4 Implicit Scenarios

Consider an implicit model of scenarios  $\mathcal{S} = \{S \subset R_2 \text{ s.t. } |S| \leq k\}$ . While this model is widely used, it poses a challenge because the number of scenarios can be exponential. Therefore, even computing the worst-case second stage cost, for a given first stage solution, might not be possible in polynomial time and we can no longer assume that we can guess  $OPT_2$ . Note that the worst-case scenarios have size exactly  $k$ . Our analysis for this model depends on the balance between supply (drivers) and demand (riders). We define the surplus  $\ell$  as the excess in the number of available drivers for matching first-stage riders and a second-stage scenario:  $\ell = |D| - |R_1| - k$ . As a warm-up, we study the case of no surplus ( $\ell = 0$ ). Then, we address the more general case with a small surplus of drivers.

### 4.1 Warm-up: no surplus

When the number of drivers equals the number of first stage riders plus the size of scenarios (i.e.,  $\ell = 0$ ), we show a 3-approximation by simply solving a single scenario TSRMB with any of the scenarios. In fact, since  $\ell = 0$ , all scenarios are matched to the same set of drivers in the optimal solution. Hence, between any two scenarios, there exists a matching where all edge weights are less than  $2OPT_2$ . So by solving TSRMB with only one of these scenarios, we can recover a solution and bound the cost of the other scenarios within  $OPT_1 + 3OPT_2$  using the triangle inequality. The algorithm and proof are presented below.

■ **Algorithm 4** Implicit scenarios with no surplus.

---

**Input:** First stage riders  $R_1$ , second stage riders  $R_2$ , size  $k$  and drivers  $D$ .

**Output:** First stage decision  $D_1$ .

- 1:  $S_1 :=$  a second stage scenario of size  $k$ .
  - 2:  $D_1 :=$  TSRMB-1-Scenario( $R_1, S_1, D$ ).
  - 3: **return**  $D_1$ .
- 

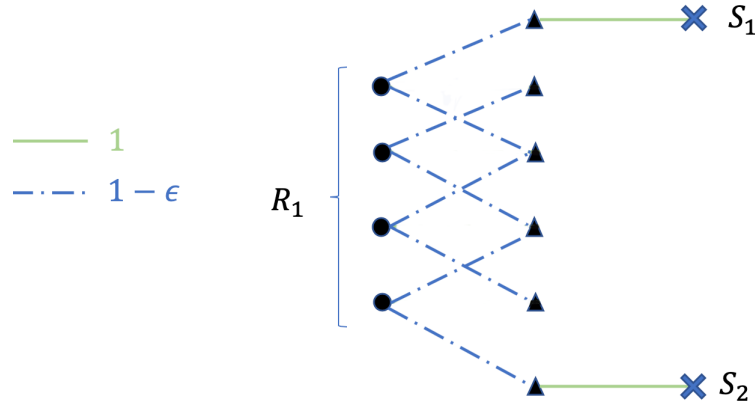
► **Lemma 7.** *Algorithm 4 yields a solution with total cost less than  $OPT_1 + 3OPT_2$  for TSRMB with implicit scenarios and no surplus.*

## 12:12 Matching Drivers to Riders: A Two-Stage Robust Approach

**Proof of Lemma 7.** Let  $OPT_1$  and  $OPT_2$  be the first and second stage cost of the optimal solution. Let  $f(D_1)$  be the total cost of the solution returned by the algorithm. We claim that  $f(D_1) \leq OPT_1 + 3OPT_2$ . It is clear that  $cost_1(D_1, R_1) + cost_2(D \setminus D_1, S_1) \leq OPT_1 + OPT_2$ . Let  $S \in \mathcal{S}$  be another scenario. Because  $|D| = |R_1| + k$ , the optimal solution uses exactly the same  $k$  drivers to match all the second stage scenarios. This implies that we can use the triangular inequality to find a matching between  $S$  and  $S_1$  of bottleneck cost less than  $2OPT_2$ . Hence for any scenario  $S$ ,

$$\begin{aligned} cost_1(D_1, R_1) + cost_2(D \setminus D_1, S) &\leq cost_1(D_1, R_1) + cost_2(D \setminus D_1, S_1) + 2OPT_2 \\ &\leq OPT_1 + 3OPT_2. \end{aligned} \quad \blacktriangleleft$$

If the surplus is strictly greater than 0, the above procedure can have an approximation ratio of  $\Omega(m)$ . Consider the example in Figure 3, with  $k = 1$  and two second stage riders. The single scenario solution for  $S_1$  uses the optimal second stage driver of  $S_2$ . Hence, if  $S_2$  is realized, the cost of matching  $S_2$  to the closest available driver is  $\Omega(m)$ . Similarly, the single scenario problem for  $S_2$  yields a  $\Omega(m)$  cost for  $S_1$ .



■ **Figure 3** First stage riders are depicted as black dots and drivers as black triangles. The two second stage riders are depicted as blue crosses. Second stage optimum are depicted as solid green edges.  $\mathcal{S} = \{S_1, S_2\}$ ,  $k = 1$  and  $\ell = 1$ .

### 4.2 Small surplus

The TSRMB problem becomes challenging even with a unit surplus. Motivated by this, we focus on the case of a small surplus  $\ell$ . In particular, we assume that  $\ell < k$ , i.e., the excess in the total available drivers is smaller than the size of any scenario. We present a constant approximation algorithm in this regime for the implicit model of uncertainty where the size of scenarios is relatively small with respect to the size of the universe ( $k = O(\sqrt{n})$ ). This technical assumption is needed for our analysis but it is not too restrictive and still captures the regime where the number of scenarios can be exponential. Our algorithm attempts to cluster the second stage riders in different groups (a *ball* and a set of *outliers*) in order to reduce the number of possible worst-case configurations. We then solve a sequence of instances with representative riders from each group. In what follows, we present our construction for these groups of riders.

**Our construction.** First, we show that many riders are contained in a ball with radius  $3OPT_2$ . The center of this ball,  $\delta$ , can be found by selecting the driver with the least maximum distance to its closest  $k$  second-stage riders, i.e.,

$$\delta = \arg \min_{\delta' \in D} \max_{r \in R_k(\delta')} d(\delta', r), \quad (3)$$

where  $R_k(\delta')$  is the set of the  $k$  closest second stage riders to  $\delta'$ . Formally, we have the following lemma. We present the proof in Appendix B.

► **Lemma 8.** *Suppose  $k \leq \sqrt{\frac{n}{2}}$  and  $\ell < k$  and let  $\delta$  be the driver given by (3). Then, the ball  $\mathcal{B}$  centered at  $\delta$  with radius  $3OPT_2$  contains at least  $n - \ell$  second stage riders. Moreover, the distance between any of these riders and any rider in  $R_k(\delta)$  is less than  $4OPT_2$ .*

Now, we focus on the rest of second stage riders. We say that a rider  $r \in R_2$  is an *outlier* if  $d(\delta, r) > 3OPT_2$ . Denote  $\{o_1, o_2, \dots, o_\ell\}$  the farthest  $\ell$  riders from  $\delta$  with  $d(\delta, o_1) \geq d(\delta, o_2) \geq \dots \geq d(\delta, o_\ell)$ . By Lemma 8, the  $n - \ell$  riders in  $\mathcal{B}$  are not outliers and the only potential outliers can be in  $\{o_1, o_2, \dots, o_\ell\}$ . Let  $j^*$  be the threshold such that  $o_1, o_2, \dots, o_{j^*}$  are outliers and  $o_{j^*+1}, \dots, o_\ell$  are not, with the convention that  $j^* = 0$  if there is no outlier. There are  $\ell + 1$  possible values for  $j^*$ . We call each of these possibilities a *configuration*. For  $j = 0, \dots, \ell$ , let  $C_j$  be the configuration corresponding to threshold candidate  $j$ .  $C_0$  is the configuration where there is no outlier and  $C_{j^*}$  is the correct configuration.

■ **Algorithm 5** Implicit scenarios with small surplus and  $k \leq \sqrt{\frac{n}{2}}$ .

---

**Input:** First stage riders  $R_1$ , second stage riders  $R_2$ , size  $k$  and drivers  $D$ .

**Output:** First stage decision  $D_1$ .

- 1: Set  $\delta :=$  driver given by (3).
  - 2: Set  $S_1 :=$  the closest  $k$  second stage riders to  $\delta$ .
  - 3: Set  $S_2 := \{o_1, \dots, o_\ell\}$  the farthest  $\ell$  second stage riders from  $\delta$  ( $o_1$  being the farthest).
  - 4: **for**  $j = 0, \dots, \ell$  **do**
  - 5:      $D_1(j) :=$  TSRMB-1-Scenario( $R_1, S_1 \cup \{o_1 \dots o_j\}, D$ ).
  - 6: **end for**
  - 7: **return**  $D_1 = \arg \min_{D_1(j): j \in \{0, \dots, \ell\}} cost_1(D_1(j), R_1) + \max_{S \in \{S_1, S_2\}} cost_2(D \setminus D_1(j), S)$ .
- 

Recall that  $R_k(\delta)$  are the closest  $k$  second-stage riders to  $\delta$ . For the sake of simplicity, we denote  $S_1 = R_k(\delta)$  and  $S_2 = \{o_1 \dots o_\ell\}$ .  $S_2$  is a feasible scenario since  $\ell < k$ . For every configuration  $C_j$ , we form a representative scenario using  $S_1$  and  $\{o_1 \dots o_j\}$ . We solve TSRMB with this single representative scenario  $S_1 \cup \{o_1 \dots o_j\}$  and denote  $D_1(j)$  the corresponding optimal solution, i.e.,

$$D_1(j) = \text{TSRMB-1-Scenario}(R_1, S_1 \cup \{o_1 \dots o_j\}, D).$$

Since we can not evaluate the cost of  $D_1(j)$  on all scenarios, we use the two proxy scenarios  $S_1$  and  $S_2$ . We show that the candidate  $D_1(j)$  with minimum cost over  $S_1$  and  $S_2$  gives a constant approximation to our original problem. The details are presented in Algorithm 5. We state the result in the next theorem.

► **Theorem 9.** *Algorithm 5 yields a solution with total cost less than  $3OPT_1 + 17OPT_2$  for TSRMB with implicit scenarios when  $k \leq \sqrt{\frac{n}{2}}$  and  $\ell < k$ .*

## 12:14 Matching Drivers to Riders: A Two-Stage Robust Approach

Before proving the theorem, we first introduce some notation. For all  $j \in \{0, \dots, \ell\}$ , denote

$$\begin{aligned}\Omega_j &= \text{cost}_1(D_1(j), R_1) \\ \Delta_j &= \text{cost}_2(D \setminus D_1(j), S_1 \cup \{o_1, \dots, o_j\}) \\ \beta_j &= \text{cost}_1(D_1(j), R_1) + \max_{S \in \{S_1, S_2\}} \text{cost}_2(D \setminus D_1(j), S)\end{aligned}$$

Recall that  $f$  the objective function of TSRMB. In particular,

$$f(D_1(j)) = \text{cost}_1(D_1(j), R_1) + \max_{S \in \mathcal{S}} \text{cost}_2(D \setminus D_1(j), S)$$

Our proof is based on the following two claims. Claim 10 establishes a bound on the cost of  $D_1(j^*)$  when evaluated on the proxy scenarios  $S_1$  and  $S_2$  and on all the scenarios in  $\mathcal{S}$ . Recall that  $j^*$  is the threshold index for the outliers as defined earlier in our construction. Claim 11 bounds the cost of  $f(D_1(j))$  for any  $j$ .

▷ **Claim 10.**  $\Omega_{j^*} + \Delta_{j^*} \leq OPT_1 + OPT_2.$  and  $f(D_1(j^*)) \leq OPT_1 + 5OPT_2.$

Proof of Claim 10.

1. In the optimal solution of the original problem,  $R_1$  is matched to a subset  $D_1^*$  of drivers. The scenario  $S_1$  is matched to a set of drivers  $D_{S_1}$  where  $D_1^* \cap D_{S_1} = \emptyset$ . Let  $D_o$  be the set of drivers that are matched to  $o_1, \dots, o_j^*$  in a scenario that contains  $o_1, \dots, o_j^*$ . It is clear that  $D_1^* \cap D_o = \emptyset$ . We claim that  $D_o \cap D_{S_1} = \emptyset$ . In fact, suppose there is a driver  $\rho \in D_o \cap D_{S_1}$ . This implies the existence of some  $o_j$  with  $j \leq j^*$  and some rider  $r \in S_1$  such that  $d(\rho, o_j) \leq OPT_2$  and  $d(\rho, r) \leq OPT_2$ . But then  $d(\delta, o_j) \leq d(\delta, r) + d(r, \rho) + d(\rho, o_j) \leq 3OPT_2$  which contradicts the fact the  $o_j$  is an outlier. Therefore  $D_o \cap D_{S_1} = \emptyset$ . We show that  $D_1^*$  is a feasible first stage solution to the single scenario problem of  $S_1 \cup \{o_1, \dots, o_j^*\}$  with a cost less than  $OPT_1 + OPT_2$ . In fact,  $D_1^*$  can be matched to  $R_1$  with a cost less than  $OPT_1$ ,  $D_{S_1}$  to  $S_1$  and  $D_o$  to  $\{o_1, \dots, o_j^*\}$  with a cost less than  $OPT_2$ . Therefore  $\Omega_{j^*} + \Delta_{j^*} \leq OPT_1 + OPT_2$ .
2. Recall that  $\text{cost}_1(D_1(j^*), R_1) = \Omega_{j^*}$ . Consider a scenario  $S$  and a rider  $r \in S$ . Let  $\mathcal{B}'$  be the set of the  $n - \ell$  closest second stage riders to  $\delta$ . Let  $D_{S_1}(j^*)$  be set of second stage drivers matched to  $S_1$  in the single scenario problem for scenario  $S_1 \cup \{o_1, \dots, o_{j^*}\}$ . Let  $D_o(j^*)$  be the set of second stage drivers matched to  $\{o_1, \dots, o_{j^*}\}$  in the single scenario problem for scenario  $S_1 \cup \{o_1, \dots, o_{j^*}\}$ . Recall that the second stage cost for this single scenario problem is  $\Delta_{j^*}$ . We distinguish three cases:
  - a. If  $r \in \mathcal{B}'$ , then by Lemma 8,  $r$  is connected to every driver in  $D_{S_1}(j^*)$  within a distance less than  $\Delta_{j^*} + 4OPT_2$ .
  - b. If  $r \in \{o_{j^*+1}, \dots, o_\ell\}$ , then  $r$  is connected to every driver in  $D_{S_1}(j^*)$  within a distance less than  $3OPT_2 + OPT_2 + \Delta_{j^*}$ .
  - c. If  $r \in \{o_1, \dots, o_{j^*}\}$  (i.e.,  $r$  an outlier), then  $r$  can be matched to a different driver in  $D_o(j^*)$  within a distance less than  $OPT_2$ .

This means that in every case, we can match  $r$  to a driver in  $D \setminus D_1(j^*)$  with a cost less than  $4OPT_2 + \Delta_{j^*}$ . This implies that

$$\max_{S \in \mathcal{S}} \text{cost}_2(D \setminus D_1(j^*), S) \leq 4OPT_2 + \Delta_{j^*}$$

and therefore

$$\Omega_{j^*} + \max_{S \in \mathcal{S}} \text{cost}_2(D \setminus D_1(j^*), S) \leq \Omega_{j^*} + \Delta_{j^*} + 4OPT_2 \leq OPT_1 + 5OPT_2. \quad \triangleleft$$

▷ **Claim 11.** For all  $j \in \{0, \dots, l\}$  we have,  $\beta_j \leq f(D_1(j)) \leq \max\{\beta_j + 4OPT_2, 3\beta_j + 2OPT_2\}$ .

Proof of Claim 11. Let  $\alpha_j$  be the second stage cost of  $D_1(j)$  on the TSRBM instance with scenarios  $S_1$  and  $S_2$ . Formally,  $\alpha_j = \max_{S \in \{S_1, S_2\}} \text{cost}_2(D \setminus D_1(j), S)$ . Therefore  $\beta_j = \Omega_j + \alpha_j$ .

Let's consider the two sets

$$\begin{aligned} O_1 &= \{r \in \{o_1, \dots, o_\ell\} \mid d(r, \delta) > 2\alpha_j + OPT_2\}. \\ O_2 &= \{o_1, \dots, o_\ell\} \setminus O_1. \end{aligned}$$

Consider  $D_1(j)$  as a first stage decision to TSRMB with scenarios  $S_1$  and  $S_2$ . Let  $\tilde{D}_1 \subset D \setminus D_1(j)$  be the set of drivers that are matched to  $O_1$  when the scenario  $S_2 = \{o_1, \dots, o_\ell\}$  is realized. Similarly, let  $\tilde{D}_2 \subset D \setminus D_1(j)$  be the drivers matched to scenario  $S_1$ . We claim that  $\tilde{D}_1 \cap \tilde{D}_2 = \emptyset$ . Suppose that there exists some driver  $\rho \in \tilde{D}_1 \cap \tilde{D}_2$ , this implies the existence of some  $o \in O_1$  and  $r \in S_1$  such that  $d(\rho, o) \leq \alpha_j$  and  $d(\rho, r) \leq \alpha_j$ . And since  $d(r, \delta) \leq OPT_2$  by definition of  $\delta$  we would have

$$d(o, \delta) \leq d(\rho, o) + d(\rho, r) + d(r, \delta) \leq 2\alpha_j + OPT_2,$$

which contradicts the definition of  $O_1$ . Therefore  $\tilde{D}_1 \cap \tilde{D}_2 = \emptyset$ .

Now consider a scenario  $S \in \mathcal{S}$ . The riders of  $S \cap O_1$  can be matched to  $\tilde{D}_1$  with a bottleneck cost less than  $\alpha_j$ . Recall that by Lemma 8, any rider in  $R_2 \setminus \{o_1, \dots, o_\ell\}$  is within a distance less than  $4OPT_2$  from any rider in  $S_1$ . The riders  $r \in S \setminus \{o_1, \dots, o_\ell\}$  can therefore be matched to any driver  $\rho \in \tilde{D}_2$  within a distance less than

$$d(r, \rho) \leq d(r, S_1) + d(S_1, \rho) \leq 4OPT_2 + \alpha_j.$$

As for riders  $r \in S \cap O_2$ , they can also be matched to any driver  $\rho$  of  $\tilde{D}_2$  within a distance less than

$$d(r, \rho) \leq d(r, \delta) + d(\delta, S_1) + d(S_1, \rho) \leq 2\alpha_j + OPT_2 + OPT_2 + \alpha_j = 3\alpha_j + 2OPT_2.$$

Therefore we can bound the second stage cost

$$\max_{S \in \mathcal{S}} \text{cost}_2(D \setminus D_1(j), S) \leq \max\{\alpha_j + 4OPT_2, 3\alpha_j + 2OPT_2\}$$

and we get that

$$\text{cost}_1(D_1(j), R_1) + \max_{S \in \mathcal{S}} \text{cost}_2(D \setminus D_1(j), S) \leq \max\{\beta_j + 4OPT_2, 3\beta_j + 2OPT_2\}$$

The other inequality  $\beta_j \leq \text{cost}_1(D_1(j), R_1) + \max_{S \in \mathcal{S}} \text{cost}_2(D \setminus D_1(j))$  is trivial. ◁

We are now ready to prove the theorem.

**Proof of Theorem 9.** Suppose Algorithm 5 returns  $D_1(\tilde{j})$  for some  $\tilde{j}$ . From Claim 11 and the minimality of  $\beta_{\tilde{j}}$ :

$$f(D_1(\tilde{j})) \leq \max\{\beta_{\tilde{j}} + 4OPT_2, 3\beta_{\tilde{j}} + 2OPT_2\} \leq \max\{\beta_{j^*} + 4OPT_2, 3\beta_{j^*} + 2OPT_2\}.$$

From Claim 10 and Claim 11, we have  $\beta_{j^*} \leq f(D_1(j^*)) \leq OPT_1 + 5OPT_2$ . We conclude that,

$$f(D_1(\tilde{j})) \leq \max\{OPT_1 + 9OPT_2, 3OPT_1 + 17OPT_2\} = 3OPT_1 + 17OPT_2. \quad \blacktriangleleft$$

## 5 Conclusion

In this paper, we present a new two-stage robust optimization framework for matching problems under both explicit and implicit models of uncertainty. Our problem is motivated by real-life applications in the ride-hailing industry. We study the Two-Stage Robust Matching Bottleneck problem, prove its hardness, and design approximation algorithms under different settings. Our algorithms give a constant approximation if the number of scenarios is fixed, but require additional assumptions when there are polynomially or exponentially many scenarios. It is an interesting question if there exists a constant approximation in the general case that does not depend on the number of scenarios.

---

## References

- 1 Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1253–1264. SIAM, 2011.
- 2 Nikhil Bansal, Niv Buchbinder, Anupam Gupta, and Joseph Seffi Naor. An  $o(\log k^2)$ -competitive algorithm for metric bipartite matching. In *European Symposium on Algorithms*, pages 522–533. Springer, 2007.
- 3 Piotr Berman, Bhaskar DasGupta, and Eduardo Sontag. Randomized approximation algorithms for set multicover problems with applications to reverse engineering of protein and gene networks. *Discrete Applied Mathematics*, 155(6-7):733–749, 2007.
- 4 Niv Buchbinder, Kamal Jain, and Joseph Seffi Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *European Symposium on Algorithms*, pages 253–264. Springer, 2007.
- 5 Nikhil R Devanur and Thomas P Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In *Proceedings of the 10th ACM conference on Electronic commerce*, pages 71–78, 2009.
- 6 Nikhil R Devanur, Kamal Jain, and Robert D Kleinberg. Randomized primal-dual analysis of ranking for online bipartite matching. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 101–107. SIAM, 2013.
- 7 Kedar Dhamdhere, Vineet Goyal, R Ravi, and Mohit Singh. How to pay, come what may: Approximation algorithms for demand-robust covering problems. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 367–376. IEEE, 2005.
- 8 Omar El Housni and Vineet Goyal. Beyond worst-case: A probabilistic analysis of affine policies in dynamic optimization. In *Advances in neural information processing systems*, pages 4756–4764, 2017.
- 9 Bruno Escoffier, Laurent Gourvès, Jérôme Monnot, and Olivier Spanjaard. Two-stage stochastic matching and spanning tree problems: Polynomial instances and approximation. *European Journal of Operational Research*, 205(1):19–30, 2010.
- 10 Uriel Feige. A threshold of  $\ln n$  for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.
- 11 Uriel Feige, Kamal Jain, Mohammad Mahdian, and Vahab Mirrokni. Robust combinatorial optimization with exponential scenarios. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 439–453. Springer, 2007.
- 12 Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and Shan Muthukrishnan. Online stochastic matching: Beating  $1-1/e$ . In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 117–126. IEEE, 2009.
- 13 Moran Feldman, Ola Svensson, and Rico Zenklusen. Online contention resolution schemes. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 1014–1033. SIAM, 2016.

- 14 Yiding Feng and Rad Niazadeh. Batching and optimal multi-stage bipartite allocations. In *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.
- 15 Yiding Feng, Rad Niazadeh, and Amin Saberi. Two-stage stochastic matching with application to ride hailing. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2862–2877. SIAM, 2021.
- 16 Harold N Gabow and Robert E Tarjan. Algorithms for two bottleneck optimization problems. *Journal of Algorithms*, 9(3):411–417, 1988.
- 17 Robert S Garfinkel and KC Gilbert. The bottleneck traveling salesman problem: Algorithms and probabilistic analysis. *Journal of the ACM (JACM)*, 25(3):435–448, 1978.
- 18 Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA*, volume 8, pages 982–991, 2008.
- 19 Anupam Gupta, Viswanath Nagarajan, and Ramamoorthi Ravi. Thresholded covering algorithms for robust and max-min optimization. In *International Colloquium on Automata, Languages, and Programming*, pages 262–274. Springer, 2010.
- 20 Bernhard Haeupler, Vahab S Mirrokni, and Morteza Zadimoghaddam. Online stochastic weighted matching: Improved approximation algorithms. In *International workshop on internet and network economics*, pages 170–181. Springer, 2011.
- 21 Dorit S Hochbaum and David B Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM (JACM)*, 33(3):533–550, 1986.
- 22 Omar El Housni, Vineet Goyal, and David Shmoys. On the power of static assignment policies for robust facility location problems. *arXiv preprint arXiv:2011.04925*, 2020.
- 23 Patrick Jaillet and Xin Lu. Online stochastic matching: New algorithms with better bounds. *Mathematics of Operations Research*, 39(3):624–646, 2014.
- 24 Bala Kalyanasundaram and Kirk Pruhs. Online weighted matching. *Journal of Algorithms*, 14(3):478–488, 1993.
- 25 Viggo Kann. Maximum bounded 3-dimensional matching is max snp-complete. *Information Processing Letters*, 37(1):27–35, 1991.
- 26 Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 587–596, 2011.
- 27 Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358, 1990.
- 28 Irit Katriel, Claire Kenyon-Mathieu, and Eli Upfal. Commitment under uncertainty: Two-stage stochastic matching problems. *Theoretical Computer Science*, 408(2-3):213–223, 2008.
- 29 Samir Khuller, Stephen G Mitchell, and Vijay V Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science*, 127(2):255–267, 1994.
- 30 Nan Kong and Andrew J Schaefer. A factor 12 approximation algorithm for two-stage stochastic matching problems. *European Journal of Operational Research*, 172(3):740–746, 2006.
- 31 Nitish Korula and Martin Pál. Algorithms for secretary problems on graphs and hypergraphs. In *International Colloquium on Automata, Languages, and Programming*, pages 508–520. Springer, 2009.
- 32 Euiwoong Lee and Sahil Singla. Maximum matching in the online batch-arrival model. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 355–367. Springer, 2017.
- 33 Lyft. Matchmaking in lyft line - part 1. <https://eng.lyft.com/matchmaking-in-lyft-line-9c2635fe62c4>, 2016.
- 34 Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 597–606, 2011.

- 35 Vahideh H Manshadi, Shayan Oveis Gharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics. *Mathematics of Operations Research*, 37(4):559–573, 2012.
- 36 Jannik Matuschke, Ulrike Schmidt-Kraepelin, and José Verschae. Maintaining perfect matchings at low cost. *arXiv preprint arXiv:1811.10580*, 2018.
- 37 Aranyak Mehta. Online matching and ad allocation. *Theoretical Computer Science*, 8(4):265–368, 2012.
- 38 Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5):22–es, 2007.
- 39 Aranyak Mehta, Bo Waggoner, and Morteza Zadimoghaddam. Online stochastic matching with unequal probabilities. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 1388–1404. SIAM, 2014.
- 40 Adam Meyerson, Akash Nanavati, and Laura Poplawski. Randomized online algorithms for minimum metric bipartite matching. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 954–959. Society for Industrial and Applied Mathematics, 2006.
- 41 Sharath Raghvendra. A robust and optimal online algorithm for minimum metric bipartite matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.
- 42 Uber. Uber marketplace and matching. <https://marketplace.uber.com/matching>, 2020.
- 43 Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.
- 44 Lingyu Zhang, Tao Hu, Yue Min, Guobin Wu, Junying Zhang, Pengcheng Feng, Pinghua Gong, and Jieping Ye. A taxi order dispatch model based on combinatorial optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 2151–2159, 2017.

## A NP-Hardness proofs for TSRMB

We start by presenting the 3-Dimensional Matching (3-DM) and Set Cover problems, that we use in our reductions to show Theorem 1. Both problems are known to be strongly NP-hard [10, 25].

**title3-Dimensional Matching (3-DM).** Given three sets  $U$ ,  $V$ , and  $W$  of equal cardinality  $n$ , and a subset  $T$  of  $U \times V \times W$ , is there a subset  $M$  of  $T$  with  $|M| = n$  such that whenever  $(u, v, w)$  and  $(u', v', w')$  are distinct triples in  $M$ ,  $u \neq u'$ ,  $v \neq v'$ , and  $w \neq w'$ ?

**Set Cover Problem.** Given a set of elements  $\mathcal{U} = \{1, 2, \dots, n\}$  (called the universe), a collection  $S_1, \dots, S_m$  of  $m$  sets whose union equals the universe and an integer  $p$ .

Question: Is there a set  $C \subset \{1, \dots, m\}$  such that  $|C| \leq p$  and  $\bigcup_{i \in C} S_i = \mathcal{U}$ ?

### Proof of Theorem 1.

**Explicit uncertainty.** Consider an instance of the 3-Dimensional Matching Problem. We can use it to construct (in polynomial time) an instance of TSRMB with 2 scenarios as follows:

- Create two scenarios of size  $n$ :  $S_1 = U$  and  $S_2 = V$ .
- Set  $D = T$ , every driver corresponds to a triple in  $T$ .
- For every  $w \in W$ , let  $d_T(w)$  be the number of sets in  $T$  that contain  $w$ . We create  $d_T(w) - 1$  first stage riders, that are all copies of  $w$ . The total number of first stage riders is therefore  $|R_1| = |T| - n$ .



- For  $(w, e) \in R_1 \times D$ ,  $d(w, e) = \begin{cases} 1 & \text{if } w \in e \\ 3 & \text{otherwise.} \end{cases}$
- For  $(u, e) \in S_1 \cup S_2 \times D$ ,  $d(u, e) = \begin{cases} 1 & \text{if } u \in e \\ 3 & \text{otherwise.} \end{cases}$
- For  $u, v \in R_1 \cup S_1 \cup S_2$ ,  $d(u, v) = \min_{e \in D} d(u, e) + d(v, e)$ .
- For  $e, f \in D$ ,  $d(e, f) = \min_{u \in R_1 \cup S_1 \cup S_2} d(u, e) + d(u, f)$ .

This choice of distances induces a metric graph. We claim that there exists a 3-dimensional matching if and only if there exists a solution to this TSRMB instance with total cost equal to 2. Suppose that  $M = \{e_1, \dots, e_n\} \subset T$  is a 3-Dimensional matching. Let  $e_1, \dots, e_n$  be the drivers that correspond to  $M$  in the TSRMB instance. We show that by using  $D_1 = D \setminus \{e_1, \dots, e_n\}$  as a first stage decision, we ensure that the total cost for the TSRMB instance is equal to 2. For any rider  $u$  in scenario  $S_1$ , by definition of  $M$ , there exists a unique edge  $e_i \in M$  that covers  $u$ . The corresponding driver  $e_i \notin D_1$  can be matched to  $u$  with a distance equal to 1. Furthermore,  $e_i$  cannot be matched to any other rider in  $S_1$  with a cost less than 1. Similarly, for any rider  $v$  in scenario  $S_2$ , since there exists a unique edge  $e_j \in M$  that covers  $v$ , the corresponding driver can be matched to  $v$  with a cost of 1. The second stage cost is therefore equal to 1. As for the first stage cost, we know by definition of  $M$ , that every element  $w \in W$  is covered exactly once. Therefore, for every  $w \in W$ , there exists  $d_T(w) - 1$  edges that contain  $w$  in  $T \setminus M$ . This means that every 1st stage rider can be matched to a driver in  $D_1$  with a cost equal to 1. Hence the total cost of this two-stage matching is equal to 2.

Suppose now that there exists a solution to the TSRMB instance with a cost equal to 2. This means that the first and second stage costs are both equal to 1. Let  $M = \{e_1, \dots, e_n\}$  be the set of drivers used in the second stage of this solution. We show that  $M$  is a 3-dimensional matching. Let  $e_i = (u, v, w)$  and  $e_j = (u', v', w')$  be distinct triples in  $M$ . Since the second stage cost is equal to 1, the driver  $e_i$  (resp.  $e_j$ ) must be matched to  $u$  (resp.  $u'$ ) in  $S_1$ . Since we have exactly  $n$  second stage drivers and  $n$  riders in  $S_1$ , this means that  $e_i$  and  $e_j$  have to be matched to different second stage riders in  $S_1$ . Therefore we get  $u' \neq u$ . Similarly we see that  $v' \neq v$ . Assume now that  $w = w'$ , this means that the TSRMB solution has used two drivers (triples)  $e_i$  and  $e_j$  that contain  $w$  in the second stage. It is therefore impossible to match all the  $d_T(w) - 1$  copies of  $w$  in the first stage with a cost equal to 1. Therefore  $w \neq w'$ . The above construction can be performed in polynomial time of the 3-DM input, and therefore shows that TSRMB with two scenarios is NP-hard.

Now, to show that TSRMB is hard to approximate within a factor better than 2, we consider three scenarios. Consider an instance of 3-DM. We can use it to construct an instance of TSRMB with 3 scenarios as follows:

- Create 3 scenarios of size  $n$ :  $S_1 = U$ ,  $S_2 = V$  and  $S_3 = W$ .
- Set  $D = T$ .
- Create  $|R_1| = |T| - n$  first stage riders.
- For  $(w, e) \in R_1 \times D$ ,  $d(w, e) = 1$ .
- For  $(u, e) \in S_1 \cup S_2 \cup S_3 \times D$ ,  $d(u, e) = \begin{cases} 1 & \text{if } u \in e \\ 3 & \text{otherwise.} \end{cases}$
- For  $u, v \in R_1 \cup S_1 \cup S_2 \cup S_3$ ,  $d(u, v) = \min_{e \in D} d(u, e) + d(v, e)$ .
- For  $e, f \in D$ ,  $d(e, f) = \min_{u \in R_1 \cup S_1 \cup S_2 \cup S_3} d(u, e) + d(u, f)$ .

This choice of distances induces a metric graph. Similarly to the proof of 2 scenarios, we can show that there exists a 3-dimensional matching if and only if there exists a TSRMB solution with cost equal to 2. Furthermore, any solution for this TSRMB instance has

## 12:20 Matching Drivers to Riders: A Two-Stage Robust Approach

either a total cost of 2 or 4 (the first stage cost is always equal to 1). We show that if a  $(2 - \epsilon)$ -approximation (for some  $\epsilon > 0$ ) to the TSRMB exists then 3-Dimensional Matching is decidable. We know that this instance of TSRMB has a solution with total cost equal to 2 if and only if there is a 3-dimensional matching. Furthermore, if there is no 3-dimensional matching, the cost of the optimal solution to TSRMB must be 4. Therefore, if an algorithm guarantees a ratio of  $(2 - \epsilon)$  and a 3-dimensional matching exists, the algorithm delivers a solution with total cost equal to 2. If there is no 3-dimensional matching, then the solution produced by the algorithm has a total cost of 4.

**Implicit uncertainty.** We prove the hardness for  $k = 1$ . We start from an instance of the Set Cover problem and construct an instance of the TSRMB problem. Consider an instance of the decision problem of set cover. We can use it to construct the following TSRMB instance:

- Create  $m$  drivers  $D = \{1, \dots, m\}$ . For each  $j \in \{1, \dots, m\}$ , driver  $j$  corresponds to set  $S_j$ .
- Create  $m - p$  first stage riders,  $R_1 = \{1, \dots, m - p\}$ .
- Create  $n$  second stage riders,  $R_2 = \{1, \dots, n\}$ .
- Set  $\mathcal{S} = \{\{1\}, \dots, \{n\}\}$ . Every scenario is of size 1.

As for the distances between riders and drivers, we define them as follows:

- For  $(i, j) \in R_1 \times D$ ,  $d(i, j) = 1$ .
- For  $(i, j) \in R_2 \times D$ ,  $d(i, j) = \begin{cases} 1 & \text{if } i \in S_j \\ 3 & \text{otherwise.} \end{cases}$
- For  $i, i' \in R_1 \cup R_2$ ,  $d(i, i') = \min_{j \in D} d(i, j) + d(i', j)$ .
- For  $j, j' \in D$ ,  $d(j, j') = \min_{i \in R_1 \cup R_2} d(i, j) + d(i, j')$ .

This choice of distances induces a metric graph. Moreover, every feasible solution to this TSRMB instance has a first stage cost of exactly 1. We show that a set cover of size  $\leq p$  exists if and only if there is a TSRMB solution with total cost equal to 2. Suppose without loss of generality that  $S_1, \dots, S_p$  is a set cover. Then by using the drivers  $\{1, \dots, p\}$  in the second stage, we ensure that every scenario is matched with a cost of 1. This implies the existence of a solution with total cost equal to 2. Now suppose there is a solution to the TSRMB problem with cost equal to 2. Let  $D_2$  be the set of second stage drivers of this solution, then we have  $|D_2| = p$ . We claim that the sets corresponding to drivers in  $D_2$  form a set cover. In fact, since the total cost of the TSRMB solution is equal to 2, the second stage cost is equal to 1. This means that for every scenario  $i \in \{1, \dots, n\}$ , there is a driver  $j \in D_2$  within a distance 1 from  $i$ . Therefore  $i \in S_j$  and  $\{S_j : j \in D_2\}$  is a set cover.

Next we show that if  $(2 - \epsilon)$ -approximation (for some  $\epsilon > 0$ ) to the TSRMB exists then Set Cover is decidable. We know that the TSRMB problem has a solution of cost 2 if and only if there is a set cover of size less than  $p$ . Furthermore, if there is no such set cover, the cost of the optimal solution must be 4. Therefore, if the algorithm guarantees a ratio of  $(2 - \epsilon)$  and there is a set cover of size less than  $p$ , the algorithm delivers a solution with a total cost of 2. If there is no set cover, then clearly the solution produced by the algorithm has a cost of 4. ◀

► **Remark 12.** For  $k \geq 2$ , we can use a generalization of Set Cover to show that the problem is hard for any  $k$ . We use a reduction from the Set MultiCover Problem ([3, 43]) defined below.

**Set MultiCover Problem.** Given a set of elements  $\mathcal{U} = \{1, 2, \dots, n\}$  (called the universe) and a collection  $S_1, \dots, S_m$  of  $m$  sets whose union equals the universe. A “coverage factor” (positive integer)  $k$  and an integer  $p$ . Is there a set  $C \subset \{1, \dots, m\}$  such that  $|C| \leq p$  and for each element  $x \in \mathcal{U}$ ,  $|j \in C : x \in S_j| \geq k$ ?

We can create an instance of TSRMB from a Set MultiCover instance similarly to Set Cover with the exception that  $\mathcal{S} = \{S \subset R_2 \text{ s.t. } |S| = k\}$ . The hardness result follows similarly.

## B Implicit scenarios: small surplus

**Proof of Lemma 8.** Let  $\delta$  be the driver given by (3). We claim that the  $k$  closest riders to  $\delta$  are all within a distance less than  $OPT_2$  from  $\delta$ . Consider  $D_2^*$  to be the  $k + \ell$  drivers left for the second stage in the optimal solution. Every driver in  $D_2^*$  can be matched to a set of different second stage riders over different scenarios. Let us rank the drivers in  $D_2^*$  according to how many different second stage riders they are matched to over all scenarios, in descending order. Formally, let  $D_2^* = \{\delta_1, \delta_2, \dots, \delta_{k+\ell}\}$  and let  $R^*(\delta_i)$  be the second stage riders that are matched to  $\delta_i$  in the optimal solution in some scenario, such that

$$|R^*(\delta_1)| \geq \dots \geq |R^*(\delta_{k+\ell})|.$$

We claim that  $|R^*(\delta_1)| \geq k$ . In fact, we have  $\sum_{i=1}^{k+\ell} |R^*(\delta_i)| \geq n$  because every second stage rider is matched to at least one driver in some scenario. Therefore

$$|R^*(\delta_1)| \geq \frac{n}{k + \ell} \geq \frac{n}{2k} \geq k.$$

We know that all the second stage riders in  $R^*(\delta_1)$  are within a distance less than  $OPT_2$  from  $\delta_1$ . Therefore  $\max_{r \in R_k(\delta_1)} d(\delta_1, r) \leq OPT_2$ . But we know that by definition of  $\delta$ ,

$$\max_{r \in R_k(\delta)} d(\delta, r) \leq \max_{r \in R_k(\delta_1)} d(\delta_1, r) \leq OPT_2$$

This proves that the  $k$  closest second stage riders to  $\delta$  are within a distance less than  $OPT_2$ . Let  $R(\delta)$  be the set of all second stage riders that are within a distance less than  $OPT_2$  from  $\delta$ . Recall that  $R_k(\delta)$  is the set of the  $k$  closest second stage riders to  $\delta$ . In the optimal solution, the scenario  $R_k(\delta)$  is matched to a set of at least new  $k - 1$  drivers  $\{\delta_{i_1}, \dots, \delta_{i_{k-1}}\} \subset D_2^* \setminus \{\delta\}$ . We show a lower bound on the size of  $R(\delta)$  and the number of riders matched to  $\{\delta_{i_1}, \dots, \delta_{i_{k-1}}\}$  over all scenarios in the optimal solution.

$$\triangleright \text{Claim 13. } |R(\delta) \cup \bigcup_{j=1}^{k-1} R^*(\delta_{i_j})| \geq n - \ell$$

*Proof.* Suppose the opposite, suppose that at least  $\ell + 1$  riders from  $R_2$  are not in the union. Let  $F$  be the set of these  $\ell + 1$  riders. Since  $\ell + 1 \leq k$ , we can construct a scenario  $S$  that includes  $F$ . In the optimal solution, and in particular, in the second stage matching of  $S$ , at least one rider from  $F$  needs to be matched to a driver from  $\{\delta, \delta_{i_1}, \dots, \delta_{i_{k-1}}\}$ . Otherwise there are only  $\ell$  second stage drivers left to match all of  $F$ . Therefore there exists  $r \in F$  such that either  $r \in R(\delta)$  or there exists  $j \in \{1, \dots, k - 1\}$  such that  $r \in R^*(\delta_{i_j})$ . This shows that  $r \in R(\delta) \cup \bigcup_{j=1}^{k-1} R^*(\delta_{i_j})$ , which is a contradiction. Therefore, at most  $\ell$  second stage riders are not in the union.  $\triangleleft$

## 12:22 Matching Drivers to Riders: A Two-Stage Robust Approach

▷ **Claim 14.** For any rider  $r \in R(\delta) \bigcup_{j=1}^{k-1} R^*(\delta_{i_j})$ , we have  $d(r, \delta) \leq 3OPT_2$ .

*Proof.* If  $r \in R(\delta)$  then by definition we have  $d(r, \delta) \leq OPT_2$ . Now suppose  $r \in R^*(\delta_{i_j})$  for  $j \in [k-1]$ . Let  $r'$  be the rider from scenario  $R_k(\delta)$  that was matched to  $\delta_{i_j}$  in the optimal solution. Then by the triangular inequality

$$d(r, \delta) \leq d(r, \delta_{i_j}) + d(\delta_{i_j}, r') + d(r', \delta) \leq 3OPT_2. \quad \triangleleft$$

From Claim 14, we see that the ball centered at  $\delta$ , with radius  $3OPT_2$ , contains at least  $n - \ell$  second stage riders in  $R(\delta) \bigcup_{j=1}^{k-1} R^*(\delta_{i_j})$ . This proves the first part of the lemma. The second part is proved in the next claim.

▷ **Claim 15.** For  $r_1 \in R_k(\delta)$  and  $r_2 \in R(\delta) \bigcup_{j=1}^{k-1} R^*(\delta_{i_j})$ , we have  $d(r_1, r_2) \leq 4OPT_2$ .

*Proof.* Let  $r_1 \in R_k(\delta)$ . If  $r_2 \in R(\delta)$  then  $d(r_1, r_2) \leq d(r_1, \delta) + d(\delta, r_2) \leq 2OPT_2$ . If  $r_2 \in R^*(\delta_{i_j})$  for some  $j$ , and  $r'$  is the rider from scenario  $R_k(\delta)$  that was matched to  $\delta_{i_j}$

$$d(r_1, r_2) \leq d(r_1, \delta) + d(\delta, r') + d(r', \delta_{i_j}) + d(\delta_{i_j}, r_2) \leq 4OPT_2. \quad \triangleleft$$

Claim 13 shows that the number of riders included in  $R(\delta) \bigcup_{j=1}^{k-1} R^*(\delta_{i_j})$  is at least  $n - \ell$ . Claim 14 shows that each one of this rider has distance less than  $3OPT_2$  from  $\delta$ . Finally, Claim 15 shows that the distance between any one of this riders and any rider in  $R_k(\delta)$  is less than  $3OPT_2$ . This concludes the proof of Lemma 8. ◀

# Secretary Matching Meets Probing with Commitment

Allan Borodin ✉

Department of Computer Science, University of Toronto, Canada

Calum MacRury ✉

Department of Computer Science, University of Toronto, Canada

Akash Rakheja ✉

Department of Computer Science, University of Toronto, Canada

---

## Abstract

We consider the online bipartite matching problem within the context of stochastic probing with commitment. This is the one-sided online bipartite matching problem where edges adjacent to an online node must be probed to determine if they exist based on edge probabilities that become known when an online vertex arrives. If a probed edge exists, it must be used in the matching. We consider the competitiveness of online algorithms in the adversarial order model (AOM) and the secretary/random order model (ROM). More specifically, we consider an unknown bipartite stochastic graph  $G = (U, V, E)$  where  $U$  is the known set of offline vertices,  $V$  is the set of online vertices,  $G$  has edge probabilities  $(p_e)_{e \in E}$ , and  $G$  has edge weights  $(w_e)_{e \in E}$  or vertex weights  $(w_u)_{u \in U}$ . Additionally,  $G$  has a downward-closed set of probing constraints  $(\mathcal{C}_v)_{v \in V}$ , where  $\mathcal{C}_v$  indicates which sequences of edges adjacent to an online vertex  $v$  can be probed. This model generalizes the various settings of the classical bipartite matching problem (i.e. with and without probing). Our contributions include the introduction and analysis of probing within the random order model, and our generalization of probing constraints which includes budget (i.e. knapsack) constraints. Our algorithms run in polynomial time assuming access to a membership oracle for each  $\mathcal{C}_v$ .

In the vertex weighted setting, for adversarial order arrivals, we generalize the known  $\frac{1}{2}$  competitive ratio to our setting of  $\mathcal{C}_v$  constraints. For random order arrivals, we show that the same algorithm attains an asymptotic competitive ratio of  $1 - 1/e$ , provided the edge probabilities vanish to 0 sufficiently fast. We also obtain a strict competitive ratio for non-vanishing edge probabilities when the probing constraints are sufficiently simple. For example, if each  $\mathcal{C}_v$  corresponds to a patience constraint  $\ell_v$  (i.e.,  $\ell_v$  is the maximum number of probes of edges adjacent to  $v$ ), and any one of following three conditions is satisfied (each studied in previous papers), then there is a conceptually simple greedy algorithm whose competitive ratio is  $1 - \frac{1}{e}$ .

- When the offline vertices are unweighted.
- When the online vertex probabilities are “vertex uniform”; i.e.,  $p_{u,v} = p_v$  for all  $(u, v) \in E$ .
- When the patience constraint  $\ell_v$  satisfies  $\ell_v \in \{1, |U|\}$  for every online vertex; i.e., every online vertex either has unit or full patience.

Finally, in the edge weighted case, we match the known optimal  $\frac{1}{e}$  asymptotic competitive ratio for the classic (i.e. without probing) secretary matching problem.

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

**Keywords and phrases** Stochastic probing, Online algorithms, Bipartite matching, Optimization under uncertainty

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.13

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/abs/2008.09260> [4]

**Acknowledgements** We would like to thank Denis Pankratov, Rajan Udhwani, and David Wajc for their very constructive comments on this paper.



© Allan Borodin, Calum MacRury, and Akash Rakheja;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 13; pp. 13:1–13:23



Leibniz International Proceedings in Informatics

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

## 1 Introduction

Stochastic probing problems are part of the larger area of decision making under uncertainty and more specifically, stochastic optimization. Unlike more standard forms of stochastic optimization, it is not just that there is some possible stochastic uncertainty in the set of inputs, stochastic probing problems involve inputs that cannot be determined without probing (at some cost and/or within some constraint) so as to reveal the inputs. Applications of stochastic probing occur naturally in many settings, such as in matching problems where compatibility (for example, in online dating and kidney exchange applications) or legality (for example, a financial transaction that must be authorized before it can be completed) cannot be determined without some trial or investigation. Amongst other applications, the online bipartite stochastic matching problem notably models online advertising where the probability of an edge can correspond to the probability of a purchase in online stores or to pay-per-click revenue in online searching. Commitment reflects the fact that one usually chooses the next probe based on some concept of expected value but in many applications (e.g. kidney exchanges) the cost or invasiveness of probing makes it practically necessary to commit. In some applications, there may be a legal requirement to commit (e.g., if a contract is possibly being offered and commitment is required).

The (*offline*) *stochastic matching* problem was introduced by Chen et al. [9]. In this problem, the input is an adversarially generated *stochastic graph*  $G = (V, E)$  with a probability  $p_e$  associated with each edge  $e$  and a patience (or time-out) parameter  $\ell_v$  associated with each vertex  $v$ . An algorithm probes edges in  $E$  within the constraint that at most  $\ell_v$  edges are probed incident to any particular vertex  $v \in V$ . Also, when an edge  $e$  is probed, it is guaranteed to exist with probability exactly  $p_e$ . If an edge  $(u, v)$  is found to exist, it is added to the matching and then  $u$  and  $v$  are no longer available. The goal is to maximize the expected size of a matching constructed in this way. Chen et al. showed that by probing edges in non-increasing order of edge probability, one attains an approximation ratio of  $1/4$ . The analysis was later improved by Adameczyk [1], who showed that this algorithm in fact attains an approximation ratio of  $1/2$ . This problem can be generalized to vertices or edges having weights.

Mehta and Panigrahi [22] adapted the offline stochastic matching model to online bipartite matching as originally studied in the classical (non-stochastic) adversarial order online model. That is, they consider the setting where the stochastic graph is unknown and online vertices are determined by an adversary. More specifically, they studied the problem in the case of an unweighted stochastic graph  $G = (U, V, E)$  where  $U$  is the set of known offline vertices and the vertices in  $V$  arrive online without knowledge of future online node arrivals. They considered the special case of uniform edge probabilities (i.e.  $p_e = p$  for all  $e \in E$ ) and *unit* patience values, that is  $\ell_v = 1$  for all  $v \in V$ . They considered a greedy algorithm which attains a competitive ratio of  $\frac{1}{2}(1 + (1 - p)^{2/p})$ , which limits to  $\frac{1}{2}(1 + e^{-2}) \approx .567$  as  $p \rightarrow 0$ . Mehta et al. [23] considered the unweighted online stochastic bipartite setting with arbitrary edge probabilities, attaining a competitive ratio of 0.534, and recently, Huang and Zhang [16] additionally handled the case of arbitrary offline vertex weights, while improving this ratio to 0.572. However, as in [22], both [23] and [16] are restricted to unit patience values, and moreover require edge probabilities which are *vanishingly small*<sup>1</sup>. Goyal and Udvani [12] improved on both of these works by showing a 0.596 competitive ratio in the same setting.

---

<sup>1</sup> Vanishingly small edge probabilities must satisfy  $\max_{e \in E} p_e \rightarrow 0$ , where the asymptotics are with respect to the size of  $G$ .

In all our results we will assume *commitment*; that is, when an edge is probed and found to exist, it must be included in the matching (if possible without violating the matching constraint). The patience constraint can be viewed as a simple form of a budget (equivalently, knapsack) constraint for the online vertices. We generalize patience and budget constraints by associating a downward-closed set  $\mathcal{C}_v$  of probing sequences for each online node  $v$  where  $\mathcal{C}_v$  indicates which sequences of edges adjacent to vertex  $v$  can be probed. In the general query and commit framework of Gupta and Nagarajan [14], the  $\mathcal{C}_v$  constraints are the outer constraints.

## 1.1 Preliminaries

An input to the **(online) stochastic matching problem** is a **(bipartite) stochastic graph**, specified in the following way. Let  $G = (U, V, E)$  be a bipartite graph with edge weights  $(w_e)_{e \in E}$  and edge probabilities  $(p_e)_{e \in E}$ . We draw an independent Bernoulli random variable of parameter  $p_e$  for each  $e \in E$ . We refer to this Bernoulli as the **state** of the edge  $e$ , and denote it by  $\text{st}(e)$ . If  $\text{st}(e) = 1$ , then we say that  $e$  is **active**, and otherwise we say that  $e$  is **inactive**. For each  $v \in V$ , denote  $\partial(v)$  as the edges of  $G$  which include  $v$ . Define  $\partial(v)^{(*)}$  as the collection of strings (tuples) formed from the edges of  $\partial(v)$  whose characters (entries) are all distinct. Note that we use string/tuple notation and terminology interchangeably. Each  $v \in V$  has an **online probing constraint**  $\mathcal{C}_v \subseteq \partial(v)^{(*)}$  which is **downward-closed**. That is,  $\mathcal{C}_v$  has the property that if  $e \in \mathcal{C}_v$ , then so is any substring or permutation of  $e$ . Thus, in particular, our setting encodes the case when  $v$  has a patience value  $\ell_v$ , and more generally, when  $\mathcal{C}_v$  corresponds to a matroid or budgetary constraint<sup>2</sup> on  $\partial(v)$ . Note that we will often assume w.l.o.g. that  $E = U \times V$ , as we can always set  $p_{u,v} := 0$ .

A solution to the online stochastic matching problem is an **online probing algorithm**. An online probing algorithm is initially only aware of the identity of the offline vertices  $U$  of  $G$ . We think of  $G$ , as well as the relevant edges probabilities, weights, and probing constraints, as being generated by an adversary. An ordering on  $V$  is then generated either through an adversarial process or uniformly at random. We refer to the former case as the **adversarial order model (AOM)** and the latter case as the **random order model (ROM)**.

Based on whichever ordering is generated on  $V$ , the nodes are then presented to the online probing algorithm one by one. When an online node  $v \in V$  arrives, the online probing algorithm sees all the adjacent edges and their associated probabilities, as well as  $\mathcal{C}_v$ . However, the edge states  $(\text{st}(e))_{e \in \partial(v)}$  remain hidden to the algorithm. Instead, the algorithm must perform a **probing operation** on an adjacent edge  $e$  to reveal/expose its state,  $\text{st}(e)$ . Moreover, the online probing algorithm must **respect commitment**. That is, if an edge  $e = (u, v)$  is probed and turns out to be active, then  $e$  must be added to the current matching, provided  $u$  and  $v$  are both currently unmatched. The probing constraint  $\mathcal{C}_v$  of the online node then restricts which sequences of probes can be made to  $\partial(v)$ . As in the classical problem, an online probing algorithm must decide on a possible match for an online node  $v$  before seeing the next online node. The goal of the online probing algorithm is to return a matching whose expected weight is as large as possible. Since  $\mathcal{C}_v$  may be exponentially large in the size of  $U$ , in order to discuss the efficiency of an online probing algorithm, we work in the **membership oracle model**. That is, upon receiving the online vertex  $v \in V$ , we

<sup>2</sup> In the case of a budget  $B_v$  and edge probing costs  $(b_e)_{e \in \partial(v)}$ , any subset of  $\partial(v)$  may be probed, provided its cumulative cost does not exceed  $B_v$ .

assume the online probing algorithm has access to a **membership oracle**. The algorithm may **query** any string  $e \in \partial(v)^{(*)}$ , thus determining in a single operation whether or not  $e \in \partial(v)^{(*)}$  is in  $\mathcal{C}_v$ .

It is easy to see we cannot hope to obtain a non-trivial competitive ratio against the expected value of an optimal matching of the stochastic graph. Consider a single online vertex with patience 1, and  $k \geq 1$  offline (unweighted) vertices where each edge  $e$  has probability  $\frac{1}{k}$  of being present. The expectation of an online probing algorithm will be at most  $\frac{1}{k}$  while the expected size of an optimal matching will be  $1 - (1 - \frac{1}{k})^k \rightarrow 1 - \frac{1}{e}$ . This example clearly shows that no constant ratio is possible if the patience is sublinear in  $k = |U|$ . Thus, the standard in the literature is to instead benchmark the performance of an online probing algorithm against an *optimal offline probing algorithm*. An **offline probing algorithm** knows  $G = (U, V, E)$ , but initially the edge states  $(\text{st}(e))_{e \in E}$  are hidden. It can adaptively probe the edges of  $E$  in any order, but must satisfy the probing constraints  $(\mathcal{C}_v)_{v \in V}$  at each step of its execution<sup>3</sup>, while respecting commitment; that is, if a probed edge  $e = (u, v)$  turns out to be active, then  $e$  is added to the matching (if possible). The goal of an offline probing algorithm is to construct a matching with optimal weight in expectation. We define the **committal benchmark**  $\text{OPT}(G)$  for  $G$  as the value of an optimal offline probing algorithm. We abuse notation slightly, and also use  $\text{OPT}(G)$  to refer to the *strategy* of the committal benchmark on  $G$ . In the arXiv version of the paper [4], we introduce the stronger **non-committal benchmark**, and indicate which of our results hold against it.

## 1.2 Our Results

We first consider the case when the stochastic graph  $G = (U, V, E)$  has **(offline) vertex weights** – i.e., there exists  $(w_u)_{u \in U}$  such that  $w_{u,v} = w_u$  for each  $v \in N(u)$ , and arbitrary downward-closed probing constraints  $(\mathcal{C}_v)_{v \in V}$ . We consider a *greedy* online probing algorithm. That is, upon the arrival of  $v$ , the probes to  $\partial(v)$  are made in such a way that  $v$  gains as much value as possible (in expectation), provided the currently unmatched nodes of  $U$  are equal to  $R \subseteq U$ . As such, we must follow the probing strategy of the committal benchmark when restricted to the **induced stochastic graph**<sup>4</sup>  $G[\{v\} \cup R]$ , which we denote by  $\text{OPT}(R, v)$  for convenience.

Observe that if  $v$  has unit patience, then  $\text{OPT}(R, v)$  reduces to probing the adjacent edge  $(u, v) \in R \times \{v\}$  such that the value  $w_u \cdot p_{u,v}$  is maximized. Moreover, if  $v$  has unlimited patience, then  $\text{OPT}(R, v)$  corresponds to probing the adjacent edges of  $R \times \{v\}$  in non-increasing order of the associated vertex weights. Building on a result in Purohit et al. [24], Brubach et al. [8] showed how to devise an *efficient* probing strategy for  $v$  whose expected value matches  $\text{OPT}(R, v)$ , for any patience value. Using this probing strategy, they devised an online probing algorithm which achieves a competitive ratio of  $1/2$  for arbitrary patience values. The challenge in extending this competitive ratio to more general probing constraints comes from the fact that it is unclear how to compute  $\text{OPT}(R, v)$  efficiently. We show that this is possible to do when the probing constraints are downward-closed, and provide a primal-dual proof of the following theorem:

<sup>3</sup> Edges  $e \in E^{(*)}$  may be probed in the order specified by  $e$ , provided  $e^v \in \mathcal{C}_v$  for each  $v \in V$ , where  $e^v$  is the substring of  $e$  restricted to edges of  $\partial(v)$ .

<sup>4</sup> Given  $R \subseteq U, V' \subseteq V$ , the induced stochastic graph  $G[R \cup V']$  is formed by restricting the edges weights and probabilities of  $G$  to those edges within  $R \times V'$ . Similarly, each probing constraint  $\mathcal{C}_v$  is restricted to those strings whose entries lie entirely in  $R \times \{v\}$ .



► **Theorem 1.1.** *Suppose the adversary presents a vertex weighted stochastic graph  $G = (U, V, E)$ , with downward-closed probing constraints  $(\mathcal{C}_v)_{v \in V}$ . If  $\mathcal{M}$  is the matching returned by Algorithm 1 when executing on  $G$ , then*

$$\mathbb{E}[w(\mathcal{M})] \geq \frac{1}{2} \cdot OPT(G),$$

*provided the vertices of  $V$  arrive in adversarial order. Moreover, Algorithm 1 can be implemented efficiently in the membership oracle model.*

Since Algorithm 1 is deterministic, the  $1/2$  competitive ratio is best possible for deterministic algorithms in the adversarial arrival setting. One direction is thus to instead consider what can be done if the online probing algorithm is allowed randomization, which has received much attention in the case of unit patience [22, 23, 12, 16]. We instead make partial progress to understanding the performance of Algorithm 1 for downward-closed probing constraints in the ROM setting. However, unlike the adversarial setting, the complexity of the constraints greatly impacts what we are able to prove. The first part of our result is asymptotic in that it yields a good competitive ratio when applied to a stochastic graph whose maximum edge probability  $p_v := \max_{e \in \partial(v)} p_e$  vanishes sufficiently fast relevant to the maximum string length of  $\mathcal{C}_v$ , namely  $c_v := \max_{e \in \mathcal{C}_v} |e|$ , for each  $v \in V$ . Note that the vanishing probability setting is similar in spirit to the small bid to budget assumption in the Adwords problem (see Goyal and Udvani [12] for details). The second part of our result applies to stochastic graphs which we refer to as **rankable**. Roughly speaking, a vertex  $v \in V$  of  $G$  is **rankable**, provided there exists a fixed/non-adaptive ranking  $\pi_v$  of  $\partial(v)$  which can be used to specify the greedy strategy  $OPT(v, R)$  of  $v$ , no matter which vertices  $R \subseteq U$  are available when  $v$  is processed. For example, this includes the well-studied unit patience setting, in which case  $v$  ranks its adjacent edges in non-increasing order of  $(w_u p_{u,v})_{u \in U}$ , as well as when  $G$  is unweighted and has arbitrary patience values, in which case  $v$  ranks its adjacent edges in non-increasing order of edge probability. A stochastic graph is rankable if all its online vertices are rankable. We defer the precise definition to Section 2.

► **Theorem 1.2.** *Suppose Algorithm 1 returns the matching  $\mathcal{M}$  when executing on the vertex weighted stochastic graph  $G = (U, V, E)$  with downward-closed constraints  $(\mathcal{C}_v)_{v \in V}$ , and the vertices of  $V$  arrive u.a.r.. We then have the following two results:*

1. *If  $c_v := \max_{e \in \mathcal{C}_v} |e|$  and  $p_v := \max_{e \in \partial(v)} p_e$ , then*

$$\mathbb{E}[w(\mathcal{M})] \geq \min_{v \in V} (1 - p_v)^{c_v} \cdot \left(1 - \frac{1}{e}\right) \cdot OPT(G).$$

*Thus, if  $c_v \cdot p_v \rightarrow 0$  (as  $|G| \rightarrow \infty$ ) for each  $v \in V$ , then  $\mathbb{E}[w(\mathcal{M})] \geq (1 - o(1))(1 - 1/e) \cdot OPT(G)$ .*

2. *If  $G$  is rankable (which includes the specific cases outlined in the abstract), then*

$$\mathbb{E}[w(\mathcal{M})] \geq \left(1 - \frac{1}{e}\right) \cdot OPT(G).$$

► **Remark 1.3.** The analysis of Algorithm 1 is tight, as an execution of Algorithm 1 corresponds to the seminal Karp et al. [17] RANKING algorithm for unweighted non-stochastic (i.e.,  $p_e \in \{0, 1\}$  for all  $e \in E$ ) bipartite matching.

In the unit patience setting of [22], Mehta and Panigrahi showed that  $.621 < 1 - \frac{1}{e}$  is a randomized inapproximation with regard to guarantees made against LP-std-unit, the LP introduced by [22] to upper bound/relax the committal benchmark in the unit patience

setting. This hardness result led Goyal and Udvani [12] to consider a new unit patience LP that is a tighter relaxation of  $\text{OPT}(G)$  than LP-std-unit, thereby allowing them to prove a  $1 - 1/e$  competitive ratio for the case of **vertex-decomposable**<sup>5</sup> edge probabilities. However, they also discuss the difficulty of extending this result to the case of arbitrary edge probabilities in the context of the Adwords problem with arbitrary budget to bid ratios. It remains open whether a randomized algorithm can attain a competitive ratio of  $1 - 1/e$  against the committal benchmark for adversarial arrivals and arbitrary edge probabilities. A corollary of Theorem 1.2 is that in the ROM setting these difficulties do *not* arise.

► **Corollary 1.4.** *Suppose the adversary presents a vertex weighted stochastic graph  $G = (U, V, E)$ , with unit patience values. If  $\mathcal{M}$  is the matching returned by Algorithm 1 when executing on  $G$ , then*

$$\mathbb{E}[w(\mathcal{M})] \geq \left(1 - \frac{1}{e}\right) \text{OPT}(G),$$

*provided the vertices of  $V$  arrive in random order.*

► **Remark 1.5.** The guarantee of Theorem 1.2 is proven against a new LP relaxation (LP-DP) whose optimum value we denote by  $\text{LPOPT}_{\text{DP}}(G)$ . In the special case when  $G$  has unit patience,  $\text{LPOPT}_{\text{std}}(G) \leq \text{LPOPT}_{\text{DP}}(G)$ . Thus, the 0.621 inapproximation of Mehta and Panigrahi against LP-std-unit does not apply (even for deterministic probing algorithms) to the ROM setting. Corollary 1.4 therefore implies that deterministic probing algorithms in the ROM setting have strictly more power than randomized probing algorithms in the adversarial order model. This contrasts with the classic ROM setting where it is unknown whether a deterministic algorithm can improve upon  $1 - 1/e$ , the optimal competitive attainable by randomized algorithms in the adversarial setting.

We next consider the unknown stochastic matching problem in the most general setting of arbitrary edge weights, and downward-closed probing constraints. Since no non-trivial competitive ratio can be proven in the case of adversarial arrivals, even in the classical setting, we work in the ROM setting. We generalize the matching algorithm of Kesselheim et al. [18] so as to apply to the stochastic probing setting.

► **Theorem 1.6.** *Suppose the adversary presents an edge-weighted stochastic graph  $G = (U, V, E)$ , with downward-closed probing constraints  $(\mathcal{C}_v)_{v \in V}$ . If  $\mathcal{M}$  is the matching returned by Algorithm 2 when executing on  $G$ , then*

$$\mathbb{E}[w(\mathcal{M})] \geq \left(\frac{1}{e} - \frac{1}{|V|}\right) \cdot \text{OPT}(G),$$

*provided the vertices of  $V$  arrive uniformly at random (u.a.r.). Moreover, Algorithm 2 can be implemented efficiently in the membership oracle model.*

► **Remark 1.7.** For context, the previous best known approximation ratio known for the offline bipartite stochastic matching problem with two-sided or one-sided patience is 0.352 due to Adamczyk et al. [3]. Since  $1/e > 0.352$ , Theorem 1.6 in fact improves on this result for the case of one-sided patience, despite the fact that Algorithm 2 works in the unknown graph setting and for more general one-sided probing constraints. Very recently, Brubach et al. [7] proved an approximation ratio of 0.382 for general stochastic graphs.

<sup>5</sup> Vertex-decomposable means that there exists probabilities  $(p_u)_{u \in U}$  and  $(p_v)_{v \in V}$ , such that  $p_{(u,v)} = p_u \cdot p_v$  for each  $(u, v) \in E$ .

### 1.3 Our Technical Contributions

In the vertex weighted setting, the first challenge is to establish a greedy strategy for a single online vertex which runs efficiently for general probing constraints. We provide a dynamic programming based algorithm (DP-OPT) for solving this problem, which builds upon the work of Brubach et al. [8], and before that, Purohit et al. [24] (see Theorem 2.1). In the adversarial arrival setting, we prove a competitive ratio of  $1/2$  by comparing the performance of Algorithm 1 to the dual of LP-DP, an extension of the LP considered by Brubach et al. [8] from patience values to general probing constraints.

We next move to the ROM/secretary setting. In the unit patience setting of Corollary 1.4, DP-OPT reduces to probing a single edge which yields the largest value in expectation, and LP-DP is a relaxation of LP-std-unit (upper bounds its optimum value). While we do not show this, one could work directly with LP-std-unit and follow the primal-dual argument of Devanur et al. [10]. In contrast, Theorem 1.2 applies to downward-closed probing constraints which comes with two main technical challenges. First, Brubach et al. [8] showed that even the offline committal benchmark has a 0.544 inapproximation against the generalization of LP-std-unit to arbitrary patience (LP-std). Moreover, this inapproximation applies to a stochastic graph which is both rankable and has vanishingly small edge probabilities. Thus, Theorem 1.2 cannot be proven by comparing the performance of Algorithm 1 to LP-std and its dual, even for patience values. Our solution is to instead work with LP-DP and its dual, LP-dual-DP. When a match between  $u \in U$  and  $v \in V$  is successfully made, we apply the well-studied cost sharing function  $g(z) := \exp(z - 1)$  to split the weight of  $u$ , as in [10]. However, LP-dual-DP contains variables which do *not* have an analogue in the classical setting. Specifically, the online vertices are associated with exponentially many variables, and we cost share with the offline vertices which were available when  $v$  was matched to  $u$ , opposed to just  $v$  itself. The second main technical challenge is that when moving away from the unit patience setting, the executions of Algorithm 1 become **non-monotonic**. Specifically, while  $v$  may get matched to  $u$ , if a new online vertex  $v^*$  is added to the graph ahead of  $v$ , then  $u$  may not be matched at all. This complicates the analysis, and is the reason the competitive ratio of Theorem 1.2 does not hold unconditionally, as we explain in Section 2.

In the edge weighted setting, upon receiving the online vertices  $V_t := \{v_1, \dots, v_t\}$ , in order to generalize the matching algorithm of Kesselheim et al. [18], Algorithm 2 would ideally probe the edges of  $\partial(v_t)$  suggested by  $\text{OPT}(G_t)$ , where  $G_t := G[U \cup V_t]$  is the induced stochastic graph on  $U \cup V_t$ . However, since we wish for our algorithms to be efficient in addition to attaining optimal competitive ratios, this strategy is not feasible. We instead make use of a new LP (LP-config) recently introduced by the authors in [5] and independently by Brubach et al. in [6, 13] for the special case of patience values, an updated version of [8]. This LP has exponentially many variables which accounts for the many probing strategies available to an arriving vertex  $v$  with probing constraint  $C_v$ . We solve this LP efficiently by using DP-OPT as a deterministic separation oracle for LP-config-dual, the dual of LP-config, in conjunction with the ellipsoid algorithm [26, 11]. This LP closely resembles what the committal benchmark is capable of doing, and thus leads to a probing algorithm with an optimum competitive ratio.

## 2 Vertex Weights

In this section, we define Algorithm 1 and introduce the techniques needed to prove Theorems 1.1 and 1.2. However, for space considerations, we defer the dual-fitting argument used in the adversarial arrival setting of Theorem 1.1 to Appendix B.

Suppose that  $G = (U, V, E)$  is a vertex weighted stochastic graph with weights  $(w_u)_{u \in U}$ . Let us now fix  $s \in V$ , and define  $\text{val}(e)$  to be the expected weight of the edge matched, provided the edges of  $e$  are probed in order, where  $e \in \mathcal{C}_s$ . Observe then the following claim:

► **Theorem 2.1.** *There exists a dynamic programming (DP) based algorithm DP-OPT, which given access to  $G[\{s\} \cup U]$ , computes a tuple  $e' \in \mathcal{C}_s$ , such that  $\text{OPT}(s, U) = \text{val}(e')$ . Moreover, DP-OPT executes in time  $O(|U|^2)$ , assuming access to a membership oracle for the downward-closed constraint  $\mathcal{C}_s$ .*

**Proof of Theorem 2.1.** It will be convenient to denote  $w_{u,s} := w_u$  for each  $u \in U$  such that  $(u, s) \in \partial(s)$ . We first must show that there exists some  $e' \in \mathcal{C}_s$  such that  $\text{val}(e') = \text{OPT}(s, U)$ , where

$$\text{val}(e) := \sum_{i=1}^{|\mathbf{e}|} p_{e_i} w_{e_i} \prod_{j=1}^{i-1} (1 - p_{e_j}), \quad (2.1)$$

for  $e \in \mathcal{C}_s$ , and  $\text{OPT}(s, U)$  is the value of the committal benchmark on  $G[\{s\} \cup U]$ . Since the committal benchmark must respect commitment – i.e., match the first edge to  $s$  which it reveals to be active – it is clear that  $e'$  exists.

Our goal is to now show that  $e'$  can be computed efficiently. Now, for any  $e \in \mathcal{C}_s$ , let  $e^r$  be the rearrangement of  $e$ , based on the non-increasing order of the weights  $(w_e)_{e \in e}$ . Since  $\mathcal{C}_s$  is downward-closed, we know that  $e^r$  is also in  $\mathcal{C}_s$ . Moreover,  $\text{val}(e^r) \geq \text{val}(e)$  (following observations in [24, 8]). Hence, let us order the edges of  $\partial(s)$  as  $e_1, \dots, e_m$ , such that  $w_{e_1} \geq \dots \geq w_{e_m}$ , where  $m := |\partial(s)|$ . Observe then that it suffices to maximize (2.1) over those strings within  $\mathcal{C}_s$  which respect this ordering on  $\partial(s)$ . Stated differently, let us denote  $\mathcal{I}_s$  as the family of subsets of  $\partial(s)$  induced by  $\mathcal{C}_s$ , and define the set function  $f : 2^{\partial(s)} \rightarrow [0, \infty)$ , where  $f(B) := \text{val}(\mathbf{b})$  for  $B = \{b_1, \dots, b_{|B|}\} \subseteq \partial(s)$ , such that  $\mathbf{b} = (b_1, \dots, b_{|B|})$  and  $w_{b_1} \geq \dots \geq w_{b_{|B|}}$ . Our goal is then to efficiently maximize  $f$  over the set-system  $(\partial(s), \mathcal{I}_s)$ . Observe that  $\mathcal{I}_s$  is downward-closed and that we can simulate oracle access to  $\mathcal{I}_s$ , based on our oracle access to  $\mathcal{C}_s$ .

For each  $i = 0, \dots, m - 1$ , denote  $\partial(s)^{>i} := \{e_{i+1}, \dots, e_m\}$ , and  $\partial(s)^{>m} := \emptyset$ . Moreover, define the family of subsets  $\mathcal{I}_s^{>i} := \{B \subseteq \partial(s)^{>i} : B \cup \{e_i\} \in \mathcal{I}_s\}$  for each  $1 \leq i \leq m$ , and  $\mathcal{I}_s^{>0} := \mathcal{I}_s$ . Observe then that  $(\partial(s)^{>i}, \mathcal{I}_s^{>i})$  is a downward-closed set system, as  $\mathcal{I}_s$  is downward-closed. Moreover, we may simulate oracle access to  $\mathcal{I}_s^{>i}$  based on our oracle access to  $\mathcal{I}_s$ .

Denote  $\text{OPT}(\mathcal{I}_s^{>i})$  as the maximum value of  $f$  over constraints  $\mathcal{I}_s^{>i}$ . Observe then that for each  $0 \leq i \leq m - 1$ , the following recursion holds:

$$\text{OPT}(\mathcal{I}_s^{>i}) := \max_{j \in \{i+1, \dots, m\}} (p_{e_j} \cdot w_{e_j} + (1 - p_{e_j}) \cdot \text{OPT}(\mathcal{I}_s^{>j})) \quad (2.2)$$

Hence, given access to the values  $\text{OPT}(\mathcal{I}_s^{>i+1}), \dots, \text{OPT}(\mathcal{I}_s^{>m})$ , we can compute  $\text{OPT}(\mathcal{I}_s^{>i})$  efficiently. Moreover,  $\text{OPT}(\mathcal{I}_s^{>m}) = 0$  by definition. Thus, it is clear that we can use (2.2) to recover an optimal solution to  $f$ . We can define DP-OPT to be a memoization based implementation of (2.2). It is clear DP-OPT can be implemented in the claimed time complexity. ◀

Given  $R \subseteq U$ , consider the induced stochastic graph,  $G[\{s\} \cup R]$  for  $R \subseteq U$  which has probing constraint  $\mathcal{C}_s^R \subseteq \mathcal{C}_v$ , constructed by restricting  $\mathcal{C}_s$  to those strings whose entries all lie in  $R \times \{s\}$ . Moreover, denote the output of executing DP-OPT on  $G[\{s\} \cup R]$  by  $\text{DP-OPT}(s, R)$ . Consider now the following online probing algorithm:

---

**Algorithm 1** Greedy-DP.

**Input:** offline vertices  $U$  with vertex weights  $(w_u)_{u \in U}$ .

**Output:** a matching  $\mathcal{M}$  of active edges of the unknown stochastic graph  $G = (U, V, E)$ .

```

1:  $\mathcal{M} \leftarrow \emptyset$ .
2:  $R \leftarrow U$ .
3: for  $t = 1, \dots, n$  do
4:   Let  $v_t$  be the current online arrival node, with constraint  $\mathcal{C}_{v_t}$ .
5:   Set  $e \leftarrow \text{DP-OPT}(v_t, R)$ 
6:   for  $i = 1, \dots, |e|$  do
7:     Probe  $e_i$ .
8:     if  $\text{st}(e_i) = 1$  then
9:       Add  $e_i$  to  $\mathcal{M}$ , and update  $R \leftarrow R \setminus \{u_i\}$ , where  $e_i = (u_i, v_t)$ .
10: return  $\mathcal{M}$ .

```

---

In general, the behaviour of the committal benchmark, namely  $\text{OPT}(s, R)$ , can change very much, even for minor changes to  $R$ . For instance, if  $R = U$ , then  $\text{OPT}(s, U)$  may probe the edge  $(u, s)$  first – thus giving it highest priority – whereas if  $u^* \in U$  is removed from  $U$  (where  $u^* \neq u$ ),  $\text{OPT}(s, U \setminus \{u^*\})$  may not probe  $(u, v)$  at all (see Example B.1 for an explicit instance of this behaviour). As a result, it is easy to consider an execution of Algorithm 1 on  $G$  where  $v$  is matched to  $u$ , but if a new vertex  $v^*$  is added to  $G$  ahead of  $v$ ,  $u$  is never matched. We thus refer to Algorithm 1 as being non-monotonic. This contrasts with the classical setting, in which the deterministic greedy algorithm in the ROM setting does not exhibit this behaviour, and thus is **monotonic**. The absence of monotonicity isn't problematic in the adversarial setting of Theorem 1.1 because our primal-dual charging assignment does not depend on the order of the online vertex arrivals (see Appendix B). This contrasts with the ROM setting, in which Example B.1 can be extended to show that the cost sharing rule  $g(z) := \exp(z - 1)$  will not work in general. Our approach is thus to restrict our attention to stochastic graphs in which executions of Algorithm 1 are either monotonic, or monotonic with high probability. This leads us to the definition of rankability, which characterizes a large number of settings in which Algorithm 1 is monotonic.

Given a vertex  $v \in V$ , and an ordering  $\pi_v$  on  $\partial(v)$ , if  $R \subseteq U$ , then define  $\pi_v(R)$  to be the longest string constructible by iteratively appending the edges of  $R \times \{v\}$  via  $\pi_v$ , subject to respecting constraint  $\mathcal{C}_v^R$ . More precisely, given  $e'$  after processing  $e_1, \dots, e_i$  of  $R \times \{v\}$  ordered according to  $\pi_v$ , if  $(e', e_{i+1}) \in \mathcal{C}_v^R$ , then update  $e'$  by appending  $e_{i+1}$  to its end, otherwise move to the next edge  $e_{i+2}$  in the ordering  $\pi_v$ , assuming  $i + 2 \leq |R|$ . If  $i + 2 > |R|$ , return the current string  $e'$  as  $\pi_v(R)$ . We say that  $v$  is **rankable**, provided there exists a choice of  $\pi_v$  which depends *solely* on  $(p_e)_{e \in \partial(v)}$ ,  $(w_e)_{e \in \partial(v)}$  and  $\mathcal{C}_v$ , such that for *every*  $R \subseteq U$ , the strings  $\text{DP-OPT}(v, R)$  and  $\pi_v(R)$  are equal. Crucially, if  $v$  is rankable, then when vertex  $v$  arrives while executing Algorithm 1, one can compute the ranking  $\pi_v$  on  $\partial(v)$  and probe the adjacent edges of  $R \times \{v\}$  based on this order, subject to not violating the constraint  $\mathcal{C}_v^R$ . By following this probing strategy, the optimality of DP-OPT ensures that the expected weight of the match made to  $v$  will be  $\text{OPT}(v, R)$ . We consider three (non-exhaustive) examples of rankability:

► **Proposition 2.2.** *Let  $G = (U, V, E)$  be a stochastic graph, and suppose that  $v \in V$ . If  $v$  satisfies either of the following conditions, then  $v$  is rankable:*

1.  $v$  has unit patience or unlimited patience; that is,  $\ell_v \in \{1, |U|\}$ .
2.  $v$  has patience  $\ell_v$ , and for each  $u_1, u_2 \in U$ , if  $p_{u_1, v} \leq p_{u_2, v}$  then  $w_{u_1} \leq w_{u_2}$ .
3.  $G$  is unweighted, and  $v$  has a budget  $B_v$  with edge probing costs  $(b_{u, v})_{u \in U}$ , and for each  $u_1, u_2 \in U$ , if  $p_{u_1, v} \leq p_{u_2, v}$  then  $b_{u_1, v} \geq b_{u_2, v}$ .

## 13:10 Secretary Matching Meets Probing with Commitment

► **Remark 2.3.** Note that the cases of Proposition 2.2 subsume all the settings listed in the abstract. The rankable assumption is similar to assumptions referred to as laminar, agreeable and compatible in other applications.

We refer to the stochastic graph  $G$  as **rankable**, provided all of its vertices are themselves rankable. We emphasize that distinct vertices of  $V$  may each use their own separate rankings of their adjacent edges.

As discussed in Subsection 1.3, the 0.544 inapproximation against LP-std [8] prevents us from proving a performance guarantee against LP-std, even for patience values. We instead upper bound  $\text{OPT}(G)$  using a tighter LP relaxation that comes with the additional benefit of applying to downward-closed probing constraints. For each  $u \in U$  and  $v \in V$ , let  $x_{u,v}$  be a decision variable corresponding to the probability that  $\text{OPT}(G)$  probes the edge  $(u, v)$ .

$$\text{maximize} \quad \sum_{u \in U} \sum_{v \in V} w_u \cdot p_{u,v} \cdot x_{u,v} \quad (\text{LP-DP})$$

$$\text{subject to} \quad \sum_{v \in V} p_{u,v} \cdot x_{u,v} \leq 1 \quad \forall u \in U \quad (2.3)$$

$$\sum_{u \in R} w_u \cdot p_{u,v} \cdot x_{u,v} \leq \text{OPT}(v, R) \quad \forall v \in V, R \subseteq U \quad (2.4)$$

$$x_{u,v} \geq 0 \quad \forall u \in U, v \in V \quad (2.5)$$

Denote  $\text{LPOPT}_{\text{DP}}(G)$  as the optimal value of this LP. Constraint (2.3) can be viewed as ensuring that the expected number of matches made to  $u \in U$  is at most 1. Similarly, (2.4) can be interpreted as ensuring that expected stochastic reward of  $v$ , suggested by the solution  $(x_{u,v})_{u \in U, v \in V}$ , is actually attainable by the committal benchmark. Thus,  $\text{OPT}(G) \leq \text{LPOPT}_{\text{DP}}(G)$  (a formal proof specific to patience values is proven in [8]).

### 2.0.1 Defining the Primal-Dual Charging Schemes

In order to prove Theorems 1.1 and 1.2, we employ primal-dual charging arguments based on the dual of LP-DP. For each  $u \in U$ , define the variable  $\alpha_u$ . Moreover, for each  $R \subseteq U$  and  $v \in V$ , define the variable  $\phi_{v,R}$  (these latter variables correspond to constraint (2.4)).

$$\text{minimize} \quad \sum_{u \in U} \alpha_u + \sum_{v \in V} \sum_{R \subseteq U} \text{OPT}(v, R) \cdot \phi_{v,R} \quad (\text{LP-dual-DP})$$

$$\text{subject to} \quad p_{u,v} \cdot \alpha_u + \sum_{\substack{R \subseteq U: \\ u \in R}} w_u \cdot p_{u,v} \cdot \phi_{v,R} \geq w_u \cdot p_{u,v} \quad \forall u \in U, v \in V \quad (2.6)$$

$$\alpha_u \geq 0 \quad \forall u \in U \quad (2.7)$$

$$\phi_{v,R} \geq 0 \quad \forall v \in V, R \subseteq U \quad (2.8)$$

The dual-fitting argument used to prove Theorem 1.2 has an initial set-up which proceeds similarly to the argument in Devanur et al. [10]. Specifically, first define  $g : [0, 1] \rightarrow [0, 1]$  where  $g(z) := \exp(z - 1)$  for  $z \in [0, 1]$ . We shall use  $g$  to perform our charging/cost sharing. Moreover, recall that given  $v \in V$ , we defined  $c_v := \max_{e \in \mathcal{C}_v} |e|$  and  $p_v := \max_{e \in \partial(v)} p_e$ . Using these definitions, we define  $F = F(G)$ , where

$$F(G) := \begin{cases} 1 - \frac{1}{e} & G \text{ is rankable} \\ (1 - \frac{1}{e}) \cdot \min_{v \in V} (1 - p_v)^{c_v} & \text{otherwise} \end{cases} \quad (2.9)$$

In order to prove Theorem 1.2, we shall prove that Algorithm 1 returns a matching of expected weight at least  $F(G) \cdot \text{LPOPT}_{\text{DP}}(G)$  when executing on the stochastic graph  $G$  in the ROM setting. Clearly, we may assume  $F(G) > 0$ , as otherwise there is nothing to prove, so we shall make this assumption for the rest of the section. Note that  $F(G) \leq 1 - 1/e$  no matter the stochastic graph  $G$ .

For each  $v \in V$ , draw  $Y_v \in [0, 1]$  independently and uniformly at random. We assume that the vertices of  $V$  are presented to Algorithm 1 in a non-decreasing order, based on the values of  $(Y_v)_{v \in V}$ . We now describe how the charging assignments are made while Algorithm 1 executes on  $G$ . First, we initialize a dual solution  $((\alpha_u)_{u \in U}, (\phi_{v,R})_{v \in V, R \subseteq U})$  where all the variables are set equal to 0. Next, we take  $v \in V, u \in U$ , and  $R \subseteq U$ , where  $u \in R$ . If  $R$  consists of the unmatched vertices of  $v$  when it arrives at time  $Y_v$ , then suppose that Algorithm 1 matches  $v$  to  $u$  while making its probes to a subset of the edges of  $R \times \{v\}$ . In this case, we **charge**  $w_u \cdot (1 - g(Y_v))/F$  to  $\alpha_u$  and  $w_u \cdot g(Y_v)/(F \cdot \text{OPT}(v, R))$  to  $\phi_{v,R}$ . Observe that each subset  $R \subseteq U$  is charged at most once, as is each  $u \in U$ . Thus,

$$\mathbb{E}[w(\mathcal{M})] = F \cdot \left( \sum_{u \in U} \mathbb{E}[\alpha_u] + \sum_{v \in V} \sum_{R \subseteq U} \text{OPT}(v, R) \cdot \mathbb{E}[\phi_{v,R}] \right), \quad (2.10)$$

where the expectation is over the random variables  $(Y_v)_{v \in V}$  and  $(\text{st}(e))_{e \in E}$ . If we now set  $\alpha_u^* := \mathbb{E}[\alpha_u]$  and  $\phi_{v,R}^* := \mathbb{E}[\phi_{v,R}]$  for  $u \in U, v \in V$  and  $R \subseteq U$ , then (2.10) implies the following lemma:

► **Lemma 2.4.** *Suppose  $G = (U, V, E)$  is a stochastic graph for which Algorithm 1 returns the matching  $\mathcal{M}$  when presented  $V$  based on  $(Y_v)_{v \in V}$  generated u.a.r. from  $[0, 1]$ . In this case, if the variables  $((\alpha_u^*)_{u \in U}, (\phi_{v,R}^*)_{v \in V, R \subseteq U})$  are defined through the above charging scheme, then*

$$\mathbb{E}[w(\mathcal{M})] = F \cdot \left( \sum_{u \in U} \alpha_u^* + \sum_{v \in V} \sum_{R \subseteq U} \text{OPT}(v, R) \cdot \phi_{v,R}^* \right).$$

We claim the following regarding  $((\alpha_u^*)_{u \in U}, (\phi_{v,R}^*)_{v \in V, R \subseteq U})$ :

► **Lemma 2.5.** *If the online nodes of  $G = (U, V, E)$  are presented to Algorithm 1 based on  $(Y_v)_{v \in V}$  generated u.a.r. from  $[0, 1]$ , then the solution  $((\alpha_u^*)_{u \in U}, (\phi_{v,R}^*)_{v \in V, R \subseteq U})$  is a feasible solution to LP-dual-DP.*

Since LP-DP is a relaxation of the committal benchmark, Theorem 1.2 follows from Lemmas 2.4 and 2.5 in conjunction with weak duality.

## 2.0.2 Proving Dual Feasibility: Lemma 2.5

Let us suppose that the variables  $((\alpha_u)_{u \in U}, (\phi_{v,R})_{v \in V, R \subseteq U})$  are defined as in the charging scheme of Section 2.0.1. In order to prove Lemma 2.5, we must show that for each fixed  $u_0 \in U$  and  $v_0 \in V$ , we have that

$$\mathbb{E}[p_{u_0, v_0} \cdot \alpha_{u_0} + w_{u_0} \cdot p_{u_0, v_0} \sum_{\substack{R \subseteq U: \\ u_0 \in R}} \phi_{v_0, R}] \geq w_{u_0} \cdot p_{u_0, v_0}. \quad (2.11)$$

Our strategy for proving (2.11) first involves the same approach as used in Devanur et al. [10]. Specifically, we define the stochastic graph  $\tilde{G} := (U, \tilde{V}, \tilde{E})$ , where  $\tilde{V} := V \setminus \{v_0\}$  and  $\tilde{G} := G[U \cup \tilde{V}]$ . We wish to compare the execution of the algorithm on the instance  $\tilde{G}$  to its execution on the instance  $G$ . It will be convenient to couple the randomness between these two executions by making the following assumptions:

## 13:12 Secretary Matching Meets Probing with Commitment

1. For each  $e \in \tilde{E}$ ,  $e$  is active in  $\tilde{G}$  if and only if it is active in  $G$ .
2. The same random variables,  $(Y_v)_{v \in \tilde{V}}$ , are used in both executions.

If we now focus on the execution of  $\tilde{G}$ , then define the random variable  $\tilde{Y}_c$  where  $\tilde{Y}_c := Y_{v_c}$  if  $u_0$  is matched to some  $v_c \in \tilde{V}$ , and  $\tilde{Y}_c := 1$  if  $u_0$  remains unmatched after the execution on  $\tilde{G}$ . We refer to the random variable  $\tilde{Y}_c$  as the **critical time** of vertex  $u_0$  with respect to  $v_0$ . We claim the following lower bounds on  $\alpha_{u_0}$  in terms of the critical time  $\tilde{Y}_c$ .

► **Proposition 2.6.**

- If  $G$  is rankable, then  $\alpha_{u_0} \geq (1 - \frac{1}{e})^{-1} w_{u_0} (1 - g(\tilde{Y}_c))$ .
- Otherwise,  $\mathbb{E}[\alpha_{u_0} \mid (Y_v)_{v \in V}, (st(e))_{e \in \tilde{E}}] \geq (1 - \frac{1}{e})^{-1} w_{u_0} (1 - g(\tilde{Y}_c))$ .

► **Remark 2.7.** Note that Proposition 2.6 is the only part of the proof of Theorem 1.2 which is affected by whether or not  $G$  is rankable. We defer the proof of Proposition 2.6 to Appendix B.

By taking the appropriate conditional expectation, we can also lower bound the random variables  $(\phi_{v_0, R})_{\substack{R \subseteq U: \\ u_0 \in R}}$ .

► **Proposition 2.8.**

$$\sum_{\substack{R \subseteq U: \\ u_0 \in R}} \mathbb{E}[\phi_{v_0, R} \mid (Y_v)_{v \in \tilde{V}}, (st(e))_{e \in \tilde{E}}] \geq \frac{1}{F} \int_0^{\tilde{Y}_c} g(z) dz.$$

**Proof of Proposition 2.8.** We first define  $R_{v_0}$  as the unmatched vertices of  $U$  when  $v_0$  arrives (this is a random subset of  $U$ ). We also once again use  $\mathcal{M}$  to denote the matching returned by Algorithm 1 when executing on  $G$ . If we now take a *fixed* subset  $R \subseteq U$ , then the charging assignment to  $\phi_{v_0, R}$  ensures that

$$\phi_{v_0, R} = w(\mathcal{M}(v_0)) \cdot \frac{g(Y_{v_0})}{F \cdot \text{OPT}(v_0, R)} \cdot \mathbf{1}_{[R_{v_0} = R]},$$

where  $w(\mathcal{M}(v_0))$  corresponds to the weight of the vertex matched to  $v_0$  (which is zero if  $v_0$  remains unmatched after the execution on  $G$ ). In order to make use of this relation, let us first condition on the values of  $(Y_v)_{v \in V}$ , as well as the states of the edges of  $\tilde{E}$ ; that is,  $(st(e))_{e \in \tilde{E}}$ . Observe that once we condition on this information, we can determine  $g(Y_{v_0})$ , as well as  $R_{v_0}$ . As such,

$$\mathbb{E}[\phi_{v_0, R} \mid (Y_v)_{v \in V}, (st(e))_{e \in \tilde{E}}] = \frac{g(Y_{v_0})}{F \cdot \text{OPT}(v_0, R)} \mathbb{E}[w(\mathcal{M}(v_0)) \mid (Y_v)_{v \in V}, (st(e))_{e \in \tilde{E}}] \cdot \mathbf{1}_{[R_{v_0} = R]}.$$

On the other hand, the only randomness which remains in the conditional expectation involving  $w(\mathcal{M}(v_0))$  is over the states of the edges adjacent to  $v_0$ . Observe now that since Algorithm 1 behaves optimally on  $G[\{v_0\} \cup R_{v_0}]$ , we get that

$$\mathbb{E}[w(\mathcal{M}(v_0)) \mid (Y_v)_{v \in V}, (st(e))_{e \in \tilde{E}}] = \text{OPT}(v_0, R_{v_0}), \tag{2.12}$$

and so for the *fixed* subset  $R \subseteq U$ ,

$$\mathbb{E}[w(\mathcal{M}(v_0)) \mid (Y_v)_{v \in V}, (st(e))_{e \in \tilde{E}}] \cdot \mathbf{1}_{[R_{v_0} = R]} = \text{OPT}(v_0, R) \cdot \mathbf{1}_{[R_{v_0} = R]}$$

after multiplying each side of (2.12) by the indicator random variable  $\mathbf{1}_{[R_{v_0} = R]}$ . Thus,

$$\mathbb{E}[\phi_{v_0, R} \mid (Y_v)_{v \in V}, (st(e))_{e \in \tilde{E}}] = \frac{g(Y_{v_0})}{F} \mathbf{1}_{[R_{v_0} = R]},$$



after cancellation. We therefore get that

$$\sum_{\substack{R \subseteq U: \\ u_0 \in R}} \mathbb{E}[\phi_{v_0, R} \mid (Y_v)_{v \in V}, (\text{st}(e))_{e \in \tilde{E}}] = \frac{g(Y_{v_0})}{F} \sum_{\substack{R \subseteq U: \\ u_0 \in R}} \mathbf{1}_{[R_{v_0} = R]}.$$

Let us now focus on the case when  $v_0$  arrives before the critical time; that is,  $0 \leq Y_{v_0} < \tilde{Y}_c$ . Up until the arrival of  $v_0$ , the executions of the algorithm on  $\tilde{G}$  and  $G$  proceed identically, thanks to the coupling between the executions. As such,  $u_0$  must be available when  $v_0$  arrives. We interpret this observation in the above notation as saying the following:

$$\mathbf{1}_{[Y_{v_0} < \tilde{Y}_c]} \leq \sum_{\substack{R \subseteq U: \\ u_0 \in R}} \mathbf{1}_{[R_{v_0} = R]}.$$

As a result,

$$\sum_{\substack{R \subseteq U: \\ u_0 \in R}} \mathbb{E}[\phi_{v_0, R} \mid (Y_v)_{v \in V}, (\text{st}(e))_{e \in \tilde{E}}] \geq \frac{g(Y_{v_0})}{F} \mathbf{1}_{[Y_{v_0} < \tilde{Y}_c]}.$$

Now, if we take expectation over  $Y_{v_0}$ , while still conditioning on the random variables  $(Y_v)_{v \in \tilde{V}}$ , then we get that

$$\mathbb{E}[g(Y_{v_0}) \cdot \mathbf{1}_{[Y_{v_0} < \tilde{Y}_c]} \mid (Y_v)_{v \in \tilde{V}}, (\text{st}(e))_{e \in \tilde{E}}] = \int_0^{\tilde{Y}_c} g(z) dz,$$

as  $Y_{v_0}$  is drawn uniformly from  $[0, 1]$ , independently from  $(Y_v)_{v \in \tilde{V}}$  and  $(\text{st}(e))_{e \in \tilde{E}}$ . Thus, after applying the law of iterated expectations,

$$\sum_{\substack{R \subseteq U: \\ u_0 \in R}} \mathbb{E}[\phi_{v_0, R} \mid (Y_v)_{v \in \tilde{V}}, (\text{st}(e))_{e \in \tilde{E}}] \geq \frac{1}{F} \int_0^{\tilde{Y}_c} g(z) dz,$$

and so the claim holds. ◀

With Propositions 2.6 and 2.8, the proof of Lemma 2.5 follows easily (see Appendix B), and so Theorem 1.2 is proven.

### 3 Edge Weights

Let us suppose that  $G = (U, V, E)$  is a stochastic graph with arbitrary edge weights, probabilities and downward-closed probing constraints  $(\mathcal{C}_v)_{v \in V}$ . For each  $k \geq 1$  and  $\mathbf{e} = (e_1, \dots, e_k) \in E^{(k)}$ , define  $g(\mathbf{e}) := \prod_{i=1}^k (1 - p_{e_i})$ . Notice that  $g(\mathbf{e})$  corresponds to the probability that all the edges of  $\mathbf{e}$  are inactive, where  $g(\lambda) := 1$  for the empty string  $\lambda$ . We also define  $\mathbf{e}_{< e_i} := (e_1, \dots, e_{i-1})$  for each  $2 \leq i \leq k$ , which we denote by  $\mathbf{e}_{< i}$  when clear. By convention,  $\mathbf{e}_{< 1} := \lambda$ . Observe then that  $\text{val}(\mathbf{e}) := \sum_{i=1}^{|\mathbf{e}|} p_{e_i} w_{e_i} \cdot g(\mathbf{e}_{< i})$  corresponds to the expected weight of the first active edge if  $\mathbf{e}$  is probed in order of its indices, where  $\text{val}(\lambda) := 0$ .

For each  $v \in V$ , we introduce a decision variable denoted  $x_v(\mathbf{e})$ , which may loosely be interpreted as the likelihood the committal benchmark probes the edges in the order specified

## 13:14 Secretary Matching Meets Probing with Commitment

by  $\mathbf{e} = (e_1, \dots, e_k)$ <sup>6</sup>. With this notation, we express the following LP:

$$\text{maximize} \quad \sum_{v \in V} \sum_{\mathbf{e} \in \mathcal{C}_v} \text{val}(\mathbf{e}) \cdot x_v(\mathbf{e}) \quad (\text{LP-config})$$

$$\text{subject to} \quad \sum_{v \in V} \sum_{\substack{\mathbf{e} \in \mathcal{C}_v: \\ (u,v) \in \mathbf{e}}} p_{u,v} \cdot g(\mathbf{e}_{<(u,v)}) \cdot x_v(\mathbf{e}) \leq 1 \quad \forall u \in U \quad (3.1)$$

$$\sum_{\mathbf{e} \in \mathcal{C}_v} x_v(\mathbf{e}) = 1 \quad \forall v \in V, \quad (3.2)$$

$$x_v(\mathbf{e}) \geq 0 \quad \forall v \in V, \mathbf{e} \in \mathcal{C}_v \quad (3.3)$$

Denote  $\text{LPOPT}_{\text{conf}}(G)$  as the optimal value of LP-config. This LP was developed from insights relevant to both the secretary and prophet settings. Specifically, the DP-OPT algorithm of Theorem 2.1 can be used as a (deterministic) polynomial time separation oracle for the dual of LP-config. This ensures that LP-config can be solved in polynomial time as a consequence of how the ellipsoid algorithm [26, 11] executes (see Theorem A.1 in Appendix A for details). In [5], we prove that LP-config is a relaxation of the committal benchmark. Unlike previous LP relaxations of the committal benchmark, we are not aware of an easy proof of this fact, and we consider it to be a technical contribution.

We now define a *fixed vertex* probing algorithm, called VERTEXPROBE, which is applied to an online vertex  $s$  of an arbitrary stochastic graph (potentially distinct from  $G$ ) with probing constraints  $\mathcal{C}_s$  on  $\partial(s)$ . Specifically, given non-negative values  $(z(\mathbf{e}))_{\mathbf{e} \in \mathcal{C}_s}$  which satisfy  $\sum_{\mathbf{e} \in \mathcal{C}_s} z(\mathbf{e}) = 1$ , draw  $\mathbf{e}'$  with probability  $z(\mathbf{e}')$ . If  $\mathbf{e}' = (e'_1, \dots, e'_k)$  for  $k := |\mathbf{e}'| \geq 1$ , then probe the edges of  $\mathbf{e}'$  in order, and match  $s$  to the first edge revealed to be active. If no such edge exists, or  $\mathbf{e}' = \lambda$ , then return  $\emptyset$ .

► **Lemma 3.1.** *Suppose VERTEXPROBE is passed a fixed online node  $s$  of a stochastic graph, and values  $(z(\mathbf{e}))_{\mathbf{e} \in \mathcal{C}_s}$  which satisfy  $\sum_{\mathbf{e} \in \mathcal{C}_s} z(\mathbf{e}) = 1$ . If for each  $e \in \partial(s)$ ,*

$$\tilde{z}_e := \sum_{\substack{\mathbf{e}' \in \mathcal{C}_v: \\ e \in \mathbf{e}'}} g(\mathbf{e}'_{<e}) \cdot z_v(\mathbf{e}'),$$

*then  $e$  is probed with probability  $\tilde{z}_e$ , and returned by the algorithm with probability  $p_e \cdot \tilde{z}_e$ .*

► **Remark 3.2.** If VERTEXPROBE outputs the edge  $e = (u, s)$  when executing on the fixed node  $s$ , then we say that  $s$  **commits** to the edge  $e = (u, s)$ , or that  $s$  commits to  $u$ .

Returning to the problem of designing an online probing algorithm for  $G$ , let us assume that  $n := |V|$ , and that the online nodes of  $V$  are denoted  $v_1, \dots, v_n$ , where the order is generated *u.a.r.* Denote  $V_t$  as the set of first  $t$  arrivals of  $V$ ; that is,  $V_t := \{v_1, \dots, v_t\}$ . Moreover, set  $G_t := G[U \cup V_t]$ , and  $\text{LPOPT}_{\text{conf}}(G_t)$  as the value of an optimal solution to LP-config (this is a random variable, as  $V_t$  is a random subset of  $V$ ). The following inequality then holds:

► **Lemma 3.3.** *For each  $t \geq 1$ ,  $\mathbb{E}[\text{LPOPT}_{\text{conf}}(G_t)] \geq \frac{t}{n} \text{LPOPT}_{\text{conf}}(G)$ .*

In light of this observation, we design an online probing algorithm which makes use of  $V_t$ , the currently known nodes, to derive an optimal LP solution with respect to  $G_t$ . As such, each time an online node arrives, we must compute an optimal solution for the LP associated to  $G_t$ , distinct from the solution computed for that of  $G_{t-1}$ .

<sup>6</sup> While this is the natural interpretation of the decision variables of LP-config, to the best of our knowledge, formally defining the variables in this way does not lead to a proof that LP-config relaxes the committal benchmark. We discuss this in detail in [5].

■ **Algorithm 2** Unknown Stochastic Graph ROM.

**Input:**  $U$  and  $n := |V|$ .

**Output:** a matching  $\mathcal{M}$  from the (unknown) stochastic graph  $G = (U, V, E)$  of active edges.

---

```

1: Set  $\mathcal{M} \leftarrow \emptyset$ .
2: Set  $G_0 = (U, \emptyset, \emptyset)$ 
3: for  $t = 1, \dots, n$  do
4:   Input  $v_t$ , with  $(w_e)_{e \in \partial(v_t)}$ ,  $(p_e)_{e \in \partial(v_t)}$  and  $\mathcal{C}_{v_t}$ .
5:   Compute  $G_t$ , by updating  $G_{t-1}$  to contain  $v_t$  (and its relevant information).
6:   if  $t < \lfloor n/e \rfloor$  then
7:     Pass on  $v_t$ .
8:   else
9:     Solve LP-config for  $G_t$  and find an optimal solution  $(x_v(e))_{v \in V_t, e \in \mathcal{C}_v}$ .
10:    Set  $e_t \leftarrow \text{VERTEXPROBE}(v_t, \partial(v_t), (x_v(e))_{e \in \mathcal{C}_{v_t}})$ .
11:    if  $e_t = (u_t, v_t) \neq \emptyset$  and  $u_t$  is unmatched then
12:      Add  $e_t$  to  $\mathcal{M}$ .
13: return  $\mathcal{M}$ .

```

---

► **Remark 3.4.** Unlike the algorithm of Kesselheim et al., our algorithm is randomized, and we do not know whether the polytope LP-config always admits an optimum integral solution. We leave it as an interesting open question as to whether or not Algorithm 2 can be derandomized.

Let us consider the matching  $\mathcal{M}$  returned by the algorithm, as well as its weight, which we denote by  $w(\mathcal{M})$ . Set  $\alpha := 1/e$  for clarity, and take  $t \geq \lceil \alpha n \rceil$ . For each  $\alpha n \leq t \leq n$ , define  $R_t$  as the *unmatched vertices* of  $U$  when vertex  $v_t$  arrives. Note that committing to  $e_t = (u_t, v_t)$  is necessary, but not sufficient, for  $v_t$  to match to  $u_t$ . With this notation, we have that  $\mathbb{E}[w(\mathcal{M})] = \sum_{t=\alpha n}^n \mathbb{E}[w(u_t, v_t) \cdot \mathbf{1}_{[u_t \in R_t]}]$ . Moreover, we claim the following:

► **Lemma 3.5.** For each  $t \geq \lceil \alpha n \rceil$ ,  $\mathbb{E}[w(e_t)] \geq LPOPT_{conf}(G)/n$ .

► **Lemma 3.6.** For each  $t \geq \lceil \alpha n \rceil$ , define  $f(t, n) := \lfloor \alpha n \rfloor / (t - 1)$ . In this case,  $\mathbb{P}[u_t \in R_t \mid V_t, v_t] \geq f(t, n)$ , where  $V_t = \{v_1, \dots, v_t\}$  and  $v_t$  is the  $t^{\text{th}}$  arriving node of  $V$ <sup>7</sup>.

The proofs of Lemmas 3.5 and 3.6 mostly follow the analogous claims as proven by Kesselheim et al. in the classic secretary matching problem. We present formal proofs in the arXiv version [4]. With these lemmas, together with the efficient solvability of LP-config, the proof of Theorem 1.6 follows easily (see Appendix C).

## 4 Conclusion and Open Problems

We considered the online stochastic bipartite matching with commitment in a number of different settings establishing several competitive bounds against the committal benchmark. Our work leaves open a number of challenging problems. For context we note that currently, even for the classical (i.e., non-probing) setting,  $1 - \frac{1}{e}$  is the best known ratio for deterministic algorithms operating on unweighted or vertex weighted graphs with random order vertex arrivals. The best known ROM inapproximation of 0.823 (due to Manshadi et al. [21]) comes from the classical i.i.d. unweighted graph setting for a known distribution and applies to randomized as well as deterministic algorithms.

---

<sup>7</sup> Note that since  $V_t$  is a set, conditioning on  $V_t$  only reveals which vertices of  $V$  encompass the first  $t$  arrivals, *not* the order they arrived in. Hence, conditioning on  $v_t$  as well reveals strictly more information.

- What is the best ratio that a deterministic or randomized online algorithm can obtain for *all* vertex weighted stochastic graphs in the ROM setting? That is, what competitive ratio can be achieved without the rankable assumption? Is there an online probing algorithm which can surpass the  $1 - 1/e$  “barrier” with or without the rankable assumption? Here we note that in the classical ROM setting, the RANKING algorithm achieves a 0.696 ratio for unweighted graphs (due to Mahdian and Yan [20]) and a 0.6534 ratio (due to Huang et al. [15]) for vertex weighted graphs. Thus, randomization seems to significantly help in the classical ROM setting.
- What is the best ratio that a randomized online algorithm can obtain for stochastic graphs in the adversarial arrival model? The Mehta and Panigrahi [22] 0.621 inapproximation shows that randomized probing algorithms (even for unweighted graphs and unit patience) cannot achieve a  $1 - 1/e$  performance guarantee against LP-std-unit, however the work of Goyal and Udvani [12] suggests that this is because LP-std-unit is too loose a relaxation of the committal benchmark.
- For edge weighted graphs, can we achieve a  $\frac{1}{e}$  competitive ratio (or any constant ratio) by a combinatorial (and more efficient) algorithm? Our vertex weighted algorithm can be viewed as a truthful online (or random order) posted price mechanism. Can we modify the edge weighted algorithm to be a truthful mechanism thereby generalizing the truthful mechanism of Reiffenhauser [25]? Note that unlike the vertex weighted algorithm, our algorithm for edge weights does not necessarily make an optimal social welfare decision for each online node.

---

## References

- 1 Marek Adamczyk. Improved analysis of the greedy algorithm for stochastic matching. *Inf. Process. Lett.*, 111(15):731–737, 2011.
- 2 Marek Adamczyk, Fabrizio Grandoni, Stefano Leonardi, and Michal Włodarczyk. When the optimum is also blind: a new perspective on universal optimization. In *ICALP*, 2017.
- 3 Marek Adamczyk, Fabrizio Grandoni, and Joydeep Mukherjee. Improved approximation algorithms for stochastic matching. *CoRR*, abs/1505.01439, 2015. [arXiv:1505.01439](https://arxiv.org/abs/1505.01439).
- 4 Allan Borodin, Calum MacRury, and Akash Rakheja. Greedy approaches to online stochastic matching. *CoRR*, abs/2008.09260, 2020. URL: <https://arxiv.org/abs/2008.09260>.
- 5 Allan Borodin, Calum MacRury, and Akash Rakheja. Prophet inequality matching meets probing with commitment. *CoRR*, abs/2102.04325, 2021. URL: <https://arxiv.org/abs/2102.04325>.
- 6 Brian Brubach, Nathaniel Grammel, Will Ma, and Aravind Srinivasan. Follow your star: New frameworks for online stochastic matching with known and unknown patience. *CoRR*, abs/1907.03963, 2021.
- 7 Brian Brubach, Nathaniel Grammel, Will Ma, and Aravind Srinivasan. Improved guarantees for offline stochastic matching via new ordered contention resolution schemes. *CoRR*, abs/2106.06892, 2021. [arXiv:2106.06892](https://arxiv.org/abs/2106.06892).
- 8 Brian Brubach, Nathaniel Grammel, and Aravind Srinivasan. Vertex-weighted online stochastic matching with patience constraints. *2019*, 1907.03963, 2019. [arXiv:1907.03963](https://arxiv.org/abs/1907.03963).
- 9 Ning Chen, Nicole Immorlica, Anna R. Karlin, Mohammad Mahdian, and Atri Rudra. Approximating matches made in heaven. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming: Part I*, ICALP '09, pages 266–278, 2009.
- 10 Nikhil R. Devanur, Kamal Jain, and Robert D. Kleinberg. Randomized primal-dual analysis of ranking for online bipartite matching. In *Proceedings of the Twenty-fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '13, pages 101–107, Philadelphia, PA, USA, 2013. Society for Industrial and Applied Mathematics. URL: <http://dl.acm.org/citation.cfm?id=2627817.2627824>.

- 11 Bernd Gärtner and Jirí Matousek. *Understanding and using linear programming*. Universitext. Springer, 2007.
- 12 Vineet Goyal and R. Udvani. Online matching with stochastic rewards: Optimal competitive ratio via path based formulation. *Proceedings of the 21st ACM Conference on Economics and Computation*, 2020.
- 13 Nathaniel Grammel, Brian Brubach, Will Ma, and Aravind Srinivasan. Follow your star: New frameworks for online stochastic matching with known and unknown patience. In *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, pages 2872–2880, 2021.
- 14 Anupam Gupta and Viswanath Nagarajan. A stochastic probing problem with applications. In Michel X. Goemans and José R. Correa, editors, *Integer Programming and Combinatorial Optimization - 16th International Conference, IPCO 2013, Valparaíso, Chile, March 18-20, 2013. Proceedings*, volume 7801 of *Lecture Notes in Computer Science*, pages 205–216. Springer, 2013.
- 15 Zhiyi Huang, Zhihao Gavin Tang, Xiaowei Wu, and Yuhao Zhang. Online vertex-weighted bipartite matching: Beating  $1-1/e$  with random arrivals, 2018. [arXiv:1804.07458](https://arxiv.org/abs/1804.07458).
- 16 Zhiyi Huang and Qiankun Zhang. Online primal dual meets online matching with stochastic rewards: Configuration lp to the rescue. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020*, page 1153–1164, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3357713.3384294.
- 17 Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 352–358, 1990.
- 18 Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions. In Hans L. Bodlaender and Giuseppe F. Italiano, editors, *Algorithms – ESA 2013*, pages 589–600, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- 19 Euiwoong Lee and Sahil Singla. Optimal Online Contention Resolution Schemes via Ex-Ante Prophet Inequalities. In Yossi Azar, Hannah Bast, and Grzegorz Herman, editors, *26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 57:1–57:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ESA.2018.57.
- 20 Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: An approach based on strongly factor-revealing lps. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing, STOC '11*, pages 597–606, New York, NY, USA, 2011. ACM. doi:10.1145/1993636.1993716.
- 21 Vahideh H. Manshadi, Shayan Oveis Gharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics. *Math. Oper. Res.*, 37(4):559–573, 2012.
- 22 Aranyak Mehta and Debmalya Panigrahi. Online matching with stochastic rewards. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 728–737. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.65.
- 23 Aranyak Mehta, Bo Waggoner, and Morteza Zadimoghaddam. Online stochastic matching with unequal probabilities. In *SODA*, pages 1388–1404, 2015.
- 24 Manish Purohit, Sreenivas Gollapudi, and Manish Raghavan. Hiring under uncertainty. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5181–5189. PMLR, June 09–15 2019.
- 25 Rebecca Reiffenhäuser. An optimal truthful mechanism for the online weighted bipartite matching problem. 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 1982–1993. SIAM, 2019.

- 26 D. Seese. Groetschel, m., l. lovasz, a. schrijver: Geometric algorithms and combinatorial optimization. (algorithms and combinatorics. eds.: R. l. graham, b. korte, l. lovasz. vol. 2), springer-verlag 1988, xii, 362 pp., 23 figs., dm 148,-. isbn 3-540-13624-x. *Biometrical Journal*, 32(8):930-930, 1990. doi:10.1002/bimj.4710320805.
- 27 Jan Vondrák, Chandra Chekuri, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing, STOC '11*, page 783-792, New York, NY, USA, 2011. Association for Computing Machinery. doi:10.1145/1993636.1993740.
- 28 David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, USA, 1st edition, 2011.

## A Solving LP-config Efficiently

Suppose that we are given an arbitrary stochastic graph  $G = (U, V, E)$ . We contrast LP-config with LP-std, which is defined only when  $G$  has patience values  $(\ell_v)_{v \in V}$ :

$$\begin{array}{ll} \text{maximize} & \sum_{e \in E} w_e \cdot p_e \cdot x_e \end{array} \quad (\text{LP-std})$$

$$\begin{array}{ll} \text{subject to} & \sum_{e \in \partial(u)} p_e \cdot x_e \leq 1 \quad \forall u \in U \end{array} \quad (\text{A.1})$$

$$\sum_{e \in \partial(v)} p_e \cdot x_e \leq 1 \quad \forall v \in V \quad (\text{A.2})$$

$$\sum_{e \in \partial(v)} x_e \leq \ell_v \quad \forall v \in V \quad (\text{A.3})$$

$$0 \leq x_e \leq 1 \quad \forall e \in E. \quad (\text{A.4})$$

Observe that LP-config and LP-std are the same LP in the case of unit patience:

$$\begin{array}{ll} \text{maximize} & \sum_{v \in V} \sum_{e \in \partial(v)} w_e \cdot p_e \cdot x_e \end{array} \quad (\text{LP-std-unit})$$

$$\begin{array}{ll} \text{subject to} & \sum_{e \in \partial(u)} p_e \cdot x_e \leq 1 \quad \forall u \in U \end{array} \quad (\text{A.5})$$

$$\sum_{e \in \partial(v)} x_e \leq 1 \quad \forall v \in V \quad (\text{A.6})$$

$$x_e \geq 0 \quad \forall e \in E \quad (\text{A.7})$$

### A.1 Solving LP-config Efficiently

We now show how LP-config be solved efficiently under the assumptions of Theorem 1.6.

► **Theorem A.1.** *Suppose that  $G = (U, V, E)$  in a stochastic graph with downward-closed probing constraints  $(\mathcal{C}_v)_{v \in V}$ . In the membership oracle model, LP-config is efficiently solvable in  $|G|$ .*

We prove Theorem A.1 by first considering the dual of LP-config. Note, that in the below LP formulation, if  $\mathbf{e} = (e_1, \dots, e_k) \in \mathcal{C}_v$ , then we set  $e_i = (u_i, v)$  for  $i = 1, \dots, k$  for convenience.

$$\begin{aligned}
& \text{minimize} && \sum_{u \in U} \alpha_u + \sum_{v \in V} \beta_v && \text{(LP-config-dual)} \\
& \text{subject to} && \beta_v + \sum_{j=1}^{|\mathbf{e}|} p_{e_j} \cdot g(\mathbf{e}_{<j}) \cdot \alpha_{u_j} \geq \sum_{j=1}^{|\mathbf{e}|} p_{e_j} \cdot w_{e_j} \cdot g(\mathbf{e}_{<j}) && \forall v \in V, \mathbf{e} \in \mathcal{C}_v \\
& && \alpha_u \geq 0 && \forall u \in U \\
& && \beta_v \in \mathbb{R} && \forall v \in V
\end{aligned}$$

Observe that to prove Theorem A.1, it suffices to show that LP-config-dual has a (deterministic) polynomial time separation oracle, as a consequence of how the ellipsoid algorithm [26, 11] executes (see [28, 27, 2, 19] for more detail).

Suppose that we are presented a particular selection of dual variables, say  $(\alpha_u)_{u \in U}$  and  $(\beta_v)_{v \in V}$ , which may or may not be a feasible solution to LP-config-dual. Our separation oracle must determine efficiently whether these variables satisfy all the constraints of LP-config-dual. In the case in which the solution is *infeasible*, the oracle must additionally return a constraint which is violated.

It is clear that we can accomplish this for the non-negativity constraints, so let us fix a particular  $v \in V$  in what follows. We wish to determine whether there exists some  $\mathbf{e} = (e_1, \dots, e_{|\mathbf{e}|}) \in \mathcal{C}_v$ , such that if  $e_i = (u_i, v)$  for  $i = 1, \dots, k$ , then

$$f(\mathbf{e}) := \sum_{j=1}^{|\mathbf{e}|} (w_{e_j} - \alpha_{u_j}) \cdot p_{e_j} \cdot g(\mathbf{e}_{<j}) > \beta_v, \quad (\text{A.8})$$

where  $f(\mathbf{e}) := 0$  if  $\mathbf{e} = \lambda$ .

► **Lemma A.2.** *In the membership oracle model, DP-OPT of Proposition 2.1 can be used to efficiently check whether  $f(\mathbf{e}') > \beta_v$  for some  $\mathbf{e}' \in \mathcal{C}_v$ , provided  $\mathcal{C}_v$  is downward-closed. Moreover, if such a tuple exists, then it can be found efficiently.*

**Proof.** In order to make this statement, it suffices to show how one can use DP-OPT to maximize the function  $f$  efficiently.

Compute  $\tilde{w}_e := w_e - \alpha_u$  for each  $e = (u, v) \in \partial(v)$ , and define  $P := \{e \in \partial(v) : \tilde{w}_e \geq 0\}$ . First observe that if  $P = \emptyset$ , then (A.8) is maximized by the empty-string  $\lambda$ . Thus, for now on assume that  $P \neq \emptyset$ . Since  $\mathcal{C}_v$  is downward-closed, it suffices to consider those  $\mathbf{e} \in \mathcal{C}_v$  whose edges all lie in  $P$ . As such, for notational convenience, let us hereby assume that  $\partial(v) = P$ . Observe then that maximizing  $f$  corresponds to executing DP-OPT on the stochastic graph  $G[U \cup \{v\}]$ , with edge weights replaced by  $(\tilde{w}_e)_{e \in \partial(v)}$ . ◀

## B Proofs and Additions to Section 2

**Proof of Theorem 1.1.** Let  $G = (U, V, E)$  be a vertex weighted stochastic graph, and assume that Algorithm 1 returns the matching  $\mathcal{M}$  when the online vertices of  $G$  are presented to the algorithm in adversarial order.

We now define a charging assignment as Algorithm 1 executes on  $G$ . First, initialize a dual solution  $((\alpha_u)_{u \in U}, (\phi_{v,R})_{v \in V, R \subseteq U})$  where all the variables are set equal to 0. Let us now take  $v \in V, u \in U$ , and  $R \subseteq U$ , where  $u \in R$ . If  $R$  consists of the unmatched vertices when  $v$  it arrives, then suppose that Algorithm 1 matches  $v$  to  $u$  while making its probes to

## 13:20 Secretary Matching Meets Probing with Commitment

a subset of the edges of  $R \times \{v\}$ . In this case, we **charge**  $w_u$  to  $\alpha_u$  and  $w_u/\text{OPT}(v, R)$  to  $\phi_{v,R}$ . Observe that each subset  $R \subseteq U$  is charged at most once, as is each  $u \in U$ . Thus,

$$\mathbb{E}[w(\mathcal{M})] = \frac{1}{2} \cdot \left( \sum_{u \in U} \mathbb{E}[\alpha_u] + \sum_{v \in V} \sum_{R \subseteq U} \text{OPT}(v, R) \cdot \mathbb{E}[\phi_{v,R}] \right), \quad (\text{B.1})$$

where the expectation is over  $(\text{st}(e))_{e \in E}$ . Let us now set  $\alpha_u^* := \mathbb{E}[\alpha_u]$  and  $\phi_{v,R}^* := \mathbb{E}[\phi_{v,R}]$  for  $u \in U, v \in V$  and  $R \subseteq U$ . We claim that  $((\alpha_u^*)_{u \in U}, (\phi_{v,R}^*)_{v \in V, R \subseteq U})$  is a feasible solution to LP-dual-DP. To show this, we must prove that for each fixed  $u_0 \in U$  and  $v_0 \in V$ , we have that

$$\mathbb{E}[p_{u_0, v_0} \cdot \alpha_{u_0} + w_{u_0} \cdot p_{u_0, v_0} \sum_{\substack{R \subseteq U: \\ u_0 \in R}} \phi_{v_0, R}] \geq w_{u_0} \cdot p_{u_0, v_0}. \quad (\text{B.2})$$

We first define  $R_{v_0}$  as the unmatched vertices of  $U$  when  $v_0$  arrives (this is a random subset of  $U$ ). Moreover, define  $\tilde{E} := E \setminus \partial(v_0)$ . We claim the following inequality:

$$\sum_{\substack{R \subseteq U: \\ u_0 \in R}} \mathbb{E}[\phi_{v_0, R} \mid (\text{st}(e))_{e \in \tilde{E}}] = \mathbf{1}_{[u_0 \in R_{v_0}]}$$

To see this, observe that if we take a *fixed* subset  $R \subseteq U$ , then the charging assignment to  $\phi_{v_0, R}$  ensures that

$$\phi_{v_0, R} = w(\mathcal{M}(v_0)) \cdot \frac{1}{\text{OPT}(v_0, R)} \cdot \mathbf{1}_{[R_{v_0} = R]},$$

where  $w(\mathcal{M}(v_0))$  corresponds to the weight of the vertex matched to  $v_0$  (which is zero if  $v_0$  remains unmatched after the execution on  $G$ ). In order to make use of this relation, let us first condition on  $(\text{st}(e))_{e \in \tilde{E}}$ . Observe that once we condition on this information, we can determine  $R_{v_0}$ . As such,

$$\mathbb{E}[\phi_{v_0, R} \mid (\text{st}(e))_{e \in \tilde{E}}] = \frac{1}{\text{OPT}(v_0, R)} \mathbb{E}[w(\mathcal{M}(v_0)) \mid (\text{st}(e))_{e \in \tilde{E}}] \cdot \mathbf{1}_{[R_{v_0} = R]}.$$

On the other hand, the only randomness which remains in the conditional expectation involving  $w(\mathcal{M}(v_0))$  is over  $(\text{st}(e))_{e \in \partial(v_0)}$ . However, since Algorithm 1 behaves optimally on  $G[\{v_0\} \cup R_{v_0}]$ , we get that

$$\mathbb{E}[w(\mathcal{M}(v_0)) \mid (Y_v)_{v \in V}, (\text{st}(e))_{e \in \tilde{E}}] = \text{OPT}(v_0, R_{v_0}), \quad (\text{B.3})$$

and so for the *fixed* subset  $R \subseteq U$ ,

$$\mathbb{E}[w(\mathcal{M}(v_0)) \mid (\text{st}(e))_{e \in \tilde{E}}] \cdot \mathbf{1}_{[R_{v_0} = R]} = \text{OPT}(v_0, R) \cdot \mathbf{1}_{[R_{v_0} = R]}$$

after multiplying each side of (B.3) by the indicator random variable  $\mathbf{1}_{[R_{v_0} = R]}$ . Thus,

$$\mathbb{E}[\phi_{v_0, R} \mid (\text{st}(e))_{e \in \tilde{E}}] = \mathbf{1}_{[R_{v_0} = R]},$$

after cancellation. We therefore get that

$$\sum_{\substack{R \subseteq U: \\ u_0 \in R}} \mathbb{E}[\phi_{v_0, R} \mid (\text{st}(e))_{e \in \tilde{E}}] = \sum_{\substack{R \subseteq U: \\ u_0 \in R}} \mathbf{1}_{[R_{v_0} = R]} = \mathbf{1}_{[u_0 \in R_{v_0}]},$$



as claimed. On the other hand, if we focus on the vertex  $u_0$ , then observe that if  $u_0 \notin R_{v_0}$ , then  $\alpha_{u_0}$  must have been charged  $w_u$ . In other words,  $\alpha_{u_0} \geq w_u \cdot \mathbf{1}_{[u_0 \notin R_{v_0}]}$ . As a result,

$$\mathbb{E}[p_{u_0, v_0} \alpha_{u_0} + w_{u_0} p_{u_0, v_0} \sum_{\substack{R \subseteq U: \\ u_0 \in R}} \phi_{v, R} | (\text{st}(e))_{e \in \tilde{E}}] \geq w_{u_0} p_{u_0, v_0} \cdot \mathbf{1}_{[u_0 \notin R_{v_0}]} + w_{u_0} p_{u_0, v_0} \cdot \mathbf{1}_{[u_0 \in R_{v_0}]},$$

and so (B.2) follows after taking expectations. The solution  $((\alpha_u^*)_{u \in U}, (\phi_{v, R}^*)_{v \in V, R \subseteq U})$  is therefore feasible, and so since  $\text{OPT}(G) \leq \text{LPOPT}_{\text{DP}}(G)$ , the proof is complete after applying weak duality and (B.1).  $\blacktriangleleft$

► **Example B.1.** Let  $G = (U, V, E)$  be a bipartite graph with  $U = \{u_1, u_2, u_3, u_4\}$ ,  $V = \{v\}$  and  $\ell_v = 2$ . Set  $p_{u_1, v} = 1/3$ ,  $p_{u_2, v} = 1$ ,  $p_{u_3, v} = 1/2$ ,  $p_{u_4, v} = 2/3$ . Fix  $\varepsilon > 0$ , and let the weights of offline vertices be  $w_{u_1} = 1 + \varepsilon$ ,  $w_{u_2} = 1 + \varepsilon/2$ ,  $w_{u_3} = w_{u_4} = 1$ . We assume that  $\varepsilon$  is sufficiently small – concretely,  $\varepsilon \leq 1/12$ . If  $R_1 := U$ , then  $\text{OPT}(v, R_1)$  probes  $(u_1, v)$  and then  $(u_2, v)$  in order. On the other hand, if  $R_2 = R_1 \setminus \{v_2\}$ , then  $\text{OPT}(v, R_2)$  does *not* probe  $(u_1, v)$ . Specifically,  $\text{OPT}(v, R_2)$  probes  $(u_3, v)$  and then  $(u_4, v)$ .

**Proof of Proposition 2.6.** For each  $v \in V$ , denote  $R_v^{\text{af}}(G)$  as the unmatched (remaining) vertices of  $U$  right after  $v$  is processed (attempts its probes) in the execution on  $G$ . We emphasize that if a probe of  $v$  yields an active edge, thus matching  $v$ , then this match is excluded from  $R_v^{\text{af}}(G)$ . Similarly, define  $R_v^{\text{af}}(\tilde{G})$  in the same way for the execution on  $\tilde{G}$  (where  $v$  is now restricted to  $\tilde{V}$ ).

We first consider the case when  $G$  is rankable, and so  $F(G) = 1 - 1/e$ . Observe that since the constraints  $(\mathcal{C}_v)_{v \in V}$  are substring-closed, we can use the coupling between the two executions to inductively prove that

$$R_v^{\text{af}}(G) \subseteq R_v^{\text{af}}(\tilde{G}), \quad (\text{B.4})$$

for each  $v \in \tilde{V}$ <sup>8</sup>. Now, since  $g(1) = 1$  (by assumption), there is nothing to prove if  $\tilde{Y}_c = 1$ . Thus, we may assume that  $\tilde{Y}_c < 1$ , and as a consequence, that there exists some vertex  $v_c \in V$  which matches to  $u_0$  at time  $\tilde{Y}_c$  in the execution on  $\tilde{G}$ .

On the other hand, by assumption we know that  $u_0 \notin R_{v_c}^{\text{af}}(\tilde{G})$  and thus by (B.4), that  $u_0 \notin R_{v_c}^{\text{af}}(G)$ . As such, there exists some  $v' \in V$  which probes  $(u_0, v')$  and succeeds in matching to  $u_0$  at time  $Y_{v'} \leq \tilde{Y}_c$ . Thus, since  $g$  is monotone,

$$\alpha_{u_0} \geq \left(1 - \frac{1}{e}\right)^{-1} w_{u_0} \cdot (1 - g(Y_{v'})) \cdot \mathbf{1}_{[\tilde{Y}_c < 1]} \geq \left(1 - \frac{1}{e}\right)^{-1} w_{u_0} \cdot (1 - g(\tilde{Y}_c)),$$

and so the rankable case is complete.

We now consider the case when  $G$  is not rankable. Suppose that  $\mathcal{M}(v_0)$  is the vertex matched to  $v_0$  when the algorithm executes on  $G$ , where  $\mathcal{M}(v_0) := \emptyset$  provided no match is made. Observe then that if no match is made to  $v_0$  in this execution, then the execution proceeds identically to the execution on  $\tilde{G}$ . As a result, we get the following relation:

$$\alpha_{u_0} \geq \frac{w_{u_0}}{F} (1 - g(\tilde{Y}_c)) \cdot \mathbf{1}_{[\mathcal{M}(v_0) = \emptyset]}.$$

Now, let us condition on  $(\text{st}(e))_{e \in \tilde{E}}$  and  $(Y_v)_{v \in V}$ , and recall the definitions of  $p_{v_0} := \max_{e \in \mathcal{C}(v_0)} p_e$  and  $c_{v_0} := \max_{e \in \mathcal{C}_{v_0}} |e|$ . Observe that if every probe involving an edge of

<sup>8</sup> Example B.1 shows why (B.4) will not hold if  $G$  is not rankable.

## 13:22 Secretary Matching Meets Probing with Commitment

$\partial(v_0)$  is inactive, then  $\mathcal{M}(v_0) = \emptyset$ . On the other hand, each probe independently fails with probability at least  $(1 - p_{v_0})$ , and there are at most  $c_{v_0}$  probes made to  $\partial(v_0)$ . Thus,

$$\mathbb{P}[\mathcal{M}(v_0) = \emptyset \mid (\text{st}(e))_{e \in \tilde{E}}, (Y_v)_{v \in V}] \geq (1 - p_{v_0})^{c_{v_0}}$$

Now, since  $F(G) = (1 - 1/e) \cdot \min_{v \in V} (1 - p_v)^{c_v}$ , we get that

$$\mathbb{E}[\alpha_{u_0} \mid (Y_v)_{v \in V}, (\text{st}(e))_{e \in \tilde{E}}] \geq \left(1 - \frac{1}{e}\right)^{-1} w_{u_0} (1 - g(\tilde{Y}_c)),$$

and so the proof is complete.  $\blacktriangleleft$

**Proof of Lemma 2.5.** We first observe that by taking the appropriate conditional expectation, Proposition 2.6 ensures that

$$\mathbb{E}[\alpha_{u_0} \mid (Y_v)_{v \in \tilde{V}}, (\text{st}(e))_{e \in \tilde{E}}] \geq \left(1 - \frac{1}{e}\right)^{-1} w_{u_0} \cdot (1 - g(\tilde{Y}_c)),$$

where the right-hand side follows since  $\tilde{Y}_c$  is entirely determined from  $(Y_v)_{v \in \tilde{V}}$  and  $(\text{st}(e))_{e \in \tilde{E}}$ . Thus, combined with Proposition 2.8,

$$\mathbb{E}[p_{u_0, v_0} \cdot \alpha_{u_0} + w_{u_0} \cdot p_{u_0, v_0} \cdot \sum_{\substack{R \subseteq U: \\ u_0 \in R}} \phi_{v, R} \mid (Y_v)_{v \in \tilde{V}}, (\text{st}(e))_{e \in \tilde{E}}], \quad (\text{B.5})$$

is lower bounded by

$$\left(1 - \frac{1}{e}\right)^{-1} w_{u_0} \cdot p_{u_0, v_0} \cdot (1 - g(\tilde{Y}_c)) + \frac{w_{u_0} p_{u_0, v_0}}{F} \int_0^{\tilde{Y}_c} g(z) dz. \quad (\text{B.6})$$

However,  $g(z) := \exp(z - 1)$  for  $z \in [0, 1]$  by assumption, so

$$(1 - g(\tilde{Y}_c)) + \int_0^{\tilde{Y}_c} g(z) dz = \left(1 - \frac{1}{e}\right),$$

no matter the value of the critical time  $\tilde{Y}_c$ . Thus,

$$\left(1 - \frac{1}{e}\right)^{-1} \left( (1 - g(\tilde{Y}_c)) + \frac{1 - 1/e}{F} \int_0^{\tilde{Y}_c} g(z) dz \right) \geq 1, \quad (\text{B.7})$$

as  $F \leq 1 - 1/e$  by definition (see (2.9)). If we now lower bound (B.6) using (B.7) and take expectations over (B.5), it follows that

$$\mathbb{E}[p_{u_0, v_0} \cdot \alpha_{u_0} + w_{u_0} \cdot p_{u_0, v_0} \cdot \sum_{\substack{R \subseteq U: \\ u_0 \in R}} \phi_{v, R}] \geq w_{u_0} \cdot p_{u_0, v_0}.$$

As the vertices  $u_0 \in U$  and  $v_0 \in V$  were chosen arbitrarily, the proposed dual solution of Lemma 2.5 is feasible, and so the proof is complete.  $\blacktriangleleft$

## C Proofs and Additions to Section 3

**Proof of Theorem 1.6.** Clearly, Algorithm 2 can be implemented efficiently, since LP-config is efficiently solvable. Thus, we focus on proving the algorithm attains the desired asymptotic competitive ratio.

Let us consider the matching  $\mathcal{M}$  returned by the algorithm, as well as its weight, which we denote by  $w(\mathcal{M})$ . Set  $\alpha := 1/e$  for clarity, and take  $t \geq \lceil \alpha n \rceil$ , where we define  $R_t$  to be the *unmatched vertices* of  $U$  when vertex  $v_t$  arrives. Moreover, define  $e_t$  as the edge  $v_t$  commits to, which is the empty-set by definition if no such commitment is made. Observe that

$$\mathbb{E}[w(\mathcal{M})] = \sum_{t=\lceil \alpha n \rceil}^n \mathbb{E}[w(u_t, v_t) \cdot \mathbf{1}_{[u_t \in R_t]}]. \quad (\text{C.1})$$

Fix  $\lceil \alpha n \rceil \leq t \leq n$ , and first observe that  $w(u_t, v_t)$  and  $\{u_t \in R_t\}$  are conditionally independent given  $(V_t, v_t)$ , as the probes involving  $\partial(v_t)$  are independent from those of  $v_1, \dots, v_{t-1}$ . Thus,

$$\mathbb{E}[w(u_t, v_t) \cdot \mathbf{1}_{[u_t \in R_t]} \mid V_t, v_t] = \mathbb{E}[w(u_t, v_t) \mid V_t, v_t] \cdot \mathbb{P}[u_t \in R_t \mid V_t, v_t].$$

Moreover, Lemma 3.6 implies that

$$\mathbb{E}[w(u_t, v_t) \mid V_t, v_t] \cdot \mathbb{P}[u_t \in R_t \mid V_t, v_t] \geq \mathbb{E}[w(u_t, v_t) \mid V_t, v_t] f(t, n),$$

and so  $\mathbb{E}[w(u_t, v_t) \mathbf{1}_{[u_t \in R_t]} \mid V_t, v_t] \geq \mathbb{E}[w(u_t, v_t) \mid V_t, v_t] f(t, n)$ . Thus, by the law of iterated expectations<sup>9</sup>

$$\begin{aligned} \mathbb{E}[w(u_t, v_t) \cdot \mathbf{1}_{[u_t \in R_t]}] &= \mathbb{E}[\mathbb{E}[w(u_t, v_t) \cdot \mathbf{1}_{[u_t \in R_t]} \mid V_t, v_t]] \\ &\geq \mathbb{E}[\mathbb{E}[w(u_t, v_t) \mid V_t, v_t] f(t, n)] = f(t, n) \mathbb{E}[w(u_t, v_t)]. \end{aligned}$$

As a result, using (C.1), we get that

$$\mathbb{E}[w(\mathcal{M})] = \sum_{t=\lceil \alpha n \rceil}^n \mathbb{E}[w(u_t, v_t) \mathbf{1}_{[u_t \in R_t]}] \geq \sum_{t=\lceil \alpha n \rceil}^n f(t, n) \mathbb{E}[w(u_t, v_t)].$$

We may thus conclude that

$$\mathbb{E}[w(\mathcal{M})] \geq \text{LPOPT}_{\text{conf}}(G) \sum_{t=\lceil \alpha n \rceil}^n \frac{f(t, n)}{n},$$

after applying Lemma 3.5. As  $\sum_{t=\lceil \alpha n \rceil}^n f(t, n)/n \geq (1/e - 1/n)$ , the result holds.  $\blacktriangleleft$

<sup>9</sup>  $\mathbb{E}[w(u_t, v_t) \cdot \mathbf{1}_{[u_t \in R_t]} \mid V_t, v_t]$  is a random variable which depends on  $V_t$  and  $v_t$ , and so the outer expectation is over the randomness in  $V_t$  and  $v_t$ .



# Semi-Streaming Algorithms for Submodular Function Maximization Under $b$ -Matching Constraint

Chien-Chung Huang ✉

CNRS, DI ENS, École normale supérieure, Université PSL, France

François Sellier ✉

École polytechnique, Institut Polytechnique de Paris, France

---

## Abstract

We consider the problem of maximizing a submodular function under the  $b$ -matching constraint, in the semi-streaming model. Our main results can be summarized as follows.

- When the function is linear, *i.e.* for the maximum weight  $b$ -matching problem, we obtain a  $2 + \varepsilon$  approximation. This improves the previous best bound of  $3 + \varepsilon$  [12].
- When the function is a non-negative monotone submodular function, we obtain a  $3 + 2\sqrt{2} \approx 5.828$  approximation. This matches the currently best ratio [12].
- When the function is a non-negative non-monotone submodular function, we obtain a  $4 + 2\sqrt{3} \approx 7.464$  approximation. This ratio is also achieved in [12], but only under the simple matching constraint, while we can deal with the more general  $b$ -matching constraint.

We also consider a generalized problem, where a  $k$ -uniform hypergraph is given with an extra matroid constraint imposed on the edges, with the same goal of finding a  $b$ -matching that maximizes a submodular function. We extend our technique to this case to obtain an algorithm with an approximation of  $\frac{8}{3}k + O(1)$ .

Our algorithms build on the ideas of the recent works of Levin and Wajc [12] and of Garg, Jordan, and Svensson [9]. Our main technical innovation is to introduce a data structure and associate it with each vertex and the matroid, to record the extra information of the stored edges. After the streaming phase, these data structures guide the greedy algorithm to make better choices.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Streaming, sublinear and near linear time algorithms; Theory of computation  $\rightarrow$  Approximation algorithms analysis

**Keywords and phrases** Maximum Weight Matching, Submodular Function Maximization, Streaming, Matroid

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.14

**Category** APPROX

**Funding** This work was funded by the grants ANR-19-CE48-0016 and ANR-18-CE40-0025-01 from the French National Research Agency (ANR).

**Acknowledgements** The authors thank David Wajc and the anonymous reviewers for their helpful comments.

## 1 Introduction

Let  $G = (V, E)$  be a multi-graph (with no self-loop) where each vertex  $v \in V$  is associated with a *capacity*  $b_v \in \mathbb{Z}_+$ . A  $b$ -*matching* is a subset of edges  $M \subseteq E$  where each vertex  $v$  has at most  $b_v$  incident edges contained in  $M$ .

In the maximum weight  $b$ -matching problem, edges are given weights  $w : E \rightarrow \mathbb{R}_+$  and we need to compute a  $b$ -matching  $M$  so that  $w(M) = \sum_{e \in M} w(e)$  is maximized. A generalization of the problem is that of maximizing a non-negative *submodular function*  $f : 2^E \rightarrow \mathbb{R}_+$  under



© Chien-Chung Huang and François Sellier;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 14; pp. 14:1–14:18



Leibniz International Proceedings in Informatics

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

the  $b$ -matching constraint, namely, we look for a  $b$ -matching  $M$  so that  $f(M)$  is maximized. Here we recall the definition of a submodular function:

$$\forall X \subseteq Y \subsetneq E, \forall e \in E \setminus Y, f(X \cup \{e\}) - f(X) \geq f(Y \cup \{e\}) - f(Y).$$

Additionally  $f$  is *monotone* if  $\forall X \subseteq Y \subseteq E, f(X) \leq f(Y)$ , otherwise it is *non-monotone*. Observe that the maximum weight matching is the special case where  $f$  is a linear sum of the weights associated with the edges.

In the traditional offline setting, both problems are extensively studied. The maximum weight  $b$ -matching can be solved in polynomial time [16]; maximizing a non-negative monotone submodular function under  $b$ -matching constraint is NP-hard and the best approximation ratios so far are  $2 + \varepsilon$  and  $4 + \varepsilon$ , for the monotone and non-monotone case, respectively [8].

In this work, we consider the problem in the semi-streaming model [14]. Here the edges in  $E$  arrive over time but we have only limited space (ideally proportional to the output size) and cannot afford to store all edges in  $E$  – this rules out the possibility of applying known offline algorithms.

## 1.1 Our Contribution

We start with the maximum weight ( $b$ -)matching. For this problem, a long series of papers [4, 5, 6, 10, 12, 13, 15, 17] have proposed semi-streaming algorithms, with progressively improved approximation ratios, culminating in the work of Paz and Schwartzman [15], where  $2 + \varepsilon$  approximation is attained, for the simple matching. For the general  $b$ -matching, very recently, Levin and Wajc [12] gave a  $3 + \varepsilon$  approximation algorithm. We close the gap between the simple matching and the general  $b$ -matching.

► **Theorem 1.** *For the maximum weight  $b$ -matching problem, we obtain a  $2 + \varepsilon$  approximation algorithm using  $O(\log_{1+\varepsilon}(W/\varepsilon) \cdot |M_{max}|)$  variables in memory, and another using  $O(\log_{1+\varepsilon}(1/\varepsilon) \cdot |M_{max}| + \sum_{v \in V} b_v)$  variables, where  $M_{max}$  denotes the maximum cardinality  $b$ -matching and  $W$  denotes the maximum ratio between two non-zero weights.*

Next we consider the general case of submodular functions, for whom approximation algorithms have been proposed in [2, 3, 12, 7]. The current best ratios obtained by Levin and Wajc [12] are  $3 + 2\sqrt{2} \approx 5.828$  and  $4 + 2\sqrt{3} \approx 7.464$ , for the monotone and non-monotone functions respectively. We propose an alternative algorithm to achieve the same bounds.

► **Theorem 2.** *To maximize a non-negative submodular function under  $b$ -matching constraint, we obtain algorithms providing a  $3 + 2\sqrt{2} \approx 5.828$  approximation for monotone functions and a  $4 + 2\sqrt{3} \approx 7.464$  approximation for non-monotone functions, using  $O(\log W \cdot |M_{max}|)$  variables, where  $M_{max}$  denotes the maximum cardinality  $b$ -matching and  $W$  denotes the maximum quotient  $\frac{f(e|Y)}{f(e'|X)}$ , for  $X \subseteq Y \subseteq E$ ,  $e, e' \in E$ ,  $f(e'|X) > 0$ .*

It should be pointed out that in [12], for the case of non-monotone functions, their algorithm only works for the simple matching, and it is unclear how to generalize it to the general  $b$ -matching [11], while our algorithm lifts this restriction<sup>1</sup>. Another interesting thing to observe is that even though the achieved ratios are the same and our analysis borrows ideas from [12], our algorithm is not really just the same algorithm disguised under a different form. See Appendix B for a concrete example where the two algorithms behave differently.

<sup>1</sup> However, when the graph is bipartite, this ratio of  $4 + 2\sqrt{3} \approx 7.464$  is already obtained by Garg et al. [9], even for the general  $b$ -matching constraint.

We also consider an extension, where a matroid<sup>2</sup> is imposed on the edges. Specifically, here  $G = (V, E)$  is a  $k$ -uniform hypergraph, where each edge  $e \in E$  contains  $k$  vertices in  $V$ . In addition to the capacities  $b_v$ , a matroid  $\mathcal{M} = (E, \mathcal{I})$  is given. A  $b$ -matching  $M$  is feasible only if  $M$  is an independent set in  $\mathcal{I}$ . The objective here is to find a feasible  $b$ -matching  $M$  that maximizes  $f(M)$  for  $f$  a non-negative submodular function<sup>3</sup>.

To see our problem in a larger context, observe that it is a particular case of the  $(k + 1)$ -matchoid<sup>4</sup>. Using the current best algorithm of Chekuri et al. [3] and Feldman et al. [7], one can obtain  $4(k + 1)$  and  $2(k + 1) + 2\sqrt{(k + 1)(k + 2)} + 1$  for monotone and non-monotone functions, respectively. We obtain the following.

► **Theorem 3.** *To maximize a non-negative submodular function under the  $b$ -matching constraint along with an additional matroid constraint, we design an algorithm providing an approximation ratio bounded by  $\frac{8}{3}k + O(1)$  for both monotone and non-monotone functions, using  $O(\log W \cdot k \cdot \min\{r_{\mathcal{M}}, |M_{\max}|\})$  variables in memory, where  $M_{\max}$  denotes the maximum cardinality  $b$ -matching,  $r_{\mathcal{M}}$  is the rank of the matroid and  $W$  denotes the maximum quotient  $\frac{f(e|Y)}{f(e'|X)}$ , for  $X \subseteq Y \subseteq E$ ,  $e, e' \in E$ ,  $f(e'|X) > 0$ .*

The exact expressions for the approximation ratios are formally stated in Theorems 27 and 29. Given  $k = 2, 3$ , and  $4$ , we obtain the approximation ratios of 13.055, 15.283, and 17.325 for the monotone function. Our ratio is in general better when  $k \geq 3$  compared to the known technique of [3, 7]. When the function is non-monotone, given  $k = 2, 3$ , and  $4$ , we obtain the ratios of 14.857, 17.012, and 18.999, respectively. Our ratio is better when  $k \geq 4$  compared to the known technique of [7].

## 1.2 Our Technique

We use a local-ratio technique to choose to retain or discard a newly arrived edge during the streaming phase. After this phase, a greedy algorithm, according to the reverse edge arrival order, is then applied to add edges one by one into the solution while guaranteeing feasibility.

This is in fact the same framework used in [12, 15]. Our main technical innovation is to introduce a data structure, which takes the form of a set of queues. Such a set of queues is associated with each vertex (and with the imposed matroid). Every edge, if retained, will appear as an element<sup>5</sup> in one of these queues for each of its endpoints (and for the imposed matroid). These queues will guide the greedy algorithm to make better choices and are critical in our improvement over previous results. Here we give some intuition behind these queues. Consider the maximum weight  $b$ -matching problem. Similar to [9], we compute, for every edge, a *gain*. The sum of the gains of the retained edges can be shown to be at least half of the real weight of the unknown optimal matching. The question then boils down

<sup>2</sup> Recall that  $\mathcal{M} = (E, \mathcal{I})$  is a matroid if the following three conditions hold: (1)  $\emptyset \in \mathcal{I}$ , (2) if  $X \subseteq Y \in \mathcal{I}$ , then  $X \in \mathcal{I}$ , and (3) if  $X, Y \in \mathcal{I}$ ,  $|Y| > |X|$ , there exists an element  $e \in Y \setminus X$  so that  $X \cup \{e\} \in \mathcal{I}$ . The sets in  $\mathcal{I}$  are the *independent sets* and the *rank*  $r_{\mathcal{M}}$  of the matroid  $\mathcal{M}$  is defined as  $\max_{X \in \mathcal{I}} |X|$ .

<sup>3</sup> It is natural to ask what if the submodular function is just a linear function. We observe that in this case, our problem reduces to maximizing a weighted rank function (recall that a *weighted rank function* over a matroid  $\mathcal{M}$  is a function  $f : 2^E \rightarrow \mathbb{R}_+$  that such that for  $X \subseteq E$ ,  $f(X) = \max_{Y \subseteq X, Y \in \mathcal{I}} w(Y)$ ), a special case of a monotone submodular function, under the  $b$ -matching constraint. Our algorithm mentioned in Theorem 2 can be trivially generalized to give an approximation ratio of  $k + 1 + 2\sqrt{k}$  for this.

<sup>4</sup> Recall that a  $p$ -matchoid  $\mathcal{M}$  is a collection  $(\mathcal{M}_i = (E_i, \mathcal{I}_i))_{i=1}^s$  of matroids, each defined on some (possibly distinct) subset  $E_i \subseteq E$ , in which each element  $e \in E$  appears in at most  $p$  of the sets  $E_i$ . To see our problem is a  $(k + 1)$ -matchoid, observe that we can define a uniform matroid on each vertex to replace the capacity constraint; as each edge appears in  $k$  vertices, in total it appears in  $k + 1$  matroids.

<sup>5</sup> In this article, we will often use “edge” and “element” interchangeably.

to how to “extract” a matching whose real weight is large compared to these gains of the retained edges. In our queues, the elements are stacked in such a way that the weight of an element  $e$  is the sum of the gains of all the elements preceding  $e$  in any queue containing  $e$ . This suggests that if  $e$  is taken by the greedy algorithm, we can as well ignore all elements that are underneath  $e$  in the queues, as their gains are already “paid for” by  $e$ .

## 2 Maximum Weight $b$ -Matching

### 2.1 Description of the Algorithm

For ease of description, we explain how to achieve 2 approximation, ignoring the issue of space complexity for the moment. We will explain how a slight modification can ensure the desired space complexity, at the expense of an extra  $\varepsilon$  term in the approximation ratio (see Appendix A).

The formal algorithm for the streaming phase is shown in Algorithm 1. We give an informal description here. Let  $S$ , initially empty, be the set of edges that have been stored so far. For each vertex  $v \in V$ , a set  $Q_v = \{Q_{v,1}, \dots, Q_{v,b_v}\}$  of queues are maintained. These queues contain the edges incident to  $v$  that are stored in  $S$  and respect the arrival order of edges (newer edges are higher up in the queues). Each time a new edge  $e$  arrives, we compute its *gain*  $g(e)$  (see Lines 5 and 8). Edge  $e$  is added into  $S$  only if its gain is strictly positive. If this is the case, for each endpoint  $u$  of  $e$ , we put  $e$  in one of  $u$ 's queues (see Lines 6 and 13) and define a *reduced weight*  $w_u(e)$  (Line 11). It should be noted that  $w_u(e)$  will be exactly the sum of the gains of the edges preceding (and including)  $e$  in the queue. We refer to the last element inserted in a queue  $Q$  as the top element of that queue, denoted  $Q.top()$ . To insert an element  $e$  on top of a queue  $Q$ , we use the instruction  $Q.push(e)$ . By convention, for an empty queue  $Q$  we have  $Q.top() = \perp$ . We also set  $w_u(\perp) = 0$ . Notice that each element  $e$  also has, for each endpoint  $v \in e$ , a pointer  $r_v(e)$  to indicate its immediate predecessor in the queue of  $v$ , where it appears.

■ **Algorithm 1** Streaming phase for weighted matching.

---

```

1:  $S \leftarrow \emptyset$ 
2:  $\forall v \in V : Q_v \leftarrow (Q_{v,1} = \emptyset, \dots, Q_{v,b_v} = \emptyset)$  ▷  $b_v$  queues for a vertex  $v$ 
3: for  $e = e_t$ ,  $1 \leq t \leq |E|$  an edge from the stream do
4:   for  $u \in e$  do
5:      $w_u^*(e) \leftarrow \min\{w_u(Q_{u,q}.top()) : 1 \leq q \leq b_u\}$ 
6:      $q_u(e) \leftarrow q$  such that  $w_u(Q_{u,q}.top()) = w_u^*(e)$ 
7:   if  $w(e) > \sum_{u \in e} w_u^*(e)$  then
8:      $g(e) \leftarrow w(e) - \sum_{u \in e} w_u^*(e)$ 
9:      $S \leftarrow S \cup \{e\}$ 
10:   for  $u \in e$  do
11:      $w_u(e) \leftarrow w_u^*(e) + g(e)$ 
12:      $r_u(e) \leftarrow Q_{u,q_u(e)}.top()$  ▷  $r_u(e)$  is the element below  $e$  in the queue
13:      $Q_{u,q_u(e)}.push(e)$  ▷ add  $e$  on the top of the smallest queue

```

---

After the streaming phase, our greedy algorithm, formally described in Algorithm 2, constructs a  $b$ -matching based on the stored set  $S$ .

The greedy proceeds based on the reverse edge arrival order – but with an important modification. Once an edge  $e$  is taken as part of the  $b$ -matching, all edges preceding  $e$  that are stored in the same queue as  $e$  will be subsequently ignored by the greedy algorithm. The variables  $z_e$  are used to mark this fact.



---

**Algorithm 2** Greedy construction phase.
 

---

```

1:  $M \leftarrow \emptyset$ 
2:  $\forall e \in S : z_e \leftarrow 1$ 
3: for  $e \in S$  in reverse order do
4:   if  $z_e = 0$  then continue ▷ skip edge  $e$  if it is marked
5:    $M \leftarrow M \cup \{e\}$ 
6:   for  $u \in e$  do
7:      $c \leftarrow e$ 
8:     while  $c \neq \perp$  do
9:        $z_c \leftarrow 0$  ▷ mark elements below  $e$  in each queue
10:       $c \leftarrow r_u(c)$ 
11: return  $M$ 

```

---

## 2.2 Analysis for Maximum Weight $b$ -Matching

For analysis, for each discarded element  $e \in E \setminus S$ , we set  $g(e) = 0$  and  $w_u(e) = w_u^*(e)$  for each  $u \in e$ . The *weight* of a queue,  $w_u(Q_{u,i})$ , is defined as the reduced weight of its top element, namely,  $w_u(Q_{u,i}.top())$ . Let  $w_u(Q_u) = \sum_{i=1}^{b_u} w_u(Q_{u,i})$ . We write  $S^{(t)}$  as the value of  $S$  at the end of the iteration  $t$  of the streaming phase, and by convention  $S^{(0)} = \emptyset$ . This notation  $^{(t)}$  will also be used for other sets such as  $Q_u$  and  $Q_{u,i}$ . Through this paper,  $M^{opt}$  will always refer to the best solution for the considered problem.

The following proposition follows easily by induction.

► **Proposition 4.**

- (i) For all  $v \in V$  we have  $g(\delta(v)) = g(\delta(v) \cap S) = w_v(Q_v)$ .
- (ii) The set  $\{Q_{v,q}.top() : 1 \leq q \leq b_v\}$  contains the  $b_v$  heaviest elements of  $S \cap \delta(v)$  in terms of reduced weights.

► **Lemma 5.** At the end of Algorithm 1, for all  $b$ -matching  $M'$  and for all  $v \in V$ , we have  $w_v(Q_v) \geq w_v(M' \cap \delta(v))$ .

**Proof.** By Proposition 4(ii),  $w_v(Q_v)$  is exactly the sum of the reduced weights of the  $b_v$  heaviest elements in  $S \cap \delta(v)$  (which are on top of the queues of  $Q_v$ ). If we can show that for each element  $e = e_t \in M' \setminus S$ ,  $w_u(e_t) \leq \min\{w_v(Q_{v,q}^{[E]}) : 1 \leq q \leq b_v\}$ , the proof will follow. Indeed, as  $e_t$  is discarded, we know that  $w_v(e_t) = \min\{w_v(Q_{v,q}^{(t-1)}) : 1 \leq q \leq b_v\} \leq \min\{w_v(Q_{v,q}^{[E]}) : 1 \leq q \leq b_v\}$ , where the inequality holds because the weight of a queue is monotonically increasing. ◀

► **Lemma 6.**  $2g(S) \geq w(M^{opt})$ .

**Proof.** It is clear that for  $e = \{u, v\}$  we have  $w_u(e) + w_v(e) \geq w(e)$ . Therefore

$$\begin{aligned}
 w(M^{opt}) &\leq \sum_{e=\{u,v\} \in M^{opt}} w_u(e) + w_v(e) = \sum_{u \in V} w_u(M^{opt} \cap \delta(u)) \\
 &\leq \sum_{u \in V} w_u(Q_u) = \sum_{u \in V} g(S \cap \delta(u)) = 2g(S),
 \end{aligned}$$

where the second inequality follows from Lemma 5 and the subsequent equality from Proposition 4(i). The last equality comes from the fact that an edge is incident to 2 vertices. ◀

Recall that  $q_v(e)$  refers to the index of the particular queue in  $Q_v$  where a new edge  $e$  will be inserted (Line 6 of Algorithm 1).

► **Lemma 7.** *Algorithm 2 outputs a feasible  $b$ -matching  $M$  with weight  $w(M) \geq g(S)$ .*

**Proof.** By an easy induction, we know that for a given  $e = e_t \in S$  and  $v \in e$ , we have:

$$w_v(e) = w_v^*(e) + g(e) = \sum_{e' \in Q_{v, q_v(e)}^{(t)}} g(e') \quad \text{and} \quad w(e) = g(e) + \sum_{u \in e} \sum_{e' \in Q_{u, q_u(e)}^{(t-1)}} g(e'). \quad (1)$$

Moreover, observe that  $S \cap \delta(v)$  can be written as a disjoint union of the  $Q_{v, q}$  for  $1 \leq q \leq b_v$ :  $S \cap \delta(v) = \bigcup_{1 \leq q \leq b_v} Q_{v, q}$ . One can also observe that Algorithm 2 takes at most one element in each queue  $Q_{v, i}$ . In fact, an element can be added only if no element above it in any of the queues where it appears has already been added into  $M$ ; and no element below it in the queues can be already part of  $M$  because  $S$  is read in the reverse arrival order. Consequently  $M$  respects the capacity constraint and is thus a feasible  $b$ -matching. We now make a critical claim from which the correctness of the lemma follows easily.

▷ **Claim 8.** Given an edge  $e \in S$ , either  $e \in M$ , or there exists another edge  $e'$  arriving later than  $e$ , such that  $e' \in M$  and there exists a queue belonging to a common endpoint of  $e$  and  $e'$ , which contains both of them.

Observe that if the claim holds, by (1), the gain  $g(e)$  of any edge  $e \in S$  will be “paid” for by some edge  $e' \in M$  and the proof will follow.

To prove the claim, let  $e = \{u, v\}$  and assume that  $e \notin M$ . Consider the two queues  $Q_{u, q_u(e)}$  and  $Q_{v, q_v(e)}$ . The edges stored above  $e$  in these two queues must have arrived later than  $e$  in  $S$  and have thus already been considered by Algorithm 2. The only reason that  $e \notin M$  must be that  $z_e = 0$  when  $e$  is processed, implying that one of these edges was already part of  $M$ . Hence the claim follows. ◀

Lemmas 6 and 7 give the following theorem:

► **Theorem 9.** *Algorithms 1 and 2 provide a 2 approximation for the maximum weight  $b$ -matching problem.*

We refer the readers to Appendix A for the details on how to handle the memory consumption of the algorithm.

► **Remark 10.** It is straightforward to extend our algorithm to a  $k$ -uniform hypergraph, where we can get an approximation ratio of  $k$ . Notice that if the  $k$ -uniform hypergraph is also  $k$ -partite, then the problem becomes that of finding a maximum weight intersection of  $k$  partition matroids. It can be shown that our stored edge set is exactly identical to the one stored by the algorithm of Garg et al. [9]. They have conjectured that for  $k$  arbitrary general matroids, their stored edge set always contains a  $k$  approximation. Our result thus proves their conjecture to be true when all matroids are partition matroids.

## 3 Submodular Function Maximization

### 3.1 Description of the Algorithm

For submodular function maximization, the streaming algorithm, formally described in Algorithm 3, is quite similar to the one for the weighted  $b$ -matching in the preceding section. Here notice that the element weight  $w(e)$  is replaced by the marginal value  $f(e | S)$  (see Lines 7 and 10). We use a similar randomization method to that of Levin and Wajc [12] for non-monotone functions (adding an element to  $S$  only with probability  $p$ , see Lines 8-9), and our analysis will bear much similarity to theirs. The greedy algorithm to build a solution  $M$  from  $S$  is still Algorithm 2.

■ **Algorithm 3** Streaming phase for submodular function maximization.

---

```

1:  $S \leftarrow \emptyset$ 
2:  $\forall v \in V : Q_v \leftarrow (Q_{v,1} = \emptyset, \dots, Q_{v,b_v} = \emptyset)$ 
3: for  $e = e_t, 1 \leq t \leq |E|$  an edge from the stream do
4:   for  $u \in e$  do
5:      $w_u^*(e) \leftarrow \min\{w_u(Q_{u,q}.top()) : 1 \leq q \leq b_u\}$ 
6:      $q_u(e) \leftarrow q$  such that  $w_u(Q_{u,q}.top()) = w_u^*(e)$ 
7:   if  $f(e|S) > \alpha \sum_{u \in e} w_u^*(e)$  then
8:      $\pi \leftarrow$  a random variable equal to 1 with probability  $p$  and 0 otherwise
9:     if  $\pi = 0$  then continue ▷ skip edge  $e$  with probability  $1 - p$ 
10:     $g(e) \leftarrow f(e|S) - \sum_{u \in e} w_u^*(e)$ 
11:     $S \leftarrow S \cup \{e\}$ 
12:    for  $u \in e$  do
13:       $w_u(e) \leftarrow w_u^*(e) + g(e)$ 
14:       $r_u(e) \leftarrow Q_{u,q_u(e)}.top()$ 
15:       $Q_{u,q_u(e)}.push(e)$ 

```

---

Algorithm 3 uses, for  $\alpha = 1 + \varepsilon$ ,  $O(\log_{1+\varepsilon}(W/\varepsilon) \cdot |M_{max}|)$  variables, where  $M_{max}$  denotes the maximum cardinality  $b$ -matching and  $W$  denotes the maximum quotient  $\frac{f(e|Y)}{f(e'|X)}$ , for  $X \subseteq Y \subseteq E$ ,  $e, e' \in E$ ,  $f(e'|X) > 0$  (in Appendix A we explain how to guarantee such space complexity when  $f$  is linear – the general case of a submodular function follows similar ideas).

### 3.2 Analysis for Monotone Submodular Function Maximization

Let  $\alpha = 1 + \varepsilon$ . In this section,  $p = 1$  (so we have actually a deterministic algorithm for the monotone case). The following two lemmas relate the total gain  $g(S)$  with the marginal values  $f(S|\emptyset)$  and  $f(M^{opt}|S)$ .

► **Lemma 11.** *It holds that  $g(S) \geq \frac{\varepsilon}{1+\varepsilon} f(S|\emptyset)$ .*

**Proof.** For an element  $e = e_t \in S$  we have  $f(e|S^{(t-1)}) \geq (1 + \varepsilon) \sum_{u \in e} w_u^*(e)$  so

$$g(e) = f(e|S^{(t-1)}) - \sum_{u \in e} w_u^*(e) \geq f(e|S^{(t-1)}) \left(1 - \frac{1}{1 + \varepsilon}\right),$$

implying that

$$g(S) = \sum_{e \in S} g(e) \geq \sum_{e = e_t \in S} f(e|S^{(t-1)}) \left(1 - \frac{1}{1 + \varepsilon}\right) = \frac{\varepsilon}{1 + \varepsilon} f(S|\emptyset). \quad \blacktriangleleft$$

As in the previous section, if an edge  $e$  is discarded, we assume that  $w_v^*(e) = w_v(e)$  for each  $v \in e$ .

► **Lemma 12.** *It holds that  $2(1 + \varepsilon)g(S) \geq f(M^{opt}|S)$ .*

**Proof.** The only elements  $e = e_t$  missing in  $S$  are the ones satisfying the inequality  $f(e|S^{(t-1)}) \leq (1 + \varepsilon) \sum_{u \in e} w_u^*(e)$ . So by submodularity,

$$\begin{aligned} f(M^{opt}|S) &\leq \sum_{e \in M^{opt} \setminus S} f(e|S) \leq \sum_{e = e_t \in M^{opt} \setminus S} f(e|S^{(t-1)}) \\ &\leq \sum_{e \in M^{opt} \setminus S} (1 + \varepsilon) \sum_{u \in e} w_u^*(e) = (1 + \varepsilon) \sum_{e \in M^{opt} \setminus S} \sum_{u \in e} w_u(e) \end{aligned}$$

$$\begin{aligned}
 &= (1 + \varepsilon) \sum_{u \in V} w_u((M^{opt} \setminus S) \cap \delta(u)) \leq (1 + \varepsilon) \sum_{u \in V} w_u(Q_u) \\
 &\leq 2(1 + \varepsilon)g(S),
 \end{aligned}$$

similar to the proof of Lemma 6.  $\blacktriangleleft$

► **Lemma 13.** *Algorithm 2 outputs a feasible  $b$ -matching with  $f(M) \geq g(S) + f(\emptyset)$ .*

**Proof.** As argued in the proof of Lemma 7,  $M$  respects the capacities and so is feasible. Now, suppose that  $M = \{e_{t_1}, \dots, e_{t_{|M|}}\}$ ,  $t_1 < \dots < t_{|M|}$ . Then

$$f(M) = f(\emptyset) + \sum_{i=1}^{|M|} f(e_{t_i} | \{e_{t_1}, \dots, e_{t_{i-1}}\}) \geq f(\emptyset) + \sum_{i=1}^{|M|} f(e_{t_i} | S^{(t_i-1)}) \geq f(\emptyset) + g(S),$$

as the values  $f(e_{t_i} | S^{(t_i-1)})$  play the same role as the weights in Lemma 7.  $\blacktriangleleft$

► **Theorem 14.** *Algorithms 3 and 2 provide a  $3 + 2\sqrt{2}$  approximation if we set  $\varepsilon = \frac{1}{\sqrt{2}}$ .*

**Proof.** By Lemmas 11 and 12, we derive  $(2 + 2\varepsilon + \frac{1+\varepsilon}{\varepsilon})g(S) \geq f(M^{opt} | S) + f(S | \emptyset) = f(M^{opt} \cup S | \emptyset) \geq f(M^{opt} | \emptyset)$ , where the last inequality is due to the monotonicity of  $f$ . By Lemma 13, the output  $b$ -matching  $M$  guarantees that  $f(M) \geq g(S) + f(\emptyset)$ . As a result,  $(3 + 2\varepsilon + \frac{1}{\varepsilon})f(M) \geq f(M^{opt} | \emptyset) + f(\emptyset) = f(M^{opt})$ . Setting  $\varepsilon = \frac{1}{\sqrt{2}}$  gives the result.  $\blacktriangleleft$

► **Remark 15.** When  $b_v = 1$  for all  $v \in V$  (i.e. simple matching), our algorithm behaves exactly the same as the algorithm of Levin and Wajc [12]. Therefore their tight example also applies to our algorithm. In other words, our analysis of approximation ratio is tight.

### 3.3 Analysis for Non-Monotone Submodular Function Maximization

In this section, we suppose that  $\frac{1}{3+2\varepsilon} \leq p \leq \frac{1}{2}$ .

► **Lemma 16.** *It holds that*

$$\left(2(1 + \varepsilon) + \frac{1 + \varepsilon}{\varepsilon}\right) \mathbb{E}[g(S)] \geq \mathbb{E}[f(S \cup M^{opt} | \emptyset)].$$

**Proof.** From Lemma 11 we have that for any execution of the algorithm (a realization of randomness), the inequality  $\frac{1+\varepsilon}{\varepsilon}g(S) \geq f(S | \emptyset)$  holds, so it is also true in expectation. We will try to prove in the following that  $2(1 + \varepsilon)\mathbb{E}[g(S)] \geq \mathbb{E}[f(M^{opt} | S)]$ , which is the counterpart of Lemma 12.

First, we show that for any  $e \in M^{opt}$ :

$$(1 + \varepsilon)\mathbb{E}\left[\sum_{u \in e} w_u(e)\right] \geq \mathbb{E}[f(e | S)] \tag{2}$$

We will use a conditioning similar to the one used in [12]. Let  $e = e_t \in M^{opt}$ . We consider the event  $A_e = [f(e | S^{(t-1)}) \leq (1 + \varepsilon)\sum_{u \in e} w_u^*(e)]$ . Notice that if  $A_e$  holds,  $e$  is not part of  $S$  and  $w_v^*(e) = w_v(e)$  for each  $v \in e$ . Now by submodularity,

$$\begin{aligned}
 \mathbb{E}[f(e | S) | A_e] &\leq \mathbb{E}[f(e | S^{(t-1)}) | A_e] \leq \mathbb{E}\left[(1 + \varepsilon)\sum_{u \in e} w_u^*(e) | A_e\right] \\
 &= (1 + \varepsilon)\mathbb{E}\left[\sum_{u \in e} w_u(e) | A_e\right]
 \end{aligned}$$

Next we consider the condition  $\overline{A_e}$  (where the edge  $e$  should be added into  $S$  with probability  $p$ ). As  $p \leq \frac{1}{2}$ , and for  $e = e_t = \{u, v\}$  we have  $w_u(e) + w_v(e) = 2f(e | S^{(t-1)}) - w_u^*(e) - w_v^*(e)$  when  $e$  is added to  $S$ , we get

$$\begin{aligned} \mathbb{E} \left[ \sum_{u \in e} w_u(e) | \overline{A_e} \right] &= p \cdot \mathbb{E} \left[ 2f(e | S^{(t-1)}) - \sum_{u \in e} w_u^*(e) | \overline{A_e} \right] + (1-p) \cdot \mathbb{E} \left[ \sum_{u \in e} w_u^*(e) | \overline{A_e} \right] \\ &= 2p \cdot \mathbb{E} \left[ f(e | S^{(t-1)}) | \overline{A_e} \right] + (1-2p) \cdot \mathbb{E} \left[ \sum_{u \in e} w_u^*(e) | \overline{A_e} \right] \\ &\geq 2p \cdot \mathbb{E} \left[ f(e | S^{(t-1)}) | \overline{A_e} \right]. \end{aligned}$$

As a result, for  $p \geq \frac{1}{3+2\varepsilon}$ ,

$$\begin{aligned} (1+\varepsilon) \mathbb{E} \left[ \sum_{u \in e} w_u(e) | \overline{A_e} \right] &\geq 2p(1+\varepsilon) \cdot \mathbb{E} \left[ f(e | S^{(t-1)}) | \overline{A_e} \right] \\ &\geq (1-p) \cdot \mathbb{E} \left[ f(e | S^{(t-1)}) | \overline{A_e} \right] \\ &\geq \mathbb{E} \left[ f(e | S) | \overline{A_e} \right], \end{aligned}$$

where the last inequality holds because with probability  $p$  we have  $f(e | S) = 0$  (as  $e \in S$ ) and with probability  $1-p$ ,  $f(e | S) \leq f(e | S^{(t-1)})$  (by submodularity).

So we have proven inequality (2) and it follows that

$$\begin{aligned} \mathbb{E} [f(M^{opt} | S)] &\leq \sum_{e \in M^{opt}} \mathbb{E} [f(e | S)] \leq (1+\varepsilon) \sum_{e \in M^{opt}} \mathbb{E} \left[ \sum_{u \in e} w_u(e) \right] \\ &= (1+\varepsilon) \sum_{u \in V} \sum_{e \in M^{opt} \cap \delta(u)} \mathbb{E} [w_u(e)] \leq (1+\varepsilon) \sum_{u \in V} \mathbb{E} [w_u(Q_u)] \\ &= 2(1+\varepsilon) \mathbb{E} [g(S)], \end{aligned}$$

where in the last inequality we use the fact that Lemma 5 holds for every realization of randomness.

Now the bounds on  $\mathbb{E} [f(M^{opt} | S)]$  and the bound on  $\mathbb{E} [f(S | \emptyset)]$  argued in the beginning give the proof of the lemma.  $\blacktriangleleft$

Then we will use the following lemma, due to Buchbinder et al. [1]:

► **Lemma 17** (Lemma 2.2 in [1]). *Let  $h : 2^N \rightarrow \mathbb{R}_+$  be a non-negative submodular function, and let  $B$  be a random subset of  $N$  containing every element of  $N$  with probability at most  $p$  (not necessarily independently), then  $\mathbb{E}[h(B)] \geq (1-p)h(\emptyset)$ .*

► **Theorem 18.** *Algorithm 3 run with  $p = \frac{1}{3+2\varepsilon}$  provides a set  $S$ , upon which Algorithm 2 outputs a  $b$ -matching  $M$  satisfying:*

$$\left( \frac{4\varepsilon^2 + 8\varepsilon + 3}{2\varepsilon} \right) \mathbb{E}[f(M)] \geq f(M^{opt}).$$

*This ratio is optimized when  $\varepsilon = \frac{\sqrt{3}}{2}$ , which gives a  $4 + 2\sqrt{3}$  approximation.*

**Proof.** Combining Lemma 13 and Lemma 16,

$$\left( 2(1+\varepsilon) + \frac{1+\varepsilon}{\varepsilon} \right) \mathbb{E}[f(M)] \geq \mathbb{E}[f(S \cup M^{opt})].$$

## 14:10 Semi-Streaming Submodular Function Maximization Under $b$ -Matching Constraint

Now we can apply Lemma 17 by defining  $h : 2^E \rightarrow \mathbb{R}_+$  as, for any  $X \subseteq E$ ,  $h(X) = f(X \cup M^{opt})$  (trivially  $h$  is non-negative and submodular). As any element of  $E$  has the probability of at most  $p$  to appear in  $S$ , we derive  $\mathbb{E}[f(S \cup M^{opt})] = \mathbb{E}[h(S)] \geq (1-p)h(\emptyset) = (1-p)f(M^{opt})$ . Therefore,

$$\left(3 + 2\varepsilon + \frac{1}{\varepsilon}\right) \mathbb{E}[f(M)] \geq \mathbb{E}[f(S \cup M^{opt})] \geq (1-p)f(M^{opt}).$$

As  $p = \frac{1}{3+2\varepsilon}$ , we have

$$\left(\frac{4\varepsilon^2 + 8\varepsilon + 3}{2\varepsilon}\right) \cdot \mathbb{E}[f(M)] \geq f(M^{opt}).$$

This ratio is optimized when  $\varepsilon = \frac{\sqrt{3}}{2}$ , which gives a  $4 + 2\sqrt{3} \approx 7.464$  approximation. ◀

### 4 Matroid-constrained Maximum Submodular $b$ -Matching

In this section we consider the more general case of a  $b$ -matching on a  $k$ -uniform hypergraph and we impose a matroid constraint  $\mathcal{M} = (E, \mathcal{I})$ . A matching  $M \subseteq E$  is feasible only if it respects the capacities of the vertices and is an independent set in  $\mathcal{M}$ .

#### 4.1 Description of the Algorithm

For the streaming phase, our algorithm, formally described in Algorithm 4, is a further generalization of Algorithm 3 in the last section.

We let  $\alpha = 1 + \varepsilon$  and  $\gamma > 1$ . For the matroid  $\mathcal{M}$ , we maintain a set  $Q_{\mathcal{M}} = \{Q_{\mathcal{M},1}, \dots, Q_{\mathcal{M},r_{\mathcal{M}}}\}$ , where  $r_{\mathcal{M}}$  is the rank of  $\mathcal{M}$ , to store the elements (so if an edge  $e$  is part of  $S$ , it appears in a total of  $k+1$  queues,  $k$  of them corresponding to the vertices in  $e$ , and the remaining one corresponding to the matroid).

To facilitate the presentation, we write  $Top(Q_{\mathcal{M}})$  to denote the set of the elements on top of the queues of  $Q_{\mathcal{M}}$ . Lines 8-13 will guarantee that  $Top(Q_{\mathcal{M}})$  is an independent set (in fact a maximum weight base among all elements arrived so far, according to the reduced weights – see Lemma 20). In the end of the algorithm (Lines 26-27), we erase all elements that are not part of  $Top(Q_{\mathcal{M}})$  and let the final output  $S_f$  be simply  $Top(Q_{\mathcal{M}})$ .  $S_f$  is then fed into the greedy, Algorithm 2, to produce the  $b$ -matching. The pointers  $r_v$  are updated (Line 27), so that the queues could be regarded as if they contained only elements of  $S_f$ .

Here we give some intuition. We retain only the elements  $Top(Q_{\mathcal{M}})$  because they are independent (hence any subset of them chosen by the Greedy algorithm), releasing us from the worry that the output is not independent. We set  $\gamma > 1$  to ensure that the gain of new edges in the same queue of  $Q_{\mathcal{M}}$  grows quickly. By doing this,  $Top(Q_{\mathcal{M}})$ , by itself, contributes to a significant fraction of all gains in  $g(S)$  (see Lemma 24). However, an overly large  $\gamma$  causes us to throw away too many edges (see Line 14), thus hurting the final approximation ratio. To optimize, we thus need to choose  $\gamma$  carefully.

The number of variables used by this algorithm is  $O(\min\{k \cdot \log_{\gamma \cdot (1+\varepsilon)}(W/\varepsilon) \cdot r_{\mathcal{M}}, k \cdot \log_{1+\varepsilon}(W/\varepsilon) \cdot |M_{max}|\})$ , where  $M_{max}$  denotes the maximum-cardinality  $b$ -matching  $W$  denotes the maximum quotient  $\frac{f(e|Y)}{f(e'|X)}$ , for  $X \subseteq Y \subseteq E$ ,  $e, e' \in E$ ,  $f(e'|X) > 0$ .

#### 4.2 Analysis for Monotone Submodular Function Maximization

In this section,  $p = 1$ . For each discarded elements  $e \in E \setminus S$ , similarly as before, we set  $w_{\mathcal{M}}(e) = w_{\mathcal{M}}^*(e)$ .

■ **Algorithm 4** Streaming phase for Matroid-constrained Maximum Submodular  $b$ -Matching.

---

```

1:  $S \leftarrow \emptyset$ 
2:  $Q_{\mathcal{M}} \leftarrow (Q_{\mathcal{M},1} = \emptyset, \dots, Q_{\mathcal{M},r_{\mathcal{M}}} = \emptyset)$ 
3:  $\forall v \in V : Q_v \leftarrow (Q_{v,1} = \emptyset, \dots, Q_{v,b_v} = \emptyset)$ 
4: for  $e = e_t, 1 \leq t \leq |E|$  an edge from the stream do
5:   for  $u \in e$  do
6:      $w_u^*(e) \leftarrow \min\{w_u(Q_{u,q}.top()) : 1 \leq q \leq b_u\}$ 
7:      $q_u(e) \leftarrow q$  such that  $w_u(Q_{u,q}.top()) = w_u^*(e)$ 
8:   if  $Top(Q_{\mathcal{M}}) \cup \{e\} \in \mathcal{I}$  then
9:      $w_{\mathcal{M}}^*(e) \leftarrow 0$ 
10:     $q_{\mathcal{M}}(e) \leftarrow q$  such that  $Q_{\mathcal{M},q}$  is empty
11:   if  $Top(Q_{\mathcal{M}}) \cup \{e\}$  contains a circuit  $C$  then
12:      $w_{\mathcal{M}}^*(e) \leftarrow \min_{e' \in C \setminus \{e\}} w_{\mathcal{M}}(e')$ 
13:      $q_{\mathcal{M}}(e) \leftarrow q$  such that  $w_{\mathcal{M}}(Q_{\mathcal{M},q}.top())$  is equal to  $\min_{e' \in C \setminus \{e\}} w_{\mathcal{M}}(e')$  and
14:      $Q_{\mathcal{M},q}.top() \in C$ 
15:   if  $f(e|S) > \alpha(\sum_{u \in e} w_u^*(e) + \gamma \cdot w_{\mathcal{M}}^*(e))$  then
16:      $\pi \leftarrow$  a random variable equal to 1 with probability  $p$  and 0 otherwise
17:     if  $\pi = 0$  then continue ▷ skip edge  $e$  with probability  $1 - p$ 
18:      $g(e) \leftarrow f(e|S) - \sum_{u \in e} w_u^*(e) - w_{\mathcal{M}}^*(e)$ 
19:      $S \leftarrow S \cup \{e\}$ 
20:     for  $u \in e$  do
21:        $w_u(e) \leftarrow w_u^*(e) + g(e)$ 
22:        $r_u(e) \leftarrow Q_{u,q_u(e)}.top()$ 
23:        $Q_{u,q_u(e)}.push(e)$ 
24:      $w_{\mathcal{M}}(e) \leftarrow w_{\mathcal{M}}^*(e) + g(e)$ 
25:      $r_{\mathcal{M}}(e) \leftarrow Q_{\mathcal{M},q_{\mathcal{M}}(e)}.top()$ 
26:      $Q_{\mathcal{M},q_{\mathcal{M}}(e)}.push(e)$ 
27:  $S_f \leftarrow Top(Q_{\mathcal{M}})$ 
28: update the values of  $r_v$  for  $v \in V$  as necessary so that only the elements in  $S_f$  are
29: considered as present in the queues (elements not in  $S_f$  are skipped in the sequences of
30: recursive values of  $r_v$ )

```

---

We introduce some basic facts in matroid theory, e.g., see [16].

► **Proposition 19.** *Given a matroid  $\mathcal{M} = (E, \mathcal{I})$  with weight  $w : E \rightarrow \mathbb{R}_+$ , then*

- (i) *An independent set  $I \in \mathcal{I}$  is a maximum weight base if and only if, for every element  $e \in E \setminus I$ ,  $I \cup \{e\}$  contains a circuit and  $w(e) \leq \min_{e' \in C \setminus \{e\}} w(e')$ .*
- (ii) *If  $I \in \mathcal{I}$ ,  $I \cup \{e\}$  contains a circuit  $C_1$  and  $I \cup \{e'\}$  contains a circuit  $C_2$  and  $C_1$  and  $C_2$  contain a common element  $e'' \in I$ , then there exists another circuit  $C_3 \subseteq (C_1 \cup C_2) \setminus \{e''\}$ .*

► **Lemma 20.** *Let  $\{e_1, \dots, e_t\}$  be the set of edges arrived so far. Then  $Top(Q_{\mathcal{M}}) = Top(Q_{\mathcal{M}}^{(t)})$  forms a maximum weight base in  $\{e_1, \dots, e_t\}$  with regard to the reduced weight  $w_{\mathcal{M}}$ .*

**Proof.** This can be easily proved by induction on the number of edges arrived so far and Proposition 19. ◀

► **Corollary 21.** *At the end of the algorithm,  $w_{\mathcal{M}}(Q_{\mathcal{M}}) \geq w_{\mathcal{M}}(M^{opt})$ .*

## 14:12 Semi-Streaming Submodular Function Maximization Under $b$ -Matching Constraint

The next two lemmas relate the total gain  $g(S)$  with  $f(S | \emptyset)$  and  $f(M^{opt} | S)$ .

► **Lemma 22.** *It holds that  $g(S) \geq \frac{\varepsilon}{1+\varepsilon} f(S | \emptyset)$ .*

**Proof.** Same proof as for Lemma 11. ◀

► **Lemma 23.** *It holds that  $(1 + \varepsilon)(k + \gamma)g(S) \geq f(M^{opt} | S)$ .*

**Proof.** By the same argument as in the proof of Lemma 12, we have

$$\sum_{e \in M^{opt} \setminus S} (1 + \varepsilon) \sum_{u \in e} w_u^*(e) \leq (1 + \varepsilon)k \cdot g(S).$$

Moreover, Corollary 21 shows that  $g(S) = w_{\mathcal{M}}(Q_{\mathcal{M}}) \geq w_{\mathcal{M}}(M^{opt})$  and we know that

$$w_{\mathcal{M}}(M^{opt}) \geq \sum_{e \in M^{opt} \setminus S} w_{\mathcal{M}}(e) = \sum_{e \in M^{opt} \setminus S} w_{\mathcal{M}}^*(e).$$

As a result, we obtain

$$\begin{aligned} f(M^{opt} | S) &\leq \sum_{e \in M^{opt} \setminus S} f(e | S) \leq \sum_{e=e_t \in M^{opt} \setminus S} f(e | S^{(t-1)}) \\ &\leq (1 + \varepsilon) \sum_{e \in M^{opt} \setminus S} \sum_{u \in e} w_u^*(e) + \gamma \cdot w_{\mathcal{M}}^*(e) \\ &\leq (1 + \varepsilon)(k + \gamma)g(S). \end{aligned} \quad \blacktriangleleft$$

The following lemma states that  $S_f$  retains a reasonably large fraction of the gains compared to  $S$ .

► **Lemma 24.** *It holds that  $\left(1 + \frac{1}{\gamma \cdot (1 + \varepsilon) - 1}\right) g(S_f) \geq g(S)$ .*

**Proof.** We have, for all element  $e = e_t \in S_f$ ,

$$\begin{aligned} g(e) &= f(e | S^{(t-1)}) - \sum_{u \in e} w_u^*(e) - w_{\mathcal{M}}^*(e) \\ &\geq (1 + \varepsilon - 1) \sum_{u \in e} w_u^*(e) + (\gamma \cdot (1 + \varepsilon) - 1)w_{\mathcal{M}}^*(e) \\ &\geq (\gamma \cdot (1 + \varepsilon) - 1)w_{\mathcal{M}}^*(e) = (\gamma \cdot (1 + \varepsilon) - 1) \sum_{e' \in Q_{\mathcal{M}, q_{\mathcal{M}}(e)}^{(t-1)}} g(e'). \end{aligned}$$

Recalling that  $q_{\mathcal{M}}(e)$  is the index of the queues in  $Q_{\mathcal{M}}$  where  $e$  is put (see Lines 13 and 25 of Algorithm 4),

$$g(S) = \sum_{e=e_t \in S_f} \left( g(e) + \sum_{e' \in Q_{\mathcal{M}, q_{\mathcal{M}}(e)}^{(t-1)}} g(e') \right) \leq \sum_{e \in S_f} \left( 1 + \frac{1}{\gamma \cdot (1 + \varepsilon) - 1} \right) g(e),$$

and the proof follows. ◀

► **Lemma 25.** *It holds that  $\left(1 + \frac{1}{\gamma \cdot (1 + \varepsilon) - 1}\right) \left((1 + \varepsilon)(k + \gamma) + 1 + \frac{1}{\varepsilon}\right) g(S_f) \geq f(M^{opt} | \emptyset)$ .*

**Proof.** By Lemmas 22 and 23, we have that  $\left((1 + \varepsilon)(k + \gamma) + 1 + \frac{1}{\varepsilon}\right) g(S) \geq f(M^{opt} | S) + f(S | \emptyset) = f(M^{opt} \cup S | \emptyset) \geq f(M^{opt} | \emptyset)$  because  $f$  is monotone. Then we use Lemma 24. ◀



► **Lemma 26.** *With  $S_f$  as input, Algorithm 2 returns a feasible  $b$ -matching  $M$  with  $f(M) \geq g(S_f) + f(\emptyset)$ .*

**Proof.** As argued in Lemma 7,  $M$  respects the capacities. Furthermore, as  $S_f$  is by construction an independent set in  $\mathcal{M}$  and  $M \subseteq S_f \in \mathcal{I}$ , we have  $M \in \mathcal{I}$ . So  $M$  is a feasible  $b$ -matching. Finally, using an analysis similar to the one in the proof of Lemma 13, we have  $f(M) \geq g(S_f) + f(\emptyset)$  (the only difference being that now the “weight”  $f(e_t | S^{(t-1)})$  of an element can be larger than the sum of the gains of the elements below it in the queues, which is not an issue for the analysis). ◀

As a result, we get the following theorem (the same way we obtained Theorem 14):

► **Theorem 27.** *For non-negative monotone submodular functions, Algorithm 4 with  $p = 1$  combined with Algorithm 2 provides a feasible  $b$ -matching such that*

$$\left(1 + \frac{1}{\gamma \cdot (1 + \varepsilon) - 1}\right) \left((1 + \varepsilon)(k + \gamma) + 1 + \frac{1}{\varepsilon}\right) f(M) \geq f(M^{opt}).$$

By setting  $\varepsilon = 1$  and  $\gamma = 2$ , we attain the approximation ratio of  $\frac{4}{3}(2k + 6)$  for all  $k$ .

It is possible to obtain better ratios for a fixed  $k$  by a more careful choice of the parameters  $\varepsilon$  and  $\gamma$ . For instance when  $k = 2, 3$ , and  $4$ , we have the respective ratios of 13.055, 15.283, and 17.325.

### 4.3 Analysis for Non-Monotone Submodular Function Maximization

In this section, we assume  $\frac{1}{1+(k+\gamma)(1+\varepsilon)} \leq p \leq \frac{1}{k+\gamma}$ . The following lemma is the counterpart of Lemma 16, whose proof again uses the technique of conditioning (in a more general form).

► **Lemma 28.** *It holds that  $((1 + \varepsilon)(k + \gamma) + \frac{1+\varepsilon}{\varepsilon}) \mathbb{E}[g(S)] \geq \mathbb{E}[f(S \cup M^{opt} | \emptyset)]$ .*

**Proof.** By Lemma 22, for any realization on randomness, we have  $\frac{1+\varepsilon}{\varepsilon} g(S) \geq f(S | \emptyset)$ , so the inequality also holds in expectation.

We next show that, for any  $e \in M^{opt}$  we have

$$\mathbb{E}[f(e | S)] \leq (1 + \varepsilon) \mathbb{E} \left[ \sum_{u \in e} w_u(e) + \gamma \cdot w_{\mathcal{M}}(e) \right]. \quad (3)$$

Let  $e \in M^{opt}$ . Conditioning on  $A_e = [f(e | S^{(t-1)}) \leq (1 + \varepsilon)(\sum_{u \in e} w_u^*(e) + \gamma \cdot w_{\mathcal{M}}^*(e))]$  (i.e.  $e$  cannot be part of  $S$ ), we have

$$\begin{aligned} \mathbb{E}[f(e | S) | A_e] &\leq \mathbb{E}[f(e | S^{(t-1)}) | A_e] \\ &\leq \mathbb{E} \left[ (1 + \varepsilon) \left( \sum_{u \in e} w_u^*(e) + \gamma \cdot w_{\mathcal{M}}^*(e) \right) | A_e \right] \\ &= (1 + \varepsilon) \mathbb{E} \left[ \sum_{u \in e} w_u(e) + \gamma \cdot w_{\mathcal{M}}(e) | A_e \right]. \end{aligned}$$

For the condition  $\overline{A_e}$ , recall that it means that with probability  $p$ , the edge is added into  $S$  and with probability  $1 - p$ , it is not. So

$$\mathbb{E} \left[ \sum_{u \in e} w_u(e) + \gamma \cdot w_{\mathcal{M}}(e) | \overline{A_e} \right]$$

## 14:14 Semi-Streaming Submodular Function Maximization Under $b$ -Matching Constraint

$$\begin{aligned}
&= p \cdot \mathbb{E} \left[ (k + \gamma)f(e | S^{(t-1)}) - (k + \gamma - 1) \left( \sum_{u \in e} w_u^*(e) \right) - kw_{\mathcal{M}}^*(e) \mid \overline{A_e} \right] \\
&\quad + (1 - p) \cdot \mathbb{E} \left[ \sum_{u \in e} w_u^*(e) + \gamma \cdot w_{\mathcal{M}}^*(e) \mid \overline{A_e} \right] \\
&\geq p \cdot \mathbb{E} \left[ (k + \gamma)f(e | S^{(t-1)}) - (k + \gamma - 1) \left( \sum_{u \in e} w_u^*(e) \right) - (k + \gamma - 1)\gamma \cdot w_{\mathcal{M}}^*(e) \mid \overline{A_e} \right] \\
&\quad + (1 - p) \cdot \mathbb{E} \left[ \sum_{u \in e} w_u^*(e) + \gamma \cdot w_{\mathcal{M}}^*(e) \mid \overline{A_e} \right] \\
&= (k + \gamma) \cdot p \cdot \mathbb{E} \left[ f(e | S^{(t-1)}) \mid \overline{A_e} \right] + (1 - (k + \gamma) \cdot p) \cdot \mathbb{E} \left[ \sum_{u \in e} w_u^*(e) + \gamma \cdot w_{\mathcal{M}}^*(e) \mid \overline{A_e} \right] \\
&\geq (k + \gamma) \cdot p \cdot \mathbb{E} \left[ f(e | S^{(t-1)}) \mid \overline{A_e} \right],
\end{aligned}$$

where in the second step we use the fact that  $\gamma > 1$  and in the last inequality that  $p \leq \frac{1}{k+\gamma}$ . Now as  $p \geq \frac{1}{1+(k+\gamma)(1+\varepsilon)}$ , we have

$$\begin{aligned}
(1 + \varepsilon)\mathbb{E} \left[ \sum_{u \in e} w_u(e) + \gamma \cdot w_{\mathcal{M}}(e) \mid \overline{A_e} \right] &\geq (1 + \varepsilon)(k + \gamma) \cdot p \cdot \mathbb{E} \left[ f(e | S^{(t-1)}) \mid \overline{A_e} \right] \\
&\geq (1 - p) \cdot \mathbb{E} \left[ f(e | S^{(t-1)}) \mid \overline{A_e} \right] \\
&\geq \mathbb{E} \left[ f(e | S) \mid \overline{A_e} \right],
\end{aligned}$$

and we have established (3).

Similar to the proof of Lemma 16 we get

$$\begin{aligned}
\mathbb{E} [f(M^{opt} | S)] &\leq \sum_{e \in M^{opt}} \mathbb{E} [f(e | S)] \\
&\leq (1 + \varepsilon) \sum_{e \in M^{opt}} \mathbb{E} \left[ \sum_{u \in e} w_u(e) \right] + (1 + \varepsilon)\gamma \cdot \mathbb{E}[w_{\mathcal{M}}(M^{opt})] \\
&\leq k(1 + \varepsilon)\mathbb{E}[g(S)] + (1 + \varepsilon)\gamma \cdot \mathbb{E}[w_{\mathcal{M}}(M^{opt})].
\end{aligned}$$

By Lemma 21 we know that for any realization of randomness,  $w_{\mathcal{M}}(M^{opt}) \leq w_{\mathcal{M}}(S_f) = g(S)$ . Thus we get  $\mathbb{E} [f(M^{opt} | S)] \leq (k + \gamma)(1 + \varepsilon)\mathbb{E}[g(S)]$ .

Now the bound on  $\mathbb{E} [f(M^{opt} | S)]$  and the bound on  $\mathbb{E} [f(S | \emptyset)]$  (argued in the beginning of the proof) give us the lemma.  $\blacktriangleleft$

Finally, using Lemma 17 we obtain:

► **Theorem 29.** *For non-negative submodular functions, Algorithm 4 with  $p = \frac{1}{1+(k+\gamma)(1+\varepsilon)}$  combined with Algorithm 2 provides a  $b$ -matching  $M$  independent in  $\mathcal{M}$  such that:*

$$\frac{1 + (k + \gamma)(1 + \varepsilon)}{(k + \gamma)(1 + \varepsilon)} \left( 1 + \frac{1}{\gamma \cdot (1 + \varepsilon) - 1} \right) \left( (1 + \varepsilon)(k + \gamma) + 1 + \frac{1}{\varepsilon} \right) \mathbb{E}[f(M)] \geq f(M^{opt}).$$

Setting  $\varepsilon = 1$  and  $\gamma = 2$  we obtain the ratio of  $\frac{2k+5}{2k+4} \cdot \frac{4}{3} \cdot (2k + 6)$  for all  $k$ .

As in the last section, it is possible to obtain better ratios for a fixed  $k$  by a more careful choice of the parameters  $\varepsilon$  and  $\gamma$ . For instance when  $k = 2, 3$ , and  $4$ , we have the respective ratios of 14.857, 17.012, and 18.999.

## References

- 1 Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *Proc. 25th SODA*, pages 1433–1452, 2014.
- 2 Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: matchings, matroids, and more. *Math. Program.*, 154(1-2):225–247, 2015.
- 3 Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. Streaming algorithms for submodular function maximization. In *Proc. 42nd ICALP*, pages 318–330, 2015.
- 4 Michael Crouch and D.M. Stubbs. Improved streaming algorithms for weighted matching, via unweighted matching. *Leibniz International Proceedings in Informatics, LIPIcs*, 28:96–104, September 2014. doi:10.4230/LIPIcs.APPROX-RANDOM.2014.96.
- 5 Leah Epstein, Asaf Levin, Julián Mestre, and Danny Segev. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM Journal on Discrete Mathematics*, 25, July 2009. doi:10.4230/LIPIcs.STACS.2010.2476.
- 6 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348:207–216, December 2005. doi:10.1016/j.tcs.2005.09.013.
- 7 Moran Feldman, Amin Karbasi, and Ehsan Kazemi. Do less, get more: Streaming submodular maximization with subsampling. In *NeurIPS*, pages 730–740, 2018.
- 8 Moran Feldman, Joseph (Seffi) Naor, Roy Schwartz, and Justin Ward. Improved approximations for k-exchange systems. In *Proc. 19th ESA*, pages 784–798, 2011. URL: <http://portal.acm.org/citation.cfm?id=2040572.2040658&coll=DL&dl=GUIDE&CFID=95852163&CFTOKEN=76709828>.
- 9 Paritosh Garg, Linus Jordan, and Ola Svensson. Semi-streaming algorithms for submodular matroid intersection. In *IPCO*, 2021.
- 10 Mohsen Ghaffari and David Wajc. Space-optimal semi-streaming for  $(2 + \varepsilon)$ -approximate matching, 2019.
- 11 Roie Levin and David Wajc, 2021. private communication.
- 12 Roie Levin and David Wajc. Streaming submodular matching meets the primal-dual method. In *SODA*, pages 1914–1933, 2021.
- 13 Andrew McGregor. Finding graph matchings in data streams. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques. APPROX 2005, RANDOM 2005.*, pages 170–181, 2005. doi:10.1007/11538462\_15.
- 14 S. Muthukrishnan. *Data Streams: Algorithms and Applications*. Now Foundations and Trends, 2005. doi:10.1561/9781933019604.
- 15 Ami Paz and Gregory Schwartzman. A  $(2 + \varepsilon)$ -approximation for maximum weight matching in the semi-streaming model. In *Proceedings of the 2017 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2153–2161, 2017. doi:10.1137/1.9781611974782.140.
- 16 Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, 2003.
- 17 Mariano Zelke. Weighted matching in the semi-streaming model. *Algorithmica*, 62:1–20, February 2008. doi:10.1007/s00453-010-9438-5.

## A Making Algorithm 1 Memory-Efficient

We explain how to guarantee the space requirement promised in Theorem 1. In this section,  $w_{\min}$  denotes the minimum non-zero value of the weight of an edge, and  $w_{\max}$  the maximum weight of an edge. Moreover, we set  $W = w_{\max}/w_{\min}$ . We also define  $M_{\max}$  as a given maximum cardinality  $b$ -matching.

Let  $\alpha = (1 + \varepsilon) > 1$ . In Algorithm 5 we add an edge  $e$  to  $S$  only if  $w(e) > \alpha \sum_{u \in e} w_u^*(e)$  (Line 7). For the moment we set  $d = 0$  in our analysis, ignoring Lines 14-16 of the algorithm.

► **Lemma 30.** *The set  $S$  obtained at the end of algorithm 5 when  $d = 0$  guarantees that  $2\alpha g(S) \geq w(M^{\text{opt}})$ .*

■ **Algorithm 5** Streaming phase for weighted matching, memory-efficient.

---

```

1:  $S \leftarrow \emptyset$ 
2:  $\forall v \in V : Q_v \leftarrow (Q_{v,1} = \emptyset, \dots, Q_{v,b_v} = \emptyset)$ 
3: for  $e = e_t, 1 \leq t \leq |E|$  an edge from the stream do
4:   for  $u \in e$  do
5:      $w_u^*(e) \leftarrow \min\{w_u(Q_{u,q}.top()) : 1 \leq q \leq b_u\}$ 
6:      $q_u(e) \leftarrow q$  such that  $w_u(Q_{u,q}.top()) = w_u^*(e)$ 
7:   if  $w(e) > \alpha \sum_{u \in e} w_u^*(e)$  then ▷ stricter condition here
8:      $g(e) \leftarrow w(e) - \sum_{u \in e} w_u^*(e)$ 
9:      $S \leftarrow S \cup \{e\}$ 
10:   for  $u \in e$  do
11:      $w_u(e) \leftarrow w_u^*(e) + g(e)$ 
12:      $r_u(e) \leftarrow Q_{u,q_u(e)}.top()$ 
13:      $Q_{u,q_u(e)}.push(e)$ 
14:   if  $d = 1$  and  $Q_{u,q_u(e)}.length() > \beta$  then ▷ remove some small element
15:     let  $e'$  be the  $\beta + 1$ -th element from the top of  $Q_{u,q_u(e)}$ 
16:     mark  $e'$  as erasable, so that when it will no longer be on the top of any
    queue, it will be removed from  $S$  and from all the queues it appears in

```

---

**Proof.** We proceed as in [9]. Let  $w_\alpha : E \rightarrow \mathbb{R}$  such that  $w_\alpha(e) = w(e)$  for  $e \in S$  and  $w_\alpha(e) = \frac{w(e)}{\alpha}$  for  $e \in E \setminus S$ . We can observe that with the weights  $w_\alpha$ , Algorithm 1 gives the same set  $S$  as Algorithm 5 with the weights  $w$ . We deduce that  $w_\alpha(M^{opt}) \leq 2g(S)$  and then, as  $w \leq \alpha w_\alpha$ , we get  $2\alpha g(S) \geq w(M^{opt})$ . ◀

Hence, using same arguments as in [10, 12] we obtain the following:

► **Theorem 31.** *Algorithm 5 (with  $d = 0$ ) combined with Algorithm 2 gives a  $2 + \varepsilon$  approximation algorithm by using  $O(\log_{1+\varepsilon}(W/\varepsilon) \cdot |M_{max}|)$  variables.*

**Proof.** In a given queue, the minimum non-zero value that can be attained is  $\frac{\varepsilon}{1+\varepsilon} w_{min}$  and the maximum value that can be attained is  $w_{max}$ . As the value of the top element of the queue increases at least by a factor  $1 + \varepsilon$  for each inserted element, a given queue contains at most  $\log_{1+\varepsilon}(W/\varepsilon) + 1$  edges. Hence, a vertex  $v \in V$  contains at most  $\min\{|\delta(v)|, b_v \cdot (\log_{1+\varepsilon}(W/\varepsilon) + 1)\}$  elements of  $S$  at the end of the algorithm.

Then let  $U \subseteq V$  be the set of *saturated* vertices of  $V$  by  $M_{max}$ , *i.e.* the set of vertices  $v \in V$  such that  $|\delta(v) \cap M_{max}| = \min\{|\delta(v)|, b_v\}$ . By construction,  $U$  is a vertex cover and  $\sum_{v \in U} \min\{|\delta(v)|, b_v\} \leq 2|M_{max}|$ . As all edges of  $S$  have at least one endpoint in  $U$ , we get:

$$\begin{aligned}
|S| &\leq \sum_{v \in U} |\delta(v) \cap S| \leq \sum_{v \in U} \min\{|\delta(v)|, b_v \cdot (\log_{1+\varepsilon}(W/\varepsilon) + 1)\} \\
&\leq \sum_{v \in U} \min\{|\delta(v)|, b_v\} \cdot (\log_{1+\varepsilon}(W/\varepsilon) + 1) \leq 2(\log_{1+\varepsilon}(W/\varepsilon) + 1) \cdot |M_{max}|,
\end{aligned}$$

so the memory consumption of the algorithm is  $O(\log_{1+\varepsilon}(W/\varepsilon) \cdot |M_{max}|)$  ◀

Modifying an idea from Ghaffari and Wajc [10] we can further improve the memory consumption of the algorithm, especially if  $W$  is not bounded. For that, set  $\beta = 1 + \frac{2 \log(1/\varepsilon)}{\log(1+\varepsilon)} = 1 + \log_{1+\varepsilon}(1/\varepsilon^2)$  in Algorithm 5 as the maximum size of a queue, and set  $d = 1$  (so that we now consider the whole code).

Consider an endpoint  $u$  of the newly-inserted edge  $e$ . If the queue  $Q_{u,q_u(e)}$  becomes too long (more than  $\beta$  elements), it means the gain  $g(e')$  of the  $\beta + 1$ -th element from the top of the queue (we will call that element  $e'$ ) is very small compared to  $g(e)$ , so we then can “potentially” remove  $e'$  from  $S$  and from the queues without hurting too much  $g(S)$ . In the code, we will mark this edge  $e'$  as *erasable*, so that when  $e'$  will no longer be on top of any queue, it will be removed from  $S$  and all the queues it appears in. To be able to do these eviction operations, the queues have to be implemented with doubly linked lists.

If an edge  $e = \{u, v\}$  is marked as erasable by Algorithm 5 ( $d = 1$ ) because an edge  $e' = \{u, v'\}$  is added to  $S$ , then we say that  $e'$  *evicted*  $e$  (and that  $e$  was *evicted* by  $e'$ ).

► **Lemma 32.** *If  $e = \{u, v\}$  is evicted by  $e' = \{u, v'\}$ , then  $g(e') \geq g(e)/\varepsilon$ .*

**Proof.** We have  $g(e') \geq \varepsilon(w_u^*(e') + w_{v'}^*(e')) \geq \varepsilon w_u^*(e') \geq \varepsilon(1 + \varepsilon)^{\beta-1}g(e) \geq g(e)/\varepsilon$  because after  $e = e_t$  is added to  $Q_{u,q_u(e)}$  we have  $w_u(Q_{u,q_u(e)}^{(t)}) \geq g(e)$  and each time an element is added to  $Q_{u,q_u(e)}$  the value  $w_u(Q_{u,q_u(e)})$  is multiplied at least by  $(1 + \varepsilon)$ . ◀

► **Theorem 33.** *For  $\varepsilon \leq \frac{1}{4}$ , Algorithm 5 with  $d = 1$  combined with Algorithm 2 gives a  $2 + \varepsilon$  approximation algorithm by using  $O(\log_{1+\varepsilon}(1/\varepsilon) \cdot |M_{max}| + \sum_{v \in V} b_v)$  variables.*

**Proof.** For an element  $e$  that was not evicted from  $S$  in Algorithm 5, denote by  $\mathcal{E}_e$  the elements that were evicted by  $e$  directly or indirectly (in a chain of evictions). This set  $\mathcal{E}_e$  contains at most 2 elements that were directly evicted when  $e$  was inserted in  $S$ , and their associated gain is at most  $\varepsilon g(e)$  for each, and at most 4 elements indirectly evicted by  $e$  when these 2 evicted elements were inserted in  $S$ , and their associated gain is at most equal to  $\varepsilon^2 g(e)$  for each, and so on. Then, as  $\varepsilon \leq \frac{1}{4}$ ,

$$\sum_{e' \in \mathcal{E}_e} g(e') \leq \sum_{i=1}^{\infty} (2\varepsilon)^i g(e) \leq 2\varepsilon g(e) \sum_{i=0}^{\infty} (1/2)^i = 4\varepsilon g(e)$$

Therefore, if  $S_0$  denotes the set  $S$  obtained by Algorithm 5 when  $d = 0$  and  $S_1$  denotes the set  $S$  obtained by Algorithm 5 when  $d = 1$ , we get:

$$g(S_0) - g(S_1) \leq 4\varepsilon g(S_1)$$

because the elements inserted in  $S$  are exactly the same for the two algorithms, the only difference being that some elements are missing in  $S_1$  (but these elements were removed when they no longer had any influence on the values of  $w^*$  and thereby no influence on the choice of the elements inserted in  $S$  afterwards). We have then:

$$w(M^{opt}) \leq 2(1 + \varepsilon)g(S_0) \leq 2(1 + \varepsilon)(1 + 4\varepsilon)g(S_1) \leq 2(1 + 6\varepsilon)g(S_1)$$

and Algorithm 2 will provide a  $2(1 + 6\varepsilon)$  approximation of the optimal  $b$ -matching. In fact, the analysis of Algorithm 2 with  $S_1$  as input is almost the same as in the proof of Lemma 7, the only difference being that now the weight of an element is no longer necessarily equal to the sum of the gains of elements below it but can be higher (which is not an issue).

Regarding the memory consumption of the algorithm, one can notice that, using the same notation  $U$  for the set of the elements saturated by  $M_{max}$  as previously, and because there are only up to  $\sum_{v \in V} b_v$  elements on top of the queues that cannot be deleted (and thus these edges are the ones making some queues in  $U$  exceed the limit  $\beta$ ), we obtain:

$$|S| \leq \sum_{v \in V} b_v + \sum_{v \in U} \beta \cdot \min\{|\delta(v)|, b_v\} \leq \sum_{v \in V} b_v + 2\beta \cdot |M_{max}|,$$

and therefore the algorithm uses  $O(\sum_{v \in V} b_v + \log_{1+\varepsilon}(1/\varepsilon) \cdot |M_{max}|)$  variables. ◀

## 14:18 Semi-Streaming Submodular Function Maximization Under $b$ -Matching Constraint

► Remark 34. Ideas presented here for the maximum weight  $b$ -matching can be easily extended to submodular function maximization and for hypergraphs under a matroid constraint, namely for the algorithms presented in Sections 3 and 4.

### **B** Example of Different Behavior Compared to [12]

Here is an example to show the difference of behavior between our algorithm and the one proposed in [12]. Consider a set of four vertices  $V = \{v_1, v_2, v_3, v_4\}$ . We set  $b_{v_1} = 2$  and  $b_{v_i} = 1$  for  $2 \leq i \leq 4$ . Let  $E = \{e_1, e_2, e_3\}$  with  $e_1 = \{v_1, v_2\}$  and  $w(e_1) = 2$ ,  $e_2 = \{v_1, v_3\}$  and  $w(e_2) = 7$ ,  $e_3 = \{v_1, v_4\}$  and  $w(e_3) = 4$ . Using only one dual variable for each vertex, the algorithm of Levin and Wajc [12] takes  $e_1$  and  $e_2$  but discards  $e_3$  because the dual variable  $\phi_{v_1}$  is equal to 4 when  $e_3$  is processed. On the other hand, our algorithm, when processing  $e_3$ , compares  $w(e_3)$  with  $w_{v_1}(e_1) = 2$ . Therefore,  $e_3$  is added into  $S$  and our algorithm provides a  $b$ -matching of weight 11 instead of 9.

# General Knapsack Problems in a Dynamic Setting

**Yaron Fairstein** ✉

Computer Science Department, Technion, Haifa, Israel

**Ariel Kulik** ✉

Computer Science Department, Technion, Haifa, Israel

**Joseph (Seffi) Naor** ✉

Computer Science Department, Technion, Haifa, Israel

**Danny Raz** ✉

Computer Science Department, Technion, Haifa, Israel

---

## Abstract

The world is dynamic and changes over time, thus any optimization problem used to model real life problems must address this dynamic nature, taking into account the cost of changes to a solution over time. The multistage model was introduced with this goal in mind. In this model we are given a series of instances of an optimization problem, corresponding to different times, and a solution is provided for each instance. The strive for obtaining near-optimal solutions for each instance on one hand, while maintaining similar solutions for consecutive time units on the other hand, is quantified and integrated into the objective function. In this paper we consider the Generalized Multistage  $d$ -Knapsack problem, a generalization of the multistage variants of the Multiple Knapsack problem, as well as the  $d$ -Dimensional Knapsack problem. We present a PTAS for Generalized Multistage  $d$ -Knapsack.

**2012 ACM Subject Classification** Theory of computation → Packing and covering problems; Theory of computation → Problems, reductions and completeness

**Keywords and phrases** Multistage, Multiple-Knapsacks, Multidimensional Knapsack

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.15

**Category** APPROX

**Related Version** *Full Version:* <https://arxiv.org/abs/2105.00882>

## 1 Introduction

In many optimization settings, the problem of interest is defined over a time horizon in which the actual setting evolves, resulting in changes over time to the problem constraints and the objective function. Thus, even if the optimization problem at hand can be solved efficiently for a single time unit, it may not be clear how to extend this solution to a time-evolving setting.

An example of such a setting comes from the world of cloud management. A cloud provider maintains a data center with servers and offers clients virtual machines having different processing capabilities. Each client demands a virtual machine (with certain properties), and if provided it must pay for it. It would be naïve to assume that the demand of clients is static over time. Factors, such as peak vs. off-hours, and the day of the week, might affect client demand. Also, the cloud provider might either turn off servers to reduce hardware deterioration and electricity usage, or open more servers to meet higher demand. Thus, the optimization problem is partitioned into multiple *stages*, where in each stage there are different constraints and possibly a different optimization goal.



© Yaron Fairstein, Ariel Kulik, Joseph Naor, and Danny Raz;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 15; pp. 15:1–15:18



Leibniz International Proceedings in Informatics

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

A simple solution is to ignore the dynamicity of the problem, and solve each stage separately and independently of other stages. Thus, profit at each stage is maximized, ignoring the solutions computed for the other stages. Such a solution may result in disgruntled clients, as it can lead to intermittent service between stages. Instead, we will aim for a *multistage* solution that balances between the optimum of each stage, while preserving some continuity between consecutive stages. This will be achieved by incorporating the continuity of the solution into the overall profit.

The multistage model was first introduced by Gupta et al. [18] and Eisenstat et al. [9] to address dynamic environments. Since its introduction, it has received growing attention (examples include [1, 12, 2, 4, 15, 8]). In the multistage model we are given a sequence of instances of an optimization problem. A solution constitutes of a series of solutions, one for each instance.

Two different ideas were used to enforce a balance between single stage optimality and continuity. In [9, 18] a change cost is charged for the dissimilarity of consecutive solutions, while in [4] additional gains were given for their similarity. In the aforementioned cloud management problem, the change cost can be interpreted as installation costs and eviction costs charged when a client is initially served, and then its service is discontinued. The gains can be modeled as increased costs the client is charged to guarantee the continuity of its service.

The cloud management problem described above can be viewed as a multistage problem where the underlying optimization problem is the Multiple Knapsack problem (MKP). In MKP we are given a set of items, each associated with a weight and a profit. Also, we are given a set of bins, each one having a capacity. A feasible solution for MKP is an assignment of items to bins such that the total weight of the items assigned to each bin does not exceed its capacity. The objective is to find a feasible solution maximizing the profit accrued from the assigned items. In the context of the cloud management setting, the items are the virtual machine demands of the clients and the bins are the available servers.

## 1.1 Problem Definition

We study the Generalized Multistage  $d$ -Knapsack problem. We begin with an informal description of the problem. An instance of the problem consists of  $T$  stages, where in each stage we are given an instance of a generalization of the classic knapsack problem. While the instances differ between stages, in all stages the same set of items  $I$  can be packed. The continuity of the solution is enforced by quantifying the similarity of consecutive solutions and integrating it into the objective function.

We quantify continuity by four types of values. The first two values specify gains earned for the similarity of solutions. For example, if an item  $i$  is packed in stages  $t - 1$  and  $t$ , gain  $g_{i,t}^+$  is awarded. Similarly,  $g_{i,t}^-$  is awarded if  $i$  is not packed in  $t - 1$  and  $t$ . The other two values define the cost of changes between consecutive solutions. For example, if an item  $i$  was not packed in stage  $t - 1$ , and it is decided to pack it in stage  $t$ , a change cost of  $c_{i,t}^+$  is charged. Similarly,  $c_{i,t}^-$  is charged if  $i$  is packed in  $t$ , but not in  $t + 1$ .

The packing problem at each stage generalizes the Multiple Knapsack problem, as well as the  $d$ -Dimensional Knapsack problem. In each instance of the problem we are given  $d$  sets of bins, and the weight an item occupies in a bin depends on the set to which the bin belongs to. The profit of an item is accrued once it is assigned to some bin in all  $d$  sets of bins. This problem is called  $d$ -Multiple Knapsack Constraints Problem and is formally defined below.

A Multiple Knapsack Constraint (MKC) is a tuple  $K = (w, B, W)$  defined over a set of items  $I$ . The function  $w : I \rightarrow \mathbb{R}_+$  defines the weight of the items,  $B$  is a set of bins, each equipped with a capacity defined by the function  $W : B \rightarrow \mathbb{R}_+$ . An *assignment* is a



function  $A : B \rightarrow 2^I$ , defining which items are assigned to each of the bins. An assignment is feasible if  $w(A(b)) = \sum_{i \in A(b)} w_i \leq W(b)$  for each bin  $b \in B$ . Similarly, given a tuple of MKCs  $\mathcal{K} = (K_j)_{j=1}^d$  over  $I$ , a tuple of  $d$  assignments  $\mathcal{A} = (A^j)_{j=1}^d$  is feasible for  $\mathcal{K}$  if for each  $j = 1, \dots, d$  assignment  $A^j$  is a feasible assignment for  $K_j$ . We say  $A$  is an assignment of set  $S \subseteq I$  if  $S = \cup_{b \in B} A_b$ .

In  $d$ -Multiple Knapsack Constraints Problem ( $d$ -MKCP), a problem first introduced in [11], we are given a tuple  $(I, \mathcal{K}, p)$ , where  $I$  is a set of items,  $\mathcal{K}$  is a tuple of  $d$  MKCs and  $p : I \rightarrow \mathbb{R}_{\geq 0}$  defines the profit of each item. A feasible solution for  $d$ -MKCP is a set  $S \subseteq I$  and a tuple of feasible assignments  $\mathcal{A}$  (w.r.t  $\mathcal{K}$ ) of  $S$ . The goal is to find a feasible solution that maximizes  $p(S) = \sum_{i \in S} p(i)$ . We note that if there exists an item with negative profit it can be discarded in advance. This fact is used later on, in Section 2.1.

The Generalized Multistage  $d$ -Knapsack problem ( $d$ -GMK), is the multistage model of  $d$ -MKCP. The problem is defined over a time horizon of  $T$  stages as follows. An instance of the problem is a tuple  $((\mathcal{P}_t)_{t=1}^T, g^+, g^-, c^+, c^-)$ , where  $\mathcal{P}_t = (I, \mathcal{K}_t, p_t)$  is a  $d_t$ -MKCP instance with  $d_t \leq d$  for  $t \in [T]$ ,  $g^+, g^- \in \mathbb{R}_+^{I \times [2, T]}$  are the gain vectors and  $c^+, c^- \in \mathbb{R}_+^{I \times [1, T]}$  are the change cost vectors.<sup>1</sup> We use  $g_{i,t}^+$  and  $g_{i,t}^-$  to denote the gain of item  $i$  at stage  $t$ . Similarly, we use  $c_{i,t}^+$  and  $c_{i,t}^-$  to denote the change cost of item  $i$  at stage  $t$ .

A feasible solution for  $d$ -GMK is a tuple  $(S_t, \mathcal{A}_t)_{t=1}^T$ , where  $(S_t, \mathcal{A}_t)$  is a feasible solution for  $\mathcal{P}_t$  (note that  $\mathcal{A}_t$  is a tuple of assignments of  $S_t$ ). Throughout the paper we assume  $S_0 = S_{T+1} = \emptyset$  and denote the objective function of instance  $\mathcal{Q}$  by  $f_{\mathcal{Q}} : I^T \rightarrow \mathbb{R}$ , where

$$f_{\mathcal{Q}}((S_t)_{t=1}^T) = \sum_{t=1}^T \sum_{i \in S_t} p_t(i) + \sum_{t=2}^T \left( \sum_{i \in S_{t-1} \cap S_t} g_{i,t}^+ + \sum_{i \notin S_{t-1} \cup S_t} g_{i,t}^- \right) - \sum_{t=1}^T \left( \sum_{i \in S_t \setminus S_{t-1}} c_{i,t}^+ + \sum_{i \in S_t \setminus S_{t+1}} c_{i,t}^- \right).$$

The goal is to find a feasible solution that maximizes the objective function  $f_{\mathcal{Q}}$ .

A study of  $d$ -GMK reveals it does not admit a constant factor approximation algorithm (see Section 3). We found that in hard instances the change costs are much larger than the profits. Thus we consider an important parameter of the problem, the *profit-cost ratio*. It is defined as the maximum ratio, over all items, between the change cost  $(c^+, c^-)$  and the profit of an item over all stages. It is denoted by  $\phi_{\mathcal{Q}}$  for any instance  $\mathcal{Q}$ , and is formally defined as

$$\phi_{\mathcal{Q}} = \min \left( \left\{ \infty \right\} \cup \left\{ r \geq 0 \mid \forall i \in I, t_1, t_2 \in [T] : \max \{ c_{i,t_1}^+, c_{i,t_1}^- \} \leq r \cdot p_{t_2}(i) \right\} \right)$$

We show that  $d$ -GMK instances where the profit-cost ratio is bounded by a constant admit a PTAS.

We also consider Submodular  $d$ -GMK, a submodular variant of  $d$ -GMK where the profit functions are replaced with monotone submodular set functions. A set function  $p : 2^I \rightarrow \mathbb{R}$  is submodular if for every  $A \subseteq B \subseteq I$  and  $i \in I \setminus B$  it holds that  $p(A \cup \{i\}) - p(A) \geq p(B \cup \{i\}) - p(B)$ . Submodular functions appear naturally in many settings such as coverage [13], matroid rank [5] and cut functions [14]. We use similar techniques to develop the algorithms for  $d$ -GMK and Submodular  $d$ -GMK. Thus, we focus on  $d$ -GMK and defer the formal definition as well as the algorithm for Submodular  $d$ -GMK to the full version of this paper [10].

<sup>1</sup> We use the notations  $[n, m] = \{i \in \mathbb{N} \mid n \leq i \leq m\}$  and  $[n] = [1, n]$  for  $n, m \in \mathbb{N}$ .

Both  $d$ -GMK and Submodular  $d$ -GMK generalize the Multistage Knapsack problem recently considered by Bampis et al. [4]. There are several aspects by which it is generalized. First, handling multiple knapsack constraints as well as  $d$ -dimensional knapsack vs a single knapsack in [4]. Second, the profit earned from assigning items can be described as a submodular function, not only by a modular function. Third, [4] considered only symmetric gains, i.e., the same gain is earned whether an item is assigned or not assigned in consecutive stages. Lastly, change costs were not considered in [4].

## 1.2 Our Results

Our main result is stated in the following theorem.

► **Theorem 1.** *For any fixed  $d \in \mathbb{N}$  and  $\phi \geq 1$  there exists a randomized PTAS for  $d$ -GMK with a profit-cost ratio bounded by  $\phi$ .*

The result uses the general framework of [4], in which the authors first presented an algorithm for instances with bounded time horizon, and then showed how it can be scaled for general instances. To handle bounded time horizons we show an approximation factor preserving reduction (as defined in [22]<sup>2</sup>) from  $d$ -GMK to a generalization of  $q$ -MKCP. The reduction illuminates the relationship between  $d$ -GMK and  $q$ -MKCP. As  $q$ -MKCP admits a PTAS [11], this results in a PTAS for  $d$ -GMK instances with a bounded time horizon.

We note the reduction can be applied to the problem considered in [4] as well. In this case the target optimization problem is  $d$ -dimensional knapsack with a matroid constraint. As the latter problem is known to admit a PTAS [17], this suggests a simpler solution for bounded time horizon in comparison to the one given in [4].

To generalize the result to unbounded time horizon we use an approach similar to [4], though a more sophisticated analysis was required to handle the change costs. The generalization is achieved by cutting the time horizon into sub-instances with a fixed time horizon. Each sub-instance is solved separately, and then the solutions are combined to create a solution for the full instance. Handling change costs is trickier as cutting an instance may lead to an excessive charge of change costs at the cut points. We must compensate for these additional costs, or we will not be able to bound the value of the solution.

The results for the modular variant generalizes the PTAS for Multistage Knapsack [4]. For  $d \geq 2$ , we cannot expect better results as even  $d$ -KP, also generalized by  $d$ -GMK, does not admit an *efficient* PTAS (EPTAS). Theorem 2 shows an EPTAS cannot be obtained for 1-GMK as well

► **Theorem 2.** *Unless  $W[1] = FPT$ , there is no EPTAS for 1-GMK, even if the length of the time horizon is  $T = 2$ , the set of bins in each MKC contains one bin and there are no change costs.*

The theorem is proved using a simple reduction from 2-dimensional knapsack. Bampis et al. [4] considered a similar withered down instance and proved that even if the gains are symmetric (i.e.,  $g_{i,t}^+ = g_{i,t}^-$ ) a *Fully* PTAS (FPTAS) does not exist for the problem.

Using a reduction from multidimensional knapsack we show that 1-GMK, in its general form, cannot be approximated to any constant factor.

► **Theorem 3.** *For any  $d \geq 1$ , there is no polynomial time approximation algorithm for  $d$ -GMK with a constant approximation ratio, unless  $NP = ZPP$ .*

---

<sup>2</sup> A formal definition is provided in Appendix A for completeness

This result justifies our study of the special cases of  $d$ -GMK in which the profit-cost ratio is bounded by a constant.

The techniques used to develop the algorithm for  $d$ -GMK can be adjusted slightly to produce an approximation algorithm for Submodular  $d$ -GMK. The complete details are given in the full version of this paper [10].

► **Theorem 4.** *For any fixed  $d \in \mathbb{N}$  and  $\epsilon > 0$  there exists a randomized  $(1 - \frac{1}{e} - \epsilon)$ -approximation algorithm for Submodular  $d$ -GMK.*

In the submodular variant one cannot hope for vast improvement over our results as the algorithm is almost tight. This is due to the hardness results for submodular maximization subject to a cardinality constraint presented by Nemhauser and Wolsey [20].

### 1.3 Related Work

In the multistage model we are given a series of instances of an optimization problem, and we search for a solution which optimizes each instance while maintains some similarity between solutions. Many optimization problem were considered under this framework. These include matching [2, 7], clustering [8], subset sum [3], vertex cover [15] and minimum  $s - t$  path [16].

The multistage model was first presented by both Eisenstat et al. [9] and Gupta et al. [18]. In [9] the Multistage Facility Location problem was considered, where the underlying metric in which clients and facility reside changes over time. A logarithmic approximation algorithm was presented for two variants of the problem; the hourly opening costs where the opening cost of a facility is charged at each stage in which it is open, and the fixed opening costs where a facility is open at all stages after its opening costs is paid. A logarithmic hardness result was also presented for the fixed opening costs variant. An et al. [1] improved the result for the hourly opening costs variant and presented a constant factor approximation. Fairstein et al. [12] proved that the logarithmic hardness result does not hold if only the client locations change over time and the facilities are static.

As mentioned, the multistage model was also introduced by Gupta et al. [18], where the Multistage Matroid Maintenance (MMM) problem was considered. In MMM we are given a set of elements equipped with costs that change over time. In addition, we are given a matroid. The goal is to select a base of minimum costs at each stage whilst minimizing the cost of the difference of bases selected for consecutive stages. Gupta et al. [18] presented a logarithmic approximation algorithm for the problem, as well as fitting lower bound, proving this result is tight.

### Organization

Section 2 provides the approximation schemes from Theorem 1. Hardness results are presented in Section 3.

## 2 Approximation Scheme for $d$ -GMK

In this section we derive the approximation scheme for  $d$ -GMK with bounded profit-cost ratio. In Section 2.1 we show how a PTAS for instances with bounded time horizon can be obtain via a reduction to a variant of  $d$ -MKCP. Subsequently, in Section 2.2 we show how the algorithm for bounded time horizon can be used to approximate general instance.

## 2.1 Bounded Time Horizon

In this section we provide a reduction from an instance of  $q$ -GMK to a generalization of  $d$ -MKCP (for specific values of  $d$  and  $q$ ). The generalization was presented in [11] and is called  $d$ -MKCP With A Matroid Constraint ( $d$ -MKCP+) and is defined by a tuple  $(I, \mathcal{K}, p, \mathcal{I})$ , where  $(I, \mathcal{K}, p)$  forms an instance of  $d$ -MKCP. Also, the set  $\mathcal{I} \subseteq 2^I$  defines a matroid<sup>3</sup> constraint. A feasible solution for  $d$ -MKCP+ is a set  $S \in \mathcal{I}$  and a tuple of feasible assignments  $\mathcal{A}$  (w.r.t  $\mathcal{K}$ ) of  $S$ . The goal is to find a feasible solution which maximizes  $\sum_{i \in S} p(i)$ . The following definition presents the construction of the reduction.

► **Definition 5.** Let  $\mathcal{Q} = ((\mathcal{P}_t)_{t=1}^T, g^+, g^-, c^+, c^-)$  be an instance of  $d$ -GMK, where  $\mathcal{P}_t = (I, \mathcal{K}_t, p_t)$  and  $\mathcal{K}_t = (K_{t,j})_{j=1}^{d_t}$ . Define  $R(\mathcal{Q}) = (E, \tilde{\mathcal{K}}, \tilde{p}, \mathcal{I})$  where

- $E = I \times 2^{[T]}$
- $\mathcal{I} = \{S \subseteq E \mid \forall i \in I: |S \cap (\{i\} \times 2^{[T]})| \leq 1\}$
- For  $t \in [T], j \leq d_t$  set MKC  $\tilde{K}_{t,j} = (\tilde{w}_{t,j}, B_{t,j}, W_{t,j})$  over  $E$ , where  $K_{t,j} = (w_{t,j}, B_{t,j}, W_{t,j})$  and

$$\tilde{w}_{t,j}((i, D)) = \begin{cases} w_{t,j}(i) & t \in D \\ 0 & \text{otherwise} \end{cases}$$

- For  $t = 1, \dots, T, d_t < j \leq d$  set MKC  $\tilde{K}_{t,j} = (w_0, \{b\}, W_0)$  over  $E$ , where  $w_0 : 2^E \rightarrow \{0\}$ ,  $W_0(b) = 0$  and  $b$  is an arbitrary bin (object).
- $\tilde{\mathcal{K}} = (\tilde{K}_{t,j})_{t \in [T], j \in [d]}$ .
- The objective function  $\tilde{p}$  is defined as follows.

$$\tilde{p}(S) = \sum_{(i,D) \in S} \left( \sum_{t \in D} p_t(i) + \sum_{t \in D: t-1 \in D} g_{i,t}^+ + \sum_{t \notin D: t-1 \notin D} g_{i,t}^- - \sum_{t \in D: t-1 \notin D} c_{i,t}^+ - \sum_{t \in D: t+1 \notin D} c_{i,t}^- \right)$$

Each element  $(i, D) \in E$  states the subset of stages in which item  $i$  is assigned. I.e.,  $i$  is only assigned in stages  $t \in D$ . Thus any solution should include at most one element  $(i, D)$  for each  $i \in I$ . This constraint is fully captured by the partition matroid constraint defined by the set of independent sets  $\mathcal{I}$ . Finally, if an element  $(i, D)$  is selected, we must assign  $i$  in each MKC  $K_{t,j}$  for  $j \in [d_t], t \in D$ . This is captured by the weight function  $\tilde{w}$ , as an element  $(i, D)$  weighs  $w_{t,j}(i)$  if and only if  $t \in D$  (otherwise its weight is zero and it can be assigned for “free”).

► **Lemma 6.** For any  $d$ -GMK instance  $\mathcal{Q}$  with time horizon  $T$ , it holds that  $R(\mathcal{Q})$  is a  $dT$ -MKCP+ instance.

**Proof.** Let  $\mathcal{Q} = ((\mathcal{P}_t)_{t=1}^T, g^+, g^-, c^+, c^-)$  be an instance of  $d$ -GMK, where  $\mathcal{P}_t = (I, \mathcal{K}_t, p_t)$  and  $\mathcal{K}_t = (K_{t,j})_{j=1}^{d_t}$ . Also, let  $R(\mathcal{Q}) = (E, \tilde{\mathcal{K}}, \tilde{p}, \mathcal{I})$  be the reduced instance of  $\mathcal{Q}$  as defined in Definition 5. It is easy to see that the set  $\mathcal{I}$  is the independent sets of a partition matroid, as for each item  $i$  at most one element  $(i, D)$  can be chosen. Thus,  $\mathcal{I}$  is the family of independent sets of a matroid as required.

Next,  $\mathcal{K}$  defines a tuple of MKCs, so all that is left to prove is that  $\tilde{p}$  is non-negative and modular. For each element  $(i, D) \in E$  we can define a fixed value

$$v((i, D)) = \sum_{t \in D} p_t(i) + \sum_{t \in D: t-1 \in D} g_{i,t}^+ + \sum_{t \notin D: t-1 \notin D} g_{i,t}^- - \sum_{t \in D: t-1 \notin D} c_{i,t}^+ + \sum_{t \in D: t+1 \notin D} c_{i,t}^-$$

It immediately follows that  $\tilde{p}(S) = \sum_{e \in S} v(e)$  and that  $\tilde{p}$  is modular. As stated in Section 1.1, elements with negative values are discarded in advance such that  $\tilde{p}$  is also non-negative. ◀

<sup>3</sup> A formal definition for matroid can be found in [21]

► **Lemma 7.** *Let  $\mathcal{Q}$  be an instance of  $d$ -GMK with time horizon  $T$ . For any feasible solution  $(S_t, \mathcal{A}_t)_{t=1}^T$  of  $\mathcal{Q}$  there exists a feasible solution  $(S, (\tilde{A}_{t,j})_{t \in [T], j \in [d]})$  of  $R(\mathcal{Q})$  such that  $f_{\mathcal{Q}}((S_t)_{t=1}^T) = \tilde{p}(S)$ .*

**Proof.** Let  $\mathcal{Q} = ((\mathcal{P}_t)_{t=1}^T, g^+, g^-, c^+, c^-)$  be an instance of  $d$ -GMK, where  $\mathcal{P}_t = (I, \mathcal{K}_t, p_t)$  and  $\mathcal{K}_t = (K_{t,j})_{j=1}^{d_t}$ . Also, let  $R(\mathcal{Q}) = (E, \tilde{\mathcal{K}}, \tilde{p}, \mathcal{I})$  be the reduced instance of  $\mathcal{Q}$ , where  $\tilde{\mathcal{K}} = (\tilde{K}_{t,j})_{t \in [T], j \in [d]}$  and  $\tilde{K}_{t,j} = (\tilde{w}, B_{t,j}, W_{t,j})$  (see Definition 5). Consider some feasible solution  $(S_t, \mathcal{A}_t)_{t=1}^T$  for  $\mathcal{Q}$ , where  $\mathcal{A}_t = (A_{t,j})_{j=1}^{d_t}$ . In the following we define a solution  $(S, (\tilde{A}_{t,j})_{t \in [T], j \in [d]})$  for  $R(\mathcal{Q})$ . Let

$$S = \{(i, D) \mid i \in I, D = \{t \in [T] \mid i \in S_t\}\}$$

It can be easily verified that  $S \in \mathcal{I}$ . The value of the subset  $S$  is

$$\begin{aligned} \tilde{p}(S) &= \\ & \sum_{(i,D) \in S} \left( \sum_{t \in D} p_t(i) + \sum_{t \in D: t-1 \in D} g_{i,t}^+ + \sum_{t \notin D: t-1 \notin D} g_{i,t}^- - \sum_{t \in D: t-1 \notin D} c_{i,t}^+ - \sum_{t \in D: t+1 \notin D} c_{i,t}^- \right) = \\ & \sum_{t=1}^T \sum_{i \in S_t} p_t(i) + \sum_{t=2}^T \left( \sum_{i \in S_{t-1} \cap S_t} g_{i,t}^+ + \sum_{i \notin S_{t-1} \cup S_t} g_{i,t}^- \right) - \sum_{t=1}^T \left( \sum_{i \in S_t \setminus S_{t-1}} c_{i,t}^+ + \sum_{i \in S_t \setminus S_{t+1}} c_{i,t}^- \right) = \\ & f_{\mathcal{Q}}((S_t)_{t=1}^T). \end{aligned}$$

Next, for each  $t \in [T], j \in [d]$  we present an assignment  $\tilde{A}_{t,j}$  of  $S$ . Consider the following two cases:

1. If  $j > d_t$ , recall  $\tilde{K}_{t,j} = (w_0, \{b\}, W_0)$  where  $w_0(i, D) = 0$  for all  $(i, D) \in E$  and  $W_0(b) = 0$ . We define  $\tilde{A}_{t,j}$  by  $\tilde{A}_{t,j}(b) = S$ . It thus holds that  $w_0(\tilde{A}_{t,j}(b)) = 0 = W_0$ . That is,  $\tilde{A}_{t,j}$  is feasible.
2. If  $j \leq d_t$ , let  $b^* \in B_{t,j}$  be some unique bin in  $B_{t,j}$  and define assignment  $\tilde{A}_{t,j} : B_{t,j} \rightarrow 2^E$  by

$$\begin{aligned} \tilde{A}_{t,j}(b) &= \left( A_{t,j}(b) \times 2^{[T]} \right) \cap S & \forall b \in B_{t,j} \setminus \{b^*\} \\ \tilde{A}_{t,j}(b^*) &= \left( \left( A_{t,j}(b^*) \times 2^{[T]} \right) \cap S \right) \cup \{(i, D) \in S \mid t \notin D\} \end{aligned} \quad (1)$$

The assignment  $\tilde{A}_{t,j}$  is a feasible assignment w.r.t  $\tilde{K}_{t,j}$  since for each bin  $b \in B_{t,j}$  it holds that

$$\sum_{(i,D) \in \tilde{A}_{t,j}(b)} \tilde{w}_{t,j}((i, D)) = \sum_{i \in A_{t,j}(b)} w_{t,j}(i) \leq W_{t,j}(b)$$

Let  $(i, D) \in S$ . If  $i \in S_t$  there is  $b \in B_{t,j}$  such that  $i \in A_{t,j}(b)$ , hence  $(i, D) \in \tilde{A}_{t,j}(b)$  by (1). If  $i \notin S_t$  then  $t \notin D$  and thus  $(i, D) \in \tilde{A}_{t,j}(b^*)$ . Overall, we have  $S \subseteq \bigcup_{b \in B_{t,j}} \tilde{A}_{t,j}(b)$ .

By (1) it follows that  $S \supseteq \bigcup_{b \in B_{t,j}} \tilde{A}_{t,j}(b)$  as well, thus  $S = \bigcup_{b \in B_{t,j}} \tilde{A}_{t,j}(b)$ . I.e.,  $\tilde{A}_{t,j}$  is an assignment of  $S$ .

Note that the assignments can be constructed in polynomial time. We can conclude that  $(S, (\tilde{A}_{t,j})_{t \in [T], j \in [d]})$  is a feasible solution for  $R(\mathcal{Q})$ , and its value is  $f_{\mathcal{Q}}((S_t)_{t=1}^T)$ . ◀

► **Lemma 8.** *Let  $\mathcal{Q}$  be an instance of  $d$ -GMK (with arbitrary time horizon  $T$ ). For any feasible solution  $(S, (\tilde{A}_{t,j})_{t \in [T], j \in [d]})$  for  $R(\mathcal{Q})$  a feasible solution  $(S_t, \mathcal{A}_t)_{t=1}^T$  for  $\mathcal{Q}$  such that  $f_{\mathcal{Q}}((S_t)_{t=1}^T) = \tilde{f}(S)$  can be constructed in polynomial time.*

## 15:8 General Knapsack Problems in a Dynamic Setting

The proof of Lemma 8 is similar to the proof of Lemma 7, thus it is deferred to Appendix A.

For any  $d$ -GMK instance with a fixed time horizon  $T$ , the reduction  $R(\mathcal{Q})$  can be constructed in polynomial (as  $|E| = |I| \cdot 2^{|T|}$ ). The next corollary follows from this observation and lemmas 7 and 8.

► **Corollary 9.** *For any fixed  $T \in \mathbb{N}$ , there exists an approximation factor preserving reduction from  $d$ -GMK with a time horizon bounded by  $T$  to  $dT$ -MKCP+.*

In [11] a PTAS for  $d$ -MKCP+ is presented. Thus, the next lemma follows from the above corollary.

► **Lemma 10.** *For any fixed  $T \in \mathbb{N}$  there exists a randomized PTAS for  $d$ -GMK with a time horizon bounded by  $T$ .*

### 2.2 General Time Horizon

In this section we present an algorithm for  $d$ -GMK with a general time horizon  $T$ . This is done by cutting the time horizon at several stages into sub-instances. Each sub-instance is optimized independently and then the solutions are combined to create a solution for the complete instance. A somewhat similar technique was used in [4]. However, they considered a model without change costs which is much simpler. Our analysis is more delicate as it requires local consideration of the assignment of each item to ensure that any additional costs charged are covered by profit and gains earned.

Given an instance  $\mathcal{Q} = ((\mathcal{P}_t)_{t=1}^T, g^+, g^-, c^+, c^-)$  we define a sub-instance for the sub-range  $[t_1, t_2]$ , denoted throughout this section by  $((\mathcal{P}_t)_{t=t_1}^{t_2}, g^+, g^-, c^+, c^-)$  without shifting or truncating the gain and change costs vectors. For example, for  $t \in [t_1, t_2]$  gain  $g_{i,t}^+$  is earned for assigning item  $i$  in stages  $t$  and  $t+1$ . Also, observe that stages  $t_1 - 1$  and  $t_2 + 1$  are outside the scope of the instance. Thus, when evaluating a solution for the sub-instance it is assumed that  $S_{t_1-1} = S_{t_2+1} = \emptyset$ .

Given an integer  $T \in \mathbb{N}$ , a set of *cut points*  $U = \{u_0, \dots, u_k\}$  of  $T$  is a set of integers such that for every  $j = 0, \dots, k-1$  it holds that  $u_j < u_{j+1}$  and  $1 = u_0 < u_k = T + 1$ .

► **Definition 11.** *Let  $\mathcal{Q} = ((\mathcal{P}_t)_{t=1}^T, g^+, g^-, c^+, c^-)$  be an instance of  $d$ -GMK, where  $\mathcal{P}_t = (I, \mathcal{K}_t, p_t)$ . Also, let  $U = \{u_0, \dots, u_k\}$  be a set of cut points. The tuple of  $d$ -GMK instances  $\mathcal{Q}_U = \left( (\mathcal{P}_t)_{t=u_j}^{u_{j+1}-1}, g^+, g^-, c^+, c^- \right)_{j=0}^{k-1}$  is defined as the cut instances of  $\mathcal{Q}$  w.r.t  $U$ .*

► **Definition 12.** *Let  $\mathcal{Q}$  be an instance of  $d$ -GMK,  $U = \{u_0, \dots, u_k\}$  be a set of cut points and  $\mathcal{Q}_U$  be the respective cut instances. Also, let  $\left( (S_t, \mathcal{A}_t)_{t=u_j}^{u_{j+1}-1} \right)_{j=0}^{k-1}$  be a tuple of feasible solutions for the tuple of cut instances  $\mathcal{Q}_U$ . Then, the solution  $(S_t, \mathcal{A}_t)_{t=1}^T$  for  $\mathcal{Q}$  is called a cut solution.*

The next corollary elaborates on the relationship between a cut solution and the cut instance solutions from which it is constructed.

► **Corollary 13.** *Given any  $d$ -GMK instance  $\mathcal{Q}$ , cut points  $U$ , cut instances  $\mathcal{Q}_U$  and feasible solutions for the cut instances, the respective cut solution is a feasible solution for  $\mathcal{Q}$ , and its value is at least the sum of values of the solutions for the cut instances.*

The proof of the corollary is fairly simple, and is deferred to Appendix A. We are now ready to present the algorithm for  $d$ -GMK with general time horizon length.

---

**Algorithm 1** General Time Horizon.

- 
- Input :**  $0 < \epsilon < \frac{1}{4}$ ,  $\phi \geq 1$ , a  $d$ -GMK instance  $\mathcal{Q}$  with time horizon  $T$  such that  $\phi_{\mathcal{Q}} \leq \phi$ , and  $\alpha$ -approximation algorithm  $A$  for  $d$ -GMK with time horizon  $T \leq \frac{2\phi}{\epsilon^2}$ .
- 1 Set  $\mu = \frac{\epsilon^2}{\phi}$ .
  - 2 **for**  $j = 1, \dots, \frac{1}{\mu}$  **do**
  - 3     Set  $U_j = \left\{ \frac{a}{\mu} + j - 1 \mid a \in \mathbb{N}, a \geq 1, \frac{a}{\mu} + j - 1 \leq T - \frac{1}{\mu} \right\} \cup \{1, T + 1\}$ .
  - 4     Find a solution for each cut instance in  $\mathcal{Q}_{U_j}$  using algorithm  $A$  and set  $\mathcal{S}_j$  as the respective cut solution.
  - 5 **Return** the solution  $\mathcal{S}_j$  which maximizes the objective function  $f_{\mathcal{Q}}$ .
- 

Before analysing the algorithm we present several definitions and lemmas that are essential for the proof. First, we start by reformulating the solution. Instead of describing the assignment of items by the tuple  $(S_t)_{t=1}^T$ , we define a new set of elements  $E = I \times [T] \times [T]$ , where each element  $(i, t_1, t_2) \in E$  states that item  $i$  is assigned in the interval  $[t_1, t_2]$ . Given a feasible solution  $(S_t, \mathcal{A}_t)_{t=1}^T$  for  $\mathcal{Q}$  we denote the representation of  $(S_t)_{t=1}^T$  as a subset of  $E$  by  $E((S_t)_{t=1}^T)$  and it is equal to

$$E((S_t)_{t=1}^T) = \{(i, t_1, t_2) \in E \mid \forall t \in [t_1, t_2] : i \in S_t \text{ and } i \notin S_{t_1-1} \cup S_{t_2+1}\}.$$

If  $\tilde{S} = E((S_t)_{t=1}^T)$ , we define the reverse mapping as  $\tilde{S}(t) = \{i \in I \mid \exists(i, t_1, t_2) \in \tilde{S} : t \in [t_1, t_2]\} = S_t$ . Now, we can define a solution for  $d$ -GMK using our new representation as  $(\tilde{S}, \mathcal{A}_t)_{t=1}^T$ .

► **Definition 14.** The value,  $v(e)$ , of element  $e = (i, t_1, t_2)$  is defined as the total value earned from assigning  $i$  in the range  $[t_1, t_2]$  minus the change costs charge for assigning and discarding it. Formally,

$$v(e) = \sum_{t=t_1}^{t_2} p_t(i) + \sum_{t=t_1+1}^{t_2} g_{i,t}^+ - c_{i,t_1}^+ - c_{i,t_2}^-$$

The value of solution  $\tilde{S} \subseteq E$  for  $d$ -GMK instance  $\mathcal{Q}$  is  $\sum_{e \in \tilde{S}} v(e) + \sum_{t=2}^T \sum_{i \notin \tilde{S}(t-1) \cup \tilde{S}(t)} g_{i,t}^-$  and it is equal to  $f_{\mathcal{Q}}((\tilde{S}(t))_{t=1}^T)$ .

In Algorithm 1 we consider a solution for a tuple of cut instances created by cutting an instance at a set of cut points. Here we consider the opposite action, the effect of cutting a solution at these cut points. We start by considering a single cut point.

► **Definition 15.** Given an element  $e = (i, t_1, t_2)$  and a cut point  $u \in (t_1, t_2]$  we define the outcome of cutting  $e$  at  $u$  as the set of intervals  $u(e) = \{(i, t_1, u-1), (i, u, t_2)\}$ . Also, the loss caused by cutting  $e$  at  $u$  is defined as the difference between the value of  $e$  and the sum of value of elements in  $u(e)$ . It is denoted by  $\ell(e, u)$  and is equal to

$$\ell(e, u) = v(e) - \sum_{e' \in u(e)} v(e') = g_{i,u}^+ + c_{i,u}^+ + c_{i,u-1}^- \quad (2)$$

We can similarly extend the definition to include more than one cut point as follows.

► **Definition 16.** Given a set of cut points  $U = \{u_0, u_1, \dots, u_k\}$  and an element  $e = (i, t_1, t_2)$  define  $U(e)$  as the set of elements created by cutting the  $e$  at all cut points in  $U$ . Formally,

$$U((i, t_1, t_2)) = \left\{ (i, \max\{t_1, u_{r-1}\}, \min\{u_r - 1, t_2\}) \mid r \in [k] \text{ and } [u_{r-1}, u_r - 1] \cap [t_1, t_2] \neq \emptyset \right\}$$

## 15:10 General Knapsack Problems in a Dynamic Setting

Consider the following example as a demonstration of the above definition. If  $e = (i, t_1, t_2)$  and  $(t_1, t_2) \cap U_j = \{u_2, u_3\}$ , then  $U_j(e) = \{(i, t_1, u_2 - 1), (i, u_2, u_3 - 1), (i, u_3, t_2)\}$ .

The definition of loss caused by cutting an element can be extended to a set of cut points  $U$ . If element  $e = (i, t_1, t_2)$  is cut by a of cut points  $U$  the loss is

$$\ell(e, U) = v(e) - \sum_{e' \in U(e)} v(e') = \sum_{u \in U \cap (t_1, t_2]} (g_{i,u}^+ + c_{i,u}^+ + c_{i,u-1}^-) = \sum_{u \in U \cap (t_1, t_2]} \ell(e, u) \quad (3)$$

since only gains  $g^+$  are lost due to cutting as well as change cost for splitting an assignment into two intervals. This means that even if an element is cut multiple times, the loss due to each cut point can be considered separately.

► **Lemma 17.** *Let  $0 < \epsilon < \frac{1}{4}$ ,  $\phi \geq 1$  and  $A$  be an  $\alpha$ -approximation algorithm for  $d$ -GMK with time horizon  $T \leq \frac{2\phi}{\epsilon^2}$ . Also, let  $\mathcal{Q}$  be an instance of  $d$ -GMK such that  $\phi_{\mathcal{Q}} \leq \phi$ . Algorithm 1 approximates  $\mathcal{Q}$  within a factor of  $(1 - \epsilon)\alpha$ .*

**Proof.** Let  $\mathcal{Q} = ((P_t)_{t=1}^T, g^+, g^-, c^+, c^-)$  be an instance of  $d$ -GMK with time horizon  $T$  and profit-cost ratio  $\phi_{\mathcal{Q}} \leq \phi$ . We assume for simplicity  $\phi$  is integral. Let  $0 < \epsilon < \frac{1}{4}$  and  $\mu = \frac{\epsilon^2}{\phi}$ . Also, let  $A$  be an  $\alpha$ -approximation algorithm for  $d$ -GMK with time horizon  $T' \leq \frac{2\phi}{\epsilon^2} = \frac{2}{\mu}$ . Note, if  $T \leq \frac{2}{\mu}$ , the cut points set  $U_0$  is an empty set, and in this case  $A$  returns an  $\alpha$ -approximation solution for  $\mathcal{Q}$  as required.

Let  $U_j = \{u_1^j, \dots, u_{k_j}^j\}$  for  $j = 1, \dots, \frac{1}{\mu}$ . We show that there exists a set of cut points  $U_j$  and a tuple of solutions  $\left( (S_t, \mathcal{A}_t)_{t=u_r^j}^{u_{r+1}^j-1} \right)_{r=0}^{k_j-1}$  for each cut instance in  $\mathcal{Q}_{U_j} = (q_j^r)_{r=0}^{k_j-1}$ , such that the sum of values of the solutions,  $\sum_{r=0}^{k_j-1} f_{q_j^r} \left( (S_t)_{t=u_r^j}^{u_{r+1}^j-1} \right)$ , is sufficiently large. From Corollary 13 it follows that the value of a cut solution is larger than the sum of its parts (due to lost gains and change costs saved if an item is assigned in adjacent instances). Thus, this also proves that the maximum cut solution found is sufficiently large as well.

Let  $(S_t^*, \mathcal{A}_t^*)_{t=1}^T$  be an optimal solution for  $\mathcal{Q}$ , and let  $\tilde{S}^* = E \left( (S_t^*)_{t=1}^T \right)$ . We partition  $\tilde{S}^*$  into two subsets by the length of the interval they describe. Formally,  $X = \left\{ (i, t_1, t_2) \in \tilde{S}^* \mid t_2 - t_1 < \frac{\phi}{\epsilon} \right\}$  and  $Y = \tilde{S}^* \setminus X$ . So  $X$  contains short intervals, and  $Y$  contains long intervals.

Define  $\tilde{S}_j$  as the subset of elements longer than  $\phi$  in  $\cup_{e \in Y} U_j(e)$  as well as short elements  $e \in X$  that are not cut by  $U_j$ . I.e.,

$$\tilde{S}_j = \{e \in X \mid U_j(e) = \{e\}\} \cup \bigcup_{e \in Y} \{(i, t_1, t_2) \in U_j(e) \mid t_2 - t_1 \geq \phi\}$$

At each stage  $t \in [T]$  it holds that  $\tilde{S}_j(t) \subseteq S_t^*$ . Thus there exists a tuple of assignments, denoted by  $\mathcal{A}_t^j$ , such that  $(\tilde{S}_j, \mathcal{A}_t^j)_{t=1}^T$  is a feasible solution for  $\mathcal{Q}$ . We partition set  $\tilde{S}_j$  as follows. Let  $\tilde{S}_{j,r} = \left\{ (i, t_1, t_2) \in \tilde{S}_j \mid [t_1, t_2] \subseteq [u_r^j, u_{r+1}^j - 1] \right\}$ . It holds that  $\tilde{S}_j = \cup_{r=0}^{k_j-1} \tilde{S}_{j,r}$  as each element  $(i, t_1, t_2)$  is contained in exactly one interval  $[u_r^j, u_{r+1}^j - 1]$ . Thus  $\left( (\tilde{S}_{j,r}, \mathcal{A}_t^j)_{t=u_r^j}^{u_{r+1}^j-1} \right)_{r=0}^{k_j-1}$  is a tuple of feasible solutions for the cut instances in  $\mathcal{Q}_{U_j}$  such that  $(\tilde{S}_{j,r}, \mathcal{A}_t^j)_{t=u_r^j}^{u_{r+1}^j-1}$  is a solution for the  $r$ -th instance. The value of all elements in the defined solutions for the cut instances is

$$\sum_{r=0}^{k_j-1} \sum_{e \in \tilde{S}_{j,r}} v(e) = \sum_{e \in \tilde{S}_j} v(e) = \sum_{e \in X: U_j(e) = \{e\}} v(e) + \sum_{e \in Y} \sum_{e' = (i, t_1, t_2) \in U_j(e): t_2 - t_1 \geq \phi} v(e')$$



Thus, after including the value of gains  $g^-$  earned by solutions  $\tilde{S}_{j,r}$ , we can bound the total value of optimal solutions for all cut instances  $\mathcal{Q}_{U_j}$  by

$$\sum_{e \in X: U_j(e) = \{e\}} v(e) + \sum_{e \in Y} \sum_{\substack{e' = (i, t_1, t_2) \in U_j(e): \\ t_2 - t_1 \geq \phi}} v(e') + \sum_{\substack{t \in [u_r^j + 1, u_{r+1}^j - 1]: \\ u_r^j \in U_j}} \sum_{i \notin \tilde{S}_{j,r}(t-1) \cup \tilde{S}_{j,r}(t)} g_{i,t}^- \quad (4)$$

We define  $\mathcal{B}$  as the total sum of values of optimal solutions for the cut instances  $(\mathcal{Q}_{U_j})_{j=1}^{\frac{1}{\mu}}$ . By utilizing Equation (4) we can bound  $\mathcal{B}$  as follows.

$$\begin{aligned} \mathcal{B} &\geq \sum_{j=1}^{\frac{1}{\mu}} \sum_{r=0}^{k_j-1} \left( \sum_{e \in \tilde{S}_{j,r}} v(e) + \sum_{t \in [u_r^j + 1, u_{r+1}^j - 1]: u_r^j \in U_j} \sum_{i \notin \tilde{S}_{j,r}(t-1) \cup \tilde{S}_{j,r}(t)} g_{i,t}^- \right) \\ &= \sum_{j=1}^{\frac{1}{\mu}} \sum_{e \in \tilde{S}_j} v(e) + \sum_{j=1}^{\frac{1}{\mu}} \sum_{t \in [2, T] \setminus U_j} \sum_{i \notin \tilde{S}_j(t-1) \cup \tilde{S}_j(t)} g_{i,t}^- \\ &= \sum_{e \in X} \sum_{j \in [\frac{1}{\mu}]: U_j(e) = \{e\}} v(e) + \sum_{e \in Y} \sum_{j=1}^{\frac{1}{\mu}} \sum_{e' = (i, t_1, t_2) \in U_j(e): t_2 - t_1 \geq \phi} v(e') \\ &\quad + \sum_{j=1}^{\frac{1}{\mu}} \sum_{t \in [2, T] \setminus U_j} \sum_{i \notin \tilde{S}_j(t-1) \cup \tilde{S}_j(t)} g_{i,t}^- \end{aligned} \quad (5)$$

We bound the value of each of the three terms separately by comparing it to the value of the optimal solution.

Consider the first term, value earned from short elements, i.e., elements  $e = (i, t_1, t_2) \in X$ . It holds that  $e \in \tilde{S}_j$  if and only if  $U_j(e) = \{e\}$  which means that  $(t_1, t_2] \cap U_j = \emptyset$ . Since for every  $j_1 \neq j_2$  it holds that  $U_{j_1} \cap U_{j_2} = \{1, T+1\}$  and since there are  $\frac{1}{\mu}$  sets of cut point, for each element  $e \in X$  it holds that  $e \in \tilde{S}_j$  for at least  $\frac{1}{\mu} - \frac{\phi}{\epsilon}$  values of  $j \in [\frac{1}{\mu}]$ . Thus,

$$\sum_{e \in X} \sum_{j \in [\frac{1}{\mu}]: U_j(e) = \{e\}} v(e) \geq \left( \frac{1}{\mu} - \frac{\phi}{\epsilon} \right) \sum_{e \in X} v(e) = \frac{1}{\mu} (1 - \epsilon) \sum_{e \in X} v(e) \quad (6)$$

Next, we bound the second term, the value earned from long elements,  $e \in Y$ . Consider the set of cut points  $U_j$ . Two operators are applied to each long element. First, it is cut and the subset  $U_j(e)$  is defined. Second, short elements are discarded from  $U_j(e)$ . The resulting subset is  $\{(i, t_1, t_2) \in U_j(e) \mid t_2 - t_1 \geq \phi\}$  and therefore we would like to bound the difference

$$\sum_{e \in Y} \left( v(e) - \sum_{e' \in \{(i, t_1, t_2) \in U_j(e) \mid t_2 - t_1 \geq \phi\}} v(e') \right)$$

Consider an element  $e = (i, t_1, t_2) \in Y$  cut by cut points set  $U_j$ . As shown in Equation (3), the loss caused by cutting  $e$  at cut point  $u \in (t_1, t_2]$  is independent of other cuts that are applied to  $e$  and is equal to  $\ell(e, u)$ . Thus we can consider each cut point separately.

As mentioned above, if  $e = (i, t_1, t_2) \in Y$ , the second operator discards elements  $e' \in U_j(e)$  that are short. Since the distance between each pair of cut points in  $U_j$  is at least  $\frac{1}{\mu} = \frac{\phi}{\epsilon^2} > \phi$ , each such short element  $e'$  is either  $(i, t_1, u)$  or  $(i, u, t_2)$  for some unique cut point  $u \in U_j$ . In addition, it must hold that either  $u - t_1 < \phi$  or  $t_2 - u < \phi$ . We associate the value lost by discarding  $e'$  to this unique cut point  $u$ .

## 15:12 General Knapsack Problems in a Dynamic Setting

Let  $e = (i, t_1, t_2) \in Y$  and  $u \in U_j$  be a cut point such that  $u \in (t_1, t_2]$ , i.e.,  $u$  cuts  $e$ . There are three cases to consider.

1. If  $u - t_1 < \phi$ , element  $e' = (i, t_1, u - 1) \in U_j(e)$  is discarded and a loss of  $v(e')$  is associated with  $u$  in addition to  $\ell(e, u)$ . Thus the total loss is at most

$$\begin{aligned} v(e') + \ell(e, u) &= \sum_{t=t_1}^{u-1} p_t(i) + \sum_{t=t_1+1}^{u-1} g_{i,t}^+ - c_{i,t_1}^+ - c_{i,u-1}^- + g_{i,u}^+ + c_{i,u}^+ + c_{i,u-1}^- \leq \\ &\leq \sum_{t=t_1}^{u-1} p_t(i) + \sum_{t=t_1+1}^u g_{i,t}^+ + c_{i,u-1}^+ \leq \sum_{t=t_1}^{u+\phi-1} p_t(i) + \sum_{t=t_1+1}^{u+\phi-1} g_{i,t}^+ \end{aligned}$$

where the equality is due to Equation (2) and the last inequality is due to the profit-cost ratio.

2. If  $t_2 - u < \phi$ , element  $e' = (i, u, t_2) \in U_j(e)$  is discarded and a loss of  $v(e')$  is associated with  $u$  in addition to  $\ell(e, u)$ . Thus the total loss is at most

$$\begin{aligned} v(e') + \ell(e, u) &= \sum_{t=u}^{t_2} p_t(i) + \sum_{t=u+1}^{t_2} g_{i,t}^+ - c_{i,u}^+ - c_{i,t_2}^- + g_{i,u}^+ + c_{i,u}^+ + c_{i,u-1}^- \leq \\ &\leq \sum_{t=u}^{t_2} p_t(i) + \sum_{t=u}^{t_2} g_{i,t}^+ + c_{i,u-1}^- \leq \sum_{t=u-\phi}^{t_2} p_t(i) + \sum_{t=u-\phi+1}^{t_2} g_{i,t}^+ \end{aligned}$$

where the equality is due to Equation (2) and the last inequality is due to the profit-cost ratio.

3. If  $t_2 - u \geq \phi$  and  $u - t_1 \geq \phi$ , no elements are discarded from  $U_j(e)$ . Thus the only loss is  $\ell(e, u)$  and can be bounded by

$$\ell(e, u) = g_{i,u}^+ + c_{i,u}^+ + c_{i,u-1}^- \leq \sum_{t=u-\phi}^{u+\phi-1} p_t(i) + \sum_{t=u-\phi+1}^{u+\phi-1} g_{i,t}^+$$

Overall we can bound the loss induced by cutting long elements at cut points  $U_j$  (due to loss  $\ell(e, u)$  and discarded short elements) by

$$\sum_{(i, t_1, t_2) \in Y} \sum_{u \in (t_1, t_2] \cap U_j} \left( \sum_{t=\max\{t_1, u-\phi\}}^{\min\{t_2, u+\phi-1\}} p_t(i) + \sum_{t=\max\{t_1+1, u-\phi+1\}}^{\min\{t_2, u+\phi-1\}} g_{i,t}^+ \right)$$

This means that the total value gained from elements that were originally in  $Y$  is

$$\begin{aligned} &\sum_{e \in Y} \sum_{j=1}^{\frac{1}{\mu}} \sum_{e'=(i, t_1, t_2) \in u(e, U_j): t_2 - t_1 \geq \phi} v(e') \geq \\ &\frac{1}{\mu} \sum_{e \in Y} v(e) - \sum_{j=1}^{\frac{1}{\mu}} \sum_{(i, t_1, t_2) \in Y} \sum_{u \in [t_1+1, t_2] \cap U_j} \left( \sum_{t=\max\{t_1, u-\phi\}}^{\min\{t_2, u+\phi-1\}} p_t(i) + \sum_{t=\max\{t_1+1, u-\phi+1\}}^{\min\{t_2, u+\phi-1\}} g_{i,t}^+ \right) \geq \\ &\frac{1}{\mu} \sum_{e \in Y} v(e) - \sum_{(i, t_1, t_2) \in Y} \sum_{u \in [t_1+1, t_2]} \left( \sum_{t=\max\{t_1, u-\phi\}}^{\min\{t_2, u+\phi-1\}} p_t(i) + \sum_{t=\max\{t_1+1, u-\phi+1\}}^{\min\{t_2, u+\phi-1\}} g_{i,t}^+ \right) \geq \\ &\frac{1}{\mu} \sum_{e \in Y} v(e) - 2\phi \sum_{(i, t_1, t_2) \in Y} \left( \sum_{t \in [t_1, t_2]} p_t(i) + \sum_{t \in [t_1+1, t_2]} g_{i,t}^+ \right) \end{aligned}$$

where the second inequality follows from the fact that for every  $j_1 \neq j_2$  it holds that  $U_{j_1} \cap U_{j_2} = \{1, T+1\}$ . The last inequality is due to the fact that the profit and gains of element  $(i, t_1, t_2)$  is lost at stage  $t \in [t_1, t_2]$  if it is cut by a cut point  $u$  such that  $|u - t| \leq \phi$ . Thus, its value is lost in at most  $2\phi$  instances. Due to the profit-cost ratio, for each long element  $e = (i, t_1, t_2) \in Y$  it holds that

$$c_{i,t_1}^+ + c_{i,t_2}^- \leq 2\phi \cdot \frac{\sum_{t=t_1}^{t_2} p_t(i) + \sum_{t=t_1+1}^{t_2} g_{i,t}^+}{t_2 - t_1} \leq \epsilon \cdot \sum_{t=t_1}^{t_2} p_t(i) + \epsilon \cdot \sum_{t=t_1+1}^{t_2} g_{i,t}^+$$

By substituting  $4\phi(c_{i,t_1}^+ + c_{i,t_2}^-) \leq 4\phi\epsilon \left( \sum_{t=t_1}^{t_2} p_t(i) + \sum_{t=t_1+1}^{t_2} g_{i,t}^+ \right)$  we get that

$$\begin{aligned} & \frac{1}{\mu} \sum_{e \in Y} v(e) - 2\phi \sum_{(i,t_1,t_2) \in Y} \left( \sum_{t=t_1}^{t_2} p_t(i) + \sum_{t=t_1+1}^{t_2} g_{i,t}^+ \right) = \\ & \sum_{(i,t_1,t_2) \in Y} \left( \frac{1}{\mu} \left( \sum_{t=t_1}^{t_2} p_t(i) + \sum_{t=t_1+1}^{t_2} g_{i,t}^+ - c_{i,t_1}^+ - c_{i,t_2}^- \right) - 2\phi \left( \sum_{t=t_1}^{t_2} p_t(i) + \sum_{t=t_1+1}^{t_2} g_{i,t}^+ \right) \right) \geq \\ & \sum_{(i,t_1,t_2) \in Y} \left( \left( \frac{1}{\mu} - 2\phi - 4\phi\epsilon \right) \left( \sum_{t=t_1}^{t_2} p_t(i) + \sum_{t=t_1+1}^{t_2} g_{i,t}^+ \right) - \left( \frac{1}{\mu} - 4\phi \right) (c_{i,t_1}^+ + c_{i,t_2}^-) \right) \geq \\ & \left( \frac{1}{\mu} - 2\phi - 4\phi\epsilon \right) \sum_{e \in Y} v(e) \geq \left( \frac{1}{\mu} - \frac{\phi}{\epsilon} \right) \sum_{e \in Y} v(e) \end{aligned}$$

where the last inequality follows the fact that  $\epsilon < \frac{1}{4}$ . Overall, we get that

$$\sum_{e \in Y} \sum_{j=1}^{\frac{1}{\mu}} v(e') \geq \left( \frac{1}{\mu} - \frac{\phi}{\epsilon} \right) \sum_{e \in Y} v(e) \quad (7)$$

Lastly, we bound the third term, gains  $g^-$  earned in all cut instance solutions. Consider a cut point set  $U_j$  and gain  $g_{i,t}^-$  earned in solution  $\tilde{S}^*$ , i.e.,  $i \notin S_{t-1}^* \cup S_t^*$ . Therefore, any element  $(i, t_1, t_2)$  such that  $t \in [t_1, t_2]$  or  $t-1 \in [t_1, t_2]$  is not in  $\tilde{S}_j$ . Thus, unless  $t \in U_j$ , gain  $g_{i,t}^-$  is earned in  $\tilde{S}_j$  and in some solution  $\tilde{S}_{j,r} \subseteq \tilde{S}_j$  such that  $t \in [u_r^j, u_{r+1}^j - 1]$  and  $u_r^j \in U_j$ . Again, we can use the fact that for every  $j_1 \neq j_2$  it holds that  $U_{j_1} \cap U_{j_2} = \{1, T+1\}$  and get that

$$\sum_{j=1}^{\frac{1}{\mu}} \sum_{t \in [2, T] \setminus U_j} \sum_{i \notin \tilde{S}_j(t-1) \cap \tilde{S}_j(t)} g_{i,t}^- \geq \left( \frac{1}{\mu} - 1 \right) \sum_{t=1}^T \sum_{i \notin \tilde{S}^*(t-1) \cap \tilde{S}^*(t)} g_{i,t}^- \quad (8)$$

By substituting inequalities (6),(7) and (8) in Inequality (5) we get that

$$\mathcal{B} \geq \left( \frac{1}{\mu} - \frac{\phi}{\epsilon} \right) \left( \sum_{e \in \tilde{S}^*} v(e) + \sum_{t \in [2, T]} \sum_{i \notin \tilde{S}^*(t-1) \cap \tilde{S}^*(t)} g_{i,t}^- \right) = \left( \frac{1}{\mu} - \frac{\phi}{\epsilon} \right) f_{\mathcal{Q}}((S_t^*)_{t=1}^T)$$

For each set of values, their average is smaller or equal to their maximum value. Thus there must exist at least one set of cut points  $U_{j^*}$  such that the sum of values of the solutions  $(\tilde{S}_{j^*,r}^*)_{r=0}^{k_j-1}$  for its cut instances,  $\mathcal{Q}_{U_{j^*}}$ , is at least  $\mu \cdot \mathcal{B}$ , the average value of a set of solutions for a set of cut instance  $\mathcal{Q}_{U_j}$  (for  $j = 1, \dots, \frac{1}{\mu}$ ). We get that

$$\sum_{e \in \tilde{S}_{j^*}} v(e) + \sum_{t \in [2, T]} \sum_{i \notin \tilde{S}_{j^*}(t-1) \cap \tilde{S}_{j^*}(t)} g_{i,t}^- \geq \mu \mathcal{B} = (1 - \epsilon) f_{\mathcal{Q}}((S_t^*)_{t=1}^T)$$

## 15:14 General Knapsack Problems in a Dynamic Setting

At iteration  $j^*$ , in which Algorithm 1 considers the cut instances  $\mathcal{Q}_{U_{j^*}}$ , algorithm  $A$  provides an approximate solution for each cut instance. Thus the value of the solution returned by  $A$  for the  $r$ -th cut instance is at least

$$\alpha \cdot \left( \sum_{e \in \tilde{S}_{j^*,r}} v(e) + \sum_{t \in [u_r^j+1, u_{r+1}^j-1]: u_r^j \in U_j, i \notin \tilde{S}_{j^*,r}^{(t-1)} \cup \tilde{S}_{j^*,r}^{(t)}} g_{i,t}^- \right)$$

Summing over all cut instances in  $\mathcal{Q}_{U_{j^*}}$  provides a solution with value at least

$$\alpha \cdot \left( \sum_{e \in \tilde{S}_{j^*}} v(e) + \sum_{t \in [2,T]} \sum_{i \notin \tilde{S}_{j^*}^{(t-1)} \cap \tilde{S}_{j^*}^{(t)}} g_{i,t}^- \right) \geq (1 - \epsilon) \cdot \alpha \cdot f_{\mathcal{Q}}((S_t^*)_{t=1}^T) \quad \blacktriangleleft$$

The correctness of Theorem 1 follows immediately from Theorem 17 and Lemma 10.

### 3 Hardness Results

In this section we present two hardness results for 1-GMK. First, we show no constant approximation ratio exists for 1-GMK (with unbounded profit-cost ratio), even if there is only one bin per stage. Then, we show that even if we wither down the model by removing the change costs, limiting the time horizon length to  $T = 2$ , and only having a single bin per stage, the problem still does not admit an EPTAS.

The above results are proved by showing an approximation preserving reduction from  $d$ -Dimensional Knapsack ( $d$ -KP) and Multidimensional Knapsack. For  $d \in \mathbb{N}$ , in  $d$ -KP we are given a set of items  $I$ , each equipped with a profit  $p_i$ , as well as a  $d$ -dimensional weight vector  $\bar{w}^i \in [0, 1]^d$ . We denote  $j$ -th coordinate of  $\bar{w}^i$  by  $\bar{w}_j^i$ . In addition, we are given a single bin equipped with a  $d$ -dimensional capacity vector  $\bar{W} \in \mathbb{R}_{\geq 0}^d$ . A subset  $S \subseteq I$  is a feasible solution if  $\sum_{i \in S} \bar{w}^i \leq \bar{W}$ . The objective is to find a feasible solution  $S$  which maximizes  $\sum_{i \in S} p_i$ .

The Multidimensional Knapsack problem is the generalization of  $d$ -KP in which  $d$  is not fixed. That is, the input for the problem is a  $d$ -KP instance for some  $d \in \mathbb{N}$ . The solutions and their values are the solution and values of the  $d$ -KP instance.

Note that  $d$ -KP is a special case of  $d$ -MKCP, where the set of MKCs is  $\mathcal{K} = (K_j)_{j=1}^d$ . The  $j$ -th MKC is  $K_j = (w_j, B_j, W_j)$ , where  $w_j(i) = \bar{w}_j^i$ ,  $B_j = \{b\}$  and  $W_j(b) = \bar{W}_j$ , where  $\bar{W}_j$  is the  $j$ -th coordinate of the capacity vector  $\bar{W}$ . Finally, the profit function  $p: I \rightarrow \mathbb{R}_{\geq 0}$  is defined as  $p(i) = p_i$  for any  $i \in I$ . For simplicity we will use this notation for  $d$ -KP and Multidimensional Knapsack throughout this section.

► **Lemma 18.** *There is an approximation preserving reduction from the Multidimensional Knapsack problem to 1-GMK with a single bin in each stage.*

**Proof.** Let  $\mathcal{Q} = (I, \mathcal{K}, p)$  be an instance of Multidimensional Knapsack, where  $\mathcal{K} = (K_j)_{j=1}^d$ . We define an instance of 1-GMK as follows. Define  $T = d$ , and for  $j = 1, \dots, d$  define  $\mathcal{P}_j = (I, (K_j), h)$  with  $h(i) = \frac{p(i)}{d}$  for all  $i \in I$ . The gains vectors are defined as zero vectors,  $g^+ = g^- = \vec{0}$ . Finally, we define the change cost vectors. For all  $i \in I$  we set

$$c_{i,t}^+ = \begin{cases} p(i) & t \in [2, d] \\ 0 & \text{otherwise.} \end{cases}, \quad c_{i,t}^- = \begin{cases} p(i) & t \in [1, d-1] \\ 0 & \text{otherwise.} \end{cases}$$

The tuple  $\tilde{\mathcal{Q}} = ((\mathcal{P}_t)_{t=1}^d, g^+, g^-, c^+, c^-)$  is a 1-GMK instance with time horizon  $T = d$ .

Let  $(S, \mathcal{A})$  be a feasible solution for  $\mathcal{Q}$ , where  $\mathcal{A} = (A_j)_{j=1}^d$ . We can easily construct a solution for  $\tilde{\mathcal{Q}}$  by setting  $\mathcal{A}_j = (A_j)$  for  $j = 1, \dots, d$ . Then,  $(S_t, \mathcal{A}_t)_{t=1}^d$ , where  $S_j = S$  for  $j = 1, \dots, d$ , is a solution for  $\tilde{\mathcal{Q}}$ . Note that all items are either assigned or not assigned in all stages. Thus the value of the solution is

$$f_{\tilde{\mathcal{Q}}}((S_t)_{t=1}^d) = \sum_{t=1}^d h(S_t) - \sum_{t=1}^T \left( \sum_{i \in S_t \setminus S_{t+1}} c_{i,t}^- + \sum_{i \in S_t \setminus S_{t-1}} c_{i,t}^+ \right) = d \cdot h(S) = p(S)$$

The solution is also feasible as for every  $j \in [d]$  it holds that  $A_j$  is a feasible assignments for MKC  $K_j$ .

Next, let  $(S_t, \mathcal{A}_t)_{t=1}^d$  be a feasible solution for  $\tilde{\mathcal{Q}}$ , where  $\mathcal{A}_j = (\tilde{A}_j)$  for  $j = 1, \dots, d$ . For  $j = 1, \dots, d$  let  $B_j = \{b_j\}$ . We define the selected items set as  $S = \bigcap_{j \in [d]} S_j$  and define the assignments accordingly,  $A_j(b_j) = S$  for  $j = 1, \dots, d$  and  $\mathcal{A} = (A_j)_{j=1}^d$ . Consider some  $j \in [s]$ , the assignment  $A_j$  is feasible as  $A_j(b_j) \subseteq \tilde{A}_j(b_j)$  and

$$\sum_{i \in A_j(b_j)} w_j(i) \leq \sum_{i \in \tilde{A}_j(b_j)} w_j(i) \leq W_j(b_j)$$

The value of the solution is

$$p(S) = d \cdot h(S) \geq \sum_{t=1}^d h(S_t) - \sum_{t=1}^T \left( \sum_{i \in S_t \setminus S_{t+1}} c_{i,t}^- + \sum_{i \in S_t \setminus S_{t-1}} c_{i,t}^+ \right) = f_{\tilde{\mathcal{Q}}}((S_t)_{t=1}^d)$$

where the inequality follows from the construction of  $\tilde{\mathcal{Q}}$ . Furthermore, note that  $S$  can be constructed in polynomial time, which concludes the proof.  $\blacktriangleleft$

In [6] Chekuri and Kahanna showed that Multidimensional Knapsack does not admit any constant approximation ratio unless  $NP = ZPP$ . Theorem 3 follows immediately from the hardness result of [6] and Lemma 18.

We now proceed to the second hardness result.

**► Lemma 19.** *There is an approximation preserving reduction from 2-Dimensional Knapsack problem to 1-GMK with time horizon  $T = 2$ , no change costs and a single bin per stage.*

**Proof.** Let  $\mathcal{Q} = (I, \mathcal{K}, p)$  be an instance of 2-dimensional knapsack, where  $\mathcal{K} = (K_1, K_2)$ . Also, since  $p$  is modular, it holds that  $p(S) = \sum_{i \in S} p_i$ .

We define an instance of 1-GMK as follows. Set  $T = 2$ ,  $\mathcal{P}_1 = (I, (K_1), h)$  and  $\mathcal{P}_2 = (I, (K_2), h)$ , where  $h$  is the zero function, i.e.,  $h : I \rightarrow \{0\}$  such that  $\forall i \in I$  it holds that  $h(i) = 0$ . Since there are only two stages, gains exists only for stage for  $t = 2$ . Set  $g_{i,2}^+ = p(i)$  and  $g_{i,2}^- = 0$  for each item  $i \in I$ . Finally, we set the change cost vectors as  $c^+ = c^- = \vec{0}$ . The tuple  $\tilde{\mathcal{Q}} = ((\mathcal{P}_1, \mathcal{P}_2), g^+, g^-, c^+, c^-)$  is a 1-GMK instance with time horizon  $T = 2$ . Note that since all profits, change costs and gains  $g^-$  are zero we can write the objective function as

$$f_{\tilde{\mathcal{Q}}}((S_1, S_2)) = \sum_{i \in S_1 \cap S_2} g_{i,2}^+.$$

Let  $(S, \mathcal{A})$  be a feasible solution for  $\mathcal{Q}$ , where  $\mathcal{A} = (A_1, A_2)$ . We can easily construct a solution for the  $\tilde{\mathcal{Q}}$  by setting  $\mathcal{A}_1 = (A_1)$  and  $\mathcal{A}_2 = (A_2)$ . Then,  $(S_t, \mathcal{A}_t)_{t=1}^2$ , where  $S_1 = S_2 = S$ , is a solution for  $\tilde{\mathcal{Q}}$ . Note that all items are either assigned in both stages or not assigned in both stages. Thus the value of the solution is

$$f_{\tilde{\mathcal{Q}}}((S, S)) = \sum_{i \in S_1 \cap S_2} g_{i,2}^+ = \sum_{i \in S} g_{i,2}^+ = \sum_{i \in S} p_i = p(S)$$

## 15:16 General Knapsack Problems in a Dynamic Setting

The solution is also feasible as  $A_1$  and  $A_2$  are feasible assignments of  $K_1$  and  $K_2$  (respectively).

Next, let  $(S_t, \mathcal{A}_t)_{t=1}^2$  be a feasible solution for  $\tilde{\mathcal{Q}}$ , where  $\mathcal{A}_1 = (\tilde{A}_1)$  and  $\mathcal{A}_2 = (\tilde{A}_2)$ . Let  $B_1 = \{b_1\}$  and  $B_2 = \{b_2\}$ . We define the selected items set as  $S = S_1 \cap S_2$  and define the assignments accordingly,  $A_1(b_1) = A_2(b_2) = S$  and  $\mathcal{A} = (A_1, A_2)$ . Assignment  $A_1$  is feasible as  $A_1(b_1) \subseteq \tilde{A}_1(b_1)$  and

$$\sum_{i \in A_1(b_1)} w_1(i) \leq \sum_{i \in \tilde{A}_1(b_1)} w_1(i) \leq W_1(b_1)$$

A similar statement shows that assignment  $A_2$  is feasible as well. The value of the solution is

$$p(S) = \sum_{i \in S} p(i) = \sum_{i \in S} g_{i,2}^+ = \sum_{i \in S_1 \cap S_2} g_{i,2}^+ = f_{\tilde{\mathcal{Q}}}((S, S))$$

which concludes the proof. ◀

In [19] Kulik and Shachnai showed that there is no EPTAS for 2-KP unless  $W[1] = FPT$ . Theorem 2 follows immediately from the hardness result of [19] and Lemma 19.

---

### References

- 1 Hyung-Chan An, Ashkan Norouzi-Fard, and Ola Svensson. Dynamic facility location via exponential clocks. *ACM Transactions on Algorithms (TALG)*, 13(2):1–20, 2017.
- 2 Evripidis Bampis, Bruno Escoffier, Michael Lampis, and Vangelis Th Paschos. Multistage matchings. In *16th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2018)*, volume 101, pages 7–1, 2018.
- 3 Evripidis Bampis, Bruno Escoffier, Kevin Schewior, and Alexandre Teiller. Online multistage subset maximization problems. In *European Symposium on Algorithms (ESA)*, volume 144. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019.
- 4 Evripidis Bampis, Bruno Escoffier, and Alexandre Teiller. Multistage knapsack. In *Mathematical Foundations of Computer Science (MFCS)*, volume 138, pages 22–1. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019.
- 5 Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a submodular set function subject to a matroid constraint. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 182–196. Springer, 2007.
- 6 Chandra Chekuri and Sanjeev Khanna. On multidimensional packing problems. *SIAM journal on computing*, 33(4):837–851, 2004.
- 7 Markus Chimani, Niklas Troost, and Tilo Wiedera. Approximating multistage matching problems. *arXiv preprint arXiv:2002.06887*, 2020.
- 8 Shichuan Deng, Jian Li, and Yuval Rabani. Approximation algorithms for clustering with dynamic points. In *28th Annual European Symposium on Algorithms (ESA 2020)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020.
- 9 David Eisenstat, Claire Mathieu, and Nicolas Schabanel. Facility location in evolving metrics. In *International Colloquium on Automata, Languages, and Programming*, pages 459–470. Springer, 2014.
- 10 Yaron Fairstein, Ariel Kulik, Danny Raz, et al. General knapsack problems in a dynamic setting. *arXiv preprint arXiv:2105.00882*, 2021.
- 11 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 2021)*, 2021. (To appear).
- 12 Yaron Fairstein, Seffi Joseph Naor, and Danny Raz. Algorithms for dynamic nfv workload. In *International Workshop on Approximation and Online Algorithms*, pages 238–258. Springer, 2018.

- 13 Uriel Feige. A threshold of  $\ln n$  for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.
- 14 Uriel Feige and Michel Goemans. Approximating the value of two power proof systems, with applications to max 2sat and max dicut. In *Proceedings Third Israel Symposium on the Theory of Computing and Systems*, pages 182–189. IEEE, 1995.
- 15 Till Fluschnik, Rolf Niedermeier, Valentin Rohm, and Philipp Zschoche. Multistage vertex cover. In *14th International Symposium on Parameterized and Exact Computation (IPEC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.
- 16 Till Fluschnik, Rolf Niedermeier, Carsten Schubert, and Philipp Zschoche. Multistage st path: Confronting similarity with dissimilarity in temporal graphs. In *31st International Symposium on Algorithms and Computation (ISAAC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
- 17 Fabrizio Grandoni and Rico Zenklusen. Approximation schemes for multi-budgeted independence systems. In *European Symposium on Algorithms*, pages 536–548. Springer, 2010.
- 18 Anupam Gupta, Kunal Talwar, and Udi Wieder. Changing bases: Multistage optimization for matroids and matchings. In *International Colloquium on Automata, Languages, and Programming*, pages 563–575. Springer, 2014.
- 19 Ariel Kulik and Hadas Shachnai. There is no eptas for two-dimensional knapsack. *Information Processing Letters*, 110(16):707–710, 2010.
- 20 George L Nemhauser and Laurence A Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operations research*, 3(3):177–188, 1978.
- 21 Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.
- 22 Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

## A Omitted Proofs and Definition

► **Definition 20.** Let  $\Pi_1, \Pi_2$  be two maximization problems. An approximation factor preserving reduction from  $\Pi_1$  to  $\Pi_2$  consists of two polynomial time algorithms  $f, g$  such that for any two instances  $I_1$  of problem  $\Pi_1$  and  $I_2 = f(I_1)$  of problem  $\Pi_2$  it holds that

- $I_2 \in \Pi_2$  and  $OPT_{\Pi_2}(I_2) \geq OPT_{\Pi_1}(I_1)$ .
- for any solution  $s_2$  for  $I_2$ , solution  $s_1 = g(I_1, s_2)$  is a solution for  $I_1$  and  $obj_{\Pi_1}(I_1, s_1) \geq obj_{\Pi_2}(I_2, s_2)$ .

where  $OPT_{\Pi}(I)$  is the value of an optimal solution for instance  $I$  of problem  $\Pi$ , and  $obj_{\Pi}(I, s)$  is the value of solution  $s$  for instance  $I$  of problem  $\Pi$ .

**Proof of Lemma 8.** Let  $\mathcal{Q} = ((\mathcal{P}_t)_{t=1}^T, g^+, g^-, c^+, c^-)$  be an instance of  $d$ -GMK, where  $\mathcal{P}_t = (I, \mathcal{K}_t, p_t)$  and  $\mathcal{K}_t = (K_{t,j})_{j=1}^{d_t}$ . Also, let  $R(\mathcal{Q}) = (E, \tilde{\mathcal{K}}, \tilde{p}, \mathcal{I})$  be the reduced instance of  $\mathcal{Q}$ . and  $(S, (\tilde{A}_{t,j})_{t \in [T], j \in [d]})$  be a feasible solution for  $R(\mathcal{Q})$ . We define the solution  $(S_t, \mathcal{A}_t)_{t=1}^T$  for  $\mathcal{Q}$  as follows. For every stage  $t$  we set  $S_t = \{i \in I \mid \exists (i, D) \in S : t \in D\}$ . For every  $t = 1, \dots, T$ ,  $j = 1, \dots, d_t$  and bin  $b \in B_{t,j}$  (the set of bins in MKC  $K_{t,j}$ ) let  $A_{t,j}(b) = \{i \in I \mid \exists (i, D) \in \tilde{A}_{t,j}(b) : t \in D\}$ . Observe that the sets  $(S_t)_{t=1}^T$  and assignments  $(A_{t,j})_{t \in [T], j \in [d_t]}$  can be constructed in polynomial time as at most  $|I|$  elements can be chosen due to the matroid constraint. The assignment  $A_{t,j}$  is an assignment of  $S_t$  since

$$S_t = \{i \in I \mid \exists (i, D) \in S : t \in D\} = \bigcup_{b \in B_{t,j}} \{i \in I \mid (i, D) \in \tilde{A}_{t,j}(b) : t \in D\} = \bigcup_{b \in B_{t,j}} A_{t,j}(b),$$

where the second equality follows the feasibility of the solution for  $R(\mathcal{Q})$ . In addition,  $A_{t,j}$  is

## 15:18 General Knapsack Problems in a Dynamic Setting

a feasible assignment for MKC  $K_{t,j}$  since for every bin  $b \in B_{t,j}$  it holds that

$$\sum_{i \in A_{t,j}(b)} w_{t,j}(i) = \sum_{(i,D) \in \tilde{A}_{t,j}(b): t \in D} \tilde{w}_{t,j}((i,D)) = \sum_{(i,D) \in \tilde{A}_{t,j}(b)} \tilde{w}_{t,j}((i,D)) \leq W_{t,j}(b)$$

Thus  $(S_t, \mathcal{A}_t)_{t=1}^T$  is a feasible solution for  $\mathcal{Q}$ .

Lastly, consider the value of the solution for  $\mathcal{Q}$ . It holds that

$$\begin{aligned} f_{\mathcal{Q}}((S_t)_{t=1}^T) &= \\ &= \sum_{t=1}^T \sum_{i \in S_t} p_t(i) + \sum_{t=2}^T \left( \sum_{i \in S_{t-1} \cap S_t} g_{i,t}^+ + \sum_{i \notin S_{t-1} \cup S_t} g_{i,t}^- \right) - \sum_{t=1}^T \left( \sum_{i \in S_t \setminus S_{t-1}} c_{i,t}^+ + \sum_{i \in S_t \setminus S_{t+1}} c_{i,t}^- \right) = \\ &= \sum_{(i,D) \in S} \left( \sum_{t \in D} p_t(i) + \sum_{t \in D: t-1 \in D} g_{i,t}^+ + \sum_{t \notin D: t-1 \notin D} g_{i,t}^- - \sum_{t \in D: t-1 \notin D} c_{i,t}^+ - \sum_{t \in D: t+1 \notin D} c_{i,t}^- \right) = \\ &= \tilde{p}(S) \end{aligned}$$

◀

**Proof of Corollary 13.** Let  $\mathcal{Q}$  be an instance of  $d$ -GMK,  $U$  be a set of cut points. Also, let  $\mathcal{Q}_U = (q_j)_{j=0}^{k-1} = \left( (\mathcal{P}_t)_{t=u_j}^{u_{j+1}-1}, g^+, g^-, c^+, c^- \right)_{j=0}^{k-1}$  be the corresponding tuple of cut instances, and  $\left( (S_t, \mathcal{A}_t)_{t=u_j}^{u_{j+1}-1} \right)_{j=0}^{k-1}$  be a tuple of feasible solutions for the cut instances.

We define the solution  $(S_t, \mathcal{A}_t)_{t=1}^T$  for  $\mathcal{Q}$ . It is easy to see that the assignments  $\mathcal{A}_t$  to  $\mathcal{K}_t$  are all feasible assignments of  $S_t$ . In addition, it holds that

$$\begin{aligned} f_{\mathcal{Q}}((S_t)_{t=1}^T) &= \\ &= \sum_{t=1}^T p_t(S_t) + \sum_{t=2}^T \left( \sum_{i \in S_{t-1} \cap S_t} g_{i,t}^+ + \sum_{i \notin S_{t-1} \cup S_t} g_{i,t}^- \right) - \sum_{t=1}^T \left( \sum_{i \in S_t \setminus S_{t+1}} c_{i,t}^- + \sum_{i \in S_t \setminus S_{t-1}} c_{i,t}^+ \right) \geq \\ &= \sum_{j=0}^{k-1} \left( \sum_{t=u_j}^{u_{j+1}-1} p_t(S_t) + \sum_{t=u_{j+1}}^{u_{j+1}-1} \left( \sum_{i \in S_{t-1} \cap S_t} g_{i,t}^+ + \sum_{i \notin S_{t-1} \cup S_t} g_{i,t}^- \right) \right. \\ &\quad \left. - \sum_{t=u_{j+1}}^{u_{j+1}-1} \left( \sum_{i \in S_t \setminus S_{t+1}} c_{i,t}^- + \sum_{i \in S_t \setminus S_{t-1}} c_{i,t}^+ \right) - \sum_{i \in S_{u_{j+1}-1}} c_{i,u_{j+1}}^- - \sum_{i \in S_{u_j}} c_{i,u_j}^+ \right) = \\ &= \sum_{j=0}^{k-1} f_{q_j} \left( (S_t)_{t=u_j}^{u_{j+1}-1} \right) \end{aligned}$$

where  $f_{q_j}$  is the objective functions of cut instance  $q_j$ . This proves that a cut solution has a higher value than the sum of solutions for cut instance from which it was created. ◀



# Min-Sum Clustering (With Outliers)

**Sandip Banerjee** ✉

The Hebrew University of Jerusalem, Israel

**Rafail Ostrovsky** ✉

University of California, Los Angeles, CA, USA

**Yuval Rabani** ✉

The Hebrew University of Jerusalem, Israel

---

## Abstract

We give a constant factor polynomial time pseudo-approximation algorithm for min-sum clustering with or without outliers. The algorithm is allowed to exclude an arbitrarily small constant fraction of the points. For instance, we show how to compute a solution that clusters 98% of the input data points and pays no more than a constant factor times the optimal solution that clusters 99% of the input data points. More generally, we give the following bicriteria approximation: For any  $\epsilon > 0$ , for any instance with  $n$  input points and for any positive integer  $n' \leq n$ , we compute in polynomial time a clustering of at least  $(1 - \epsilon)n'$  points of cost at most a constant factor greater than the optimal cost of clustering  $n'$  points. The approximation guarantee grows with  $\frac{1}{\epsilon}$ . Our results apply to instances of points in real space endowed with squared Euclidean distance, as well as to points in a metric space, where the number of clusters, and also the dimension if relevant, is arbitrary (part of the input, not an absolute constant).

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Facility location and clustering

**Keywords and phrases** Clustering, approximation algorithms, primal-dual

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.16

**Category** APPROX

**Funding** *Sandip Banerjee*: Research supported in part by Yuval Rabani's NSFC-ISF grant 2553-17. *Rafail Ostrovsky*: Supported in part by DARPA under Cooperative Agreement No: HR0011-20-2-0025, NSF Grant CNS-2001096, US-Israel BSF grant 2015782, Google Faculty Award, JP Morgan Faculty Award, IBM Faculty Research Award, Xerox Faculty Research Award, OKAWA Foundation Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, the Department of Defense, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes not withstanding any copyright annotation therein.

*Yuval Rabani*: Research supported in part by NSFC-ISF grant 2553-17 and by NSF-BSF grant 2018687.

## 1 Introduction

We consider min-sum  $k$ -clustering. This is the problem of partitioning an input dataset of  $n$  points into  $k$  clusters with the objective of minimizing the sum of intra-cluster pairwise distances. We consider primarily the prevalent setting of instances of points in  $\mathbb{R}^d$  endowed with a distance function equal to the squared Euclidean distance (henceforth referred to as the  $\ell_2^2$  case). Our results apply also to the case of instances of points endowed with an explicit metric (henceforth referred to as the metric case). Note that we consider  $k$  (and  $d$ , if relevant) to be part of the input, rather than an absolute constant. In these and similar cases we give polynomial time approximation algorithms that cluster all but a negligible constant fraction



© Sandip Banerjee, Rafail Ostrovsky, and Yuval Rabani;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 16; pp. 16:1–16:16



Leibniz International Proceedings in Informatics

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

## 16:2 Min-Sum Clustering (With Outliers)

of outliers at a cost that is at most a constant factor larger than the optimum clustering. More specifically, for any  $\epsilon > 0$ , if the optimum we compete against is required to cluster any number  $n' \leq n$  of points, our algorithm clusters at least  $(1 - \epsilon)n'$  points and at most  $n'$  points, and pays a constant factor more than the optimum for  $n'$  points. The constant depends on  $\epsilon$ .

Clustering in general is a fundamental question in unsupervised learning. The question originated in the social sciences and now is widely applicable in data analysis and machine learning, in areas including bioinformatics, computer vision, pattern recognition, signal processing, fraud/spam/fake news filtering, and market/population segmentation. Clustering is also a list of fundamental discrete optimization problems in computational geometry that have been studied for decades by theoreticians, in particular (but not exclusively) as simple non-convex targets of machine learning. Some clustering problems, notably centroid-based criteria such as  $k$ -means, have been studied extensively. We currently have a fairly tight analysis of their complexity in the worst case (e.g. [4, 42, 1, 20]) and under a wide range of restrictive conditions: low dimension (e.g., [33, 21, 30]) fixed  $k$  (e.g. [41, 27, 17, 26]), various notions of stability (e.g. [44, 7, 40, 5, 22]), restrictive models of computation (e.g., [16, 3, 15, 14]), etc., as well as practically appealing heuristics (e.g., Lloyd's iteration, local search) and supportive theoretical justification (e.g., some of the afore-mentioned papers and also [6, 38]). Theoretical understanding of density-based clustering criteria, and in particular min-sum clustering, is far less developed. There are clearly cases in practice where, for instance, min-sum clustering coincides far better with the intuitive clustering objective than standard centroid-based criteria. A simple illustrative example is the case of separating two concentric dense rings of points in the plane. Moreover, min-sum clustering satisfies Kleinberg's consistency axiom, whereas a fairly large class of centroid-based criteria including  $k$ -means and  $k$ -median do not satisfy this axiom [39, 47]. However, the state-of-the-art for computing min-sum clustering remains inferior to alternatives. Min-sum  $k$ -clustering is NP-hard in the  $\ell_2^2$  case (e.g., using arguments from [4]), and also for the metric case (see [29]), even for  $k = 2$ . In the  $\ell_2^2$  case, it can be solved in polynomial time if both  $d$  and  $k$  are absolute constants [35]. In the metric case with arbitrary  $k$ , approximating min-sum clustering to within a factor better than 1.415 is NP-hard [31, 19]. If  $k$  is a fixed constant, the problem admits a PTAS, both in the  $\ell_2^2$  case and in the metric case [28]; see also [32, 36, 43, 45] for previous work in this vein. Hassin and Or [34] gave a 2-factor approximation algorithm for the metric penalized  $k$ -min-sum problem where  $k$  is a constant. If  $k = o(\log n / \log \log n)$ , then there is a constant factor approximation algorithm for the  $\ell_2^2$  case [23]. In the metric case, assuming that  $k = o(\log n / \log \log n)$  and the instance satisfies a certain clusterability/stability condition, a partition close to optimal can be computed in polynomial time [8, 9] (see also [46] for some applications and experimental results in this vein). We note that practical applications often require the number of clusters i.e.,  $k$  too many, so the above restrictions on  $k$  are unrealistic in those cases. In the worst case, and under no restrictions on the instance, the best known approximation guarantee known is an  $O(\log n)$  approximation algorithm [13] for the metric case. This improves upon a slightly worse and much earlier guarantee [12]. In both papers, the factor is derived from representing the input metric space approximately as a convex combination of hierarchically separated tree (HST) metrics [10, 11, 25]. This incurs logarithmic distortion, which is asymptotically tight in the worst case. In HST metrics, min-sum clustering can be approximated to within a constant factor. Thus, a fundamental challenge of the study of min-sum clustering is to eliminate the gap between the hardness of approximation lower bound of 1.415 and the approximation guarantee upper bound of  $O(\log n)$ . We show that a constant factor approximation is possible, if one is willing to regard as outliers a small fraction of the input dataset. For the  $\ell_2^2$  case, we are not aware of any previous non-trivial guarantee for  $k \gg \log n / \log \log n$ .

Our results are derived using a reduction from min-sum clustering to a centroid-based criterion with (soft) capacity constraints. This can be done exactly in the  $\ell_2^2$  case, and approximately in the metric case, though to get polynomial time algorithms we use an approximation in both cases. This reduction underlies also some of the above-mentioned previous results on min-sum clustering. The outcome of this reduction is a  $k$ -median or  $k$ -means problem with non-uniform capacities. If we are aiming for a constant factor approximation then we can afford to violate the capacities by a constant factor. There are nice results on approximating  $k$ -median with non-uniform capacities, for instance [24]. Unfortunately, these results do not seem applicable here, because their input is a metric space. The reduction, even for the metric case, does not generate a metric instance of capacitated  $k$ -median (the triangle inequality is violated unboundedly). Nevertheless, we do draw some ideas from this literature. Our min-sum clustering algorithm is based on the well-trodden path of using the primal-dual schema repeatedly to search for a good Lagrange multiplier in lagrangian relaxation of the problem (see [37] for the origin of this method). The dual program has a variable for every data point, and a constraint for every possible cluster. The dual ascent process requires detection of constraints that become tight. In our case, this is a non-trivial problem, which we solve only approximately. As usual, the dual values are used to “buy” the opening of the clusters that become tight, and we have to contend with points contributing simultaneously to multiple clusters. This is done, as usual, by creating a conflict graph among the tight clusters and choosing an independent set in this graph. However, in our case there are unusual complications. The connection cost is a distance (not a metric in the  $\ell_2^2$  case, but this is a minor concern) multiplied by the cardinality of the cluster. If there is a conflict between a large cluster and some small clusters, we have the following dilemma. If we open the large cluster, the unclustered points in the small clusters may lack dual “money” to connect to the large cluster; they can only afford the distance multiplied by the cardinality of their small cluster. If, on the other hand, we open (some or all of) the small clusters, assigning the unclustered points in the large cluster to those small clusters might inflate their cardinality by a super-constant factor, leaving all points with insufficient funds to connect to the inflated clusters.

We resolve this dilemma as follows (using in part some ideas from [18]). We open larger clusters first, so if a cluster is not opened, it is smaller than the conflicting cluster that blocked it. Unclustered points are not assigned to the blocking cluster, but rather aggregated around each blocking cluster to form their own clusters of appropriate cardinality. We use approximate cardinality, in scales which are powers of a constant  $b$ . As we require the Lagrange multiplier preserving (LMP) property, we must have sufficient “funds” to pay the opening costs in full (but can settle for paying just a fraction of the connection cost). This is possible if in a scale of, say,  $b^p$  we have, say, at least  $b^{2+p}$  unclustered points in clusters of scaled cardinality  $b^p$  (each set of roughly this size can afford to open its own cluster). If a blocking cluster is blocking fewer points in this scale, we can’t afford to cluster them and must discard them as outliers. This is the primary source of the excess outliers.

As usual, the search for a good Lagrange multiplier may end with two integer primal solutions, one with fewer than  $k$  clusters and one with more than  $k$  clusters, whose convex combination is a feasible fractional bipoint solution to the  $k$ -clustering problem. In our case, as we already may have to give up on some outliers, we can simply output either the  $< k$  solution or the  $k$  largest clusters in the  $> k$  solution. We point out that these extra outliers can be avoided by using a more sophisticated “rounding” of the bipoint solution, but given our loss in the primal-dual phase, it would not improve meaningfully our guarantees.

The above description sums up the algorithm in the case that  $n' = n$ . Our result also extends to the case that the optimal solution is also allowed to discard some outliers (we may have to discard some more). The main additional issue in the case  $n' < n$  is that in the primal-dual phase we may open a cluster that brings the number of clustered points from below  $n'$  to above  $n'$ . In this case, some points in this last cluster need to be discarded, but then the remaining clustered points might have insufficient “funds” to open the last cluster. If we have many clusters, we can afford to eliminate the smallest cluster, declaring its points as outliers, and use the dual values of the points in that cluster to pay for opening the last one. If there is a small number of clusters, we may assume that the primal-dual phase opened less than  $k$  clusters (to ensure this property, if  $k$  is a small constant, we employ the known PTAS; thus we can assume that  $k$  is large). Our approach in this case draws from [2], where a similar issue is addressed in the case of the sum-of-radii  $k$ -clustering problem. Though the questions are quite different, we use a similar idea of computing a (slightly) non-Lagrange multiplier preserving approximation to the lagrangian relaxation. The LMP property is regularly used in the argument that the bipoint solution is both feasible and cheap; the approach we adopt requires an extra argument to bound the cost of a bipoint solution that incorporates a non-LMP solution.

The rest of the paper is organized as follows. Section 2 introduces some basic definitions and claims. Section 3 describes the algorithm. Section 4 analyzes the algorithm. For conciseness, the paper presents the  $\ell_2^2$  case. Our main result is Theorem 8. The metric case is essentially identical, and is briefly explained in Theorem 9. We note that we made no effort to optimize the constant factor guarantees, throughout the paper.

## 2 Definitions and Preliminary Claims

Consider an instance of min-sum clustering that is defined by a set of points  $X \subset \mathbb{R}^d$  and a target number of clusters  $k \in \mathbb{N}$ . Let  $n = |X|$ . The cost of a cluster  $Y \subset X$  is

$$\text{cost}(Y) = \frac{1}{2} \cdot \sum_{x,y \in Y} \|x - y\|_2^2.$$

The center of mass (or mean) of  $Y$  is  $\text{cm}(Y) = \frac{1}{|Y|} \cdot \sum_{x \in Y} x$ . The following proposition is a well-known fact (for instance, see [35]).

► **Proposition 1.** *The following assertions hold for every finite set  $Y \subset \mathbb{R}^d$ .*

1. *The center of mass  $\text{cm}(Y)$  is the unique minimizer of  $\sum_{x \in Y} \|x - y\|_2^2$  over  $y \in \mathbb{R}^d$ .*
2.  *$\text{cost}(Y) = |Y| \cdot \sum_{x \in Y} \|x - \text{cm}(Y)\|_2^2$ .*

A *min-sum  $k$ -clustering* of  $X$  is a partition of  $X$  into  $k$  disjoint subsets  $X_1, X_2, \dots, X_k$  that minimizes over all possible partitions

$$\sum_{i=1}^k \text{cost}(X_i) = \sum_{i=1}^k |X_i| \cdot \sum_{x \in X_i} \|x - \text{cm}(X_i)\|_2^2.$$

In the version allowing outliers, we are given a target  $n' \leq n$  of the number of points to cluster, and we are required that  $|\bigcup_{i=1}^k X_i| \geq n'$ . Clearly, the version without outliers is a special case of the version with outliers with  $n' = n$ . Let  $\text{opt}(X, n', k)$  denote the optimal min-sum cost of clustering  $n'$  points in  $X$  into  $k$  clusters. Formally, we can express the goal

as a problem of optimizing an exponential size integer program:

$$\begin{aligned}
& \text{minimize} && \sum_{Y \subset X} \text{cost}(Y) \cdot z_Y \\
& \text{s.t.} && \sum_{Y \ni x} z_Y + w_x \geq 1 \quad \forall x \in X \\
& && \sum_{Y \subset X} z_Y \leq k \\
& && \sum_{x \in X} w_x \leq n - n' \\
& && z_Y \in \{0, 1\} \quad \forall Y \subset X \\
& && w_x \in \{0, 1\} \quad \forall x \in X.
\end{aligned} \tag{1}$$

Fix  $b \in \mathbb{N}$ ,  $b > 1$ . For  $i \in \mathbb{N}$ , let  $\text{floor}_b(i) = b^{\lfloor \log_b i \rfloor}$ . For  $Y \subset X$ , let  $\text{ctr}(Y)$  be a reference point that we set for now as  $\text{ctr}(Y) = \text{cm}(Y)$ . Define

$$\text{cost}_b(Y) = \text{floor}_b(|Y|) \cdot \sum_{y \in Y} \|y - \text{ctr}(Y)\|_2^2.$$

In other words, (assuming  $\text{ctr}(Y) = \text{cm}(Y)$ ) we revise  $\text{cost}(Y)$  by rounding  $|Y|$  down to the nearest power of  $b$ . Thus,  $\frac{1}{b} \cdot \text{cost}(Y) < \text{cost}_b(Y) \leq \text{cost}(Y)$ . We relax the integer program (1) as follows ( $b$  to be determined later):

$$\begin{aligned}
& \text{minimize} && \sum_{Y \subset X} \text{cost}_b(Y) \cdot z_Y \\
& \text{s.t.} && \sum_{Y \ni x} z_Y + w_x \geq 1 \quad \forall x \in X \\
& && \sum_{Y \subset X} z_Y \leq k \\
& && \sum_{x \in X} w_x \leq n - n' \\
& && z_Y \geq 0 \quad \forall Y \subset X \\
& && w_x \geq 0 \quad \forall x \in X.
\end{aligned} \tag{2}$$

Then, following a well-traveled path, we lagrangify the constraint on the number of clusters to get the following lagrangian relaxation ( $\lambda$  denotes the unknown Lagrange multiplier).

$$\begin{aligned}
& \text{minimize} && \sum_{Y \subset X} \text{cost}_b(Y) \cdot z_Y + \lambda \cdot (\sum_{Y \subset X} z_Y - k) \\
& \text{s.t.} && \sum_{Y \ni x} z_Y + w_x \geq 1 \quad \forall x \in X \\
& && \sum_{x \in X} w_x \leq n - n' \\
& && z_Y \geq 0 \quad \forall Y \subset X \\
& && w_x \geq 0 \quad \forall x \in X.
\end{aligned} \tag{3}$$

For fixed  $\lambda$ , this is a linear program, and its dual is:

$$\begin{aligned}
& \text{maximize} && \sum_{x \in X} \alpha_x - \gamma \cdot (n - n') - \lambda \cdot k \\
& \text{s.t.} && \sum_{x \in Y} \alpha_x \leq \lambda + \text{cost}_b(Y) \quad \forall Y \subset X \\
& && 0 \leq \alpha_x \leq \gamma \quad \forall x \in X.
\end{aligned} \tag{4}$$

Notice that the linear program (3) can be interpreted as a relaxation of the “facility location” version of the problem, with  $\lambda$ -uniform cluster opening costs.

► **Lemma 2.** *For any  $\lambda$ , the optimal value of the linear program (4) is a lower bound on the optimal value of the integer program (1).*

**Proof.** Consider any optimal solution  $(z, w)$  to the integer program (1). Notice that we may assume that  $\sum_{Y \subset X} z_Y = k$ , otherwise we can split some clusters to get exactly  $k$  of them. Splitting clusters cannot increase the cost of the solution. This is also a feasible solution to the linear program (3). Moreover, the Lagrange term  $\lambda \cdot (\sum_{Y \subset X} z_Y - k)$  zeroes out, and  $\sum_{Y \subset X} \text{cost}_b(Y) \cdot z_Y \leq \sum_{Y \subset X} \text{cost}(Y) \cdot z_Y$ . By weak duality, the value of any feasible solution to the dual program (4) is a lower bound on the value of any feasible solution to the linear program (3). ◀

## 16:6 Min-Sum Clustering (With Outliers)

An obvious issue with the dual program (4) is that the number of constraints is exponential in  $n$ . We want to construct a dual solution by growing the dual variables, however, it is not clear how to detect new tight dual constraints without enumerating over the  $\exp(n)$  number of constraints. We now address this issue. First consider the following fact.

► **Proposition 3.** *Let  $Y$  be a finite set of points in  $\mathbb{R}^d$ . There exists  $y \in Y$  such that  $\sum_{x \in Y} \|x - y\|_2^2 \leq 2 \cdot \sum_{x \in Y} \|x - \text{cm}(Y)\|_2^2$ . (We note that the factor of 2 can be improved to  $1 + \epsilon$ , for any  $\epsilon > 0$ , using the center of mass of  $O(1/\epsilon^2)$  points in  $Y$ , e.g. [35, 28].)*

**Proof sketch.** Notice that for every  $y \in \mathbb{R}^d$ ,

$$\sum_{x \in Y} \|x - y\|_2^2 \leq \sum_{x \in Y} \|x - \text{cm}(Y)\|_2^2 + |Y| \cdot \|y - \text{cm}(Y)\|_2^2$$

(see [44]). Thus, by picking  $y \in Y$  that minimizes  $\|y - \text{cm}(Y)\|_2^2$ , the proposition follows. ◀

An immediate consequence of Proposition 3 is that  $\mathcal{F} = X$  is a set of  $n$  points in  $\mathbb{R}^d$ , such that for every  $Y \subset X$  there exists a point  $c_Y \in \mathcal{F}$  such that

$$\sum_{x \in Y} \|x - \text{cm}(Y)\|_2^2 \leq \sum_{x \in Y} \|x - c_Y\|_2^2 \leq 2 \cdot \sum_{x \in Y} \|x - \text{cm}(Y)\|_2^2. \quad (5)$$

(We can improve the factor of 2 to any constant  $1 + \epsilon$  by increasing the size of  $\mathcal{F}$  to  $n^{O(1/\epsilon^2)}$ .) Now, given  $\mathcal{F}$ , set initially  $\text{ctr}(Y) = c_Y$  for all  $Y \subset X$ . Notice that this puts  $\text{cost}_b(Y) = \text{floor}_b(|Y|) \cdot \sum_{y \in Y} \|y - c_Y\|_2^2$ . We consider the following revised dual program.

$$\begin{aligned} & \text{maximize} && \sum_{x \in X} \alpha_x - \gamma \cdot (n - n') - \lambda \cdot k \\ & \text{s.t.} && \sum_{x \in Y} \alpha_x \leq \lambda + \text{floor}_b(|Y|) \cdot \sum_{x \in Y} \|x - y\|_2^2 \quad \forall Y \subset X, \forall y \in Y \\ & && 0 \leq \alpha_x \leq \gamma \quad \forall x \in X. \end{aligned} \quad (6)$$

► **Lemma 4.** *For any  $\lambda$ , the optimal value of the linear program (6) is at most twice the optimal value of the integer program (1).*

**Proof.** The dual of the linear program (6) is

$$\min \left\{ \sum_{Y \subset X} \sum_{y \in Y} \text{floor}_b(|Y|) \cdot \sum_{x \in Y} \|x - y\|_2^2 \cdot z_{Y,y} + \lambda \cdot \left( \sum_{Y \subset X} \sum_{y \in Y} z_{Y,y} - k \right) : \right. \\ \left. \forall x \in X, \sum_{Y \ni x} \sum_{y \in Y} z_{Y,y} + w_x \geq 1 \wedge \sum_{x \in X} w_x \leq n - n' \wedge z, w \geq 0 \right\} \quad (7)$$

Consider an optimal clustering of any  $n'$  points in  $X$  into  $k$  disjoint clusters  $Y_1, Y_2, \dots, Y_k$ . For all  $Y \subset X$ , set  $z_{Y,y}$  to be the indicator that  $Y$  is a cluster in this list and  $y = c_Y$ . Also, for all  $x \in X$  set  $w_x$  to be the indicator that  $x$  is not clustered. Clearly, this is a feasible solution to the linear program (7), so its value is an upper bound on the optimal value of the linear program (6). The Lagrange term vanishes as there are exactly  $k$  non-zero values  $Z_{Y,y}$ . Thus, the upper bound is

$$\sum_{j=1}^k \text{floor}_b(|Y_j|) \cdot \sum_{x \in Y_j} \|x - c_{Y_j}\|_2^2 \leq 2 \cdot \sum_{j=1}^k \text{floor}_b(|Y_j|) \cdot \sum_{x \in Y_j} \|x - \text{cm}(Y_j)\|_2^2 \leq 2 \cdot \sum_{j=1}^k \text{cost}(Y_j),$$

where the first inequality uses Equation (5). ◀

In the primal-dual procedure, there is an active set  $\text{active} \subset X$  of points for which it is safe to raise the dual variable  $\alpha_x$  for all  $x \in \text{active}$ . We need to detect when a new dual constraint becomes tight and requires the removal of the points that are involved from active. This can be done in polynomial time for the revised dual program (6) as follows. For every  $y \in X$  and for every  $j \in \{0, 1, 2, \dots, \log_b \text{floor}_b(n)\}$ , we check if there exists  $Y \subset X$  that satisfies (i)  $y \in Y$ ; (ii)  $Y \cap \text{active} \neq \emptyset$ ; (iii)  $\log_b \text{floor}_b(|Y|) = j$ ; (iv)  $\sum_{x \in Y} \alpha_x \geq \lambda + b^j \cdot \sum_{x \in Y} \|x - y\|_2^2$ . In order to do this, consider the set of points  $C_{y,j} = \{x \in X : \alpha_x \geq b^j \cdot \|x - y\|_2^2\}$ , and sort  $C_{y,j}$  by nonincreasing order of  $\alpha_x - b^j \cdot \|x - y\|_2^2$ .

► **Lemma 5.** *There exists a choice of  $Y, y, j$  that satisfies (i)–(iv) iff there exists a choice of  $y, j$  such that  $|C_{y,j}| \geq b^j$  and  $C_{y,j} \cap \text{active} \neq \emptyset$  and first  $\min\{|C_{y,j}|, b^{j+1} - 1\}$  points in the above order that include  $y$  and at least one point from active are a set satisfying (i)–(iv).*

**Proof.** Clearly the existence of  $y, j$  such that  $C_{y,j}$  has the listed properties implies the existence of  $Y, y, j$  that satisfy (i)–(iv). As for the other direction, consider  $Y, y, j$  that satisfy (i)–(iv). Clearly  $y \in C_{y,j}$ . Suppose that there exists a point  $x \in Y \setminus C_{y,j}$ . Then, putting  $Y' = Y \setminus \{x\}$  and  $j' = \log_b \text{floor}_b(|Y'|) \leq j$ , we have that  $Y', y, j'$  also satisfy (i)–(iv). Thus, we may assume that  $Y \subset C_{y,j}$ . Now, choice in lemma of a subset of  $C_{y,j}$  subject to conditions (i)–(iii) maximizes  $\sum_{x \in Y} (\alpha_x - b^j \cdot \|x - y\|_2^2)$ . Thus, this subset also satisfies (iv). ◀

There are  $O(n \log n)$  pairs  $y, j$ . Listing and sorting each  $C_{y,j}$  takes at most  $O(n \log n)$  operations. Listing the candidate  $Y \subset C_{y,j}$  and checking it takes  $O(|C_{y,j}|)$  operations. Thus, finding a new tight constraint can be done in polynomial time. (Trivially, we can discretize the increase of the dual variables and/or use binary search to find the increase that causes a new constraint to become tight. As we're dealing with squared Euclidean distance, if the input consists of finite precision rational numbers, then all computed values are finite precision rational numbers.)

### 3 The Algorithm

We now describe the following three-phase primal-dual algorithm (Algorithm 1: PRIMAL-DUAL, refer page 9) that can be used to solve the facility location version of min-sum clustering. In addition to the pointset  $X$ , the cluster opening cost  $\lambda$ , and the target number of points  $n'$ , the algorithm also gets a (sufficiently large, TBD) parameter  $b$  that governs the excess number of discarded outliers in its output. Throughout the algorithm, sets of points  $Y \subset X$  will maintain values  $\text{card}_b(Y)$  and  $\text{ctr}(Y)$ . Clearly, we cannot do this explicitly and efficiently for every set  $Y \subset X$ . We use Lemma 5 and its consequences to implement the operations that we need, without storing explicitly these values for more than  $n$  sets. This affects only the first phase of the algorithm. For  $x \in X$  and  $Y \subset X$ , we denote throughout the paper  $d(x, Y) = b^{\text{card}_b(Y)} \cdot \|x - \text{ctr}(Y)\|_2^2$ . This is interpreted according to the relevant values of  $\text{card}_b(Y)$  and  $\text{ctr}(Y)$ .

Phase 1 constructs a dual solution and collects candidate clusters. During phase 1, a point  $x$  is either active or inactive. Initially, for all  $x \in X$ , we set  $\alpha_x$  to 0, and we set  $x$  to be active. The set of candidate clusters preclusters is empty. We raise all active  $x$  at a uniform rate, and pause to change the status of points and clusters if one of the following events happens.

- There exists an active  $x \in X$  and a cluster  $Y \in \text{preclusters}$  such that  $\alpha_x \geq d(x, Y)$ . In this case, replace  $Y$  by  $Y \cup \{x\}$  in preclusters. The new cluster in preclusters inherits the  $\text{card}_b$  and  $\text{ctr}$  values from  $Y$ . Also set  $x$  to be inactive.

## 16:8 Min-Sum Clustering (With Outliers)

- There exists  $Y \subset X$  that contains an active point and  $y \in Y$  such that the dual constraint associated with the pair  $Y, y$  is tight. Explicitly,

$$\sum_{x \in Y} \alpha_x \geq \lambda + \text{cost}_b(Y),$$

where we set  $\text{card}_b(Y) = \log_b \text{floor}_b(|Y|)$  and  $\text{ctr}(Y) = y$ . In this case, add an inclusion-wise minimal such  $Y$  to preclusters and set all  $x \in Y$  to be inactive (and set  $\text{card}_b(Y)$  and  $\text{ctr}(Y)$  as stated above).

The first phase ends as soon as the number of active  $x \in X$  drops to  $n - n'$  or lower. If this number drops below  $n - n'$ , we do not add the last cluster  $Y_{\text{last}}$  to preclusters, but keep it separately. Note that each new tight constraint causes at least one point  $x \in X$  to become inactive, hence the number of sets  $Y$  that require storing explicitly their parameters  $\text{card}_b(Y)$  and  $\text{ctr}(Y)$  is at most  $n' \leq n$ .

In phase 2, we trim the set of candidate clusters and assign points uniquely to the clusters in the trimmed list, as follows. Note that we need the parameters  $\text{card}_b$  and  $\text{ctr}$  only for clusters for which these values were stored explicitly in phase 1. Define a conflict graph on the clusters in preclusters. Two clusters  $Y_1, Y_2 \in \text{preclusters}$  are connected by an edge in the conflict graph iff there exists  $x \in Y_1 \cap Y_2$  such that  $\alpha_x > \max\{d(x, Y_1), d(x, Y_2)\}$ . In other words, the edge  $\{Y_1, Y_2\}$  indicates that there is  $x \in Y_1 \cap Y_2$  that contributes to the opening cost  $\lambda$  of both  $Y_1$  and  $Y_2$ . Next, take a lexicographically maximal independent set  $\mathcal{I}$  in the conflict graph, ordering preclusters by non-increasing order of  $\text{card}_b(Y)$ , breaking ties arbitrarily. We group the points clustered in preclusters into meta-clusters of the form  $(Y, Y')$ , where  $Y \in \mathcal{I}$  indicates the meta-cluster, and  $Y'$  is a set of points. (Thus, the entire meta-cluster associated with  $Y$  is  $\cup_{(Y, Y') \in \text{metaclusters}} Y'$ .) In particular, for  $Y \in \mathcal{I}$ , we put  $(Y, Y) \in \text{metaclusters}$ . Any remaining points in preclusters are added as follows. If  $Y' \notin \mathcal{I}$ , then let  $Y''$  be the set of remaining points in  $\{x \in Y' : \alpha_x = \max_{y \in Y'} \alpha_y\}$ , and let  $Y \in \mathcal{I}$  be such that  $Y$  precedes  $Y'$  in the order on preclusters and  $\{Y, Y'\}$  is an edge. Add  $(Y, Y'')$  to metaclusters, with  $\text{card}_b(Y'') = \text{card}_b(Y')$  and  $\text{ctr}(Y'') = \text{ctr}(Y)$ . Finally, if the number of assigned points to metaclusters is less than  $n'$ , then we add  $(Y_{\text{last}}, Y)$  to metaclusters, where  $Y$  is a set of previously unclustered points from  $Y_{\text{last}}$  of the cardinality needed to complete the number of clustered point to  $n'$ . (Notice that at least  $n'$  points are clustered in preclusters  $\cup \{Y_{\text{last}}\}$ , so this is possible.)

Phase 3 determines the final output clustering of the points. For every meta-cluster  $(Y, \cdot)$  and for every integer  $p \leq \text{card}_b(Y)$ , let  $n_{Y,p}$  denote the number of points  $x \in X$  such that there exists  $(Y, Y') \in \text{metaclusters}$  with  $Y' \ni x$  and  $\text{card}_b(Y') = p$ . We open clusters as follows. For  $p = \text{card}_b(Y)$ , we open  $\max\{1, \lfloor \frac{n_{Y,p-2} + n_{Y,p-1} + n_{Y,p}}{b^{2+p}} \rfloor\}$  clusters and assign all the points counted in  $n_{Y,p-2}, n_{Y,p-1}, n_{Y,p}$  to these clusters, as evenly as possible.

► **Lemma 6.** *The number of points in each such cluster is at most  $2b^{2+p}$ , and if  $Y \neq Y_{\text{last}}$  then this number is at least  $b^p$ .*

**Proof.** If we open one cluster, then clearly  $n_{Y,p-2} + n_{Y,p-1} + n_{Y,p} < 2b^{2+p}$ . If we open  $s > 1$  clusters, then we must have  $sb^{2+p} \leq n_{Y,p-2} + n_{Y,p-1} + n_{Y,p} < (s+1)b^{2+p}$ . Thus, the number of points in each cluster is between  $b^{2+p}$  and  $(1 + 1/s)b^{2+p}$ . Clearly for every set  $Y \neq Y_{\text{last}}$ ,  $(Y, Y) \in \text{metaclusters}$ , and by the definition of  $p = \text{card}_b(Y)$ , it holds that  $|Y| \geq b^p$ . ◀

For  $p < \text{card}_b(Y) - 2$ , we open  $\lfloor \frac{n_{Y,p}}{b^{2+p}} \rfloor$  clusters. If this number is at least 1, we assign all the points counted in  $n_{Y,p}$  to these clusters, as evenly as possible. If this number is 0, we discard all the points counted in  $n_{Y,p}$  as outliers.



► **Lemma 7.** *In this step, if no cluster is opened then number of points that are discarded is less than  $b^{2+p}$ , else the number of points in each cluster is at least  $b^{2+p}$  and less than  $2b^{2+p}$ .*

**Proof.** The assertion is trivial. ◀

■ **Algorithm 1** Algorithm PRIMAL-DUAL.

---

```

1: procedure PRIMALDUAL( $X, \lambda, n', b$ )
2:    $\alpha, \text{preclusters}, Y_{\text{last}} \leftarrow \text{PRIMALDUALPHASE1}(X, \lambda, n', b)$ 
3:    $\text{metaclusters} \leftarrow \text{PRIMALDUALPHASE2}(X, n', b, \alpha, \text{preclusters}, Y_{\text{last}})$ 
4:    $\text{clusters} \leftarrow \text{PRIMALDUALPHASE3}(X, b, \text{metaclusters})$ 
5:   return clusters
6: end procedure
7:
8: procedure PRIMALDUALPHASE1( $X, \lambda, n', b$ )
9:    $\text{active}, \text{preclusters} \leftarrow X, \emptyset$ 
10:   $\alpha_x \leftarrow 0$  for all  $x \in X$ 
11:  while  $|\text{active}| > n - n'$  do
12:    raise  $\alpha_x$  at a uniform rate for all  $x \in \text{active}$  ▷ stop raising when one of the following two cases happens
13:    if  $\exists x \in \text{active}$  and  $Y \in \text{preclusters}$  such that  $\alpha_x \geq d(x, Y)$  then
14:       $\text{card}_b(Y \cup \{x\}), \text{ctr}(Y \cup \{x\}) \leftarrow \text{card}_b(Y), \text{ctr}(Y)$ 
15:       $\text{preclusters}, \text{active} \leftarrow \text{preclusters} \setminus \{Y\} \cup \{Y \cup \{x\}\}, \text{active} \setminus \{x\}$ 
16:    else if  $\exists Y \subset X$  and  $y \in Y$  such that  $Y \cap \text{active} \neq \emptyset$  the dual constraint for  $Y, y$  is tight then
17:       $\text{card}_b(Y), \text{ctr}(Y) \leftarrow \log_b \text{floor}_b(|Y|), y$ 
18:      if  $|\text{active} \setminus Y| < n - n'$  then ▷ use an inclusion-wise minimal such  $Y$ 
19:        return  $\alpha, \text{preclusters}, Y$ 
20:      else
21:         $\text{preclusters}, \text{active} \leftarrow \text{preclusters} \cup \{Y\}, \text{active} \setminus Y$ 
22:      end if
23:    end if
24:  end while
25:  return  $\alpha, \text{preclusters}, \emptyset$ 
26: end procedure
27:
28: procedure PRIMALDUALPHASE2( $X, n', b, \alpha, \text{preclusters}, Y_{\text{last}}$ )
29:    $\text{active}, \text{metaclusters} \leftarrow \{x \in X : x \in Y \in \text{preclusters} \vee x \in Y_{\text{last}}\}, \emptyset$ 
30:   for  $Y \in \text{preclusters}$ , by order of nonincreasing  $\text{card}_b(Y)$  do
31:     if  $\exists (Y', Y'') \in \text{metaclusters}$  with  $x \in Y \cap Y'$  and  $\alpha_x > \max\{d(x, Y), d(x, Y')\}$  then
32:        $Y'', \text{card}_b(Y''), \text{ctr}(Y'') \leftarrow \{x \in Y \cap \text{active} : \alpha_x = \max_{y \in Y} \alpha_y\}, \text{card}_b(Y), \text{ctr}(Y')$ 
33:        $\text{metaclusters} \leftarrow \text{metaclusters} \cup \{(Y', Y'')\}$ 
34:        $\text{active} \leftarrow \text{active} \setminus Y''$ 
35:     else
36:       remove each  $x \in Y$  from any  $Y'' \ni x$  with  $(Y', Y'') \in \text{metaclusters}$  ▷  $\alpha_x \leq d(x, Y'')$ ;
37:        $\text{card}_b(Y''), \text{ctr}(Y'')$  don't change
38:        $\text{metaclusters} \leftarrow \text{metaclusters} \cup \{(Y, Y)\}$ 
39:        $\text{active} \leftarrow \text{active} \setminus Y$ 
40:     end if
41:   end for
42:    $Y, \text{card}_b(Y), \text{ctr}(Y) \leftarrow \{|\text{active}| - n + n' \text{ points in } Y_{\text{last}} \cap \text{active}\}, \text{card}_b(Y_{\text{last}}), \text{ctr}(Y_{\text{last}})$ 
43:   if  $Y \neq \emptyset$  then
44:      $\text{metaclusters} \leftarrow \text{metaclusters} \cup \{(Y_{\text{last}}, Y)\}$ 
45:   end if
46:   return metaclusters
47: end procedure
48: procedure PRIMALDUALPHASE3( $X, b, \text{metaclusters}$ )
49:    $\text{clusters} \leftarrow \emptyset$ 
50:   for  $(Y, \cdot) \in \text{metaclusters}$  do
51:      $Y_{\text{max}} \leftarrow \{x \in X : \exists Y' \ni x \text{ such that } (Y, Y') \in \text{metaclusters} \wedge \text{card}_b(Y') \geq \text{card}_b(Y) - 2\}$ 
52:      $\text{clusters} \leftarrow \text{clusters} \cup \text{PARTITION}(Y_{\text{max}}, \max\{1, \lfloor |Y_{\text{max}}|/b^{2+\text{card}_b(Y)} \rfloor\})$ 
53:     for  $p < \text{card}_b(Y) - 2$  do
54:        $Y_p \leftarrow \{x \in X : \exists Y' \ni x \text{ such that } (Y, Y') \in \text{metaclusters} \wedge \text{card}_b(Y') = p\}$ 
55:       if  $|Y_p| \geq b^{2+p}$  then
56:          $\text{clusters} \leftarrow \text{clusters} \cup \text{PARTITION}(Y_p, \lfloor |Y_p|/b^{2+p} \rfloor)$ 
57:       end if
58:     end for
59:   end for
60:   return clusters
61: end procedure
62:
63: procedure PARTITION( $S, m$ ) ▷  $m \geq 1$ 
64:   partition  $S$  as evenly as possible into  $m$  disjoint subsets  $S_1, S_2, \dots, S_m$ 
65:   return  $\{S_1, S_2, \dots, S_m\}$ 
66: end procedure

```

---

## 16:10 Min-Sum Clustering (With Outliers)

We are now ready to define our min-sum  $k$ -clustering algorithm (Algorithm 2: MIN-SUM-CLUSTERING refer page 10). If  $k \leq \frac{4}{\epsilon}$ , we can run a PTAS or a constant factor approximation for fixed  $k$  (for instance [28, 23]).<sup>1</sup> Otherwise, our algorithm follows the general schema of the lagrangian relaxation method. Let  $\delta > 0$  be determined later. We run the procedure PRIMALDUAL on various values of  $\lambda$ , and if the smallest returned cluster for a particular value of  $\lambda$  which is denoted as  $Y_{\min, \lambda}$  has at most  $\frac{\epsilon}{3} \cdot n'$  points, we remove this cluster. Using binary search on the Lagrange multiplier  $\lambda$ , we find two values  $\lambda_1 < \lambda_2$ , with  $\lambda_2 - \lambda_1 < \delta$ , that satisfy the following property. The above process (running PRIMALDUAL, then removing the smallest cluster if it's sufficiently small) returns  $k_1 > k$  clusters (denoted as clusters<sub>1</sub>) for  $\lambda = \lambda_1$ , and  $k_2 \leq k$  clusters (denoted as clusters<sub>2</sub>) for  $\lambda = \lambda_2$ . If  $\frac{k-k_2}{k_1-k_2} \geq 1 - \frac{\epsilon}{4}$ , we output the  $k$  largest clusters in the solution for  $\lambda_1$  (i.e. from clusters<sub>1</sub>), and otherwise we output the solution for  $\lambda_2$  (i.e. from clusters<sub>2</sub>).

■ **Algorithm 2** Algorithm MIN-SUM-CLUSTERING.

---

```

1: procedure MINSUMCLUSTERING( $X, k, n', \epsilon$ )
2:    $\lambda_1 \leftarrow 0, \lambda_2 \leftarrow \sum_{x,y \in X} \|x - y\|_2^2$ 
3:   clusters1  $\leftarrow \{\{x\} : x \in X\},$  clusters2  $\leftarrow X$ 
4:    $b \leftarrow \frac{1+\epsilon}{\epsilon}$ 
5:    $\delta \leftarrow \frac{2}{(n+k)\lambda_2}$  ▷ we need  $\delta \leq \frac{2}{(n+k) \text{opt}(X, n', k)}$ 
6:   while  $\lambda_2 - \lambda_1 > \delta$  do
7:      $\lambda = \frac{1}{2} \cdot (\lambda_1 + \lambda_2)$ 
8:     clusters  $\leftarrow \text{PrimalDual}(X, \lambda, n', b)$ 
9:      $Y_{\min, \lambda} \leftarrow$  smallest cluster in clusters
10:     $k' \leftarrow |\text{clusters}| - 1$ 
11:    if  $|Y_{\min, \lambda}| \leq \frac{\epsilon}{3} \cdot n'$  then
12:      clusters  $\leftarrow \text{clusters} \setminus \{Y_{\min, \lambda}\}$ 
13:    end if
14:    if  $k' > k$  then
15:       $\lambda_1, \text{clusters}_1, k_1 \leftarrow \lambda, \text{clusters}, k'$ 
16:    else ▷  $k' \leq k$ 
17:       $\lambda_2, \text{clusters}_2, k_2 \leftarrow \lambda, \text{clusters}, k'$ 
18:    end if
19:  end while
20:   $\rho_1 \leftarrow \frac{k-k_2}{k_1-k_2}$  ▷  $k_1 > k \geq k_2 \geq 0$ 
21:  if  $\rho_1 \geq 1 - \frac{\epsilon}{4}$  then
22:    return  $\{k$  largest sets in clusters1 $\}$ 
23:  else
24:    return clusters2 ▷ If  $|\text{clusters}_2| < k$ , split clusters arbitrarily to get exactly  $k$ 
25:  end if
26: end procedure

```

---

<sup>1</sup> These papers consider only the case without outliers. The PTAS in [28] enumerates over cluster sizes and approximate cluster centers, then computes an optimal assignment of the data points to the approximate centers, given the corresponding cluster sizes. Clearly, the algorithm can be adapted trivially to handle the case with outliers by modifying the target sum of cluster sizes.

► **Theorem 8.** *The execution of **procedure** `MINSUMCLUSTERING`( $X, k, n', \epsilon$ ) computes a clustering of  $X' \subset X$  into  $k$  clusters such that  $|X'| \in [(1 - \epsilon)n', n']$ , and the total cost of the clustering of  $X'$  is at most  $O\left(\frac{1}{\epsilon^3}\right) \cdot \text{opt}(X, n', k)$ . The time complexity of this computation is  $\text{poly}(n, \log(1/\epsilon), \log \Delta)$ , where  $\Delta$  is ratio of largest to non-zero smallest  $\|\cdot\|_2^2$  distance in  $X$ .*

**Proof.** The performance guarantee is an immediate consequence of Corollary 11 below. The running time is a straightforward analysis of the code. ◀

► **Theorem 9.** *The same claim applies to instances of points in a metric space  $(X, \text{dist})$ , with  $\text{dist}$  replacing  $\|\cdot\|_2^2$  in the code and in the claim.*

**Proof sketch.** The  $\|\cdot\|_2^2$  distance can be replaced by any metric distance  $\text{dist}$  in all claims starting from Proposition 3. The proofs sometime require minor changes. In particular, in Lemma 13, the factor  $\frac{1}{9}$  can be improved to  $\frac{1}{3}$  on account of the triangle inequality, and this improves all the other constants that depend on it. ◀

## 4 Proofs

In this section we analyze the min-sum  $k$ -clustering algorithm. The analysis builds on the following guarantees of the primal-dual schema.

► **Theorem 10.** *For every  $\epsilon \in (0, 1]$  there exists a constant  $c = c_\epsilon$  such that the following holds. Let clusters be the output of **procedure** `PRIMALDUAL`( $X, \lambda, n', b$ ), and let  $\alpha$  be the dual solution computed during the execution of this procedure. Set  $\gamma = \max_{x \in X} \alpha_x$ . Then,*

1.  $(\alpha, \gamma)$  is a feasible solution to the dual program (6).
2.  $\sum_{Y \in \text{clusters}} |Y| \in [(1 - \frac{\epsilon}{3})n', n']$ .
3.  $c \cdot (|\text{clusters}| - 1) \cdot \lambda + \sum_{Y \in \text{clusters}} \text{cost}(Y) \leq c \cdot \sum_{Y \in \text{clusters}} \sum_{x \in Y} \alpha_x$ .

► **Corollary 11.** *Let clusters be the output of **procedure** `MINSUMCLUSTERING`. Then, the following assertions hold:*

1.  $|\text{clusters}| \leq k$ .
2.  $\sum_{Y \in \text{clusters}} |Y| \in [(1 - \epsilon)n', n']$ .
3.  $\sum_{Y \in \text{clusters}} \text{cost}(Y) \leq \frac{8(c+1)}{\epsilon} \cdot \text{opt}(X, n', k)$ .

**Proof.** The first assertion follows directly from the definition of the procedure.

For the second assertion, let  $\lambda_i, \text{clusters}_i$  (where  $i = 1, 2$ ) be the values that determine the output of the procedure. By Theorem 10,  $\sum_{Y \in \text{clusters}_i} |Y| \geq (1 - \frac{\epsilon}{3})n'$ . If  $Y_{\min, \lambda_i}$  is removed from  $\text{clusters}_i$ , then  $|Y_{\min, \lambda_i}| \leq \frac{\epsilon}{3} \cdot n'$ . Thus, if  $i = 2$  then clearly the assertion holds. If  $i = 1$ , then  $\rho_1 \geq 1 - \frac{\epsilon}{4}$ . Therefore,

$$k_1 \leq \frac{1}{\rho_1} \cdot k \leq \left(1 + \frac{\epsilon}{4 - \epsilon}\right) \cdot k \leq \left(1 + \frac{\epsilon}{3}\right) \cdot k.$$

Thus, the procedure removes from the output at most a fraction of  $\frac{\epsilon}{3}$  of the clusters in  $\text{clusters}_1$ . As the removed clusters are the smallest, they contain at most  $\frac{\epsilon}{3} \cdot n'$  points.

As for the third assertion, consider the two solutions  $\text{clusters}_1, \text{clusters}_2$  that are used to determine the procedure's output. For  $i = 1, 2$ , let  $\alpha_i, \text{preclusters}_i$  be the output of `PRIMALDUALPHASE1` during the computation of  $\text{clusters}_i$ . Put  $\gamma_i = \max_{x \in X} \alpha_{i,x}$ . Let

$$(\alpha, \gamma) = \rho_1(\alpha_1, \gamma_1) + (1 - \rho_1)(\alpha_2, \gamma_2).$$

Clearly,  $(\alpha, \gamma)$  is a feasible solution to the dual LP (6) with the constant  $\lambda = \rho_1 \lambda_1 + (1 - \rho_1) \lambda_2$ . Notice that there are exactly  $n - n'$  points that are not included in  $\text{preclusters}_i$ . Each point

## 16:12 Min-Sum Clustering (With Outliers)

$x \in X$  which is not included in preclusters $_i$  has  $\alpha_{i,x} = \gamma_i$ . (Notice that all the points that are excluded are active. This is true even for points that are discarded from the last tight cluster that gets included in preclusters $_i$ .) So, the value of the solution  $(\alpha, \gamma)$  is

$$\begin{aligned}
2 \cdot \text{opt}(X, n', k) &\geq \sum_{x \in X} \alpha_x - (n - n')\gamma - \lambda k \\
&= \rho_1 \cdot \left( \sum_{x \in X} \alpha_{1,x} - (n - n')\gamma_1 - \lambda k_1 \right) + (1 - \rho_1) \cdot \left( \sum_{x \in X} \alpha_{2,x} - (n - n')\gamma_2 - \lambda k_2 \right) \\
&= \rho_1 \cdot \left( \sum_{Y \in \text{preclusters}_1} \sum_{x \in Y} \alpha_{1,x} - \lambda k_1 \right) + (1 - \rho_1) \cdot \left( \sum_{Y \in \text{preclusters}_2} \sum_{x \in Y} \alpha_{2,x} - \lambda k_2 \right) \\
&\geq \rho_1 \cdot \left( \sum_{Y \in \text{preclusters}_1} \sum_{x \in Y} \alpha_{1,x} - \lambda_1 k_1 \right) + (1 - \rho_1) \cdot \left( \sum_{Y \in \text{preclusters}_2} \sum_{x \in Y} \alpha_{2,x} - \lambda_2 k_2 \right) \\
&\quad - \delta \cdot (k_1 + k_2),
\end{aligned}$$

where the first inequality follows from Lemma 4, and the first equality uses the fact that  $k = \rho_1 k_1 + (1 - \rho_1)k_2$ .

For  $i = 1, 2$  consider the final value of clusters $_i$ . By definition,  $k_i$  is one less than the number of clusters returned from **procedure** PRIMALDUAL, so by Theorem 10,

$$\sum_{Y \in \text{clusters}_i} \text{cost}(Y) \leq c \cdot \left( \sum_{Y \in \text{clusters}_i} \sum_{x \in Y} \alpha_x - k_i \cdot \lambda_i \right).$$

In particular, the right-hand side is non-negative. Notice that if  $\rho_1 \geq 1 - \frac{\epsilon}{4}$ , then clearly  $\rho_1 > \frac{\epsilon}{4}$  and the cost of the output clustering is at most  $\sum_{Y \in \text{clusters}_1} \text{cost}(Y)$ . Similarly, if  $\rho_1 < 1 - \frac{\epsilon}{4}$ , then  $1 - \rho_1 > \frac{\epsilon}{4}$  and the cost of the output clustering is at most  $\sum_{Y \in \text{clusters}_2} \text{cost}(Y)$ . Either way, we get that the cost of the clustering is at most  $\frac{8c}{\epsilon} \cdot \text{opt}(X, n', k) + \frac{4\delta}{\epsilon} \cdot (k_1 + k_2) \leq \frac{8(c+1)}{\epsilon} \cdot \text{opt}(X, n', k)$ .  $\blacktriangleleft$

Next we analyze primal-dual algorithm and prove Theorem 10. The notation follows Algorithm 1.

► **Lemma 12.** *At the end of executing **procedure** PRIMALDUALPHASE1, for every  $Y \in$  preclusters and for every  $x \in Y$ , we have that  $\alpha_x \geq d(x, Y)$ .*

**Proof.** When  $Y$  is added to preclusters then there exists  $y \in Y$  and  $j = \log_b \text{floor}_b(|Y|)$  such that  $Y \subset C_{y,j}$ . We set  $\text{card}_b(Y) = j$  and  $\text{ctr}(Y) = y$ , so by the definition of  $C_{y,j}$  the lemma holds. If a point  $x$  is later added to  $Y$ , the condition for doing it is that  $\alpha_x \geq d(x, Y)$ .  $\blacktriangleleft$

► **Lemma 13.** *At the end of executing **procedure** PRIMALDUALPHASE2, the following assertions hold:*

1.  $\sum_{(Y, Y') \in \text{metaclusters}} |Y'| = n'$ .
2. For every  $(Y, Y') \in \text{metaclusters}$  and for every  $x \in Y'$ , we have that  $\alpha_x \geq \frac{1}{9} \cdot d(x, Y')$ .

**Proof.** The first assertion holds as the points that are clustered in metaclusters are all the points that are clustered in preclusters plus some of the points clustered in  $Y_{\text{last}}$ . The number of such points is at most  $n'$  without  $Y_{\text{last}}$ , and at least  $n'$  with  $Y_{\text{last}}$ . The algorithm takes from  $Y_{\text{last}}$  exactly the number of points needed to complete the number in preclusters to  $n'$ .

For the second assertion, consider  $(Y, Y') \in \text{metaclusters}$  and  $x \in Y'$ . If  $Y = Y'$ , then Lemma 12 guarantees the assertion. Otherwise, consider  $Y'' \in \text{preclusters}$  that caused  $(Y, Y')$  to be added to metaclusters. In particular,  $Y' \subset Y''$ ,  $\text{card}_b(Y') = \text{card}_b(Y'') \leq \text{card}_b(Y)$ , and

there exists  $z \in Y \cap Y''$  such that  $\alpha_z > \max\{d(z, Y), d(z, Y'')\}$ . By the choice of  $Y'$  in the algorithm,  $\alpha_x = \max_{x' \in Y''} \alpha_{x'}$ , so it must be that  $\alpha_z \leq \alpha_x$ . Notice that by Lemma 12,

$$\alpha_x \geq d(x, Y'') = b^{\text{card}_b(Y'')} \cdot \|x - \text{ctr}(Y'')\|_2^2.$$

$$\begin{aligned} \text{Also, } \alpha_z &> \max\{b^{\text{card}_b(Y)} \cdot \|z - \text{ctr}(Y)\|_2^2, b^{\text{card}_b(Y'')} \cdot \|z - \text{ctr}(Y'')\|_2^2\} \\ &\geq b^{\text{card}_b(Y')} \cdot \max\{\|z - \text{ctr}(Y)\|_2^2, \|z - \text{ctr}(Y'')\|_2^2\}. \end{aligned}$$

$$\begin{aligned} \text{Thus, } d(x, Y') &= b^{\text{card}_b(Y')} \cdot \|x - \text{ctr}(Y)\|_2^2 \\ &\leq b^{\text{card}_b(Y')} \cdot (\|x - \text{ctr}(Y'')\|_2 + \|z - \text{ctr}(Y'')\|_2 + \|z - \text{ctr}(Y)\|_2)^2 \\ &\leq 9 \cdot \alpha_x. \end{aligned} \quad \blacktriangleleft$$

Let clusters be the output of **procedure** PRIMALDUALPHASE3. Recall that every cluster  $Z \in \text{clusters}$  is derived in some iteration indexed by  $(Y, \cdot) \in \text{metaclusters}$ . It holds that either  $Z \subset Y_{\max}$  or  $Z \subset Y_p$  for some  $p < \text{card}_b(Y) - 2$ . Notice that in the latter case,  $Y \neq Y_{\text{last}}$ . We will set implicitly  $\text{card}_b(Z)$  as follows.

$$\text{card}_b(Z) = \begin{cases} \text{card}_b(Y) & Z \subset Y_{\max}, \\ p & Z \subset Y_p, p < \text{card}_b(Y) - 2. \end{cases}$$

We will also set implicitly  $\text{ctr}(Z) = \text{ctr}(Y)$ .

► **Lemma 14.** *If  $Z \subset Y_p$  for some  $p < \text{card}_b(Y) - 2$ , then  $\sum_{x \in Z} \alpha_x \geq b \cdot \lambda$ . The same is true if  $Z \subset Y_{\max}$ ,  $Y \neq Y_{\text{last}}$ , and  $|Y_{\max}| \geq b^{2+\text{card}_b(Y)}$ .*

**Proof.** Consider  $x \in Z \subset Y_p$ ,  $p < \text{card}_b(Y) - 2$ . There is a pair  $(Y, Y')$  in metaclusters such that  $p = \text{card}_b(Y') < \text{card}_b(Y) - 2$ , and  $x \in Y'$ . Moreover, there is  $Y'' \in \text{preclusters}$  such that  $Y' \subset Y''$  and  $\text{card}_b(Y'') = p$  and  $\alpha_x = \max_{y \in Y''} \alpha_y$ . By the definition of  $\text{card}_b$ , we have that  $|Y''| < b^{1+p}$ . Therefore,  $\alpha_x > \frac{\lambda}{b^{1+p}}$ . By Lemma 7,  $|Z| \geq b^{2+p}$ , hence the conclusion.

A similar argument applies to  $Z \subset Y_{\max}$ , assuming that  $|Y_{\max}| \geq b^{2+\text{card}_b(Y)}$ . In this case, if  $Z \supset Y$  then we have  $\sum_{x \in Y} \alpha_x \geq \lambda$ . As  $|Y| < b^{1+\text{card}_b(Y)}$ ,  $Z$  also contains more than  $b^{2+\text{card}_b(Y)} - b^{1+\text{card}_b(Y)} = b^{2+\text{card}_b(Y)} \cdot (1 - \frac{1}{b})$  points from pairs  $(Y, Y') \in \text{metaclusters}$ ,  $Y' \neq Y$ . By the argument for  $Y_p$ , for each such point  $x$  we have  $\alpha_x > \frac{\lambda}{b^{1+\text{card}_b(Y')}} > \frac{\lambda}{b^{1+\text{card}_b(Y)}}$ . Overall, we get that  $\sum_{x \in Z} \alpha_x > \lambda + b^{2+\text{card}_b(Y)} \cdot (1 - \frac{1}{b}) \cdot \frac{\lambda}{b^{1+\text{card}_b(Y)}} = b \cdot \lambda$ . If  $Z$  does not contain  $Y$ , then the argument for  $Y_p$  holds. ◀

► **Lemma 15.** *For every  $Z \in \text{clusters}$ ,  $\text{cost}(Z) \leq 2b^2 \cdot \sum_{x \in Z} d(x, Z)$ .*

**Proof.** We have  $\text{cost}(Z) = |Z| \cdot \sum_{x \in Z} \|x - \text{cm}(Z)\|_2^2 \leq |Z| \cdot \sum_{x \in Z} \|x - \text{ctr}(Y)\|_2^2$ . By Lemmas 6 and 7,  $|Z| \leq 2b^2 \cdot \text{card}_b(Z)$ . ◀

**Proof of Theorem 10.** First, consider the feasibility of  $(\alpha, \gamma)$ . Clearly,  $\gamma$  is set in the theorem to satisfy the constraints that include it. Regarding the constraints that involve only  $\alpha$ , we prove that they are satisfied throughout the execution of **procedure** PRIMALDUALPHASE1. The proof is by induction on the number of inactive points. Clearly, the initial  $\alpha$  is feasible. Now, suppose that  $\alpha$  is feasible for some number of inactive points, and consider the next step when this number increases and a set  $A \subset \text{active}$  is removed from active (we will use active here to denote the set before the removal of  $A$ ). Let  $\alpha'$  denote the values of the dual variables just before  $A$  is removed from active. If there exist  $Y \subset X$  and  $y \in Y$  such that the constraint for the pair  $Y, y$  is violated, then clearly  $Y \cap \text{active} \neq \emptyset$ , otherwise the same constraint would

## 16:14 Min-Sum Clustering (With Outliers)

have been violated by the solution  $\alpha$ , as  $\alpha$  and  $\alpha'$  differ only on active. But then there is some intermediate value  $\alpha''$  such that  $\alpha''_x = \alpha_x$  for all  $x \notin \text{active}$  and  $\alpha_x \leq \alpha''_x < \alpha'_x$  for all  $x \in \text{active}$ , which causes this constraint (or another constraint involving active points) to become tight. Therefore, at least one point would have been removed from active before we reach the values  $\alpha'$ , in contradiction with our assumptions. Next, consider number of points clustered in the output clusters of **procedure** PRIMALDUAL. Clearly, **procedure** PRIMALDUALPHASE2 clusters in metaclusters exactly  $n'$  points. Some of these points are discarded by **procedure** PRIMALDUALPHASE3. Consider some  $(Y, \cdot) \in \text{metaclusters}$ . By Lemma 7, number of points discarded from these clusters is less than

$$\sum_{p < \text{card}_b(Y) - 2} b^{2+p} = \frac{b^{\text{card}_b(Y)} - b^2}{b - 1}.$$

On the other hand, all the points in  $Y_{\max}$  are clustered in clusters, as  $Y \neq Y_{\text{last}}$  in this case. Clearly, the number of points in  $Y_{\max}$  is at least  $|Y| \geq b^{\text{card}_b(Y)}$ . Thus, less than  $\frac{1}{b-1} \cdot n' \leq \epsilon \cdot n'$  points get discarded. Finally, let's consider the cost of the clustering. Let  $Z \in \text{clusters}$  be a cluster that satisfies the conditions of Lemma 14. Then,

$$\begin{aligned} \sum_{x \in Z} \alpha_x - \lambda &> \frac{b-1}{b} \cdot \sum_{x \in Z} \alpha_x \\ &\geq \frac{b-1}{9b} \cdot \sum_{x \in Z} d(x, Z) \\ &\geq \frac{b-1}{18b^3} \cdot \text{cost}(Z), \end{aligned}$$

where the first inequality follows from Lemma 14, the second inequality follows from Lemma 13, and the third inequality follows from Lemma 15. The remaining clusters are sets  $Y_{\max}$  with  $|Y_{\max}| < b^{2+\text{card}_b(Y)}$  and a subset of  $Y_{\text{last}}$ . Consider a cluster  $Y_{\max} \in \text{clusters}$ . We have that

$$\sum_{x \in Y_{\max}} \alpha_x - \lambda = \sum_{x \in Y} \alpha_x - \lambda + \sum_{x \in Y_{\max} \setminus Y} \alpha_x \geq \sum_{x \in Y} d(x, Y) + \frac{1}{9} \cdot \sum_{x \in Y_{\max} \setminus Y} d(x, Y) \geq \frac{1}{18b^2} \cdot \text{cost}(Y_{\max}).$$

Finally, if there's a cluster  $Z \subset Y_{\text{last}}$ , then

$$\sum_{x \in Z} \alpha_x \geq \frac{1}{9} \cdot \sum_{x \in Z} d(x, Y_{\max}) \geq \frac{1}{18b^2} \cdot \text{cost}(Z).$$

Thus, we can set  $c = c_\epsilon = \frac{18b^3}{b-1} = \frac{18(1+\epsilon)^3}{\epsilon^2}$ . ◀

---

## References

- 1 S. Ahmadian, A. Norouzi-Fard, O. Svensson, and J. Ward. Better guarantees for  $k$ -means and Euclidean  $k$ -median by primal-dual algorithms. *Proc. of the 58th Ann. IEEE Symp. on Foundations of Computer Science*, pages 61–72, 2017.
- 2 S. Ahmadian and C. Swamy. Approximation algorithms for clustering problems with lower bounds and outliers. In *Proc. of the 43rd Int'l Colloq. on Automata, Languages, and Programming*, pages 69:1–69:15, 2016.
- 3 N. Ailon, R. Jaiswal, and C. Monteleoni. Streaming  $k$ -means approximation. In *Proc. of the 23rd Ann. Conf. on Neural Information Processing Systems*, pages 10–18, 2009.
- 4 D. Aloise, A. Deshpande, P. Hansen, and P. Popat. NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, May 2009.
- 5 D. Arthur, B. Manthey, and H. Röglin. Smoothed analysis of the  $k$ -means method. *J. ACM*, 58(5):19:1–19:31, 2011.

- 6 D. Arthur and S. Vassilvitskii.  $k$ -means++: The advantages of careful seeding. In *Proc. of the 18th Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 1027–1035, 2007.
- 7 P. Awasthi, A. Blum, and O. Sheffet. Stability yields a PTAS for  $k$ -median and  $k$ -means clustering. In *Proc. of the 51st Ann. IEEE Symp. on Foundations of Computer Science*, pages 309–318, 2010.
- 8 M.-F. Balcan, A. Blum, and A. Gupta. Approximate clustering without the approximation. In *Proc. of the 20th Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 1068–1077, 2009.
- 9 M.-F. Balcan and M. Braverman. Finding low error clusterings. In *COLT 2009 - The 22nd Conference on Learning Theory*, 2009.
- 10 Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proc. of the 37th Ann. IEEE Symp. on Foundations of Computer Science*, page 184, 1996.
- 11 Y. Bartal. On approximating arbitrary metrics by tree metrics. In *Proc. of the 30th Ann. ACM Symp. on Theory of Computing*, pages 161–168, 1998.
- 12 Y. Bartal, M. Charikar, and D. Raz. Approximating min-sum  $k$ -clustering in metric spaces. In *Proc. of the 33rd Ann. ACM Symp. on Theory of Computing*, pages 11–20, 2001.
- 13 B. Behsaz, Z. Friggstad, M. R. Salavatipour, and R. Sivakumar. Approximation algorithms for min-sum  $k$ -clustering and balanced  $k$ -median. *Algorithmica*, 81(3):1006–1030, 2019.
- 14 V. Braverman, D. Feldman, and H. Lang. New frameworks for offline and streaming coresets constructions. *CoRR*, abs/1612.00889, 2016.
- 15 V. Braverman, A. Meyerson, R. Ostrovsky, A. Roytman, M. Shindler, and B. Tagiku. Streaming  $k$ -means on well-clusterable data. In *Proc. of the 22nd Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 26–40, 2011.
- 16 P. Bunn and R. Ostrovsky. Secure two-party  $k$ -means clustering. In *Proc. of the 14th Ann. ACM Conf. on Computer and Communications Security*, pages 486–497, 2007.
- 17 K. Chen. On coresets for  $k$ -median and  $k$ -means clustering in metric and Euclidean spaces and their applications. *SIAM J. Comput.*, 39:923–947, 2009.
- 18 J. Chuzhoy and Y. Rabani. Approximating  $k$ -median with non-uniform capacities. In *Proc. of the 16th Ann. ACM-SIAM Symp. on Discrete Algorithms*, page 952–958, 2005.
- 19 V. Cohen-Addad, Karthik C. S., and E. Lee. On approximability of  $k$ -means,  $k$ -median, and  $k$ -minsum clustering, 2019.
- 20 V. Cohen-Addad and Karthik C.S. Inapproximability of clustering in  $l_p$  metrics. In *Proc. of the 60th Ann. IEEE Symp. on Foundations of Computer Science*, pages 519–539, 2019.
- 21 V. Cohen-Addad, P. N. Klein, and C. Mathieu. Local search yields approximation schemes for  $k$ -means and  $k$ -median in Euclidean and minor-free metrics. *Proc. of the 57th Ann. IEEE Symp. on Foundations of Computer Science*, pages 353–364, 2016.
- 22 Vincent Cohen-Addad and Chris Schwiegelshohn. On the local structure of stable clustering instances. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 49–60. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.14.
- 23 A. Czumaj and C. Sohler. Small space representations for metric min-sum  $k$ -clustering and their applications. In *Proc. of the 24th Ann. Conf. on Theoretical Aspects of Computer Science*, pages 536–548, 2007.
- 24 H. G. Demirci and S. Li. Constant approximation for capacitated  $k$ -median with  $(1+\epsilon)$ -capacity violation. *ArXiv*, abs/1603.02324, 2016.
- 25 J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proc. of the 35th Ann. ACM Symp. on Theory of Computing*, pages 448–455, 2003.
- 26 D. Feldman and M. Langberg. A unified framework for approximating and clustering data. In *Proc. of the 43rd Ann. ACM Symp. on Theory of Computing*, pages 569–578, 2011.
- 27 D. Feldman, M. Monemizadeh, and C. Sohler. A PTAS for  $k$ -means clustering based on weak coresets. In *Proc. of the 23rd Ann. Symp. on Computational Geometry*, pages 11–18, 2007.

## 16:16 Min-Sum Clustering (With Outliers)

- 28 W. Fernandez de la Vega, M. Karpinski, C. Kenyon, and Y. Rabani. Approximation schemes for clustering problems. In *Proc. of the 35th Ann. ACM Symp. on Theory of Computing*, pages 50–58, 2003.
- 29 W. Fernandez de la Vega and C. Kenyon. A randomized approximation scheme for metric MAX-CUT. *Journal of Computer and System Sciences*, 63(4):531–541, 2001.
- 30 Z. Friggstad, M. Rezapour, and M. R. Salavatipour. Local search yields a PTAS for  $k$ -means in doubling metrics. *Proc. of the 57th Ann. IEEE Symp. on Foundations of Computer Science*, pages 365–374, 2016.
- 31 V. Guruswami and P. Indyk. Embeddings and non-approximability of geometric problems. In *Proc. of the 14th Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 537–538, 2003.
- 32 N. Guttman-Beck and R. Hassin. Approximation algorithms for min-sum  $p$ -clustering. *Discret. Appl. Math.*, 89(1-3):125–142, 1998.
- 33 S. Har-Peled and A. Kushal. Smaller coresets for  $k$ -median and  $k$ -means clustering. *Discrete Comput. Geom.*, 37(1):3–19, 2007.
- 34 R. Hassin and E. Or. Min sum clustering with penalties. *European Journal of Operational Research*, 206(3):547–554, 2010.
- 35 M. Inaba, N. Katoh, and H. Imai. Applications of weighted Voronoi diagrams and randomization to variance-based  $k$ -clustering. In *Proc. of the 10th Ann. Symp. on Computational Geometry*, pages 332–339, 1994.
- 36 P. Indyk. A sublinear time approximation scheme for clustering in metric spaces. In *Proc. of the 40th Ann. IEEE Symp. on Foundations of Computer Science*, pages 154–159, 1999.
- 37 K. Jain and V. 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.
- 38 R. Jaiswal and N. Garg. Analysis of  $k$ -means++ for separable data. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 591–602, 2012.
- 39 J. Kleinberg. An impossibility theorem for clustering. In *Proc. of the 15th Int'l Conf. on Neural Information Processing Systems*, pages 463–470, 2002.
- 40 A. Kumar and R. Kannan. Clustering with spectral norm and the  $k$ -means algorithm. In *Proc. of the 51st Ann. IEEE Symp. on Foundations of Computer Science*, pages 299–308, 2010.
- 41 A. Kumar, Y. Sabharwal, and S. Sen. Linear time algorithms for clustering problems in any dimensions. In *Proc. of the 32nd Int'l Conf. on Automata, Languages and Programming*, pages 1374–1385, 2005.
- 42 M. Mahajan, P. Nimbhorkar, and K. Varadarajan. The planar  $k$ -means problem is NP-hard. In *Proc. of the 3rd Int'l Workshop on Algorithms and Computation*, pages 274–285, 2009.
- 43 J. Matoušek. On approximate geometric  $k$ -clustering. *Discrete & Computational Geometry*, 24(1):61–84, January 2000.
- 44 R. Ostrovsky, Y. Rabani, L. J. Schulman, and C. Swamy. The effectiveness of Lloyd-type methods for the  $k$ -means problem. In *Proc. of the 47th Ann. IEEE Symp. on Foundations of Computer Science*, pages 165–176, 2006.
- 45 L. J. Schulman. Clustering for edge-cost minimization. In *Proc. of the 32nd Ann. ACM Symp. on Theory of Computing*, pages 547–555, 2000.
- 46 K. Voevodski, M.-F. Balcan, H. Röglin, S.-H. Teng, and Y. Xia. Min-sum clustering of protein sequences with limited distance information. In *Proc. of the 1st Int'l Conf. on Similarity-Based Pattern Recognition*, pages 192–206, 2011.
- 47 R. B. Zadeh and S. Ben-David. A uniqueness theorem for clustering. In *Proc. of the 25th Ann. Conf. on Uncertainty in Artificial Intelligence*, pages 639–646, 2009.



# Streaming Approximation Resistance of Every Ordering CSP

Noah Singer ✉ 

Harvard College, Cambridge, MA, USA

Madhu Sudan ✉

School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA

Santhoshini Velusamy ✉

School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA

---

## Abstract

An ordering constraint satisfaction problem (OCSP) is given by a positive integer  $k$  and a constraint predicate  $\Pi$  mapping permutations on  $\{1, \dots, k\}$  to  $\{0, 1\}$ . Given an instance of OCSP( $\Pi$ ) on  $n$  variables and  $m$  constraints, the goal is to find an ordering of the  $n$  variables that maximizes the number of constraints that are satisfied, where a constraint specifies a sequence of  $k$  distinct variables and the constraint is satisfied by an ordering on the  $n$  variables if the ordering induced on the  $k$  variables in the constraint satisfies  $\Pi$ . Ordering constraint satisfaction problems capture natural problems including “Maximum acyclic subgraph (MAS)” and “Betweenness”.

In this work we consider the task of approximating the maximum number of satisfiable constraints in the (single-pass) streaming setting, where an instance is presented as a stream of constraints. We show that for every  $\Pi$ , OCSP( $\Pi$ ) is approximation-resistant to  $o(n)$ -space streaming algorithms, i.e., algorithms using  $o(n)$  space cannot distinguish streams where almost every constraint is satisfiable from streams where no ordering beats the random ordering by a noticeable amount. This space bound is tight up to polylogarithmic factors. In the case of MAS our result shows that for every  $\epsilon > 0$ , MAS is not  $1/2 + \epsilon$ -approximable in  $o(n)$  space. The previous best inapproximability result only ruled out a  $3/4$ -approximation in  $o(\sqrt{n})$  space.

Our results build on recent works of Chou, Golovnev, Sudan, Velingker, and Velusamy who show tight, linear-space inapproximability results for a broad class of (non-ordering) constraint satisfaction problems (CSPs) over arbitrary (finite) alphabets. Our results are obtained by building a family of appropriate CSPs (one for every  $q$ ) from any given OCSP, and applying their work to this family of CSPs. To convert the resulting hardness results for CSPs back to our OCSP, we show that the hard instances from this earlier work have the following “small-set expansion” property: If the CSP instance is viewed as a hypergraph in the natural way, then for every partition of the hypergraph into small blocks most of the hyperedges are incident on vertices from distinct blocks. By exploiting this combinatorial property, in combination with the hardness results of the resulting families of CSPs, we give optimal inapproximability results for all OCSPs.

**2012 ACM Subject Classification** Mathematics of computing → Approximation algorithms; Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Discrete optimization

**Keywords and phrases** Streaming approximations, approximation resistance, constraint satisfaction problems, ordering constraint satisfaction problems

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.17

**Category** APPROX

**Related Version** *Full Version including all proofs:* [arXiv:2105.01782](https://arxiv.org/abs/2105.01782) [22]

**Funding** M. Sudan and S. Velusamy supported in part by a Simons Investigator Award and NSF Award CCF 1715187.



© Noah Singer, Madhu Sudan, and Santhoshini Velusamy;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 17; pp. 17:1–17:19



Leibniz International Proceedings in Informatics

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

## 1 Introduction

In this work we consider the complexity of “approximating” “ordering constraint satisfaction problems (OCSPs)” in the “streaming setting”. We introduce these notions below before describing our results.

### 1.1 Orderings and Constraint Satisfaction Problems

In this work we consider optimization problems where the solution space is all possible orderings of  $n$  variables. The Travelling Salesperson Problem and most forms of scheduling fit this framework, though our work considers a more restricted class of problems, namely *ordering constraint satisfaction problems (OCSPs)*. OCSPs as a class were first defined by Guruswami, Håstad, Manokaran, Raghavendra, and Charikar [10]. To describe them here, we first set up some notation and terminology, and then give some examples.

We let  $[n]$  denote the set  $\{0, \dots, n-1\}$  and  $S_n$  denote the set of permutations on  $[n]$ , i.e., the set of bijections  $\sigma : [n] \rightarrow [n]$ . We sometimes use  $[\sigma(0) \sigma(1) \dots \sigma(n-1)]$  to denote  $\sigma : [n] \rightarrow [n]$ . The solution space of ordering problems is  $S_n$ , i.e., an *assignment* to  $n$  variables is given by  $\sigma \in S_n$ . Given  $k$  distinct integers  $a_0, \dots, a_{k-1}$  we define  $\text{ord}(a_0, \dots, a_{k-1})$  to be the unique permutation in  $S_k$  which sorts  $a_0, \dots, a_{k-1}$ . In other words,  $\text{ord}(a_0, \dots, a_{k-1})$  is the unique permutation  $\pi \in S_k$  such that  $a_{\pi(0)} < \dots < a_{\pi(k-1)}$ . A *k-ary ordering constraint function* is given by a predicate  $\Pi : S_k \rightarrow \{0, 1\}$ . An *ordering constraint application* on  $n$  variables is given by a constraint function  $\Pi$  and a  $k$ -tuple  $\mathbf{j} = (j_0, j_1, \dots, j_{k-1}) \in [n]^k$  where the  $j_i$ 's are distinct. In the interest of brevity we will often skip the term “ordering” below and further refer to constraint functions as “functions” and constraint applications as “constraints”. A constraint  $(\Pi, \mathbf{j})$  is *satisfied* by an assignment  $\sigma \in S_n$  if  $\Pi(\text{ord}(\sigma|_{\mathbf{j}})) = 1$ , where  $\sigma|_{\mathbf{j}}$  is the  $k$ -tuple  $(\sigma(j_0), \dots, \sigma(j_{k-1})) \in [n]^k$ .

A *maximum ordering constraint satisfaction problem*,  $\text{Max-OCSP}(\Pi)$ , is specified by a single ordering constraint function  $\Pi : S_k \rightarrow \{0, 1\}$ , for some positive integer arity  $k$ . An *instance* of  $\text{Max-OCSP}(\Pi)$  on  $n$  variables is given by  $m$  constraints  $C_0, \dots, C_{m-1}$  where  $C_i = (\Pi, \mathbf{j}(i))$ , i.e., the application of the function  $\Pi$  to the variables  $\mathbf{j}(i) = (j(i)_0, \dots, j(i)_{k-1})$ . (We omit  $\Pi$  from the description of a constraint  $C_i$  when clear from context.) The *value* of an ordering  $\sigma \in S_n$  on an instance  $\Psi = (C_0, \dots, C_{m-1})$ , denoted  $\text{val}_\Psi(\sigma)$ , is the fraction of constraints satisfied by  $\sigma$ , i.e.,  $\text{val}_\Psi(\sigma) = \frac{1}{m} \sum_{i \in [m]} \Pi(\text{ord}(\sigma|_{\mathbf{j}(i)}))$ . The optimal value of  $\Psi$  is defined as  $\text{val}_\Psi = \max_{\sigma \in S_n} \{\text{val}_\Psi(\sigma)\}$ .

The simplest, and arguably most interesting, problem which fits the  $\text{Max-OCSP}$  framework is the *maximum acyclic subgraph (MAS)* problem. In this problem, the input is a directed graph on  $n$  vertices, and the goal is to find an ordering of the vertices which maximize the number of forward edges. A simple depth-first search algorithm can decide whether a given graph  $G$  has a *perfect* ordering (i.e., one which has *no* back edges); however, Karp [17], in his famous list of 21 NP-complete problems, proved the NP-completeness of deciding whether, given a graph  $G$  and a parameter  $k$ , there exists an ordering of the vertices such that at least  $k$  edges are forward. For our purposes, MAS can be viewed as a 2-ary  $\text{Max-OCSP}$  problem, by defining the ordering constraint predicate  $\Pi_{\text{MAS}} : S_2 \rightarrow \{0, 1\}$  given by  $\Pi_{\text{MAS}}([0 \ 1]) = 1$  and  $\Pi_{\text{MAS}}([1 \ 0]) = 0$ , and associating vertices with variables and edges with constraints. Indeed, an edge/constraint  $(u, v)$  (where  $u, v \in [n]$  are distinct variables/vertices) will be satisfied by an assignment/ordering  $\sigma \in S_n$  iff  $\Pi_{\text{MAS}}(\text{ord}(\sigma|_{(u,v)})) = 1$ , or equivalently, iff  $\sigma(u) < \sigma(v)$ .

A second natural  $\text{Max-OCSP}$  problem is the *maximum betweenness (MaxBtw)* problem. This is a 3-ary OCSP in which an ordering  $\sigma$  satisfies a constraint  $(u, v, w)$  iff  $\sigma(v)$  is between  $\sigma(u)$  and  $\sigma(w)$ , i.e., iff  $\sigma(u) < \sigma(v) < \sigma(w)$  or  $\sigma(u) > \sigma(v) > \sigma(w)$ , and the goal is again

to find the maximum number of satisfiable constraints. This is given by the constraint satisfaction function  $\Pi_{\text{BtwN}} : S_3 \rightarrow \{0, 1\}$  given by  $\Pi_{\text{BtwN}}([0\ 1\ 2]) = 1, \Pi_{\text{BtwN}}([2\ 1\ 0]) = 1$ , and  $\Pi_{\text{BtwN}}(\pi) = 0$  for all other  $\pi \in S_3$ . The complexity of maximizing betweenness was originally studied by Opatrný [21], who proved that even deciding whether a set of betweenness constraints is perfectly satisfiable is NP-complete.

## 1.2 Approximability

In this work we consider the *approximability* of ordering constraint satisfaction problems. We say that a (randomized) algorithm  $A$  is an  $\alpha$ -*approximation algorithm* for Max-OCSP( $\Pi$ ) if for every instance  $\Psi$ ,  $\alpha \cdot \text{val}_\Psi \leq A(\Psi) \leq \text{val}_\Psi$  with probability at least  $2/3$  over the internal coin tosses of  $A$ . Thus our approximation factors  $\alpha$  are numbers in the interval  $[0, 1]$ .

Given  $\Pi : S_k \rightarrow \{0, 1\}$  let  $\rho(\Pi) = \frac{|\{\pi \in S_k \mid \Pi(\pi) = 1\}|}{k!}$  denote the probability that  $\Pi$  is satisfied by a random ordering. Every instance  $\Psi$  of Max-OCSP( $\Pi$ ) satisfies  $\text{val}_\Psi \geq \rho(\Pi)$  and thus the trivial algorithm that always outputs  $\rho(\Pi)$  is a  $\rho(\Pi)$ -approximation algorithm for Max-OCSP( $\Pi$ ). Under what conditions it is possible to beat this trivial approximation is a major open question.

For MaxBtwN, the trivial algorithm is a  $\frac{1}{3}$ -approximation. Chor and Sudan [4] showed that  $(\frac{47}{48} + \epsilon)$ -approximating MaxBtwN is NP-hard, for every  $\epsilon > 0$ . The  $\frac{47}{48}$  factor was improved to  $\frac{1}{2}$  by Austrin, Manokaran, and Wenner [1]. For MAS, the trivial algorithm is a  $\frac{1}{2}$ -approximation. Newman [20] showed that  $(\frac{65}{66} + \epsilon)$ -approximating MAS is NP-hard, for every  $\epsilon > 0$ . [1] improved the  $\frac{65}{66}$  to  $\frac{14}{15}$ , and Bhangale and Khot [2] further improved the factor to  $\frac{2}{3}$ .

We could hope that for every nontrivial nontrivial Max-OCSP( $\Pi$ ), it is NP-hard to even  $(\rho(\Pi) + \epsilon)$ -approximate Max-OCSP( $\Pi$ ) for any constant factor  $\epsilon > 0$ . This property is called *approximation resistance* (and we define it more carefully in the setting of streaming algorithms below). Approximation resistance based on NP-hardness is known for certain constraint satisfaction problems which do not fall under the Max-OCSP framework; this includes the seminal result of Håstad [13] that it is NP-hard to  $(\frac{7}{8} + \epsilon)$ -approximate Max3AND for any  $\epsilon > 0$ . But as far as we know, such results are lacking for *any* Max-OCSP problem.

Given this situation, Guruswami, Håstad, Manokaran, Raghavendra, and Charikar [10] proved the “next best thing”: assuming the unique games conjecture (UGC) of Khot [18], every Max-OCSP( $\Pi$ ) is approximation-resistant. But the question of proving approximation resistance for polynomial-time algorithms without relying on unproven assumptions such as UGC and  $P \neq NP$  remains unsolved. Towards this goal, in this work, we consider the approximability of Max-OCSP’s in the (*single-pass*) *streaming model*, which we define below.

## 1.3 Streaming algorithms

A (single-pass) streaming algorithm is defined as follows. An instance  $\Psi = (C_0, \dots, C_{m-1})$  of Max-OCSP( $\Pi$ ) is presented as a stream of constraints with the  $i$ th element of the stream being  $\mathbf{j}(i)$  where  $C_i = (\Pi, \mathbf{j}(i))$ . A streaming algorithm  $A$  updates its state with each element of the stream and at the end produces the output  $A(\Psi) \in [0, 1]$  (which is supposed to estimate  $\text{val}_\Psi$ ). The measure of complexity of interest to us is the space used by  $A$  and in particular we distinguish between algorithms that use space polylogarithmic in the input length and space that grows polynomially ( $\Omega(n^\delta)$  for  $\delta > 0$ ) in the input length.

We say that a problem Max-OCSP( $\Pi$ ) is *approximable (in the streaming setting)* if we can beat the trivial  $\rho(\Pi)$ -approximation algorithm by a positive constant factor. Specifically Max-OCSP( $\Pi$ ) is said to be *approximable* if for every  $\delta > 0$  there exists  $\epsilon > 0$  and a space  $O(n^\delta)$  algorithm  $A$  that is a  $(\rho(\Pi) + \epsilon)$ -approximation algorithm for Max-OCSP( $\Pi$ ), We say Max-OCSP( $\Pi$ ) is *approximation-resistant (in the streaming setting)* otherwise.

In recent years, investigations into CSP approximability in the streaming model have been strikingly successful, resulting in tight characterizations of streaming approximability for many problems [19, 14, 15, 12, 11, 16, 8, 6, 7, 5]. Most of these papers studied approximability, not of *ordering* CSPs, but of “non-ordering CSPs” where the variables can take values in a finite alphabet. ([12] and [11] are the exceptions, and we will discuss them below.) While single-pass streaming algorithms are a weaker model than general polynomial-time algorithms, we do remark that nontrivial approximations for many problems are possible in the streaming setting. In particular, the Max2AND problem is (roughly)  $\frac{4}{9}$ -approximable in the streaming setting (whereas the trivial approximation is a  $\frac{1}{4}$ -approximation) [8].

## 1.4 Main result and comparison to prior and related works

► **Theorem 1** (Main theorem). *For every  $k \in \mathbb{N}$  and every  $\Pi : \mathcal{S}_k \rightarrow \{0, 1\}$ , Max-OCSP( $\Pi$ ) is approximation resistant in the (single-pass) streaming setting. In particular for every  $\epsilon > 0$ , every  $(\rho(\Pi) + \epsilon)$ -approximation algorithm  $A$  for Max-OCSP( $\Pi$ ) requires  $\Omega(n)$  space.*

In particular our theorem implies that MAS is not  $1/2 + \epsilon$ -approximable in  $o(n)$  space for every  $\epsilon > 0$ , and MaxBtwn is not  $1/3 + \epsilon$ -approximable. Theorem 1 is restated in Section 3 along with several necessary lemmas; it follows readily from these lemmas and its proof is omitted.

Theorem 1 parallels the classical result of [10], who prove that Max-OCSP( $\Pi$ ) is approximation resistant with respect to *polynomial-time* algorithms, for every  $\Pi$ , assuming the unique games conjecture. In our setting of streaming algorithms, the only problem that seems to have been previously explored in the literature was MAS, and even in this case a tight approximability result was not known.

In the case of MAS, Guruswami, Velingker, and Velusamy [12] proved that for every  $\epsilon > 0$ , MAS is not  $(\frac{7}{8} + \epsilon)$ -approximable in  $o(\sqrt{n})$  space using a gadget reduction from the Boolean hidden matching problem [9]. A stronger  $o(\sqrt{n})$ -space,  $3/4$ -approximation hardness for MAS is indicated in the work of Guruswami and Tao [11], who prove streaming bounds for unique games, a “non-ordering” CSP problem, and suggest a reduction from unique games to MAS.

As far as we know, our result is the first tight approximability result for Max-OCSP( $\Pi$ ) for *any* non-constant  $\Pi$  in  $\Omega(n^\delta)$  space for any  $\delta > 0$ , and it yields tight approximability results for *every*  $\Pi$  in *linear* space. We remark that this linear space bound is also optimal (up to logarithmic factors); similarly to the observation in [5] for non-ordering CSPs, Max-OCSP( $\Pi$ ) values can be approximated arbitrarily well in  $\tilde{O}(n)$  space by subsampling  $O(n)$  constraints from the input instance and then solving the Max-OCSP( $\Pi$ ) problem on this subinstance exactly.<sup>1</sup>

Chakrabarti, Ghosh, McGregor, and Vorotnikova [3] recently also studied directed graph ordering problems (e.g., acyclicity testing,  $(s, t)$ -connectivity, topological sorting) in the streaming setting. For the problems that considered in [3], their work gives *super-linear* space lower bounds even for multi-pass streaming algorithms. Note that for our problems an  $\tilde{O}(n)$  upper bound holds, suggesting that their problems are not OCSPs. Indeed this is true, but one of the problems considered is close enough to MAS to allow a more detailed comparison. The specific problem is the *minimum feedback arc set* (MFAS) problem, the goal of which is to output the fractional size of the smallest set of edges whose removal

<sup>1</sup> This assumes a definition of streaming complexity which makes no restriction on time complexity. Of course, if we restrict to polynomial time, then assuming the unique games conjecture, no nontrivial approximation will be possible.

produces an acyclic subgraph. In other words, the sum of MFAS value of a graph and the MAS value of the graph is exactly one. [3] proved that for every  $\kappa > 1$ ,  $\kappa$ -approximating<sup>2</sup> the MFAS value requires  $\Omega(n^2)$  space in the streaming setting (for a single pass, and more generally  $\Omega(n^{1+\Omega(1/p)})/p^{O(1)}$  space for  $p$  passes). Note that such lower bounds are obtained using instances with optimum MFAS values that are  $o(1)$ . Thus the MAS values in the same graph are  $1 - o(1)$  (even in the **NO** instances) and thus these results usually do not imply any hardness of approximation for MAS.

## 1.5 Techniques

Our general approach is to start with a hardness result for CSPs over alphabets of size  $q$  (i.e., constraint satisfaction problems where the variables take values in  $[q]$ ), and then to reduce these CSPs to the OCSF at hand. While this general approach is not new, the optimality of our results seems to come from the fact that we choose the CSP problem carefully, and are able to get optimal hardness results for problems of our choice thanks to a general result of Chou, Golovnev, Sudan, Velingker and Velusamy [5]. Thus whereas previous approaches towards proving hardness of MAS, for example, were unable to get optimal hardness results for MAS despite starting with optimal hardness results of the source (unique games), by choosing our source problem more carefully we manage to get optimal hardness results. In the remainder of this section, we describe and motivate this approach towards proving the approximation-resistance of Max-OCSF's.

### 1.5.1 Special case: The intuition for MAS

We start by describing our proof technique for the special case of the MAS problem. In this section, for readability, we (mostly) use the language of graphs, edges, and vertices instead of instances, constraints, and variables.

Similarly to earlier work in the setting of streaming approximability (e.g., [14]), we prove inapproximability of MAS by exhibiting a pair of distributions, which we denote  $\mathcal{G}^Y$  and  $\mathcal{G}^N$ , satisfying the following two properties:

1.  $\mathcal{G}^Y$  and  $\mathcal{G}^N$  are “indistinguishable” to streaming algorithms (to be defined formally below).
2. (With high probability)  $\mathcal{G}^Y$  has high MAS values ( $\approx 1$ ) and  $\mathcal{G}^N$  has low MAS values ( $\approx \frac{1}{2}$ ).

The existence of such distributions would suffice to establish the theorem: there cannot be any streaming approximation for MAS, since any such algorithm would be able to distinguish these distributions. But how are we to actually construct distributions  $\mathcal{G}^Y$  and  $\mathcal{G}^N$  satisfying these properties?

The strategy which has proved successful in past work for proving streaming approximation resistance of other varieties of CSPs was roughly to let the  $\mathcal{G}^N$  graphs be completely random, while  $\mathcal{G}^Y$  graphs are sampled with “hidden structure”, which is essentially a very good assignment. Then, one would show that streaming algorithms cannot detect the existence of such hidden structure, via a reduction to a communication game (typically a variant of Boolean hidden matching [9, 24]). In our setting, we might hope that the hidden structure could simply be an ordering; that is, we could hope to define  $\mathcal{G}^Y$  by first sampling a random ordering of the vertices, then sampling edges which go forward with respect to this ordering, and then perhaps adding some noise. But unfortunately, we lack the techniques to prove communication lower bounds when orderings are the hidden structure.

<sup>2</sup> For minimization problems a  $\kappa$  approximation is one whose value is at least the minimum value and at most  $\kappa$  times larger than the minimum. Thus approximation factors are larger than 1.

Hence, instead of seeking a direct proof of an indistinguishability result, in this paper, we turn back to earlier indistinguishability results proven in the context of *non-ordering CSPs*. In this setting, variables take on values in an alphabet  $[q]$ , and constraints specify allowed values of subsets of the variables. In particular, two distinct variables may take on the same value in  $[q]$ , whereas in the ordering setting, every variable in  $[n]$  must get a distinct value in  $[n]$ . (See Subsection 4.1 for a formal definition.) We will set  $q$  to be a large constant, carefully design a non-ordering CSP function, employ past results (i.e., [5]) to characterize its streaming inapproximability, examine the  $\mathcal{G}^Y$  and  $\mathcal{G}^N$  graphs created in the reduction, and then show that  $\mathcal{G}^N$  graphs have low MAS values while the hidden structure in the  $\mathcal{G}^Y$  graphs – even if it isn’t an ordering per se – guarantees high MAS values.

Why would we expect such an idea to work out, and how do we properly choose the non-ordering CSP constraint function? To begin, this constraint function will be a 2-ary function  $f : [q]^2 \rightarrow \{0, 1\}$ . Let  $\text{Max-CSP}(f)$  denote the non-ordering CSP problem of maximizing the number of  $f$  constraints satisfied by an assignment  $\mathbf{b} \in [q]^n$ . We will view an input graph  $G$  simultaneously as an instance of MAS and as an instance of  $\text{Max-CSP}(f)$ , with the same underlying set of edges/constraints. For a graph  $G$ , let  $\text{val}_G$  denote its MAS value and  $\overline{\text{val}}_G$  its value in  $\text{Max-CSP}(f)$ . We will choose  $f$  so that the indistinguishable hard distributions  $\mathcal{G}^Y$  and  $\mathcal{G}^N$  (originating from the reduction of [5]) have the following four properties:

1. With high probability over  $G \sim \mathcal{G}^Y$ ,  $\overline{\text{val}}_G \approx 1$ .
2. With high probability over  $G \sim \mathcal{G}^N$ ,  $\overline{\text{val}}_G \approx \frac{1}{2}$ .
3. For all  $G$ ,  $\text{val}_G \geq \overline{\text{val}}_G$ .
4. With high probability over  $G \sim \mathcal{G}^N$ ,  $\text{val}_G$  is not much larger than  $\overline{\text{val}}_G$ .

Together, these items will suffice to prove the theorem since item 2 and item 4 together imply that with high probability over  $G \sim \mathcal{G}^N$ ,  $\text{val}_G \approx \frac{1}{2}$ , while item 1 and item 3 together imply that with high probability over  $G \sim \mathcal{G}^Y$ ,  $\text{val}_G \approx 1$ .

Concretely, we setup the non-ordering CSP function as follows. Recall that  $\Pi_{\text{MAS}}([0\ 1]) = 1$  while  $\Pi_{\text{MAS}}([1\ 0]) = 0$ . We define the constraint function  $f_{\text{MAS}}^q : [q]^2 \rightarrow \{0, 1\}$  by  $f_{\text{MAS}}^q(x, y) = 1$  iff  $x < y$ . Note that  $f_{\text{MAS}}^q$  is supported on  $\frac{q(q-1)}{2} \approx \frac{1}{2}$  pairs in  $[q]^2$ . We first show that [5]’s results imply that  $\text{Max-CSP}(f_{\text{MAS}}^q)$  is approximation-resistant, and pick  $\mathcal{G}^Y$  and  $\mathcal{G}^N$  as the **YES** and **NO** distributions witnessing this result. This immediately yields item 1 and item 2 above. It remains to prove item 4 and item 3. In the remainder of this subsection, we sketch the proofs; see Figure 1 for a visual depiction, and Section 4 for the formal proofs.

Towards item 3, we take advantage of the fact that  $\text{Max-CSP}(f_{\text{MAS}}^q)$  captures a “ $q$ -coarsening” of MAS. We consider an arbitrary  $\text{Max-CSP}(f_{\text{MAS}}^q)$ -assignment  $\mathbf{b} \in [q]^n$  for a graph  $G$ , which assigns to the  $i$ -th vertex a value  $b_i \in [q]$ . We construct an ordering of  $G$ ’s vertices by first placing the “block” of vertices assigned value 0, then the block of vertices assigned 1, etc., finally placing the vertices assigned value  $q - 1$ . (Within any particular block, the vertices may be ordered arbitrarily.) Now whenever an edge  $(u, v)$  is satisfied by  $\mathbf{b}$  when viewing  $G$  as an instance of  $\text{Max-CSP}(f_{\text{MAS}}^q)$  – that is, whenever  $b_v > b_u$  – the same edge will be satisfied by our constructed ordering when viewing  $G$  as an instance of MAS. Hence  $\text{val}_G \geq \overline{\text{val}}_G$ .

Towards item 4, we can no longer use the results of [5] as a black box. Instead, we show that the graphs  $\mathcal{G}^N$  are “small partition expanders” in a specific sense: any partition of the constraint graph into  $q$  roughly equal sized blocks has very few edges, specifically a  $o(1)$  fraction, which lie *within* the blocks. Now, we think of an ordering  $\sigma \in S_n$  variables as dividing the  $n$  variables into  $q$  blocks with variables  $\sigma(0), \dots, \sigma(n/q - 1)$  being in the first block,  $\sigma(n/q), \dots, \sigma(2n/q - 1)$  being in the second block and so on. Whenever an edge  $(u, v)$  is satisfied by  $\sigma$  when viewing  $G$  as an instance of MAS, it will also be satisfied by our

constructed ordering when viewing  $G$  as an instance of  $\text{Max-CSP}(f_{\text{MAS}}^q)$ , unless  $u$  and  $v$  end up in the same block; but by the small partition expansion condition, this happens only for  $o(1)$  fraction of the edges. Hence  $\text{val}_G \leq \overline{\text{val}}_G + o(1)$ .

We remark in passing that our notion of coarsening is somewhat similar to, but not the same as, that used in previous works, notably [10]. In particular the techniques used to compare the OCSP value (before coarsening) with the non-ordering CSP value (after coarsening) are somewhat different: Their analysis involves more sophisticated tools such as influence of variables and Gaussian noise stability. The proof of item 4 in our setting, in contrast, uses a more elementary analysis of the type common with random graphs. Finally, we remark that in the rest of the paper, in the interest of self-containedness, our construction will “forget” about  $\text{Max-CSP}(f_{\text{MAS}}^q)$ , define the distributions  $\mathcal{G}^Y$  and  $\mathcal{G}^N$  explicitly, and treat  $\overline{\text{val}}_G$  simply as an artifact of the analysis which calculates the MAS values of  $\mathcal{G}^Y$  and  $\mathcal{G}^N$ , but we hope that this discussion has motivated the construction.

## 1.5.2 Extending to general ordering CSPs

Extending the idea to other OCSPs involves two additional steps. Given the constraint function  $\Pi$  (of arity  $k$ ) and positive integer  $q$ , we define  $f_{\Pi}^q$  analogously to  $f_{\text{MAS}}^q$ . We then explicitly describe the **YES** and **NO** distributions of  $\text{Max-CSP}(f_{\Pi}^q)$  which the general theorem of [5] shows are indistinguishable to  $o(n)$  space algorithms. Crucial to this application is the observation that  $f_{\Pi}^q$  is a “ $1 - k - 1/q$ -wide” function, where  $f_{\Pi}^q$  is  $\omega$ -wide if there exists a vector  $\mathbf{v} = (v_0, \dots, v_{k-1}) \in [q]^k$  such that for an  $\omega$ -fraction of  $a \in [q]$ , we have  $f_{\Pi}^q(v_0 + a, \dots, v_{k-1} + a) = 1$ . This would allow us to conclude that  $\text{Max-CSP}(f_{\Pi}^q)$  is hard to approximate to within factor of roughly  $\rho/\omega$ , though as in the special case of MAS we do not use this result explicitly.<sup>3</sup> Instead, the second step of our proof replicates item 4 above. We give an analysis of the partition expansion in the **NO** instances arising from the construction in [5]. Specifically we show that the constraint hypergraph is now a “small partition hypergraph expander”, in the sense that any partition into  $q$  roughly equal sized blocks would have very few hyperedges that contain even two vertices from the same block. With these two additional ingredients in place, and following the same template as in the hardness for MAS, we immediately get the approximation resistance of  $\text{Max-OCSP}(\Pi)$  for general  $\Pi$ .

### 1.5.2.1 This version

Our current results improve on a previous version of this paper [23] that gave only  $\Omega(\sqrt{n})$  space lower bounds for all OCSPs. Our improvement to  $\Omega(n)$  space lower bounds comes by invoking the more recent results of [5], whereas our previous version used the strongest lower bounds for CSPs that were available at the time from an earlier work of Chou, Golovnev, Sudan, and Velusamy [7]. The results of [7] are quantitatively weaker for the problems considered in [5], though their results apply to a broader collection of problems. Interestingly for our application, which covers *all* OCSPs, the narrower set of problems considered in [5] suffices. We also note that the proof in this version of our paper is more streamlined thanks to the notion of “wide” constraints introduced and used in [5].

We omit some proofs in this conference version due to space constraints; see the relevant sections in our full version [22].

<sup>3</sup> Indeed, the “width” observation is involved in the proof of item 1 and item 2 even in the MAS case (with  $k = 2$ ).

### 1.5.2.2 Organization of the rest of the paper

In Section 2 we introduce some notation we use and background material. In Section 3 we prove our main theorem, Theorem 1. In this section we also introduce two distributions on Max-OCSP( $\Pi$ ) instances, the **YES** distribution and the **NO** distribution, and state lemmas asserting that these distributions are concentrated on instances with high, and respectively low, OCSP value; and that these distributions are indistinguishable to single-pass small space streaming algorithms. We prove the lemmas on the OCSP values in Section 4, and describe the indistinguishability lemma in Section 5.

## 2 Preliminaries and definitions

### 2.1 Basic notation

Some of the notation we use is already introduced in Subsection 1.1. Here we introduce some more notation we use.

The *support* of an ordering constraint function  $\Pi : \mathcal{S}_k \rightarrow \{0, 1\}$  is the set  $\text{supp}(\Pi) = \{\pi \in \mathcal{S}_k \mid \Pi(\pi) = 1\}$ .

Addition of elements in  $[q]$  is implicitly taken modulo  $q$ .

Throughout this paper we will be working with  $k$ -uniform *ordered* hypergraphs, or simply  $k$ -hypergraphs, defined in the sequel. Given a finite set  $V$ , an (ordered, self-loop-free)  $k$ -hyperedge  $e = (v_1, \dots, v_k)$  is a sequence of  $k$  distinct elements  $v_1, \dots, v_k \in V$ . We stress that the ordering of vertices within an edge is important to us. An (ordered, self-loop-free, multi-)  $k$ -hypergraph  $G = (V, E)$  is given by a set of vertices  $V$  and a multiset  $E = E(G) \subseteq V^k$  of  $k$ -hyperedges. A  $k$ -hyperedge  $\mathbf{e}$  is *incident* on a vertex  $v$  if  $v$  appears in  $\mathbf{e}$ . Let  $\Gamma(\mathbf{e}) \subseteq V$  denote the set of vertices to which a  $k$ -hyperedge  $\mathbf{e}$  is incident, and let  $m = m(G)$  denote the number of  $k$ -hyperedges in  $G$ .

A  $k$ -hypergraph is a  $k$ -*hypermatching* if it has the property that no pair of (distinct)  $k$ -hyperedges is incident on the same vertex. For  $\alpha \leq \frac{1}{k}$ , an  $\alpha$ -*partial  $k$ -hypermatching* is a  $k$ -hypermatching which contains  $\alpha n$   $k$ -hyperedges. We let  $\mathcal{H}_{k,n,\alpha}$  denote the uniform distribution over all  $\alpha$ -partial  $k$ -hypermatchings on  $[n]$ .

A vector  $\mathbf{b} = (b_0, \dots, b_{n-1}) \in [q]^n$  may be viewed as a  $q$ -*partition* of  $[n]$  into *blocks*  $\mathbf{b}^{-1}(0), \dots, \mathbf{b}^{-1}(q-1)$ , where the  $i$ -th block  $\mathbf{b}^{-1}(i)$  is defined as the set of indices  $\{j \in [n] : b_j = i\}$ . Given  $\mathbf{b} = (b_0, \dots, b_{n-1}) \in [q]^n$  and an indexing vector  $\mathbf{j} = (j_0, \dots, j_{k-1}) \in [n]^k$ , we define  $\mathbf{b}|_{\mathbf{j}} = (b_{j_0}, \dots, b_{j_{k-1}})$ .

Given an instance  $\Psi$  of Max-OCSP( $\Pi$ ) on  $n$  variables, we define the *constraint hypergraph*  $G(\Psi)$  to be the  $k$ -hypergraph on  $[n]$ , where each  $k$ -hyperedge corresponds to a constraint (given by the exact same  $k$ -tuple). We also let  $m(\Psi)$  denote the number of constraints in  $\Psi$  (equiv., the number of  $k$ -hyperedges in  $G(\Psi)$ ).

### 2.2 Concentration bound

We also require the following form of *Azuma's inequality*, a concentration inequality for submartingales. For us the following form, for Boolean-valued random variables with bounded conditional expectations taken from Kapralov and Krachun [16], is particularly convenient.

► **Lemma 2** ([16, Lemma 2.5]). *Let  $X_0, \dots, X_{m-1}$  be (not necessarily independent)  $\{0, 1\}$ -valued random variables, such that for some  $p \in (0, 1)$ ,  $\mathbb{E}[X_i \mid X_0, \dots, X_{i-1}] \leq p$  for every  $i \in [m]$ . Then if  $\mu := pm$ , for every  $\nu > 0$ ,*

$$\Pr[X_0 + \dots + X_{m-1} \geq \mu + \nu] \leq \exp\left(-\frac{1}{2} \cdot \frac{\nu^2}{\mu + \nu}\right).$$



### 3 The streaming space lower bound

In this section we restate our main theorem, and state the lemmas which are necessary for its proof.

► **Theorem 1** (Main theorem). *For every  $k \in \mathbb{N}$  and every  $\Pi : \mathcal{S}_k \rightarrow \{0, 1\}$ ,  $\text{Max-OCSP}(\Pi)$  is approximation resistant in the (single-pass) streaming setting. In particular for every  $\epsilon > 0$ , every  $(\rho(\Pi) + \epsilon)$ -approximation algorithm  $A$  for  $\text{Max-OCSP}(\Pi)$  requires  $\Omega(n)$  space.*

Our lower bound is proved, as is usual for such statements, by showing that no small space algorithm can “distinguish” **YES** instances with OCSP value at least  $1 - \epsilon/2$ , from **NO** instances with OCSP value at most  $\rho(\Pi) + \epsilon/2$ . Such a statement is in turn proved by exhibiting two families of distributions, the **YES** distributions and the **NO** distributions, and showing these are indistinguishable. Specifically we choose some parameters  $q, T, \alpha$  and a permutation  $\pi \in \mathcal{S}_k$  carefully and define two distributions  $\mathcal{G}^Y = \mathcal{G}_{q,n,\alpha,T}^{Y,\pi}(\Pi)$  and  $\mathcal{G}^N = \mathcal{G}_{q,n,\alpha,T}^N(\Pi)$ . We claim that for our choice of parameters  $\mathcal{G}^Y$  is supported on instances with value at least  $1 - \epsilon/2$  – this is asserted in Lemma 5. Similarly we claim that  $\mathcal{G}^N$  is mostly supported (with probability  $1 - o(1)$ ) on instances with value at most  $\rho(\Pi) + \epsilon/2$  (see Lemma 6). Finally we assert in Lemma 7 that any algorithm that distinguishes  $\mathcal{G}^Y$  from  $\mathcal{G}^N$  with “advantage” at least  $1/8$  (i.e., accepts  $\Psi \sim \mathcal{G}^Y$  with probability  $1/8$  more than  $\Psi \sim \mathcal{G}^N$ ) requires  $\Omega(n)$  space.

#### 3.1 Distribution of hard instances

For  $\ell, k \in [q]$ , define the  $k$ -tuple of “contiguous” values  $\mathbf{v}_q^{(\ell)} = (\ell, \dots, \ell + k - 1) \in [q]^k$ . Crucially, since the addition here is taken modulo  $q$ , we may have  $\ell + k - 1 < \ell$  and in particular  $\text{ord}(\mathbf{v}_q^{(\ell)})$  may not be the identity.

For a  $k$ -tuple  $\mathbf{a} = (a_0, \dots, a_{k-1})$  and a permutation  $\pi \in \mathcal{S}_k$ , define the *permuted*  $k$ -tuple  $\mathbf{a}_\pi$  as  $(a_{\pi^{-1}(0)}, \dots, a_{\pi^{-1}(k-1)})$ . In particular, we have  $(\mathbf{v}_q^{(\ell)})_\pi = (\pi^{-1}(0) + \ell, \dots, \pi^{-1}(k-1) + \ell)$ . We define  $\mathbf{a}_\pi$  in this way because:

► **Proposition 3.** *If  $\mathbf{a}$  is a  $k$ -tuple of distinct integers, then  $\text{ord}(\mathbf{a}_\pi) = \text{ord}(\mathbf{a}) \circ \pi$  (where  $\circ$  denotes composition of permutations).*

**Proof.** Recall that  $\text{ord}(\mathbf{a})$  is the unique permutation  $\tau$  such that  $a_{\tau(0)} < \dots < a_{\tau(k-1)}$ . Let  $\tau = \text{ord}(\mathbf{a})$ , and let  $\sigma = \text{ord}(\mathbf{a}_\pi)$ , so that  $\sigma$  is the unique permutation such that  $a_{\sigma(\pi^{-1}(0))} < \dots < a_{\sigma(\pi^{-1}(k-1))}$ . Then  $\tau = \sigma \circ \pi^{-1}$ . Hence  $\tau \circ \pi = \sigma$ , as desired. ◀

We now formally define our **YES** and **NO** distributions for  $\text{Max-OCSP}(\Pi)$ .

► **Definition 4** ( $\mathcal{G}_{q,n,\alpha,T}^{Y,\pi}(\Pi)$  and  $\mathcal{G}_{q,n,\alpha,T}^N(\Pi)$ ). *For  $k \in \mathbb{N}$  and  $\Pi : \mathcal{S}_k \rightarrow \{0, 1\}$ , let  $q, n, T \in \mathbb{N}$ ,  $\alpha > 0$ , and let  $B = N$  or  $B = (Y, \pi)$  for some  $\pi \in \text{supp}(\Pi)$ . We define the distribution  $\mathcal{G}_{q,n,\alpha,T}^B$ , over  $n$ -variable  $\text{Max-OCSP}(\Pi)$  instances, as follows:*

1. *Sample a uniformly random  $q$ -partition  $\mathbf{b} = (b_0, \dots, b_{n-1}) \in [q]^n$ .*
2. *Sample  $T$  hypermatchings independently  $\tilde{G}_0, \dots, \tilde{G}_{T-1} \sim \mathcal{H}_{k,n,\alpha}$ .*
3. *For each  $t \in [T]$ , do the following:*
  - *Let  $G_t$  be an empty  $k$ -hypergraph on  $[n]$ .*
  - *For each  $k$ -hyperedge  $\tilde{\mathbf{e}} = (j_0, \dots, j_{k-1}) \in E(\tilde{G}_t)$ :*
    - *(YES) If  $B = (Y, \pi)$ , and there exists  $\ell \in [q]$  such that  $\mathbf{b}|_j = (\mathbf{v}_q^{(\ell)})_\pi$ , add  $\tilde{\mathbf{e}}$  to  $G_t$  with probability  $\frac{1}{q}$ .*
    - *(NO) If  $B = N$ , add  $\tilde{\mathbf{e}}$  to  $G_t$  with probability  $\frac{1}{q^k}$ .*

## 17:10 Streaming Approximation Resistance of Every Ordering CSP

4. Let  $G := G_0 \cup \dots \cup G_{T-1}$ .
5. Return the Max-OCSP( $\Pi$ ) instance  $\Psi$  on  $n$  variables given by the constraint hypergraph  $G$ .

We say that an algorithm **ALG** achieves advantage  $\delta$  in distinguishing  $\mathcal{G}_{q,n,\alpha,T}^{Y,\pi}(\Pi)$  from  $\mathcal{G}_{q,n,\alpha,T}^N(\Pi)$  if there exists an  $n_0$  such that for all  $n \geq n_0$ , we have

$$\left| \Pr_{\Psi \sim \mathcal{G}_{q,n,\alpha,T}^{Y,\pi}(\Pi)} [\mathbf{ALG}(\Psi) = 1] - \Pr_{\Psi \sim \mathcal{G}_{q,n,\alpha,T}^N(\Pi)} [\mathbf{ALG}(\Psi) = 1] \right| \geq \delta.$$

We make several remarks on this definition. Firstly, note that the constraints within  $\mathcal{G}_{q,n,\alpha,T}^{Y,\pi}(\Pi)$  and  $\mathcal{G}_{q,n,\alpha,T}^N(\Pi)$  do not directly depend on  $\Pi$ . We still parameterize the distributions by  $\Pi$ , since they are formally distributions over Max-OCSP( $\Pi$ ) instances;  $\Pi$  also determines the set of allowed permutations  $\pi$  in the **YES** case as well as the underlying arity  $k$ . However, we will omit the parameterization ( $\Pi$ ) when clear from context. Secondly, we note that when sampling an instance from  $\mathcal{G}_{q,n,\alpha,T}^N$ , the partition  $\mathbf{b}$  has no effect, and so  $\mathcal{G}_{q,n,\alpha,T}^N$  is completely random. Hence these instances fit into the standard paradigm for streaming lower bounds of “random graphs vs. random graphs with hidden structure”. Finally, we observe that the number of constraints in both distributions is distributed as a sum of  $m = n\alpha T$  independent Bernoulli( $\frac{1}{q^k}$ ) random variables.

In the following section we state lemmas which highlight the main properties of the distributions above. See Figure 1 in Appendix A for a visual interpretation of the distributions in the case of MAS.

### 3.2 Statement of key lemmas

Our first lemma shows that  $\mathcal{G}^Y$  is supported on instances of high value.

► **Lemma 5** ( $\mathcal{G}^Y$  has high Max-OCSP( $\Pi$ ) values). *For every ordering constraint satisfaction function  $\Pi$ , every  $\pi \in \text{supp}(\Pi)$  and  $\Psi \sim \mathcal{G}_{q,n,\alpha,T}^{Y,\pi}$ , we have  $\text{val}_\Psi \geq 1 - \frac{k-1}{q}$  (i.e., this occurs with probability 1).*

We sketch the proof of Lemma 5 in Subsection 4.2. Next we assert that  $\mathcal{G}^N$  is supported mostly on instances of low value.

► **Lemma 6** ( $\mathcal{G}^N$  has low Max-OCSP( $\Pi$ ) values). *For every  $k$ -ary ordering constraint function  $\Pi : \mathcal{S}_k \rightarrow \{0, 1\}$ , and every  $\epsilon > 0$ , there exists  $q_0 \in \mathbb{N}$  and  $\alpha_0 \geq 0$  such that for all  $q \geq q_0$  and  $\alpha \leq \alpha_0$ , there exists  $T_0 \in \mathbb{N}$  such that for all  $T \geq T_0$ , for sufficiently large  $n$ , we have*

$$\Pr_{\Psi \sim \mathcal{G}_{q,n,\alpha,T}^N} \left[ \text{val}_\Psi \geq \rho(\Pi) + \frac{\epsilon}{2} \right] \leq 0.01.$$

We discuss, and partially prove, Lemma 6 in Subsection 4.3. We note that this lemma is more technically involved than Lemma 5 and this is the proof that needs the notion of “small partition expanders”. Finally the following lemma asserts the indistinguishability of  $\mathcal{G}^Y$  and  $\mathcal{G}^N$  to small space streaming algorithms and is discussed in Section 5. We remark that this lemma follows directly from the work of [5].

► **Lemma 7**. *For every  $q, k \in \mathbb{N}$  there exists  $\alpha_0(k) > 0$  such that for every  $T \in \mathbb{N}$ ,  $\alpha \in (0, \alpha_0(k)]$  the following holds: For every  $\Pi : \mathcal{S}_k \rightarrow \{0, 1\}$  and  $\pi \in \text{supp}(\Pi)$ , every streaming algorithm **ALG** distinguishing  $\mathcal{G}_{q,n,\alpha,T}^{Y,\pi}$  from  $\mathcal{G}_{q,n,\alpha,T}^N$  with advantage  $1/8$  for all lengths  $n$  uses space  $\Omega(n)$ .*

Assuming Lemma 5, Lemma 6, and Lemma 7 the proof of Theorem 1 is straightforward and is omitted.

## 4 Bounds on Max-OCSP( $\Pi$ ) values of $\mathcal{G}^Y$ and $\mathcal{G}^N$

The goal of this section is to discuss, and at least partially prove, our technical lemmas which lower bound the Max-OCSP( $\Pi$ ) values of  $\mathcal{G}_{q,n,\alpha,T}^{Y,\pi}$  (Lemma 5) and upper bound the Max-OCSP( $\Pi$ ) values of  $\mathcal{G}_{q,n,\alpha,T}^N$  (Lemma 6).

### 4.1 CSPs and coarsening

In preparation for proving the lemmas, we recall the definition of (non-ordering) *constraint satisfaction problems* (CSPs), whose solution spaces are  $[q]^n$  (as opposed to  $\mathcal{S}_n$ ), and define an operation called *q-coarsening* on Max-OCSP's, which restricts the solution space from  $\mathcal{S}_n$  to  $[q]^n$ .

A *maximum constraint satisfaction problem*, Max-CSP( $f$ ), is specified by a single constraint function  $f : [q]^k \rightarrow \{0, 1\}$ , for some positive integer  $k$ . An *instance* of Max-CSP( $f$ ) on  $n$  variables is given by  $m$  constraints  $C_0, \dots, C_{m-1}$  where  $C_i = (f, \mathbf{j}(i))$ , i.e., the application of the function  $f$  to the variables  $\mathbf{j}(i) = (j(i)_0, \dots, j(i)_{k-1})$ . (Again,  $f$  is omitted when clear from context.) The *value* of an assignment  $\mathbf{b} \in [q]^n$  on an instance  $\Phi = (C_0, \dots, C_{m-1})$ , denoted  $\overline{\text{val}}_{\Phi}^q(\mathbf{b})$ , is the fraction of constraints satisfied by  $\mathbf{b}$ , i.e.,  $\overline{\text{val}}_{\Phi}^q(\mathbf{b}) = \frac{1}{m} \sum_{i \in [m]} f(\mathbf{b}|_{\mathbf{j}(i)})$ , where (recall)  $\mathbf{b}|_{\mathbf{j}} = (b_{j_0}, \dots, b_{j_{k-1}})$  for  $\mathbf{b} = (b_0, \dots, b_{n-1})$ ,  $\mathbf{j} = (j_0, \dots, j_{k-1})$ . The optimal value of  $\Phi$  is defined as  $\overline{\text{val}}_{\Phi}^q = \max_{\mathbf{b} \in [q]^n} \{\overline{\text{val}}_{\Phi}^q(\mathbf{b})\}$ .

► **Definition 8** (*q-coarsening*). Let  $\Pi$  be a  $k$ -ary Max-OCSP and let  $q \in \mathbb{N}$ . The *q-coarsening* of  $\Pi$  is the  $k$ -ary Max-CSP problem Max-CSP( $f_{\Pi}^q$ ) where we define  $f_{\Pi}^q : [q]^k \rightarrow \{0, 1\}$  as follows: For  $\mathbf{a} \in [q]^k$ ,  $f_{\Pi}^q(\mathbf{a}) = 1$  iff the entries in  $\mathbf{a}$  are all distinct and  $\Pi(\text{ord}(\mathbf{a})) = 1$ . The *q-coarsening* of an instance  $\Psi$  of Max-OCSP( $\Pi$ ) is the instance  $\Phi$  of Max-CSP( $f_{\Pi}^q$ ) given by the identical collection of constraints.

The following lemma captures the idea that coarsening restricts the space of possible solutions; compare to Lemma 15 below.

► **Lemma 9.** If  $q \in \mathbb{N}$ ,  $\Psi$  is an instance of Max-OCSP( $\Pi$ ), and  $\Phi$  is the *q-coarsening* of  $\Psi$ , then  $\text{val}_{\Psi} \geq \overline{\text{val}}_{\Phi}^q$ .

**Proof.** We will show that for every assignment  $\mathbf{b} \in [q]^n$  to  $\Phi$ , we can construct an assignment  $\sigma \in \mathcal{S}_n$  to  $\Psi$  such that  $\text{val}_{\Psi}(\sigma) \geq \overline{\text{val}}_{\Phi}^q(\mathbf{b})$ . Consider an assignment  $\mathbf{b} \in [q]^n$ . Let  $\sigma$  be the ordering on  $[n]$  given by placing the blocks  $\mathbf{b}^{-1}(0), \dots, \mathbf{b}^{-1}(q-1)$  in order (within each block, we enumerate the indices arbitrarily). Consider any constraint  $C = \mathbf{j} = (j_0, \dots, j_{k-1})$  in  $\Phi$  which is satisfied by  $\mathbf{b}$  in  $\Phi$ . Since  $f_{\Pi}^q(\mathbf{b}|_{\mathbf{j}}) = 1$ , by definition of  $f_{\Pi}^q$  we have that  $\Pi(\text{ord}(\mathbf{b}|_{\mathbf{j}})) = 1$  and  $b_{j_0}, \dots, b_{j_{k-1}}$  are distinct. The latter implies, by construction of  $\sigma$ , that  $\text{ord}(\mathbf{b}|_{\mathbf{j}}) = \text{ord}(\sigma|_{\mathbf{j}})$ . Hence  $\Pi(\text{ord}(\sigma|_{\mathbf{j}})) = 1$ , so  $\sigma$  satisfies  $C$  in  $\Psi$ . Hence  $\text{val}_{\Psi}(\sigma) \geq \overline{\text{val}}_{\Phi}^q(\mathbf{b})$ . ◀

### 4.2 $\mathcal{G}^Y$ has high Max-OCSP( $\Pi$ ) values

In this section, we prove Lemma 5, which states that the Max-OCSP( $\Pi$ ) values of instances  $\Psi$  drawn from  $\mathcal{G}_{q,n,\alpha,T}^{Y,\pi}$  are large. Note that we prove a bound for *every* instance  $\Psi$  in the support of  $\mathcal{G}_{q,n,\alpha,T}^{Y,\pi}$ , although it would suffice for our application to prove that such a bound holds with high probability over the choice of  $\Psi$ .

To prove Lemma 5, if  $\Phi$  is the *q-coarsening* of  $\Psi$ , by Lemma 9, it suffices to show that  $\overline{\text{val}}_{\Phi}^q \geq 1 - \frac{k-1}{q}$ . One natural approach is to consider the *q-partition*  $\mathbf{b} = (b_0, \dots, b_{n-1}) \in [q]^n$  sampled when sampling  $\Psi$  and view  $\mathbf{b}$  as an assignment to  $\Phi$ . Consider any constraint

## 17:12 Streaming Approximation Resistance of Every Ordering CSP

$C = \mathbf{j} = (j_0, \dots, j_{k-1})$  in  $\Psi$ ; by the definition of  $\mathcal{G}^{Y, \pi}$  (Definition 4), we have  $\mathbf{b}|_{\mathbf{j}} = (\mathbf{v}_q^{(\ell)})_{\pi}$  for some (unique)  $\ell \in [q]$ , which we term the *identifier* of  $C$  (recall, we defined  $\mathbf{v}_q^{(\ell)}$  as the  $k$ -tuple  $(\ell, \dots, \ell + k - 1) \in [q]^k$ ). In other words,  $\mathbf{b}|_{\mathbf{j}} = (\mathbf{v}_q^{(\ell)})_{\pi}$ . Hence,  $C$  is satisfied by  $\mathbf{b}$  iff  $\Pi(\text{ord}((\mathbf{v}_q^{(\ell)})_{\pi})) = 1$ . By Proposition 3 above,  $\text{ord}((\mathbf{v}_q^{(\ell)})_{\pi}) = \text{ord}(\mathbf{v}_q^{(\ell)}) \circ \pi$ . Hence a sufficient condition for  $\mathbf{b}$  to satisfy  $C$  (which is in fact necessary in the case  $|\text{supp}(\Pi)| = 1$ ) is that  $\text{ord}(\mathbf{v}_q^{(\ell)}) = [0 \dots k - 1]$  (since then  $\text{ord}((\mathbf{v}_q^{(\ell)})_{\pi}) = \pi$ ); this happens iff  $C$ 's identifier  $\ell \in \{0, \dots, q - k\}$ . Unfortunately, when sampling the constraints  $C$ , we might get “unlucky” and get a sample which over-represents the constraints  $C$  with identifier  $\ell \in \{q - k + 1, \dots, q - 1\}$ . We can resolve this issue using “shifted” versions of  $\mathbf{b}$ ;<sup>4</sup> the proof is omitted here.

### 4.3 $\mathcal{G}^N$ has low Max-OCSP( $\Pi$ ) values

In this section, we prove Lemma 6, which states that the Max-OCSP( $\Pi$ ) value of an instance drawn from  $\mathcal{G}^N$  does not significantly exceed the random ordering threshold  $\rho(\Pi)$ , with high probability.

Using concentration bounds (i.e., Lemma 2), one could show that a fixed solution  $\sigma \in \mathcal{S}_n$  satisfies more than  $\rho(\Pi) + \frac{1}{q}$  constraints with probability which is exponentially small in  $n$ . However, taking a union bound over all  $n!$  permutations  $\sigma$  would cause an unacceptable blowup in the probability. Instead, to prove Lemma 6, we take an indirect approach, involving bounding the Max-CSP value of the  $q$ -coarsening of a random instance and bounding the gap between the Max-OCSP value and the  $q$ -coarsened Max-CSP value. To do this, we define the following notions of small set expansion for  $k$ -hypergraphs:

► **Definition 10** (Lying on a set). *Let  $G = (V, E)$  be a  $k$ -hypergraph. Given a set  $S \subseteq V$ , a  $k$ -hyperedge  $\mathbf{e} \in E$  lies on  $S$  if it is incident on two (distinct) vertices in  $S$  (i.e., if  $|\Gamma(\mathbf{e}) \cap S| \geq 2$ ).*

► **Definition 11** (Congregating on a partition). *Let  $G = (V, E)$  be a  $k$ -hypergraph. Given a  $q$ -partition  $\mathbf{b} \in [q]^n$ , a  $k$ -hyperedge  $\mathbf{e} \in E$  congregates on  $\mathbf{b}$  if it lies on one of the blocks  $\mathbf{b}^{-1}(i)$ .*

We denote by  $N(G, S)$  the number of  $k$ -hyperedges of  $G$  which lie on  $S$ .

► **Definition 12** (Small set hypergraph expansion (SSHE) property). *A  $k$ -hypergraph  $G = (V, E)$  is a  $(\gamma, \delta)$ -small set hypergraph expander (SSHE) if it has the following property: For every subset  $S \subseteq V$  of size at most  $\gamma|V|$ ,  $N(G, S) \leq \delta|E|$  (i.e., the number of  $k$ -hyperedges in  $E$  which lie on  $S$  is at most  $\delta|E|$ ).*

► **Definition 13** (Small partition hypergraph expansion (SPHE) property). *A  $k$ -hypergraph  $G = (V, E)$  is a  $(\gamma, \delta)$ -small partition hypergraph expander (SPHE) if it has the following property: For every partition  $\mathbf{b} \in [q]^n$  where each block  $\mathbf{b}^{-1}(i)$  has size at most  $\gamma|V|$ , the number of  $k$ -hyperedges in  $E$  which congregate on  $\mathbf{b}$  is at most  $\delta|E|$ .*

In the context of Figure 1 in Appendix A, the SPHE property says that for *any* partition with small blocks, there cannot be too many “orange” edges.

Having defined the SSHE and SPHE properties, we now sketch the proof of Lemma 6. The full proof is omitted.

<sup>4</sup> Alternatively, in expectation,  $\overline{\text{val}}_{\Phi}^q(\mathbf{b}) = 1 - \frac{k-1}{q}$ . Hence with probability at least  $\frac{99}{100}$ ,  $\overline{\text{val}}_{\Phi}^q(\mathbf{b}) \geq 1 - \frac{100(k-1)}{q}$  by Markov's inequality; this suffices for a “with-high-probability” statement.

**Proof sketch of Lemma 6.** For sufficiently large  $q$ , with high probability, the Max-CSP value of the  $q$ -coarsening of a random Max-OCSP( $\Pi$ ) instance drawn from  $\mathcal{G}_q^N$  is not much larger than  $\rho(\Pi)$  (Lemma 20 below). The constraint hypergraph for a random Max-OCSP( $\Pi$ ) instance drawn from  $\mathcal{G}_q^N$  is a good SSHE with high probability (Lemma 18 below). Hypergraphs which are good SSHEs are also (slightly worse) SPHEs (Lemma 14 below). Finally, if the constraint hypergraph of a Max-OCSP( $\Pi$ ) instance is a good SPHE, its Max-OCSP( $\Pi$ ) value cannot be much larger than its  $q$ -coarsened Max-CSP value (Lemma 15 below); intuitively, this is because if we “coarsen” an optimal ordering  $\sigma$  for the Max-OCSP by lumping vertices together in small groups to get an assignment  $\mathbf{b}$  for the coarsened Max-CSP, we can view this assignment  $\mathbf{b}$  as a partition on  $V$ , and for every  $k$ -hyperedge in  $G(\Psi)$  which does not congregate on this partition, the corresponding constraint in  $\Psi$  is satisfied. ◀

We remark that the bounds on Max-CSP values of coarsened random instances (Lemma 20 below) and on SSHE in random instances (Lemma 18 below) both use concentration inequalities (i.e., Lemma 2) and union bound over a space of size only  $(O_\epsilon(1))^n$  (the space of all solutions to the coarsened Max-CSP and the space of all small subsets of  $[n]$ , respectively); this lets us avoid the issue of union-bounding over the entire space  $S_n$  directly.

In the remainder of this section, we describe the necessary lemmas.

▶ **Lemma 14** (Good SSHEs are good SPHEs). *For every  $\gamma, \delta > 0$ , if a  $k$ -hypergraph  $G = (V, E)$  is a  $(\gamma, \delta)$ -SSHE, then it is a  $(\gamma, \delta(\frac{2}{\gamma} + 1))$ -SPHE.*

**Proof.** Omitted. ◀

▶ **Lemma 15** (Coarsening roughly preserves value in SPHEs). *Let  $\Psi$  be a Max-OCSP( $\Pi$ ) instance on  $n$  variables. Suppose that the constraint hypergraph of  $\Psi$  is a  $(\gamma, \delta)$ -SPHE. Let  $\Phi$  be the  $q$ -coarsening of  $\Psi$ . Then for sufficiently large  $n$ , if  $q \geq \frac{2}{\gamma}$ ,*

$$\text{val}_\Psi \leq \overline{\text{val}}_\Phi^q + \delta.$$

**Proof.** We will show that for every assignment  $\sigma \in S_n$  to  $\Psi$ , we can construct an assignment  $\mathbf{b} = (b_0, \dots, b_{n-1}) \in [q]^n$  to  $\Phi$  such that  $\text{val}_\Psi(\sigma) \leq \overline{\text{val}}_\Phi^q(\mathbf{b}) + \delta$ . Fix  $\sigma \in S_n$ . Define  $\mathbf{b} \in [q]^n$  by  $b_i = \lfloor \sigma(i) / \lfloor \gamma n \rfloor \rfloor$  for each  $i \in [n]$ . Observe that since  $\sigma(i) \leq n - 1$ , we have  $b_i \leq \lfloor (n - 1) / \lfloor \gamma n \rfloor \rfloor < q$ , hence  $\mathbf{b}$  is a valid assignment to  $\Phi$ . Also,  $\mathbf{b}$  has the property that for every  $i, j \in [n]$ , if  $\sigma(i) < \sigma(j)$  then  $b_i \leq b_j$ ; we call this *monotonicity* of  $\mathbf{b}$ .

View  $\mathbf{b}$  as a  $q$ -partition and consider the constraint hypergraph of  $\Psi$  (which is the same as the constraint hypergraph of  $\Phi$ ). Call a constraint  $C = (j_0, \dots, j_{k-1})$  *good* if it is both satisfied by  $\sigma$ , and the  $k$ -hyperedge corresponding to it does not congregate on  $\mathbf{b}$ . If  $C$  is good, then  $b_{j_0}, \dots, b_{j_{k-1}}$  are all distinct; together with monotonicity of  $\mathbf{b}$ , we conclude that if  $C$  is good, then  $\text{ord}(\mathbf{b}|_C) = \text{ord}(\sigma(j_0), \dots, \sigma(j_{k-1}))$ .

Finally, we note that each block in  $\mathbf{b}$  has size at most  $\gamma n$  by definition; hence by the SPHE property of the constraint hypergraph of  $\Psi$ , at most  $\delta$ -fraction of the constraints of  $\Psi$  correspond to  $k$ -hyperedges which congregate on  $\mathbf{b}$ . Since  $\text{val}_\Psi(\sigma)$  fraction of the constraints of  $\Psi$  are satisfied by  $\sigma$ , at least  $(\text{val}_\Psi(\sigma) - \delta)$ -fraction of the constraints of  $\Psi$  are good, and hence  $\mathbf{b}$  satisfies at least  $(\text{val}_\Psi(\sigma) - \delta)$ -fraction of the constraints of  $\Phi$ , as desired. ◀

The construction in this lemma was called *coarsening* the assignment  $\sigma$  by [10] (cf. [10, Definition 4.1]).

We also include the following helpful lemma, which lets us restrict to the case where our sampled Max-OCSP( $\Pi$ ) instance has many constraints.

## 17:14 Streaming Approximation Resistance of Every Ordering CSP

► **Lemma 16** (Most instances in  $\mathcal{G}^N$  have many constraints). *For every  $n, \alpha, \gamma > 0$ , and  $q \in \mathbb{N}$ ,*

$$\Pr_{\Psi \sim \mathcal{G}_{q,n,\alpha,T}^N} \left[ m(\Psi) \leq \frac{n\alpha T}{2q^k} \right] \leq \exp \left( -\frac{n\alpha T}{8q^k} \right).$$

**Proof.** The number of constraints in  $\Psi$  is distributed as the sum of  $n\alpha T$  independent Bernoulli( $1/q^k$ ) random variables. The desired bound follows by applying the Chernoff bound. ◀

### 4.3.1 $\mathcal{G}^N$ is a good SSHE with high probability

Recall that for a  $k$ -hypergraph  $G = (V, E)$  and  $S \subseteq V(G)$ , we define  $N(G, S)$  to be the number of  $k$ -hyperedges in  $G$  that lie on  $S$ , and for an  $k$ -hyperedge  $\mathbf{e} \in E$ , we define  $\Gamma(\mathbf{e}) \subseteq V$  as the set of vertices incident on  $\mathbf{e}$ .

► **Lemma 17** (Random hypermatchings barely lie on small sets). *For every  $n$  and  $\alpha, \gamma > 0$  with  $\alpha \leq \frac{1}{2k}$ , and every subset  $S \subseteq [n]$  of at most  $\gamma n$  vertices, we have*

$$\Pr_{G \sim \mathcal{H}_{k,n,\alpha}} [N(G, S) \geq 8k^2\gamma^2\alpha n] \leq \exp(-\gamma^2\alpha n).$$

**Proof.** Label the hyperedges of  $G$  as  $\mathbf{e}_0, \dots, \mathbf{e}_{\alpha n-1}$ . For  $i \in [\alpha n]$ , let  $X_i$  be the indicator for the event that  $\mathbf{e}_i$  lies on  $S$ . We have  $N(G, S) = X_0 + \dots + X_{\alpha n-1}$ .

We first bound  $\mathbb{E}[X_i \mid X_0, \dots, X_{i-1}]$  for each  $i$ . Conditioned on  $\mathbf{e}_0, \dots, \mathbf{e}_{i-1}$ , the  $k$ -hyperedge  $\mathbf{e}_i$  is uniformly distributed over the set of all  $k$ -hyperedges on  $[n] \setminus (\Gamma(\mathbf{e}_0) \cup \dots \cup \Gamma(\mathbf{e}_{i-1}))$ . It suffices to union-bound, over distinct pairs  $j_1 < j_2 \in \binom{[k]}{2}$ , the probability that the  $j_1$ -st and  $j_2$ -nd vertices of  $\mathbf{e}_i$  are in  $S$  (conditioned on  $X_0, \dots, X_{i-1}$ ). We can sample the  $j_1$ -st and  $j_2$ -nd vertices of  $\mathbf{e}_i$  first (uniformly over remaining vertices outside of  $S$ ) and then sample the remaining vertices (uniformly over remaining vertices). Hence we have the upper-bound

$$\mathbb{E}[X_i \mid X_0, \dots, X_{i-1}] \leq \binom{k}{2} \cdot \frac{|S|(|S|-1)}{(n-ki)(n-ki-1)} \leq \left( \frac{|S|}{n-k\alpha n} \right)^2 \leq 4k^2\gamma^2,$$

since  $\alpha \leq \frac{1}{2k}$ .

Now, we apply the concentration bound in Lemma 2 to conclude that:

$$\Pr_{G \sim \mathcal{H}_{k,n,\alpha}} [X_0 + \dots + X_{\alpha n-1} \geq 8k^2\gamma^2\alpha n] \leq \exp(-2k^2\gamma^2\alpha n) \leq \exp(-\gamma^2\alpha n). \quad \blacktriangleleft$$

► **Lemma 18.** *For every  $n, \alpha, \gamma > 0$ , and  $q \in \mathbb{N}$  with  $\alpha \leq \frac{1}{2k}$ ,*

$$\Pr_{\Psi \sim \mathcal{G}_{q,n,\alpha,T}^N} \left[ G(\Psi) \text{ is not a } (\gamma, 8k^2\gamma^2)\text{-SSHE} \mid m(\Psi) \geq \frac{n\alpha T}{2q^k} \right] \leq \exp \left( -\left( \frac{\gamma^2\alpha T}{2q^k} - \ln 2 \right) n \right).$$

**Proof.** Let  $\alpha_0, \dots, \alpha_{T-1} \geq 0$  be such that  $\frac{\alpha T}{2q^k} \leq \alpha_0 + \dots + \alpha_{T-1} \leq \alpha T$ . It suffices to prove the bound, for every such sequence  $\alpha_0, \dots, \alpha_{T-1}$ , conditioned on the event that for every  $i \in [T]$ ,  $m(G_i) = \alpha_i n$  (where  $G_i$  is defined as in Definition 4). This is equivalent to simply sampling each  $G_i \sim \mathcal{H}_{k,n,\alpha_i}$  independently.

Fix any set  $S \subseteq [n]$  of size at most  $\gamma n$ . Applying Lemma 17, and the fact that each hypermatching  $G_i$  in  $G$  is sampled independently, we conclude that

$$\begin{aligned} & \Pr_{\Psi \sim \mathcal{G}_{q,n,\alpha,T}^N} [\exists i \in [T] \text{ s.t. } N(G_i, S) \geq 8k^2\gamma^2\alpha_i n \mid \forall i \in [T], m(G_i) = \alpha_i n] \\ & \leq \exp(-\gamma^2(\alpha_0 + \dots + \alpha_{T-1})n) \\ & \leq \exp\left(-\frac{\gamma^2\alpha T n}{2q^k}\right). \end{aligned}$$

Hence by averaging, the total fraction of  $k$ -hyperedges in  $G$  which lie on  $S$  is at most  $8k^2\gamma^2$ . Taking the union-bound over the  $\leq 2^n$  possible subsets  $S \subseteq [n]$  gives the desired bound.  $\blacktriangleleft$

### 4.3.2 $\mathcal{G}^N$ has low coarsened Max-CSP( $f_{\Pi}^q$ ) values with high probability

For  $G \sim \mathcal{H}_{k,n,\alpha}$ , we define an instance  $\Phi(G)$  of Max-CSP( $f_{\Pi}^q$ ) on  $n$  variables  $x_0, \dots, x_{n-1}$  naturally as follows: for each  $k$ -hyperedge  $\mathbf{j} = (j_0, \dots, j_{k-1}) \in E(G) \subseteq [n]^k$ , we add the constraint  $\mathbf{j}$  to  $\Phi(G)$ .

► **Lemma 19** (Satisfiability of random instances of Max-CSP( $f_{\Pi}^q$ )). *For every  $n, \alpha, \eta > 0$ , and  $\mathbf{b} \in [q]^n$ ,*

$$\Pr_{G \sim \mathcal{H}_{k,n,\alpha}} [\text{val}_{\Phi(G)}^f(\mathbf{b}) \geq \rho(\Pi) + \eta] \leq \exp\left(-\left(\frac{\eta^2\alpha}{2(\rho(\Pi) + \eta)}\right)n\right).$$

**Proof.** Omitted.  $\blacktriangleleft$

► **Lemma 20.** *For every  $n$  and  $\alpha, \eta > 0$ ,*

$$\begin{aligned} & \Pr_{\Psi \sim \mathcal{G}_{q,n,\alpha,T}^N} \left[ \text{val}_{\Phi}^f \geq \rho(\Pi) + \eta, \text{ where } \Phi \text{ is the } q\text{-coarsening of } \Psi \mid m(\Psi) \geq \frac{n\alpha T}{2q^k} \right] \\ & \leq \exp\left(-\left(\frac{\eta^2\alpha T}{4(\rho(\Pi) + \eta)q^k} - \ln q\right)n\right). \end{aligned}$$

**Proof.** Identical to the proof of Lemma 18 (using Lemma 19 instead of Lemma 17), but now union-bounding over a set of size  $q^n$  (i.e., the set of possible assignments  $\mathbf{b} \in [q]^n$  for  $\Phi$ ).  $\blacktriangleleft$

## 5 Streaming indistinguishability of $\mathcal{G}^Y$ and $\mathcal{G}^N$

In this section we remark on the proof of Lemma 7 (although the full proof is omitted). This indistinguishability follows directly from the work of [5], who introduce a  $T$ -player communication problem called *implicit randomized mask detection (IRMD)*. Once we properly situate our instances  $\mathcal{G}^Y$  and  $\mathcal{G}^N$  within the framework of [5], Lemma 7 follows immediately.

We first recall their definition of the IRMD problem, and state their lower bound. The following definition is based on [5, Definition 3.1]. In [5] the IRMD game is parametrized by two distributions  $\mathcal{D}_Y$  and  $\mathcal{D}_N$ , but hardness is proved for a specific pair of distributions which suffices for our purpose; these distributions will thus be “hardcoded” into the definition we give.

► **Definition 21** (Implicit randomized mask detection (IRMD) problem). *Let  $q, k, n, T \in \mathbb{N}, \alpha \in (0, 1/k)$  be parameters. In the  $\text{IRMD}_{\alpha,T}$  game, there are  $T$  players, indexed from 0 to  $T - 1$ , and a hidden partition encoded by a random  $\mathbf{b} \in [q]^n$ . The  $t$ -th player has two inputs:*

## 17:16 Streaming Approximation Resistance of Every Ordering CSP

(a.)  $M_t \in \{0,1\}^{\alpha kn \times n}$ , the hypermatching matrix corresponding to a uniform  $\alpha$ -partial  $k$ -hypermatching on  $n$  vertices (i.e., drawn from  $\mathcal{H}_{n,\alpha}$ ), and (b.) a vector  $\mathbf{z}_t \in [q]^{\alpha kn}$  that can be generated from one of two different distributions:

- (**YES**)  $\mathbf{z}_t = M_t \mathbf{b} + \mathbf{y}_t \pmod{q}$  where  $\mathbf{y}_t \in [q]^{\alpha kn}$  is of the form  $\mathbf{y}_t = (\mathbf{y}_{t,0}, \dots, \mathbf{y}_{t,\alpha n-1})$  and each  $\mathbf{y}_{t,i} \in [q]^k$  is sampled as  $(a, \dots, a)$  where  $a$  is sampled uniformly from  $[q]$ .
- (**NO**)  $\mathbf{z}_t = M_t \mathbf{b} + \mathbf{y}_t \pmod{q}$  where  $\mathbf{y}_t \in [q]^{\alpha kn}$  is of the form  $\mathbf{y}_t = (\mathbf{y}_{t,0}, \dots, \mathbf{y}_{t,\alpha n-1})$  and each  $\mathbf{y}_{t,i} \in [q]^k$  is sampled as  $(a_0, \dots, a_{k-1})$  where each  $a_j$  is sampled uniformly and independently from  $[q]$ .

This is a one-way game where the  $t$ -th player can send a private message to the  $(t+1)$ -st player after receiving a message from the previous player. The goal is for the  $(T-1)$ -st player to decide whether the  $\{\mathbf{z}_t\}$  have been chosen from the **YES** or **NO** distribution, and the advantage of a protocol is defined as

$$\left| \Pr_{\text{YES case}} [\text{the } (T-1)\text{-st player outputs } 1] - \Pr_{\text{NO case}} [\text{the } (T-1)\text{-st player outputs } 1] \right|.$$

Note that the definition of the IRMD problem does not depend on an underlying family of constraints. Nevertheless, we are able to leverage its hardness to prove Lemma 7 (and indeed, all hardness results in [5] itself stem from hardness for the IRMD problem). The following theorem from [5] gives a lower bound on the communication complexity of the IRMD problem:

► **Theorem 22** ([5, Theorem 3.2]). *For every  $q, k \in \mathbb{N}$  and  $\delta \in (0, 1/2)$ ,  $\alpha \in (0, 1/k)$ ,  $T \in \mathbb{N}$  there exists  $n_0 \in \mathbb{N}$  and  $\tau \in (0, 1)$  such that the following holds. For all  $n \geq n_0$ , every protocol for  $\text{IRMD}_{\alpha, T}$  on  $n$  vertices with advantage  $\delta$  requires  $\tau n$  bits of communication.*

We use this hardness result to prove Lemma 7, via a standard communication-to-streaming reduction from IRMD. Our proof is based on the reduction given by [5, Theorem 4.3], which introduces a notion called the *width* of a constraint family, which we briefly discuss. For our purposes, it suffices to define the width  $\omega(f) \in [0, 1]$  of a single constraint  $f : [q]^k \rightarrow \{0, 1\}$  as

$$\omega(f) = \max_{\mathbf{b} \in [q]^k} \left\{ \Pr_{\ell \in [q]} [f(\mathbf{b} + \ell) = 1] \right\},$$

where  $\mathbf{b} + \ell$  denotes adding  $\ell$  to each component of  $\mathbf{b}$ . [5, Theorem 4.3] states that for every  $f$  and  $\epsilon > 0$ ,  $\text{Max-CSP}(f)$  cannot be  $(\rho(f)/\omega(f) + \epsilon)$ -approximated by a sublinear-space single-pass streaming algorithm, where  $\rho(f) = \Pr_{\mathbf{b} \in [q]^k} [f(\mathbf{b}) = 1]$  is the random assignment value for  $f$ . (The approximation ratio  $\rho(f)/\omega(f)$  is derived from the fact that the **NO** instances in the reduction have values close to  $\rho(f)$ , while the **YES** instances have values close to  $\omega(f)$ .) Hence whenever  $\omega(f)$  is close to 1,  $\text{Max-CSP}(f)$  is difficult to approximate. In our setting, we have  $\omega(f_{\Pi}^q) \geq 1 - \frac{k-1}{q}$ ; indeed, simply take  $\mathbf{b} = (\pi^{-1}(0), \dots, \pi^{-1}(k-1))$ , and then for any  $\ell \in \{0, \dots, q-k\}$ , we have  $f_{\Pi}^q(\mathbf{b} + \ell) = 1$  (by the same reasoning as in Subsection 4.2). The fact that  $\omega(f_{\Pi}^q) \approx 1$  for large  $q$  is precisely what enables us to apply [5]'s lower bounds to get optimal lower bounds in our setting.

---

### References

- 1 Per Austrin, Rajsekar Manokaran, and Cenny Wenner. On the NP-hardness of approximating ordering-constraint satisfaction problems. *Theory of Computing*, 11:257–283, 2015. Conference version in APPROX 2013. doi:10.4086/toc.2015.v011a010.
- 2 Amey Bhangale and Subhash Khot. UG-Hardness to NP-Hardness by Losing Half. In *34th Computational Complexity Conference (CCC 2019, New Brunswick, New Jersey, USA, August*

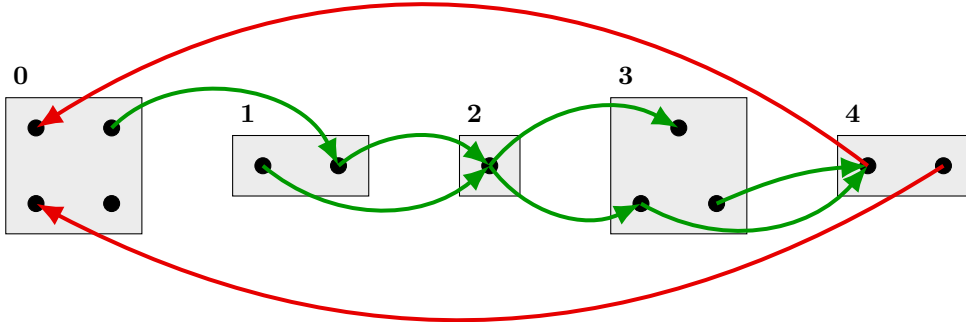


- 18-20, 2019), volume 137 of *LIPICs*. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.CCC.2019.3.
- 3 Amit Chakrabarti, Prantar Ghosh, Andrew McGregor, and Sofya Vorotnikova. Vertex ordering problems in directed graph streams. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2020, Salt Lake City, Utah, USA, January 5-9, 2020)*, pages 1786–1802. Society for Industrial and Applied Mathematics, 2020. doi:10.5555/3381089.3381198.
  - 4 Benny Chor and Madhu Sudan. A geometric approach to betweenness. *SIAM Journal on Discrete Mathematics*, 11(4):511–523, 1998. Conference version in *Algorithms, ESA 1995*. doi:10.1137/S0895480195296221.
  - 5 Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, Ameya Velingker, and Santhoshini Velusamy. Linear Space Streaming Lower Bounds for Approximating CSPs, June 2021. arXiv:2106.13078.
  - 6 Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, and Santhoshini Velusamy. Approximability of all Boolean CSPs in the dynamic streaming setting, 2021. arXiv:2102.12351.
  - 7 Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, and Santhoshini Velusamy. Approximability of all finite CSPs in the dynamic streaming setting, June 2021. arXiv:2105.01161.
  - 8 Chi-Ning Chou, Alexander Golovnev, and Santhoshini Velusamy. Optimal Streaming Approximations for all Boolean Max-2CSPs and Max- $k$ SAT. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS 2020, November 16-19, 2020)*, pages 330–341. IEEE Computer Society, 2020. doi:10.1109/FOCS46700.2020.00039.
  - 9 Dmitry Gavinsky, Julia Kempe, Iordanis Kerenidis, Ran Raz, and Ronald de Wolf. Exponential separation for one-way quantum communication complexity, with applications to cryptography. *SIAM Journal on Computing*, 38(5):1695–1708, 2008. Conference version in *STOC 2007*. doi:10.1137/070706550.
  - 10 Venkatesan Guruswami, Johan Håstad, Rajsekar Manokaran, Prasad Raghavendra, and Moses Charikar. Beating the Random Ordering is Hard: Every ordering CSP is approximation resistant. *SIAM Journal on Computing*, 40(3):878–914, 2011. Conference version in *FOCS 2008*. doi:10.1137/090756144.
  - 11 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/RANDOM 2019, Cambridge, MA, USA, September 20-22, 2019)*, volume 145 of *LIPICs*, pages 5:1–5:12. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.APPROX-RANDOM.2019.5.
  - 12 Venkatesan Guruswami, Ameya Velingker, and Santhoshini Velusamy. Streaming Complexity of Approximating Max 2CSP and Max Acyclic Subgraph. In Klaus Jansen, José D. P. Rolim, David Williamson, and Santosh S. Vempala, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017, Berkeley, CA, USA, August 16-18, 2017)*, volume 81 of *LIPICs*, pages 8:1–8:19. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICs.APPROX-RANDOM.2017.8.
  - 13 Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001. doi:10.1145/502090.502098.
  - 14 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Streaming lower bounds for approximating MAX-CUT. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015, San Diego, California, USA, January 4-6, 2015)*, pages 1263–1282. Society for Industrial and Applied Mathematics, January 2015. doi:10.1137/1.9781611973730.84.
  - 15 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. Society for Industrial and Applied Mathematics, January 2017. doi:10.5555/3039686.3039798.

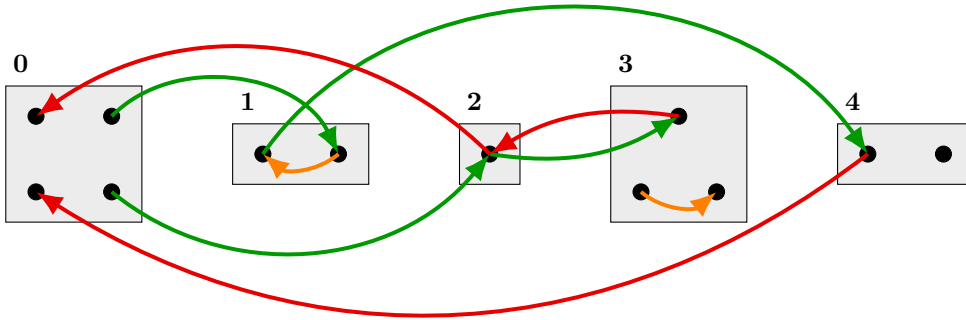
## 17:18 Streaming Approximation Resistance of Every Ordering CSP

- 16 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. Association for Computing Machinery, June 2019. doi:10.1145/3313276.3316364.
- 17 Richard M. Karp. Reducibility among Combinatorial Problems. In R.E. Miller, J.W. Thatcher, and J.D. Bohlinger, editors, *The IBM Research Symposia Series*, pages 85–103. Springer, 1972. doi:10.1007/978-1-4684-2001-2\_9.
- 18 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC 2002, Québec, Canada, May 19-21, 2002)*, pages 767–775. Association for Computing Machinery, 2002. doi:10.1145/509907.510017.
- 19 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. Association for Computing Machinery, 2015. doi:10.1145/2688073.2688093.
- 20 Alantha Newman. *Approximating the Maximum Acyclic Subgraph*. Masters Thesis, Massachusetts Institute of Technology, 2000.
- 21 Jaroslav Opatrny. Total Ordering Problem. *SIAM Journal on Computing*, 8(1):111–114, 1979. doi:10.1137/0208008.
- 22 Noah Singer, Madhu Sudan, and Santhoshini Velusamy. Streaming approximation resistance of every ordering CSP, 2021. Full version of this paper. arXiv:2105.01782.
- 23 Noah Singer, Madhu Sudan, and Santhoshini Velusamy. Streaming approximation resistance of every ordering CSP, May 2021. Original version of this paper; proved only  $o(\sqrt{n})$  space lower bounds. arXiv:2105.01782v1.
- 24 Elad Verbin and Wei Yu. The streaming complexity of cycle counting, sorting by reversals, and other problems. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2011, San Francisco, California, USA, January 23-25, 2011)*, pages 11–25. Society for Industrial and Applied Mathematics, 2011. doi:10.5555/2133036.2133038.

**A** Example hard instances for MAS



(a) Constraint graph of a sample MAS instance drawn from  $\mathcal{G}^Y$ .



(b) Constraint graph of a sample MAS instance drawn from  $\mathcal{G}^N$ .

**Figure 1** The constraint graphs of MAS instances which could plausibly be drawn from  $\mathcal{G}^Y$  and  $\mathcal{G}^N$ , respectively, for  $q = 5$  and  $n = 12$ . Recall that MAS is a binary Max-OCSP with ordering constraint function  $\Pi$  supported only on  $[0\ 1]$ . According to the definition of  $\mathcal{G}^Y$  (see Definition 4, with  $\pi = [0\ 1]$ ), instances are sampled by first sampling a  $q$ -partition  $\mathbf{b} = (b_0, \dots, b_{n-1}) \in [q]^n$ , and then sampling some edges; every sampled edge  $(u, v)$  must satisfy  $b_v = b_u + 1 \pmod{q}$ . On the other hand, there are no requirements on  $(b_u, b_v)$  for instances sampled from  $\mathcal{G}^N$ . Above, the blocks of the partition  $\mathbf{b}$  are labelled  $0, \dots, 4$ , and the reader can verify that the edges satisfy the appropriate requirements. We also color the edges in a specific way: We color an edge  $(u, v)$  green, orange, or red if  $b_v > b_u$ ,  $b_v = b_u$ , or  $b_v < b_u$ , respectively. This visually suggests important elements of our proofs that  $\mathcal{G}^Y$  has MAS values close to 1 and  $\mathcal{G}^N$  has MAS values close to  $\frac{1}{2}$  (for formal statements, see Lemma 5 and Lemma 6, respectively). Specifically, in the case of  $\mathcal{G}^Y$ , if we arbitrarily arrange the vertices in each block, we will get an ordering in which every green edge is satisfied, and we expect all but  $\frac{1}{q}$  fraction of the edges to be satisfied (i.e., all but those which go from block  $q - 1$  to block 0). On the other hand, if we executed a similar process in  $\mathcal{G}^N$ , the resulting ordering would satisfy all green edges and some subset of the orange edges; together, in expectation, these account only for  $\frac{q(q+1)}{2q^2} = \frac{q+1}{2q} \approx \frac{1}{2}$  fraction of the edges.



# Upper and Lower Bounds for Complete Linkage in General Metric Spaces

**Anna Arutyunova** ✉

Universität Bonn, Germany

**Anna Großwendt** ✉

Universität Bonn, Germany

**Heiko Röglin** ✉

Universität Bonn, Germany

**Melanie Schmidt** ✉

Universität Köln, Germany

**Julian Wargalla** ✉

Universität Köln, Germany

---

## Abstract

In a hierarchical clustering problem the task is to compute a series of mutually compatible clusterings of a finite metric space  $(P, \text{dist})$ . Starting with the clustering where every point forms its own cluster, one iteratively merges two clusters until only one cluster remains. Complete linkage is a well-known and popular algorithm to compute such clusterings: in every step it merges the two clusters whose union has the smallest radius (or diameter) among all currently possible merges. We prove that the radius (or diameter) of every  $k$ -clustering computed by complete linkage is at most by factor  $O(k)$  (or  $O(k^2)$ ) worse than an optimal  $k$ -clustering minimizing the radius (or diameter). Furthermore we give a negative answer to the question proposed by Dasgupta and Long [6], who show a lower bound of  $\Omega(\log(k))$  and ask if the approximation guarantee is in fact  $\Theta(\log(k))$ . We present instances where complete linkage performs poorly in the sense that the  $k$ -clustering computed by complete linkage is off by a factor of  $\Omega(k)$  from an optimal solution for radius and diameter. We conclude that in general metric spaces complete linkage does not perform asymptotically better than single linkage, merging the two clusters with smallest inter-cluster distance, for which we prove an approximation guarantee of  $O(k)$ .

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Facility location and clustering

**Keywords and phrases** Hierarchical Clustering, Complete Linkage, agglomerative Clustering,  $k$ -Center

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.18

**Category** APPROX

**Funding** This work has been supported by DFG grants RO 5439/1-1 and SCHM 2765/1-1.

## 1 Introduction

The  $k$ -clustering problem asks for a partition of a point set in a metric space into  $k$  subsets (or clusters). To measure whether the data is clustered well, one option is to pick a center for every cluster and compute the maximum distance between a point and the center of its cluster. This objective is to be minimized and is known as  $k$ -center. A problem which is independent of the choice of centers is the  $k$ -diameter problem, where we want to minimize the maximum distance between two points lying in the same cluster. Observe that  $k$ -center and  $k$ -diameter are related to each other in the sense that for a fixed set  $P$  the cost of an optimal  $k$ -diameter clustering on  $P$  is at most twice the cost of an optimal  $k$ -center clustering, which again costs at most as much as an optimal  $k$ -diameter clustering. There are other



© Anna Arutyunova, Anna Großwendt, Heiko Röglin, Melanie Schmidt, and Julian Wargalla; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 18; pp. 18:1–18:22



Leibniz International Proceedings in Informatics

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

objectives to measure the quality of a clustering where every point contributes to the cost of the clustering, for example *k-median* and *k-means*. Here we want to minimize the cost which equals the sum over all (squared) distances between a point and the center of its cluster.

The *k*-center problem is NP-hard to approximate with factor  $\alpha < 2$  [12, 14]. This bound is tight, as both Gonzalez [7] and Hochbaum and Shmoys [13] show. Gonzalez's 2-approximation algorithm is a simple, but elegant greedy approach. Starting with an arbitrary point  $p_1 \in P$ , one constructs an enumeration  $P = \{p_1, \dots, p_{|P|}\}$  by successively choosing as  $p_{i+1}$  a point from  $P$  whose minimum distance to any point from  $\{p_1, \dots, p_i\}$  is maximal. Assigning every point from  $P$  to its closest neighbor among  $p_1, \dots, p_k$  (the centers) yields a 2-approximation for the *k*-center problem for all  $k = 1, \dots, |P|$ . One can prove that the resulting clustering is also a 2-approximation to *k*-diameter. Another greedy approach is the reverse greedy algorithm, which starts with all data points as centers and iteratively removes a center such that the objective stays as small as possible. Hershkowitz and Kehne [11] show that this algorithm computes an  $\Theta(k)$ -approximation. Observe that both greedy algorithms compute an *incremental clustering* where the centers of a *k*-clustering are also centers of an *l*-clustering if  $l \geq k$ .

Gonzalez's algorithm [7] allows to compute good clusterings, even if one does not previously know an appropriate value for *k*. However, even successive clusterings computed by Gonzalez's algorithm and reverse greedy can be radically different and so it can be difficult to compare them and select one that seems appropriate for the task.

Another greedy approach known as *complete linkage* starts with every point in its own cluster and consecutively merges two clusters whose union has the smallest radius (or diameter when considering the *k*-diameter objective) among all possible cluster pairs. If we proceed like this until only one cluster remains, we also obtain a *k*-clustering for any possible  $1 \leq k \leq |P|$ . However, this time, the resulting clusterings are also *hierarchically compatible*: for all  $l \geq k$  the *l*-clustering is a refinement of the *k*-clustering. This makes it easier to compare such clusterings with each other and to choose an appropriate *k*-clustering. Also, this additional hierarchical structure is interesting in and of itself. Famous examples include phylogenetic trees that represent the relationship between animal species in biology.

A series of such hierarchically compatible clusterings  $\mathcal{C}_1, \dots, \mathcal{C}_{|P|}$  (with  $\mathcal{C}_k$  being a *k*-clustering for all *k*) forms a *hierarchical clustering*. Complete linkage is a common and popular bottom-up approach to compute these and can be generalized to fit any *k*-clustering objective, resulting in so called *agglomerative clustering* methods. For hierarchical *k*-means this is Wards method [16].

To evaluate a hierarchical clustering  $\mathcal{C}_1, \dots, \mathcal{C}_{|P|}$  we refer to the underlying *k*-clusterings: it is an  $\alpha$ -*approximation* if the cost of  $\mathcal{C}_k$  is at most  $\alpha$  times that of an optimal *k*-clustering for all  $1 \leq k \leq |P|$ .

**Related work.** For hierarchical *k*-center and *k*-diameter, constant factor approximations are known. For both problems Dasgupta and Long [6] and Charikar et al. [4] give a polynomial-time 8-approximation. In [15] Lin et al. introduce the concept of nesting. Using this technique, every approximation algorithm to a *k*-clustering objective that satisfies their nesting property can be converted into an algorithm for its hierarchical version. Especially *k*-median and *k*-means satisfy this property and thus (in combination with the currently best constant factor approximations for *k*-median [3] and *k*-means [2]), polynomial time constant factor approximations do indeed exist for the hierarchical *k*-median/*k*-means problem. Yet the resulting guarantees are relatively high ( $\approx 56$  for *k*-median and  $\approx 3662$  for *k*-means). Nesting can also be applied to *k*-center/*k*-diameter but does not improve upon the 8-approximation.

As optimal  $k$ -clusterings are not necessarily hierarchically compatible, even assuming unlimited computation power 1-approximations do not exist in general. Das and Kenyon-Mathieu [5] give an instance for the diameter and Großwendt [10] for the radius where the best hierarchical clustering is a 2-approximation. Using the concept of nesting by Lin et al. [15], Großwendt [10] proves an existential upper bound of 4 for hierarchical  $k$ -center.

However, greedy algorithms are more common in practical applications. There exist several theoretical results on upper and lower bounds on the approximation factor for complete linkage. For metrics induced by norms in  $\mathbb{R}^d$ , especially the Euclidean metric, Ackermann et al. [1] prove that, assuming the dimension  $d$  to be constant, complete linkage computes for both the  $k$ -center and  $k$ -diameter objective an  $O(\log(k))$  approximation. This was later improved by Großwendt and Röglin [8] to  $O(1)$ . Both works distinguish between two variants of  $k$ -center: one where centers must be from the set  $P$  and the second where they can be arbitrary points chosen from the whole space  $\mathbb{R}^d$ . In the first case the approximation factor shown in [1, 8] depends linearly on  $d$  and in the second case exponentially on  $d$ . For the  $k$ -diameter problem it even depends doubly exponentially on the dimension. Furthermore Ackermann et al. prove for the  $l_p$ -metric with  $1 \leq p < \infty$  a lower bound of  $\Omega(\sqrt[p]{\log(d)})$  for complete linkage for  $k$ -diameter and  $k$ -center with centers drawn from  $P$  [1].

Little is known about complete linkage in general metric spaces. Dasgupta and Long show in [6] that the lower bound is in  $\Omega(\log(k))$ . With an approach to upper bound the increase in cost by a complete linkage merge, which we borrow from [1], we obtain in a relatively straightforward manner an upper bound of  $O(\log(|P| - k))$  for complete linkage for  $k$ -center.

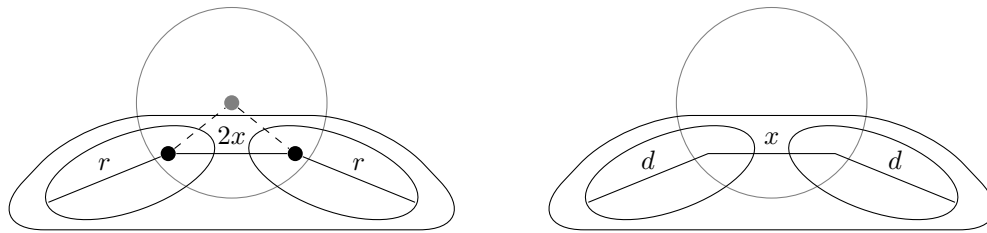
There exist few results for agglomerative clustering regarding other objectives. Großwendt et al. [9] analyze Ward's method for  $k$ -means, and show that if the clusters of an optimal  $k$ -means clustering are sufficiently far apart, Ward's method computes a 2-approximation and under some additional assumptions in fact reconstructs the optimal clustering.

**Our Results.** We study upper and lower bounds for the complete linkage algorithm in general metric spaces for the  $k$ -center and  $k$ -diameter objective. For  $k$ -center in general metric spaces it is reasonable to assume that centers can be only drawn from  $P$  and thus we only consider this variant. Our main results are:

- A lower bound of  $\Omega(k)$  for complete linkage for  $k$ -center and  $k$ -diameter, which improves the currently highest lower bound of  $\Omega(\log(k))$  by Dasgupta and Long [6] significantly.
- An upper bound of  $O(k)$  for  $k$ -center and an upper bound of  $O(k^2)$  for  $k$ -diameter, which are to the best of the authors' knowledge the first non-trivial upper bounds for complete linkage in general metric spaces.

The lower bound  $\Omega(k)$  is surprising as it shows that complete linkage does not perform asymptotically better than single linkage, which merges the two clusters with smallest distance to each other (the distance of two clusters is the smallest distance between two of their points). Dasgupta and Long [6] prove a lower bound of  $\Omega(k)$  for single linkage and we show that the approximation factor is in fact  $\Theta(k)$ . As single linkage is not designed to minimize the radius or diameter of emerging clusters, it is a natural assumption that it performs worse than complete linkage. However our results show that this assumption is generally not true. It is even still open if complete linkage for  $k$ -diameter performs as good as single linkage, as we are only able to prove an upper bound of  $O(k^2)$ .

**Techniques.** One of the biggest and most well-known issues concerning single linkage is that of chaining. If there is a sequence of points  $x_1, \dots, x_k \in P$  with  $\text{dist}(x_i, x_{i+1})$  relatively small for all  $i$ , then single linkage might merge all of them together, despite the resulting cluster



■ **Figure 1** On the left we see two  $k$ -center clusters with radius  $r$  whose centers lie in the same optimal cluster. The radius of the merged cluster is at most  $r + 2x$ . On the right we have a similar situation for  $k$ -diameter but the merged cluster has diameter at most  $2d + x$ .

being quite large. Dasgupta and Long show with their lower bound of  $\Omega(\log k)$  that a similar process of chaining can also occur when executing complete linkage. They give the example of points placed on a regular  $(k \times k)$ -grid with a spacing of 1. The distance is given by the sum of the discrete metric on the horizontal axis and the logarithm of the absolute value of the vertical axis. That is,  $\text{dist}((x, y), (x', y')) = \mathbf{1}_{x \neq x'} + \log_2(1 + |y - y'|)$ . Now, although an optimal clustering just consists of the individual rows of the grid, complete linkage might reproduce the columns instead (assuming that  $k$  is a power of 2): iteratively go from top to bottom and merge vertically neighboring clusters. Every such iteration halves the number of clusters and, due to the logarithm, only increases the cost by 1, just as when merging along the rows. Of course, we would have to pay only once to merge horizontally, whereas we have to pay  $\log_2 k$  times to merge vertically, but complete linkage cannot distinguish between these two cases. In fact, one can shift the vertical placement by arbitrarily small values to ensure that complete linkage always chooses the bad case.

We have to heavily modify the example to improve upon this  $\log_2 k$  factor. The fundamental problem is this: a vertical merge is only allowed to increase the cost by 1 to tie it with any horizontal merge, whereas the number of rows occupied by a cluster (and thus its diameter) doubles. We raise the lower bound by constructing an instance on which complete linkage iteratively merges diagonally shifted clusters. This process of merging clusters is much slower and does not require us to introduce a logarithmic scaling: merging one such cluster into the other incurs a cost of 1, while at the same time increasing the number of occupied rows only by one. The instance that we describe later is successively built from smaller components that exhibit exactly this behaviour, while ensuring that any such merge does not pay for the whole row.

Following the work of Ackermann et al. [1] one can show for complete linkage an upper bound of  $\log(|P| - k)$  for  $k$ -center. This comes from the following easy property, which is true for the radius but cannot be transferred to diameter: Suppose the optimal  $k$ -center solution  $\mathcal{O}$  has cost  $x$ . In a complete linkage clustering consisting of more than  $k$  clusters two of its centers must lie in the same optimal cluster and therefore are at distance  $\leq 2x$  to each other. Thus the merge that is performed by complete linkage increases the cost by at most  $2x$ . However if we replace  $k$ -center by  $k$ -diameter we see that the cost is more than doubled in the worst case (see Figure 1), which is not enough to obtain an upper bound polynomial in  $k$ . Thus we introduce another perspective on the cost of a cluster. A cluster is good if its cost is small enough in comparison to the number of optimal clusters from  $\mathcal{O}$  which it intersects. As  $\mathcal{O}$  consists of  $k$  clusters this already implies a sufficiently small upper bound for good clusters. For all remaining clusters we show that their number is small enough. This approach leads to an upper bound of  $O(k^2)$  for  $k$ -diameter and, in combination with the  $\log(|P| - k)$  upper bound, an upper bound of  $O(k)$  for  $k$ -center.



## 2 Preliminaries

Let  $(P, \text{dist})$  be a metric space with  $n$  points and  $1 \leq k \leq n$ . The  $k$ -center problem asks for a partition of  $P$  into  $k$  clusters  $\mathcal{C} = \{C_1, \dots, C_k\}$ . The cost of cluster  $C_i$  is given by  $\text{cost}(C_i) = \min_{c \in C_i} \max_{x \in C_i} \text{dist}(x, c)$  while the cost of the clustering  $\mathcal{C}$  is  $\text{cost}(\mathcal{C}) = \max_{i=1, \dots, k} \text{cost}(C_i)$  and is to be minimized.

In the  $k$ -diameter problem we also have to find a partition of  $P$  into  $k$  clusters  $\mathcal{C} = \{C_1, \dots, C_k\}$  and minimize the overall cost. However, we replace the cost of a cluster  $C_i$  by  $\text{cost}(C_i) = \max_{x, y \in C_i} \text{dist}(x, y)$ . For both the  $k$ -center and the  $k$ -diameter problem we denote by  $\mathcal{O}_k$  an arbitrary but fixed *optimal clustering*.

We study the hierarchical version of the above problems, where we ask for a  $k$ -clustering  $\mathcal{C}_k$  of  $P$  for every  $1 \leq k \leq n$ . The clusterings must be *hierarchically compatible*, which means that  $\mathcal{C}_{k-1}$  is obtained from  $\mathcal{C}_k$  by merging two of its clusters, i.e., for all  $2 \leq k \leq n$  there are  $A, B \in \mathcal{C}_k$  such that  $\mathcal{C}_{k-1} = \mathcal{C}_k \setminus \{A, B\} \cup \{A \cup B\}$ . A sequence of such  $k$ -clusterings  $(\mathcal{C}_k)_{k=1}^n$  is called a *hierarchical clustering*. We say that it is an  $\alpha$ -approximation if  $\text{cost}(\mathcal{C}_k) \leq \alpha \text{cost}(\mathcal{O}_k)$  for all  $1 \leq k \leq n$ . Thus the task is to find a hierarchical clustering which is a good approximation to the optimal solution on every level  $k$ .

A common class of approaches for computing such hierarchical clusterings are *agglomerative linkage algorithms*. As outlined above, a hierarchical clustering can be computed in a bottom-up fashion, where pairs of clusters are merged successively. Agglomerative linkage procedures do exactly that, with the choice of clusters to be merged at every step given by a linkage function. Such a linkage function maps all possible pairs of disjoint clusters onto  $\mathbb{R}_+$  and the algorithm chooses one pair that minimizes this value: Suppose that we have already constructed  $\mathcal{C}_k$  and are using the linkage function  $f$ . Then  $\mathcal{C}_{k-1}$  is given by merging a pair  $A \neq B \in \mathcal{C}_k$  with  $f(A, B) = \min_{A' \neq B' \in \mathcal{C}_k} f(A', B')$ . As already stated, the two linkage functions we are interested in are:

- Single linkage:  $(A, B) \mapsto \text{dist}(A, B) = \min_{(a, b) \in A \times B} \text{dist}(a, b)$ .
- Complete linkage:  $(A, B) \mapsto \text{cost}(A \cup B)$ .

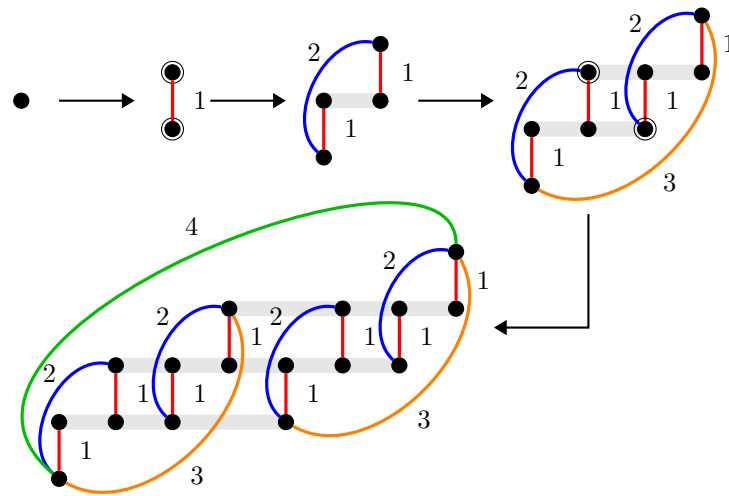
To analyze the performance of the respective agglomerative algorithms we often consider the smallest clustering from  $(\mathcal{C}_k)_{k=1}^n$  (in terms of the number of clusters) whose cost does not exceed a given bound. This perspective is already used by Großwendt and Röglin [8] and allows a better handling of the cost. For any  $x \geq 0$  let  $t_{\leq x} = \min\{k \mid \text{cost}(\mathcal{C}_k) \leq x\}$  and set  $\mathcal{H}_x = \mathcal{C}_{t_{\leq x}}$ . Observe that  $\mathcal{H}_x$  is the smallest clustering from  $(\mathcal{C}_k)_{k=1}^n$  with cost at most  $x$ . Thus it has the useful property that every merge of two clusters in  $\mathcal{H}_x$  results in a clustering of cost more than  $x$ . Furthermore, for a cluster  $C \subseteq P$  and an optimal  $k$ -clustering  $\mathcal{O} = \mathcal{O}_k$  we denote by  $\mathcal{O}_C = \{O \in \mathcal{O} \mid O \cap C \neq \emptyset\}$  the set of all optimal  $k$ -clusters hit by  $C$ .

## 3 Approximation Guarantee of Single Linkage

As outlined in [6] there are clustering instances where single linkage builds chains yielding the lower bound  $\Omega(k)$  on the approximation factor. We show in Appendix A that this is the worst case scenario, as in fact single linkage computes an  $O(k)$ -approximation for hierarchical  $k$ -center/ $k$ -diameter.

► **Theorem 1.** *Let  $(\mathcal{C}_k)_{k=1}^n$  be the hierarchical clustering computed by single linkage on  $(P, \text{dist})$  and let  $\mathcal{O}_k$  be an optimal clustering for  $k$ -center or  $k$ -diameter, respectively. We have for all  $1 \leq k \leq n$*

1.  $\text{cost}(\mathcal{C}_k) \leq (2k + 2) \cdot \text{cost}(\mathcal{O}_k)$  for the  $k$ -center cost
2.  $\text{cost}(\mathcal{C}_k) \leq 2k \cdot \text{cost}(\mathcal{O}_k)$  for the  $k$ -diameter cost.



■ **Figure 2** The progression of the first 5 components  $K_1, \dots, K_5$ . The gray sets indicate points on the same level and form the optimal clusters. When analyzing the instance for the radius, the encircled points in the  $K_2$  and  $K_4$  component indicate their optimal centers.

#### 4 Lower Bounds for Complete Linkage

In the following we show that complete linkage performs asymptotically as bad as single linkage in the worst case. That is, for every  $k \in \mathbb{N}$  we provide an instance  $P_k$  on which the diameter and radius of a  $k$ -clustering computed by complete linkage is off by a factor of  $\Omega(k)$  from the cost of an optimal solution. This improves upon the previously known lower bound of  $\Omega(\log_2(k))$  established by Dasgupta and Long. Recall from the introduction that one of the big problems preventing an improved lower bound was that any horizontal merge already paid for all the involved rows. As such, for the worst case, one was only allowed to merge vertically, but this can be done at most  $\log_2(k)$  times. We improve upon this by inductively constructing an instance from smaller components that are diagonally shifted to produce bigger ones. Merging two such diagonally shifted components incurs an additional cost of 1, while ensuring at the same time that this does not pay for any future merges of parallel components.

A  $k$ -component  $K_k = (G_k, \phi_k)$  is a combination of a graph  $G_k = (V_k, E_k)$  and a mapping  $\phi_k : V_k \rightarrow \{1, \dots, k\}$ . The mapping is necessary for the construction of the component and later on determines an optimal  $k$ -clustering on  $P_k$ . We refer to  $\phi_k(x)$  as the *level* of  $x$ . The other part of the component is an undirected graph  $G_k$ , referred to as a  $k$ -graph, on  $2^{k-1}$  points with edge weights in  $\mathbb{N}$  that describe the distances between the levels.

The 1-component  $K_1$  consists of a single point  $x$  with  $\phi_1(x) = 1$ . All higher components are constructed inductively from this 1-component. Given the  $(k - 1)$ -component  $K_{k-1}$  we construct  $K_k$  as follows: Let  $K_{k-1}^{(0)}$  and  $K_{k-1}^{(1)}$  be two copies of the  $(k - 1)$ -component  $K_{k-1}$ . For the  $k$ -graph  $G_k$  we first take the disjoint union of the graphs  $G_{k-1}^{(0)}$  and  $G_{k-1}^{(1)}$ . This already yields all the points of  $G_k$ . For the  $k$ -mapping  $\phi_k$  we set  $\phi_k(x) = \phi_{k-1}^{(i)}(x) + i$  for  $x \in V(G_{k-1}^{(i)}) \subset V(G_k)$ . That is, in the first copy the levels stay the same, whereas in the second all levels are shifted by 1. Finally, to complete  $G_k$ , we add one edge of weight  $k - 1$  from the unique point  $s \in V(G_k)$  with  $\phi_k(s) = 1$  to the unique point  $t \in V(G_k)$  with  $\phi_k(t) = k$ . The progression of the first five components is given in Figure 2.

The instance  $P_k$  is now constructed from the  $k$ -component as follows: Let  $K_k^{(1)}, \dots, K_k^{(k+1)}$  be  $k+1$  copies of  $K_k$ . Take the disjoint union of the corresponding  $k$ -graphs  $G_k^{(1)}, \dots, G_k^{(k+1)}$  and connect them by adding edges  $\{x, y\}$  of weight 1 for every two points  $x \in V(G_k^{(i)})$  and  $y \in V(G_k^{(j)})$  with  $\phi_k^{(i)}(x) = \phi_k^{(j)}(y)$ . Note that the sets of points from the same level constitute cliques of diameter and radius 1 and form an optimal solution of cost 1. To simplify notation we omit the indices and write  $\phi_k(x)$  to denote the level of a point  $x \in V(G_k^{(j)}) \subset V(P_k)$ . The distance between two points in  $V(P_k)$  is given by the length of a shortest path.

Let  $(\mathcal{C}_{k'})_{k'=1}^n$  be the clustering produced by complete linkage on  $(V(P_k), \text{dist})$  minimizing the radius or diameter. Recall that  $\mathcal{C}_{k'-1}$  arises from  $\mathcal{C}_{k'}$  by merging two clusters  $A, B \in \mathcal{C}_{k'}$  that minimize the radius or diameter of  $A \cup B$ . Remember that  $t_{\leq x} = \min\{k' \mid \text{cost}(\mathcal{C}_{k'}) \leq x\}$  and that  $\mathcal{H}_x = \mathcal{C}_{t_{\leq x}}$  denotes the smallest clustering with cost smaller or equal to  $x$ . We show in the following two subsections that  $\mathcal{H}_{k-1}$  consists exactly of the  $k+1$  different  $k$ -graphs that make up the instance resulting in the following theorem.

► **Theorem 2.** *For every  $k \in \mathbb{N}$  there exists an instance  $(V(P_k), \text{dist})$  on which complete linkage, minimizing either diameter or radius, computes a solution of diameter  $k$  or radius  $\frac{k}{2}$ , respectively, whereas the cost of an optimal solution is 1.*

### 4.1 A Lower Bound for Diameter-Based Cost

We start with the analysis for diameter-based costs and after that move on to radius-based costs.

► **Lemma 3.** *The distance between two points  $x, y \in V(P_k)$  is at least as big as the difference in levels  $|\phi_k(x) - \phi_k(y)|$ .*

**Proof.** By the inductive construction of the components, an edge of weight  $w$  can cross at most  $w$  levels. Hence the distance between  $x$  and  $y$  is at least  $|\phi_k(x) - \phi_k(y)|$ . ◀

Consider an  $\ell$ -graph  $G_\ell$ . Instead of talking about the cluster  $V(G_\ell)$  in  $(V(P_k), \text{dist})$  we slightly abuse our notation and see  $G_\ell$  as a cluster with  $\text{cost}(G_\ell) = \max_{x, y \in V(G_\ell)} \text{dist}(x, y)$ , i.e., the diameter of  $V(G_\ell)$ . Using the previous lemma we can show inductively that the diameter of any  $\ell$ -graph in  $P_k$  is  $\ell - 1$ .

► **Lemma 4.** *Let  $G_\ell$  be an  $\ell$ -graph contained in  $P_k$ . We have  $\text{cost}(G_\ell) = \ell - 1$ .*

**Proof.** We prove the upper bound  $\text{cost}(G_\ell) \leq \ell - 1$  by induction. The 1-graphs are points and so the claim follows trivially for  $\ell = 1$ . Assume now that we have shown the claim for  $\ell - 1$ . Let  $s, t \in V(G_\ell)$  be points such that  $\text{dist}(s, t) = \text{cost}(G_\ell)$ . If these points lie in the same graph, say  $G_{\ell-1}^{(0)}$ , of the two  $(\ell - 1)$ -graphs  $G_{\ell-1}^{(0)}$  and  $G_{\ell-1}^{(1)}$  that make up  $G_\ell$ , then

$$\text{cost}(G_\ell) = \text{dist}(s, t) \leq \text{cost}(G_{\ell-1}^{(0)}) \leq \ell - 2 < \ell - 1$$

by induction and we are done. Otherwise we may assume that  $s \in V(G_{\ell-1}^{(0)})$  and  $t \in V(G_{\ell-1}^{(1)})$ . This leaves us with another case analysis. If  $s$  is the unique point with level 1 and  $t$  is the unique point in level  $\ell$  in  $G_\ell$  then we are again done, since by construction there exists an edge between  $s$  and  $t$  of weight  $\ell - 1$ . Otherwise one of  $s$  or  $t$  must share a level with a point not in the same  $(\ell - 1)$ -graph as themselves. Without loss of generality we may assume that  $s$  lies in the same level as some  $u \in V(G_{\ell-1}^{(1)})$ . By induction  $\text{dist}(u, t) \leq \ell - 2$  and so

$$\text{cost}(G_\ell) = \text{dist}(s, t) \leq \text{dist}(s, u) + \text{dist}(u, t) \leq 1 + \ell - 2 = \ell - 1.$$

This concludes the proof of the upper bound  $\text{cost}(G_\ell) \leq \ell - 1$ .

To see the lower bound  $\text{cost}(G_\ell) \geq \ell - 1$ , we apply Lemma 3 to the unique point  $s$  with level 1 and the unique point  $t$  with level  $\ell$  in  $G_\ell$ . This shows that  $\text{cost}(G_\ell) \geq \text{dist}(s, t) \geq \ell - 1$ . ◀

The goal now is to show that complete linkage actually reconstructs these graphs as clusters. We already computed the cost of an  $\ell$ -graph and now it is left to observe that merging two  $\ell$ -graphs costs at least  $\ell$ .

► **Lemma 5.** *Complete linkage might merge clusters on  $(V(P_k), \text{dist})$  in such a way that for all  $\ell \leq k$ , the clustering  $\mathcal{H}_{\ell-1}$  consists exactly of the  $\ell$ -graphs that make up  $P_k$ .*

**Proof.** We again prove the claim by induction. Complete linkage always starts with every point in a separate cluster. Since those are exactly the 1-graphs and any merge costs at least 1, the claim follows for  $\ell = 1$ . Suppose now that  $\mathcal{H}_{\ell-1}$  consists exactly of the  $\ell$ -graphs of the instance. Since we are dealing with integer weights, any new merge increases the cost by at least 1 and so we may merge all pairs of  $\ell$ -graphs that form the  $(\ell + 1)$ -graphs. These are cheapest merges as they altogether increase the cost from  $\ell - 1$  to  $\ell$  (see Lemma 4). To finish the proof we are left to show that at this point there are no more free merges left. Take any two  $(\ell + 1)$ -graphs  $G_{\ell+1} \neq G'_{\ell+1}$  contained in the current clustering. If they do not exactly cover the same levels, then the distance between the point in the lowest level to the point in the highest level is strictly more than  $\ell$  by Lemma 3. Hence, we can assume that they share the same levels, say level  $\lambda$  up to level  $\ell + \lambda$ . Denote by  $s$  the unique point in  $V(G_{\ell+1})$  with  $\phi_k(s) = \lambda$  and by  $t$  the unique point in  $V(G'_{\ell+1})$  with  $\phi_k(t) = \ell + \lambda$ . A shortest path connecting  $s$  and  $t$  must contain an edge  $\{u, w\}$  with  $u \in V(G_{\ell+1})$  and  $w \in V(P_k) \setminus V(G_{\ell+1})$ . Such an edge either weights at least  $\ell + 1$  or weights 1 and connects points in the same level, i.e.,  $\phi_k(u) = \phi_k(w)$ . In the first case we directly obtain  $\text{dist}(s, t) \geq \ell + 1$ . In the second case we use Lemma 3 and obtain

$$\begin{aligned} \text{dist}(s, t) &= \text{dist}(s, u) + \text{dist}(u, w) + \text{dist}(w, t) \\ &\geq |\phi_k(s) - \phi_k(u)| + 1 + |\phi_k(w) - \phi_k(t)| \\ &= |\phi_k(s) - \phi_k(t)| + 1 \\ &= \ell + 1. \end{aligned}$$

It follows that  $\mathcal{H}_\ell$  consists exactly of the  $(\ell + 1)$ -graphs that make up  $P_k$ . ◀

**Proof of Theorem 2 (diameter).** Lemma 5 shows that  $\mathcal{H}_{k-1}$  can consist of all the  $k$ -graphs that make up  $P_k$ . There are exactly  $k + 1$  of them and so there is one merge remaining to get a  $k$ -clustering. By definition of  $\mathcal{H}_{k-1}$ , this last merge increases the cost by at least 1 and so the  $k$ -clustering produced by complete linkage costs at least  $k$ , whereas the optimal clustering consisting of the  $k$  individual levels costs 1. ◀

## 4.2 A Lower Bound for Radius-Based Costs

We show that the instance  $(V(P_k), \text{dist})$  also yields a lower bound of  $k/2$  for radius-based costs. This requires some additional work, as we now also have to keep track of the centers that induce an optimal radius. For an  $\ell$ -graph  $G_\ell$  we again slightly abuse the notation and talk about  $G_\ell$  as a cluster with  $\text{cost}(G_\ell) = \min_{c \in V(G_\ell)} \max_{x \in V(G_\ell)} \text{dist}(c, x)$ , the radius of  $V(G_\ell)$ .

To prove Lemma 6 we show that there is a point in  $P_k$  for which the following holds:

- For all but one of the  $\ell$ -graphs that constitute  $G_{2\ell}$  we can find a point that we can reach by an edge of weight 1. Since the diameter of these graphs is  $\ell - 1$ , this is sufficient.
- The remaining  $\ell$ -graph lies in the same  $(\ell + 1)$ -graph as our point and so we are again done by considering the diameter. Also there are no points that induce a smaller radius, since the diameter of  $G_{2\ell}$  is already  $2\ell - 1$ .

► **Lemma 6.** *Let  $G_{2\ell}$  be any of the  $2\ell$ -graphs that constitute  $P_k$  for  $1 \leq \ell \leq \frac{k}{2}$  arbitrary. Then it holds that  $\text{cost}(G_{2\ell}) = \ell$  and furthermore, all optimal centers that induce this cost are themselves already contained in  $G_{2\ell}$  (and not in any other  $2\ell$ -graph).*

**Proof.** By Lemma 4 we know that the diameter of  $G_{2\ell}$  is  $2\ell - 1$ . Thus the radius of  $G_{2\ell}$  is at least  $\ell$ . To show the upper bound of  $\ell$  suppose that  $G_{2\ell}$  covers the levels  $\lambda$  up to  $\lambda + 2\ell - 1$  in  $P_k$ . Consider the unique  $(\ell + 1)$ -graph  $H_{\ell+1}$  contained in  $G_{2\ell}$  covering the levels  $\lambda + \ell - 1$  to  $\lambda + 2\ell - 1$ . Let  $c$  be the unique point in  $H_{\ell+1}$  with level  $\lambda + \ell - 1$ . By Lemma 4 the diameter of  $H_{\ell+1}$  is  $\ell$ , so any point in  $H_{\ell+1}$  is at distance  $\leq \ell$  to  $c$ . Consider now a point  $x \in V(G_{2\ell}) \setminus V(H_{\ell+1})$  and the  $\ell$ -graph  $H_\ell$  containing  $x$ . We claim that  $H_\ell$  contains a point  $y$  with level  $\lambda + \ell - 1$ . If this is not true then  $H_\ell$  covers the levels  $\lambda + \ell$  up to  $\lambda + 2\ell - 1$  and therefore also contains the unique point in  $G_{2\ell}$  with level  $\lambda + 2\ell - 1$ . This is not possible as the unique point in  $G_{2\ell}$  with level  $\lambda + 2\ell - 1$  is already contained in  $H_{\ell+1}$ . So using that the diameter of  $H_\ell$  is  $\ell - 1$  and  $\phi_k(c) = \phi_k(y)$  we obtain

$$\text{dist}(c, x) \leq \text{dist}(c, y) + \text{dist}(y, x) \leq 1 + (\ell - 1) = \ell.$$

Now we prove that all optimal centers must be contained in  $G_{2\ell}$ . For all points  $c \in V(P_k) \setminus V(G_{2\ell})$  we have to show that  $\max_{x' \in V(G_{2\ell})} \text{dist}(c, x') \geq \ell + 1$ . Suppose that  $\phi_k(c) \leq \lambda + \ell - 1$ . Let  $x$  be the unique point in  $G_{2\ell}$  with level  $\lambda + 2\ell - 1$ , we claim that  $\text{dist}(c, x) \geq \ell + 1$ . Consider a shortest path between  $c$  and  $x$  and let  $\{u, w\}$  be an edge on this path with  $u \in V(P_k) \setminus V(G_{2\ell})$  and  $w \in V(G_{2\ell})$ . By construction  $\{u, w\}$  either weights at least  $2\ell$  in which case

$$\text{dist}(c, x) \geq 2\ell \geq \ell + 1$$

or it weights 1 and  $\phi_k(u) = \phi_k(w)$ , so

$$\begin{aligned} \text{dist}(c, x) &= \text{dist}(c, u) + \text{dist}(u, v) + \text{dist}(v, x) \\ &\geq |\phi_k(c) - \phi_k(u)| + 1 + |\phi_k(w) - \phi_k(x)| \\ &= |\phi_k(c) - \phi_k(x)| + 1 \\ &\geq \ell + 1. \end{aligned}$$

In case  $\phi_k(c) \geq \lambda + \ell$  we can prove analogously that  $\text{dist}(c, y) \geq \ell + 1$  for the unique point  $y$  in  $G_{2\ell}$  with level  $\lambda$ . This finishes the proof. ◀

Now we make sure that complete linkage completely reconstructs these components. In particular we show that merging  $2\ell$ -graphs which cover the same levels increases the cost of our solution. Here we make use of the fact that sets of optimal centers for any pair of  $2\ell$ -graphs do not intersect. Lemma 7 ensures that the cost indeed increases.

► **Lemma 7.** *Let  $C, D$  be two subsets of  $V(P_k)$  with  $\text{cost}(C) = \text{cost}(D)$ . Let  $Z(C)$  and  $Z(D)$  denote the set of all optimal centers for  $C$  respectively  $D$ . If  $Z(C) \cap Z(D) = \emptyset$  then  $\text{cost}(C \cup D) > \text{cost}(C)$ .*

**Proof.** Let  $x \in V(P_k)$ . Since  $Z(C) \cap Z(D) = \emptyset$  this point can be an optimal center for at most one of the sets. Assume without loss of generality that  $x \notin Z(D)$ . We have

$$\max_{y \in C \cup D} \text{dist}(y, x) \geq \max_{y \in D} \text{dist}(y, x) > \text{cost}(D) = \text{cost}(C)$$

So we have for all  $x \in V(P_k)$  that  $\max_{y \in C \cup D} \text{dist}(y, x) > \text{cost}(C)$  which proves the lemma. ◀

Now, with this we can prove that the merging behavior of complete linkage reconstructs our components. Observe that Theorem 2 is an immediate consequence of Corollary 8.

► **Corollary 8.** *Complete linkage might merge clusters in  $(V(P_k), \text{dist})$  in such a way, that for  $1 \leq \ell \leq \frac{k}{2}$ , the clustering  $\mathcal{H}_\ell$  consists exactly of the  $2\ell$ -graphs that make up  $P_k$ .*

**Proof.** The proof is an analogous induction to Lemma 5. Consider the case  $\ell = 1$ . The first merge increases the cost to 1. Observe by Lemma 6 that the cost of a 2-graph is 1. Furthermore, the same lemma shows that the sets of optimal centers for any pair of 2-graphs do not intersect and so, as shown in Lemma 7 any further merge necessarily has to increase the cost. Hence  $\mathcal{H}_1$  consists exactly of the 2-graphs.

Assume now that the claim holds for  $\mathcal{H}_\ell$ . The induction step works essentially the same as the base case. Any merge will increase the cost of the solution by at least 1 by definition of  $\mathcal{H}_\ell$  and so we might as well merge all  $2\ell$ -graphs that together compose a  $(2\ell + 2)$ -graph as this is a cheapest choice (Lemma 6). Furthermore, any additional merge would increase the cost to at least  $\ell + 2$  (again by Lemma 7) and so  $\mathcal{H}_{\ell+1}$  consist of the  $(2\ell + 2)$ -graphs. ◀

Notice that in our analysis we decided which clusters will be merged by complete linkage whenever it has to choose between two merges of the same cost. However with some adjustments on the instance  $P_k$  we can show a lower bound of  $\Omega(k)$  for both, diameter and radius, for any behavior of complete linkage on ties. For more details we refer to Appendix B.

## 5 An Upper Bound for Complete Linkage

Even though complete linkage is often used when it comes to computing a hierarchical clustering, there are no known non-trivial upper bounds for its approximation guarantee in general metric spaces, to the best of the authors' knowledge. We give an upper bound for complete linkage for hierarchical  $k$ -center and hierarchical  $k$ -diameter.

### 5.1 An Upper Bound for Radius-Based Cost

We show that the approximation ratio of the radius of any  $k$ -clustering  $\mathcal{C}_k$  produced by complete linkage relative to an optimal  $k$ -center clustering is in  $O(k)$ .

► **Theorem 9.** *Let  $(\mathcal{C}_k)_{k=1}^n$  be the hierarchical clustering computed by complete linkage on  $(P, \text{dist})$  optimizing the radius. For all  $1 \leq k \leq n$  the radius cost  $\text{cost}(\mathcal{C}_k)$  is upper bounded by  $O(k) \text{cost}(\mathcal{O}_k)$ , where  $\mathcal{O}_k$  is an optimal  $k$ -center clustering.*

To simplify the notation we fix an arbitrary  $k$  and assume that the optimal  $k$ -clustering  $\mathcal{O} = \mathcal{O}_k$  has cost  $\text{cost}(\mathcal{O}) = \frac{1}{2}$ . The latter is possible without loss of generality by scaling the metric appropriately.

We split the proof of Theorem 9 into two parts. In the first, we derive a crude upper bound for the increasing cost of clusterings produced during the execution of complete linkage. This part follows the work of Ackermann et al. [1], who use the same bound to estimate the cost of some few merge steps. Proposition 12 shows that the difference in cost between  $\mathcal{C}_k$  and  $\mathcal{C}_t$  for  $t > k$  is at most  $\lceil \log(t - k) \rceil + 1$ . That is,  $\text{cost}(\mathcal{C}_k) \leq \lceil \log(t - k) \rceil + 1 + \text{cost}(\mathcal{C}_t)$  holds for all  $1 \leq k < t \leq n$ . A clustering  $\mathcal{C}_t$  whose cost we can estimate directly (i.e. without referring to any other clustering) thus yields a proper upper bound for  $\text{cost}(\mathcal{C}_k)$ . Ideally, this clustering should consist of relatively few clusters (so that  $\lceil \log(t - k) \rceil$  is small), while at the same time not being too expensive. Of course, however, these criteria oppose each other. Naively choosing the initial clustering  $\mathcal{C}_t = \mathcal{C}_n$  is not good enough. Although its cost is minimal, the number of clusters is too high, only yielding an upper bound of  $\text{cost}(\mathcal{C}_k) \leq \lceil \log(n - k) \rceil + 1$ . In the second part of the proof we thus set out to find a different clustering to start from.

### Part 1: An estimate of the relative difference in cost

When dealing with radii, any merge done by complete linkage previous to reaching a  $k$ -clustering increases the cost by at most  $2 \text{cost}(\mathcal{O}) = 1$  (Figure 1). This is due to the fact that the centers of two of those clusters are contained in the same optimal cluster.

We show that complete linkage clusterings at times  $t_{\leq x}$  and  $t_{\leq x+1}$  can have at most  $k$  clusters in common. All other clusters from  $\mathcal{H}_x$  are merged in  $\mathcal{H}_{x+1}$ .

► **Lemma 10.** *For all  $x \geq 0$  the clustering  $\mathcal{H}_{x+1}$  contains at most  $k$  clusters of cost at most  $x$ . In particular, it holds that  $|\mathcal{H}_{x+1} \cap \mathcal{H}_x| \leq k$ .*

**Proof.** Assume on the contrary that there exist  $k+1$  pairwise different clusters  $D_1, \dots, D_{k+1}$  at time  $t_{\leq x+1}$  of cost at most  $x$ . Denote by  $d_i \in D_i$  a point that induces the smallest radius, i.e.  $\text{cost}(D_i) = \max_{d \in D_i} \text{dist}(d, d_i)$  for all  $i$ . Then two of these points, say  $d_1$  and  $d_2$ , have to be contained in the same optimal cluster  $O \in \mathcal{O}$ . Hence, we know that

$$\text{cost}(D_1 \cup D_2) \leq 1 + \max_{i \in \{1,2\}} \text{cost}(D_i) \leq 1 + x$$

because  $\text{dist}(d_1, d_2) \leq 2 \text{cost}(O) \leq 2 \text{cost}(\mathcal{O}) = 1$  and  $\text{cost}(D_i) \leq x$  for  $i = 1, 2$ . This contradicts the definition of  $\mathcal{H}_{x+1}$ , as  $D_1$  and  $D_2$  can still be merged without pushing the cost beyond  $x+1$ . ◀

With this we can upper bound  $|\mathcal{H}_{x+i}|$  in terms of  $|\mathcal{H}_x|$  for all  $i \in \mathbb{N}$ . The proof of Corollary 11 can be found in Appendix C.

► **Corollary 11.** *For all  $i \in \mathbb{N}_+$  and  $x \geq 0$  it holds that  $|\mathcal{H}_{x+i}| \leq k + \frac{1}{2^i}(|\mathcal{H}_x| - k)$ .*

► **Proposition 12.** *For all  $k < t \leq n$  it holds that  $\text{cost}(\mathcal{C}_k) \leq \lceil \log(t - k) \rceil + 1 + \text{cost}(\mathcal{C}_t)$ .*

**Proof.** Let  $x = \text{cost}(\mathcal{C}_t)$ , so that  $\mathcal{H}_x$  consists of at most  $t$  clusters. Applying Corollary 11 with  $i = \lceil \log(t - k) \rceil + 1$  then shows that

$$|\mathcal{H}_{x+i}| < k + \frac{1}{t - k}(|\mathcal{H}_x| - k) \leq k + 1.$$

That is,  $\mathcal{H}_{x+i}$  emerges from  $\mathcal{C}_k$  by merging some (or none) of its clusters and we can conclude that  $\text{cost}(\mathcal{C}_k) \leq \text{cost}(\mathcal{H}_{x+i}) \leq x + i = \text{cost}(\mathcal{C}_t) + \lceil \log(t - k) \rceil + 1$ . ◀

### Part 2: A cheap clustering with few clusters

Suppose that there exists a complete linkage clustering  $\mathcal{C}_t$  for some  $t > k$  with  $t \in O(2^k)$  clusters and  $\text{cost}(\mathcal{C}_t) \in O(k)$ . Then applying Proposition 12 shows that

$$\text{cost}(\mathcal{C}_k) \in \log(O(2^k)) + 1 + O(k) = O(k) = O(k) \text{cost}(\mathcal{O})$$

and Theorem 9 is proven (recall that  $\text{cost}(\mathcal{O}) = \frac{1}{2}$ ). We show that  $\mathcal{C}_t = \mathcal{H}_{4k+2}$  is a sufficiently good choice. To estimate the size of  $\mathcal{H}_{4k+2}$ , we distinguish between active and inactive clusters. Remember that  $\mathcal{O}_C = \{O \in \mathcal{O} \mid O \cap C \neq \emptyset\}$  is the set of optimal clusters hit by  $C$ .

► **Definition 13.** *We call a cluster  $C \in \mathcal{H}_x$  active, if  $\text{cost}(C) \leq 4 \cdot |\mathcal{O}_C|$ , or if there exists an active cluster  $C' \in \mathcal{H}_{x-1}$  such that  $\mathcal{O}_C \subseteq \mathcal{O}_{C'}$ . Otherwise,  $C$  is called inactive.*

## 18:12 Upper and Lower Bounds for Complete Linkage in General Metric Spaces

The behavior that makes complete linkage more difficult to analyze than single linkage is that the former sometimes merges clusters that are quite far apart. That is, contrary to single linkage, complete linkage can produce clusters that are very expensive relative to the number of optimal clusters hit by them. We mark such clusters as inactive and count them directly the first time they are created. We will see that the number of such clusters is small. However the number of active clusters is potentially large, but if the cost of the clustering reaches  $4k + 2$ , this number can also be bounded as we see in the following lemma.

► **Lemma 14.** *There are at most  $2^k$  active clusters in  $\mathcal{H}_{4k+2}$ .*

**Proof.** Notice that at time  $t_{\leq 4k+2}$  there cannot exist two active clusters  $C_1$  and  $C_2$  with  $\mathcal{O}_{C_1} \subseteq \mathcal{O}_{C_2}$ . Indeed, since  $C_2$  hits all the optimal clusters hit by  $C_1$  we get that

$$\text{cost}(C_1 \cup C_2) \leq \text{cost}(C_2) + 1 \leq 4|\mathcal{O}_{C_2}| + 1 \leq 4k + 1$$

and so  $C_1$  and  $C_2$  would have been merged in  $\mathcal{H}_{4k+2}$ . Now, if there are more than  $2^k$  active clusters in  $\mathcal{H}_{4k+2}$ , then at least two of them must hit exactly the same set of optimal clusters. Since we have just ruled this out, the lemma follows. ◀

We estimate the number of inactive clusters, by looking at the circumstances under which they arise. As it happens, at each step there are not many clusters whose merge yields an inactive cluster.

► **Lemma 15.** *There are at most  $4k^2 + k$  inactive clusters in  $\mathcal{H}_{4k+2}$ .*

**Proof.** Let  $m_x$  be the number of inactive clusters in  $\mathcal{H}_x$ . We show that the recurrence relation  $m_x \leq m_{x-1} + k$  holds for any  $x \in \mathbb{N}$ . In that case  $m_{4k+2} \leq (4k+1)k = 4k^2 + k$  since  $m_1 = 0$  and we are done.

To prove the recurrence relation first fix some arbitrary  $x \in \mathbb{N}$  and let  $D \in \mathcal{H}_x$  be an inactive cluster. Let  $D_1, \dots, D_\ell \in \mathcal{H}_{x-1}$  be the clusters whose merge results in  $D$ . We show that none of them can be active at time  $t_{\leq x-1}$  and have cost at least  $x - 2$ . Since this only leaves few possible clusterings, we get the recurrence inequality given above. Suppose that for one of the clusters, say  $D_i$ , it holds that  $4 \cdot |\mathcal{O}_{D_i}| \geq \text{cost}(D_i) \geq x - 2$ . Right away, notice that  $|\mathcal{O}_{D_i}| < |\mathcal{O}_D|$  since otherwise  $D$  would also be active by definition. But then

$$\text{cost}(D) \leq x \leq \text{cost}(D_i) + 2 \leq 4|\mathcal{O}_{D_i}| + 2 < 4(|\mathcal{O}_{D_i}| + 1) \leq 4|\mathcal{O}_D|$$

contradicts the assumption of  $D$  being inactive. As such, we know that all  $D_i$  ( $i = 1, \dots, \ell$ ) must be inactive or have cost less than  $x - 2$ . In other words, each inactive cluster in  $\mathcal{H}_x$  descends from the set

$$\{D \in \mathcal{H}_{x-1} \mid D \text{ is inactive}\} \cup \{D \in \mathcal{H}_{x-1} \mid \text{cost}(D) < x - 2\}.$$

The cardinality of the set on the left is  $m_{x-1}$  and, by Lemma 10, the cardinality of the set on the right is at most  $k$ . This proves the claim. ◀

► **Corollary 16.**  *$\mathcal{H}_{4k+2}$  consists of at most  $2^k + 4k^2 + k$  clusters.*

Notice that Theorem 9 is an immediate consequence of Corollary 16 and Proposition 12.



## 5.2 An Upper Bound for Diameter-Based Cost

The main challenge in proving an upper bound on the approximation guarantee of complete linkage when replacing the  $k$ -center objective by the  $k$ -diameter objective is to deal with the possibly large increase of cost after a merge step (see Figure 1).

For some arbitrary but fixed  $k$  let  $\mathcal{O}$  denote an optimal  $k$ -diameter solution and assume that  $\text{cost}(\mathcal{O}) = 1$  from now on. To motivate our approach consider the clustering  $\mathcal{H}_1$  computed by complete linkage at time  $t_{\leq 1}$ . Observe that every optimal cluster can fully contain at most one cluster from  $\mathcal{H}_1$ , as the union of such clusters would cost at most 1. Now, consider the graph  $G = (V, E)$  with  $V = \mathcal{O}$  and edges  $\{A, B\} \subset V$  for every cluster  $C \in \mathcal{H}_1$  intersecting  $A$  and  $B$ . If there is such an edge  $\{A, B\}$ , then the cost of merging  $A$  and  $B$  is upper bounded by 3. We can go even further and consider the merge of all optimal clusters in a connected component of  $G$ . Suppose the size of the connected component is  $m$ , then the resulting cluster costs at most  $2m - 1$ . There are two extreme cases in which we could end up: if  $E = \emptyset$ , then  $\mathcal{H}_1 = \mathcal{O}$  and complete linkage has successfully recovered the optimal solution. On the other hand, if  $G$  is connected, then merging all points costs at most  $2k - 1$  and we get an  $O(k)$ -approximative solution. The remaining cases are more difficult to handle. We proceed by successively adding edges between optimal clusters, while maintaining the property that for a connected component  $Z$  in  $G$  merging  $\cup_{A \in V(Z)} A$  costs at most  $|V(Z)|^2$ . This leads to an upper bound of  $k^2$  for all clusters  $C$  constructed by complete linkage with  $C \subset \cup_{A \in V(Z)} A$ . We show that the number of clusters which do not admit this property is sufficiently small, such that in the end, we are able to prove that  $\mathcal{H}_{k^2}$  consists of at most  $k$  clusters. This immediately leads the following theorem.

► **Theorem 17.** *Let  $(\mathcal{C}_k)_{k=1}^n$  be the hierarchical clustering computed by complete linkage on  $(P, \text{dist})$  optimizing the diameter. For all  $1 \leq k \leq n$  the diameter cost  $\text{cost}(\mathcal{C}_k)$  is upper bounded by  $k^2 \text{cost}(\mathcal{O}_k)$ , where  $\mathcal{O}_k$  is an optimal  $k$ -diameter clustering.*

Essential for this section is a sequence of *cluster graphs*  $G_t = (V_t, E_t)$  for  $t = 1, \dots, k^2$  constructed directly on the set  $V_t = \mathcal{O}$  of optimal  $k$ -clusters. We start with the cluster graph  $G_1$  that contains edges  $\{A, B\}$  for every two vertices  $A, B \in V_1 = \mathcal{O}$  that are hit by a common cluster from  $\mathcal{H}_1$ . To this we successively add edges based on a vertex labeling in order to create the remaining cluster graphs  $G_2, \dots, G_{k^2}$ . The labeling distinguishes vertices as being either *active* or *inactive*. We denote the set of active vertices in  $V_t$  by  $V_t^a$  and the set of inactive ones by  $V_t^i$ . In the beginning ( $t = 1$ ) the inactive vertices are set to precisely those that are isolated:  $V_1^i = \{O \in V_1 \mid \delta_{G_1}(O) = \emptyset\}$ . For  $t \geq 2$ , the labeling is outlined in Definition 18. Over the course of time, active vertices may become inactive, but inactive vertices never become active again.

Given a labeling for  $V_{t+1}$ , we construct  $G_{t+1}$  from  $G_t$  by adding additional edges: If there are two active vertices  $A, B \in V_{t+1}^a$  that are both hit by a common cluster from  $\mathcal{H}_{t+1}$ , we add an edge  $\{A, B\}$  to  $E_{t+1}$ .

► **Definition 18.** *Let  $A \in V_{t+1}$  be an arbitrary optimal cluster and  $Z_A$  the connected component in  $G_t$  that contains  $A$ . We call  $A$  inactive (i.e.,  $A \in V_{t+1}^i$ ) if  $\lceil \text{cost}(Z_A) \rceil \leq t$ , and active otherwise. Here, and in the following  $\text{cost}(Z_A) = \text{cost}(\cup_{B \in V(Z_A)} B)$  denotes the cost of merging all optimal clusters contained in  $V(Z_A)$ .*

Thus if a connected component in  $G_t$  has small cost, then all vertices in this component become inactive in  $G_{t+1}$  by definition. We state the following useful properties of inactive vertices in  $(G_t)_{t=1}^{k^2}$ .

► **Lemma 19.** *If  $Z$  is a connected component in  $G_{t+1}$  with  $V(Z) \cap V_{t+1}^i \neq \emptyset$ , then*

1.  $Z$  is also a connected component in  $G_t$  and  $\lceil \text{cost}(Z) \rceil \leq t$ ,
  2. we have  $V(Z) \subseteq V_{t+1}^i$ , i.e., all vertices in  $Z$  become inactive at the same time.
- Moreover we have  $V_t^i \subseteq V_{t+1}^i$ , so once vertices become inactive, they stay inactive. Equivalently,  $V_{t+1}^a \subseteq V_t^a$ .

**Proof.** Take any inactive vertex  $A \in V_{t+1}^i \cap V(Z)$  and consider the connected component  $Z_A$  in  $G_t$  containing  $A$ . By Definition 18, we have that  $\lceil \text{cost}(Z_A) \rceil \leq t$  and so all other vertices in  $Z_A$  have to be in  $V_{t+1}^i$  as well. We observe that  $E_{t+1} \setminus E_t$  only contains edges between vertices from  $V_{t+1}^a$  by construction. This shows  $Z = Z_A$ .

It is left to show that inactive vertices stay inactive. For  $t = 1$  the inactive vertices  $V_1^i$  are already connected components with cost at most 1. As such, they remain inactive at step  $t = 2$ . For  $t \geq 2$ , consider an inactive vertex  $A \in V_t^i$  and the connected component  $Z \subseteq G_t$  containing it. We showed previously that  $V(Z) \subseteq V_t^i$  and so  $Z$  is also a connected component in  $G_{t+1}$  with  $\lceil \text{cost}(Z) \rceil \leq t - 1 < t$  and thus  $A \in V(Z) \subseteq V_{t+1}^i$ . ◀

► **Definition 20.** *Let  $C \in \mathcal{H}_t$  for some fixed  $t \in \mathbb{N}$ . We define  $\mathcal{I}_t = \{C \in \mathcal{H}_t \mid \mathcal{O}_C \cap V_t^i \neq \emptyset\}$  as the set of all clusters in  $\mathcal{H}_t$  which hit at least one inactive vertex of  $G_t$ . We call these clusters inactive and all clusters from  $\mathcal{H}_t \setminus \mathcal{I}_t$  active.*

We prove the following easy property about active clusters.

► **Lemma 21.** *If  $C \in \mathcal{H}_t \setminus \mathcal{I}_t$ , then  $G_t[\mathcal{O}_C]$  forms a clique. In particular there exists a connected component in  $G_t$  that fully contains  $\mathcal{O}_C$ .*

**Proof.** By definition of  $\mathcal{I}_t$ ,  $\mathcal{O}_C$  must consist exclusively of active vertices. Since all of them are hit by  $C \in \mathcal{H}_t$  there exists an edge  $\{A, B\} \in E_t$  for every pair  $A, B \in \mathcal{O}_C$ . In other words,  $G_t[\mathcal{O}_C]$  forms a clique and the claim follows. ◀

This does not necessarily hold for an inactive cluster  $C \in \mathcal{I}_t$ . As  $C$  contains at least one inactive vertex, the connected component  $Z$  which contains this vertex does not grow. If later on complete linkage merges  $C$  with another cluster the result is an inactive cluster which may hit vertices outside of  $Z$ . So  $G_{t'}$  does not reflect the progression of  $C$  for  $t' \geq t$ . However, the number of such clusters cannot exceed  $|V_t^i|$ .

► **Lemma 22.** *The number of inactive clusters in  $\mathcal{H}_t$  is at most the number of inactive vertices at time  $t$ . That is,  $|\mathcal{I}_t| \leq |V_t^i|$  holds for all  $t \in \mathbb{N}$ .*

**Proof.** We prove the claim by showing that the following inductive construction defines a family of injective mappings  $\phi_t : \mathcal{I}_t \rightarrow V_t^i$ :

- Let  $C \in \mathcal{I}_1$  be an inactive cluster. By definition  $C$  thus has to intersect an inactive optimal cluster  $A \in V_1^i$ . Actually, there can only be one such cluster, as any other optimal cluster that is hit would induce an edge incident to  $A$  in  $G_1$ , making it active. Set  $\phi_1(C) = A$ , so that  $\mathcal{O}_C = \{\phi_1(C)\}$ .
- For  $t > 1$  and  $C \in \mathcal{I}_t$  we distinguish two cases: If there is no cluster in  $\mathcal{I}_{t-1}$  that is a subset of  $C$ , we pick an arbitrary but fixed  $A \in \mathcal{O}_C \cap V_t^i$  and set  $\phi_t(C) = A$ . Otherwise, we know that  $C$  must descend from some cluster  $D \in \mathcal{I}_{t-1}$  and we can set  $\phi_t(C) = \phi_{t-1}(D)$ . Since  $\phi_{t-1}(D) \in V_{t-1}^i \subseteq V_t^i$  by Lemma 19, this shows that  $\phi_t$  really maps into  $V_t^i$ .

Suppose that there exist two inactive clusters  $C, D \in \mathcal{I}_1$  that are mapped to the same inactive vertex  $A \in V_1^i$ . Then, by the construction of  $\phi_1$ ,  $\mathcal{O}_C = \{A\} = \mathcal{O}_D$  shows that  $C$  and  $D$  are actually fully contained in the same optimal cluster. The optimal cluster has diameter at most 1 and so  $C$  and  $D$  would have already been merged in  $\mathcal{H}_1$ . As this is not possible,  $\phi_1$  has to be injective.

Now, let  $t \geq 2$  be arbitrary and assume  $\phi_{t-1}$  to be injective. We show that in that case  $\phi_t$  also has to be injective. Suppose on the contrary, that there exist two different clusters  $C, D \in \mathcal{I}_t$  with  $\phi_t(C) = \phi_t(D)$ . We distinguish three cases.

**Case 1:** Both  $C$  and  $D$  descend from (i.e., contain) clusters  $C', D' \in \mathcal{I}_{t-1}$  with  $\phi_t(C) = \phi_{t-1}(C')$  and  $\phi_t(D) = \phi_{t-1}(D')$ , respectively. Then  $\phi_{t-1}(C') = \phi_t(C) = \phi_t(D) = \phi_{t-1}(D')$  entails that  $C' = D'$ , since  $\phi_{t-1}$  is assumed to be injective. Clearly,  $C' = D'$  cannot end up being a subset of two different clusters in  $\mathcal{I}_t$  and so we end up in a contradiction.

**Case 2:** Neither  $C$  nor  $D$  descend from a cluster in  $\mathcal{I}_{t-1}$ . In other words,  $C$  and  $D$  fully descend from clusters in  $\mathcal{H}_{t-1} \setminus \mathcal{I}_{t-1}$  and so there exist clusters  $C', D' \in \mathcal{H}_{t-1} \setminus \mathcal{I}_{t-1}$  contained in  $C$  and  $D$ , respectively, such that  $A = \phi_t(C) = \phi_t(D) \in \mathcal{O}_{C'} \cap \mathcal{O}_{D'}$ . Applying Lemma 21 yields the existence of a connected component  $Z$  in  $G_{t-1}$  with  $V(Z) \supset \mathcal{O}_{C'} \cup \mathcal{O}_{D'}$ . We show that this connected component has cost at most  $t-1$ . In that case,  $C'$  and  $D'$  should have already been merged in  $\mathcal{H}_{t-1}$ ; a contradiction. To show that  $\text{cost}(Z) \leq t-1$ , consider the connected component  $Z'$  in  $G_t$  containing  $A = \phi_t(C) = \phi_t(D) \in \mathcal{O}_C \cap \mathcal{O}_D \cap V_t^i$ . Since  $A$  was chosen from a subset of  $V_t^i$ , we know from Lemma 19 that  $Z'$  is also a connected component in  $G_{t-1}$  with  $\text{cost}(Z') \leq t-1$ . Now,  $A \in V(Z) \cap V(Z')$  shows that  $Z = Z'$  and so we are done.

**Case 3:**  $D$  contains a cluster  $D' \in \mathcal{I}_{t-1}$ , so that  $\phi_t(D) = \phi_{t-1}(D') \in V_{t-1}^i$ , whereas  $C$  does not. (The symmetric case with the roles of  $C$  and  $D$  swapped is left out.) Since  $C$  fully descends from  $\mathcal{H}_{t-1} \setminus \mathcal{I}_{t-1}$ , we know that  $\mathcal{O}_C \subseteq V_{t-1}^a$ . But this already yields a contradiction:  $V_{t-1}^a \ni \phi_t(C) = \phi_t(D) = \phi_{t-1}(D') \in V_{t-1}^i$ .

This covers all possible cases, with each one ending in a contradiction. Hence  $\phi_t$  has to be injective and by induction this holds for all  $t \in \mathbb{N}$ .  $\blacktriangleleft$

Active clusters from  $\mathcal{H}_t$  are nicely represented by the graph  $G_t$  as it is shown in Lemma 21. We can indirectly bound the cost of active clusters by bounding the cost of the connected components they are contained in.

► **Lemma 23.** *Let  $Z$  be a connected component in  $G_t$ . If  $V(Z) \subset V_t^a$ , we have  $\text{cost}(Z) \leq |V(Z)|^2$ .*

**Proof.** Again, we prove this via an induction over  $t$ . For  $t = 1$  and  $A, B \in V(Z)$  we want to upper bound the distance between  $p \in A$  and  $q \in B$ . Let  $A = Q_1, \dots, Q_s = B$  be a simple path connecting  $A$  and  $B$  in  $Z$ . We know by definition of  $G_1$  that for  $j = 1, \dots, s-1$  there is a pair of points  $p_j \in Q_j$  and  $q_j \in Q_{j+1}$  with  $\text{dist}(p_j, q_j) \leq 1$ . Using the triangle inequality we obtain

$$\begin{aligned} \text{dist}(p, q) &\leq \text{dist}(p, p_1) + \sum_{j=1}^{s-2} (\text{dist}(p_j, q_j) + \text{dist}(q_j, p_{j+1})) \\ &\quad + \text{dist}(p_{s-1}, q_{s-1}) + \text{dist}(q_{s-1}, q) \\ &\leq 2s - 1. \end{aligned}$$

Here we use that  $q_j$  and  $p_{j+1}$  are in the same optimal cluster, thus the distance between those points is at most one.

Since  $V(Z)$  contains only active vertices we have  $|V(Z)| \geq 2$ . Using the above upper bound on the distance between two points in  $\bigcup_{A \in V(Z)} A$  we obtain

$$\text{cost}(Z) \leq 2|V(Z)| - 1 \leq |V(Z)|^2.$$

## 18:16 Upper and Lower Bounds for Complete Linkage in General Metric Spaces

For  $t > 1$  let  $Z_1, \dots, Z_u$  denote the connected components in  $G_{t-1}$  with  $V(Z) = \bigcup_{j=1}^u V(Z_j)$ . Let  $j, j' \in \{1, \dots, u\}$ . We observe that  $V(Z_j) \subset V(Z) \subset V_t^a \subset V_{t-1}^a$ . Thus we obtain by induction that

$$\text{cost}(Z_j) \leq |V(Z_j)|^2. \quad (1)$$

Suppose that  $\lceil \text{cost}(Z_j) \rceil \leq t - 1$ . Then  $V(Z_j) \subset V_t^i$  by definition, which is a contradiction to  $V(Z) \cap V_t^i = \emptyset$ . So we must have

$$t \leq \lceil \text{cost}(Z_j) \rceil. \quad (2)$$

Combining (1) and (2) we obtain

$$t \leq \sqrt{\lceil \text{cost}(Z_j) \rceil \lceil \text{cost}(Z_{j'}) \rceil} \leq \sqrt{|V(Z_j)|^2 |V(Z_{j'})|^2} = |V(Z_j)| |V(Z_{j'})|. \quad (3)$$

For  $A, B \in V(Z)$  we want to upper bound the distance between  $p \in A$  and  $q \in B$ . Let  $A = Q_1, \dots, Q_s = B$  be a simple path connecting  $A$  and  $B$  in  $Z$  which enters and leaves every connected component  $Z_j$  for  $j \in \{1, \dots, u\}$  at most once. We divide the path into several parts such that every part lies in one connected component from  $\{Z_1, \dots, Z_u\}$ . Let  $1 = m_1 < m_2 < \dots < m_l = s$  such that  $Q_{m_j}, \dots, Q_{m_{j+1}-1}$  lie in one connected component  $Z^{(j)} \in \{Z_1, \dots, Z_u\}$  and  $Z^{(j)} \neq Z^{(j+1)}$  for all  $j \in \{1, \dots, l\}$ . Since  $(Q_{m_{j-1}}, Q_{m_j}) \in E_t$  we know that there exists a cluster in  $\mathcal{H}_t$  that intersects  $Q_{m_{j-1}}$  and  $Q_{m_j}$ , thus there is a pair of points  $p_j \in Q_{m_{j-1}}$  and  $q_j \in Q_{m_j}$  such that  $\text{dist}(p_j, q_j) \leq t$ . We obtain

$$\begin{aligned} \text{dist}(p, q) &\leq \sum_{j=1}^{l-1} (\text{cost}(Z^{(j)}) + \text{dist}(p_j, q_j)) + \text{cost}(Z^{(l)}) \leq \sum_{j=1}^l (|V(Z^{(j)})|^2 + t) \\ &\leq \left( \sum_{j=1}^l |V(Z^{(j)})| \right)^2 = |V(Z)|^2. \end{aligned}$$

For the second inequality we use (1) and  $\text{dist}(p_j, q_j) \leq t$ . For the third inequality we use (3). So we obtain the claimed upper bound on the cost of  $Z$ .  $\blacktriangleleft$

We see that a connected component in  $G_{k^2}$  cannot contain two active clusters, yielding the following upper bound.

**► Lemma 24.** *At time  $t \leq k^2$  the number of active clusters is less than or equal to the number of active vertices. In other words,  $|\mathcal{H}_{k^2} \setminus \mathcal{I}_{k^2}| \leq |V_{k^2}^a|$ .*

**Proof.** By Lemma 21 we know that every cluster  $C \in \mathcal{H}_{k^2} \setminus \mathcal{I}_{k^2}$  is fully contained in a connected component  $Z_C$  from  $G_{k^2}$ . We show that mapping any such  $C$  to an arbitrary vertex in  $Z_C$  yields an injective map  $\varphi : \mathcal{H}_{k^2} \setminus \mathcal{I}_{k^2} \hookrightarrow V_{k^2}^a$ . First, notice that  $\varphi$  is well-defined: If  $Z_C$  contains an inactive vertex, then all its vertices are inactive (Lemma 19), contradicting the choice of  $C$  as active.

Suppose now that there are two different clusters  $C, C' \in \mathcal{H}_{k^2} \setminus \mathcal{I}_{k^2}$  that are mapped to the same vertex  $\varphi(C) = \varphi(C')$ . Then the connected components  $Z_C$  and  $Z_{C'}$ , in which they are embedded, already have to coincide ( $Z_C = Z_{C'}$ ). But we have just shown (Lemma 23), that  $\text{cost}(Z_C) \leq |V(Z_C)|^2 \leq k^2$  and so  $C$  and  $C'$  would have already been merged in  $\mathcal{H}_{k^2}$ . As such the images of both cannot coincide and the map is injective.  $\blacktriangleleft$

Together with the bound for the number of inactive clusters we are now able to prove the theorem.

**Proof of Theorem 17.** Using Lemma 22&24 we obtain  $|\mathcal{H}_{k^2}| = |\mathcal{H}_{k^2} \setminus \mathcal{I}_{k^2}| + |\mathcal{I}_{k^2}| \leq |V_{k^2}^a| + |V_{k^2}^i| = k$ , yielding  $\text{cost}(\mathcal{C}_k) \leq \text{cost}(\mathcal{H}_{k^2}) \leq k^2 \text{cost}(\mathcal{C}_k)$ .  $\blacktriangleleft$

## References

- 1 Marcel R. Ackermann, Johannes Blömer, Daniel Kuntze, and Christian Sohler. Analysis of agglomerative clustering. *Algorithmica*, 69(1):184–215, 2014. doi:10.1007/s00453-012-9717-4.
- 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 J. Comput.*, 49(4), 2020. doi:10.1137/18M1171321.
- 3 Jaroslaw Byrka, Thomas W. Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for  $k$ -median and positive correlation in budgeted optimization. *ACM Trans. Algorithms*, 13(2):23:1–23:31, 2017. doi:10.1145/2981561.
- 4 Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. *SIAM J. Comput.*, 33(6):1417–1440, 2004. doi:10.1137/S0097539702418498.
- 5 Aparna Das and Claire Kenyon-Mathieu. On hierarchical diameter-clustering and the supplier problem. *Theory Comput. Syst.*, 45(3):497–511, 2009. doi:10.1007/s00224-009-9186-6.
- 6 Sanjoy Dasgupta and Philip M. Long. Performance guarantees for hierarchical clustering. *J. Comput. Syst. Sci.*, 70(4):555–569, 2005. doi:10.1016/j.jcss.2004.10.006.
- 7 Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985. doi:10.1016/0304-3975(85)90224-5.
- 8 Anna Großwendt and Heiko Röglin. Improved analysis of complete-linkage clustering. *Algorithmica*, 78(4):1131–1150, 2017. doi:10.1007/s00453-017-0284-6.
- 9 Anna Großwendt, Heiko Röglin, and Melanie Schmidt. Analysis of ward’s method. In Timothy M. Chan, editor, *Proc. of the Thirtieth Annu. ACM-SIAM Symp. on Discrete Algorithms, SODA*, pages 2939–2957. SIAM, 2019. doi:10.1137/1.9781611975482.182.
- 10 Anna-Klara Großwendt. *Theoretical Analysis of Hierarchical Clustering and the Shadow Vertex Algorithm*. PhD thesis, University of Bonn, 2020. URL: <http://hdl.handle.net/20.500.11811/8348>.
- 11 D. Ellis Hershkowitz and Gregory Kehne. Reverse greedy is bad for  $k$ -center. *Inf. Process. Lett.*, 158:105941, June 2020. doi:10.1016/j.ipl.2020.105941.
- 12 Dorit S. Hochbaum. When are np-hard location problems easy? *Ann. Oper. Res.*, 1(3):201–214, 1984. doi:10.1007/BF01874389.
- 13 Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the  $k$ -center problem. *Math. Oper. Res.*, 10(2):180–184, 1985. doi:10.1287/moor.10.2.180.
- 14 Wen-Lian Hsu and George L. Nemhauser. Easy and hard bottleneck location problems. *Discret. Appl. Math.*, 1(3):209–215, 1979. doi:10.1016/0166-218X(79)90044-1.
- 15 Guolong Lin, Chandrashekhar Nagarajan, Rajmohan Rajaraman, and David P. Williamson. A general approach for incremental approximation and hierarchical clustering. *SIAM J. Comput.*, 39(8):3633–3669, 2010. doi:10.1137/070698257.
- 16 Joe H. Ward, Jr. Hierarchical grouping to optimize an objective function. *J. of the Am. Stat. Assoc.*, 58:236–244, 1963. doi:10.1080/01621459.1963.10500845.

## A Single Linkage

Let  $(\mathcal{C}_k)_{i=1}^n$  be the hierarchical clustering computed by single linkage on  $(P, \text{dist})$ . Recall that  $\mathcal{C}_{k-1}$  arises from  $\mathcal{C}_k$  by merging two clusters  $A, B \in \mathcal{C}_k$  that minimize  $\text{dist}(A, B)$ .

We first compare the radius of  $\mathcal{C}_k$  to the cost of an optimal  $k$ -center clustering  $\mathcal{O}$ . We introduce a graph  $G$  whose vertices are the optimal clusters  $V(G) = \mathcal{O}$  and whose edges  $E(G) = \{\{O, O'\} \subseteq \mathcal{O} \mid \text{dist}(O, O') \leq 2 \text{cost}(\mathcal{O})\}$  connect all pairs of optimal clusters  $O, O' \in \mathcal{O}$  with distance at most twice the optimal radius.

We make a similar construction to compare the diameter of  $\mathcal{C}_k$  to the cost of an optimal  $k$ -diameter clustering  $\mathcal{O}'$ . We consider the graph  $G'$  with  $V(G') = \mathcal{O}'$  where two clusters in  $\mathcal{O}'$  are connected via an edge if their distance is at most  $\text{cost}(\mathcal{O}')$ .

## 18:18 Upper and Lower Bounds for Complete Linkage in General Metric Spaces

To estimate the cost of a single linkage cluster  $C \in \mathcal{C}_k$  we look at the optimal clusters hit by  $C$ . The next lemma shows that for any two points in  $C$  we can find a path connecting them that passes through a chain of optimal clusters with distance at most  $2 \text{cost}(\mathcal{O})$  or  $\text{cost}(\mathcal{O}')$  when considering the radius or diameter, respectively. One can already anticipate that this gives an upper bound of  $O(k)$  on the radius or diameter of any such cluster  $C$ . In Figure 3 we see an example of such a cluster  $C$  and the optimal clusters hit by  $C$ .

► **Lemma 25.** *Let  $C \in \mathcal{C}_t$  be a cluster computed by single linkage at a time step  $t \geq k$ . Then the graphs  $G[\mathcal{O}_C]$  and  $G'[\mathcal{O}'_C]$  induced by the vertex set of optimal clusters hit by  $C$  are connected.*

**Proof.** We prove the lemma for  $G[\mathcal{O}_C]$  by induction. At the beginning ( $t = n$ ) the lemma obviously holds, since any cluster contained in  $\mathcal{C}_n$  is a point and thus hits only one single optimal cluster. Assume now that the claim holds for  $t > k$ . By the pigeonhole principle there must exist two clusters  $C, C' \in \mathcal{C}_t$  with two points  $c \in C$  and  $c' \in C'$  lying in the same optimal cluster  $O \in \mathcal{O}$ . We know that  $\text{dist}(C, C') \leq 2 \text{cost}(O) \leq 2 \text{cost}(\mathcal{O})$ . But this value is exactly the objective that single linkage minimizes, so we know in particular that this upper bound also holds for the distance between the clusters  $D, D'$  chosen by single linkage. Combining this with the induction hypothesis that both  $G[\mathcal{O}_D]$  and  $G[\mathcal{O}_{D'}]$  are connected finishes the proof. One proves analogously that  $G'[\mathcal{O}'_C]$  is connected. ◀

As we see in Figure 3 this already yields an upper bound of  $2k \text{cost}(\mathcal{O}')$  on the diameter of  $C$ . We estimate the radius of  $C$  by looking at the paths going through optimal clusters in  $\mathcal{O}_C$  that are at distance at most  $2 \text{cost}(\mathcal{O})$  from one another. Choosing the center appropriately and uncoiling these paths in our original space  $P$  yields our upper bound of  $(2k + 2) \text{cost}(\mathcal{O})$ .

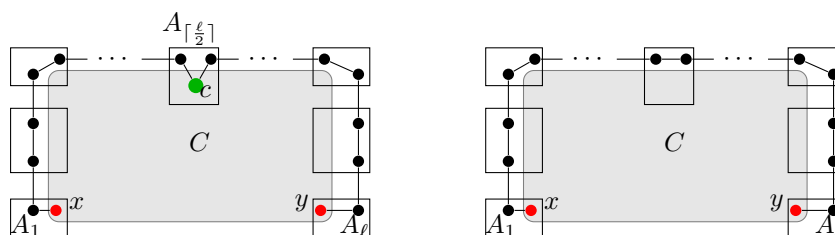
► **Theorem 1.** *Let  $(\mathcal{C}_k)_{k=1}^n$  be the hierarchical clustering computed by single linkage on  $(P, \text{dist})$  and let  $\mathcal{O}_k$  be an optimal clustering for  $k$ -center or  $k$ -diameter, respectively. We have for all  $1 \leq k \leq n$*

1.  $\text{cost}(\mathcal{C}_k) \leq (2k + 2) \cdot \text{cost}(\mathcal{O}_k)$  for the  $k$ -center cost
2.  $\text{cost}(\mathcal{C}_k) \leq 2k \cdot \text{cost}(\mathcal{O}_k)$  for the  $k$ -diameter cost.

**Proof.** We prove the statement for  $k$ -center. Fix an arbitrary time step  $1 \leq k \leq n$  and denote  $\mathcal{O} = \mathcal{O}_k$ . Let  $C \in \mathcal{C}_k$  be an arbitrary cluster and  $P$  a longest simple path in  $G[\mathcal{O}_C]$ . Choose as center for  $C$  an arbitrary vertex  $c \in C \cap O$  from an optimal cluster  $O$  lying in the middle of  $P$ . Note that by this choice every other vertex in  $G[\mathcal{O}_C]$  is reachable from  $O$  by a path of length at most  $\frac{k}{2}$ . Uncoiling such paths in  $P$  gives us an upper bound of  $2(k + 1) \text{cost}(\mathcal{O})$  for the distance between  $c$  and any other point  $z \in C$  as follows: If  $O_z \in \mathcal{O}$  is the optimal cluster containing  $z$ , then by choice of  $O$ , there exists a path  $O = O_1, \dots, O_{\ell+1} = O_z$  in  $G[\mathcal{O}_C]$  of length  $\ell \leq \frac{k}{2}$  connecting them. That means, for each  $i = 1, \dots, \ell$  there exist points  $x_i \in O_i, y_{i+1} \in O_{i+1}$  such that  $\text{dist}(x_i, y_{i+1}) \leq 2 \text{cost}(\mathcal{O})$ . Hence

$$\begin{aligned} \text{dist}(c, z) &\leq \text{dist}(c, x_1) + \sum_{i=1}^{\ell-1} (\text{dist}(x_i, y_{i+1}) + \text{dist}(y_{i+1}, x_{i+1})) \\ &\quad + \text{dist}(x_\ell, y_{\ell+1}) + \text{dist}(y_{\ell+1}, z) \\ &\leq 2(2\ell + 1) \text{cost}(\mathcal{O}) \leq 2(k + 1) \text{cost}(\mathcal{O}). \end{aligned}$$

Using Lemma 25 one proves the statement for  $k$ -diameter analogously. ◀



■ **Figure 3** Cluster  $C$  hits the optimal clusters  $A_1, \dots, A_\ell$  with  $\text{dist}(A_i, A_{i+1}) \leq 2 \text{cost}(\theta)$  when considering the radius on the left and  $\text{dist}(A_i, A_{i+1}) \leq \text{cost}(\theta')$  when considering the diameter on the right. In the left picture, we see that choosing center  $c$  in  $A_{\lceil \frac{\ell}{2} \rceil}$  leads to radius  $\leq 2(\ell + 1) \text{cost}(\theta)$ . Similarly the diameter of  $C$  in the right picture is at most  $2\ell \text{cost}(\theta)$ .

## B A Lower Bound for Complete Linkage without Bad Ties

In this section we focus on modifying the instance  $(V(P_k), \text{dist})$  such that merging two  $\ell$ -graphs  $G_\ell, G'_\ell$  which are part of the same  $(\ell + 1)$ -graph is slightly cheaper than performing any other merge in a clustering consisting of all  $\ell$ -graphs.

### B.1 Diameter-Based Cost

We explain how to adjust the construction of the  $k$ -components for the diameter. Let  $\epsilon \in (0, \frac{1}{2})$ . The definition of  $K_1$  stays the same. As before a  $k$ -component is constructed from two copies  $K_{k-1}^{(0)}, K_{k-1}^{(1)}$  of the  $(k - 1)$ -component by taking the disjoint union of the corresponding graphs and increasing the level of each point in  $K_{k-1}^{(1)}$  by one. Here we do not add an edge of weight  $k - 1$  between the unique point  $s \in V(G_{k-1}^{(0)})$  with level 1 and  $t \in V(G_{k-1}^{(1)})$  with level  $k$ . Instead we complete  $G_k$  by adding edges of weight  $(k - 1)(1 - \epsilon)$  between  $x \in V(G_{k-1}^{(0)})$  and  $y \in V(G_{k-1}^{(1)})$  if they are not on the same level, i.e.,  $\phi_k(x) \neq \phi_k(y)$ .

The instance  $P_k$  is then constructed from  $k$ -copies  $K_k^{(1)}, \dots, K_k^{(k)}$  of the  $k$ -component  $K_k$ . We take the disjoint union of the corresponding  $k$ -graphs  $G_k^{(1)}, \dots, G_k^{(k)}$  and connect them by adding edges  $\{x, y\}$  of weight 1 for every two points  $x \in V(G_k^{(i)})$  and  $y \in V(G_k^{(j)})$  with  $\phi_k^{(i)}(x) = \phi_k^{(j)}(y)$ .

We show that the clustering computed by complete linkage on  $(V(P_k), \text{dist})$  at time  $t_{\leq \ell(1-\epsilon)}$  consists exactly of the  $(\ell + 1)$ -graphs that make up the instance.

► **Lemma 26.** *The distance between two points  $x, y \in V(P_k)$  is at least  $|\phi_k(x) - \phi_k(y)|(1 - \epsilon)$ .*

**Proof.** By the inductive construction of the components, an edge which crosses  $w$  levels costs at least  $w(1 - \epsilon)$ . Hence the distance between  $x$  and  $y$  is at least  $|\phi_k(x) - \phi_k(y)|(1 - \epsilon)$ . ◀

As before we use the previous lemma to show that the diameter of any  $\ell$ -graph in  $P_k$  is  $(\ell - 1)(1 - \epsilon)$ .

► **Lemma 27.** *Let  $G_\ell$  be an  $\ell$ -graph contained in  $P_k$ . We have  $\text{cost}(G_\ell) = (\ell - 1)(1 - \epsilon)$ .*

**Proof.** We prove the upper bound  $\text{cost}(G_\ell) \leq (\ell - 1)(1 - \epsilon)$  by induction. The 1-graphs are points and so the claim follows trivially for  $\ell = 1$ . Assume now that we have shown the claim for  $\ell - 1$ . Let  $G_\ell$  be an  $\ell$ -graph and  $s, t \in V(G_\ell)$  points such that  $\text{cost}(G_\ell) = \text{dist}(s, t)$ . If

## 18:20 Upper and Lower Bounds for Complete Linkage in General Metric Spaces

these points lie in the same graph, say  $G_{\ell-1}^{(0)}$ , of the two  $(\ell-1)$ -graphs  $G_{\ell-1}^{(0)}$  and  $G_{\ell-1}^{(1)}$  that make up  $G_\ell$ , then

$$\text{dist}(s, t) \leq \text{cost}(G_{\ell-1}^{(0)}) \leq (\ell-2)(1-\epsilon) < (\ell-1)(1-\epsilon)$$

by induction and we are done. Otherwise we may assume that  $s \in V(G_{\ell-1}^{(0)})$  and  $t \in V(G_{\ell-1}^{(1)})$ . We distinguish two cases. If  $\phi_k(s) = \phi_k(t)$  these points are connected by an edge of weight one by construction. Notice that an  $\ell$ -graph does not contain points with the same level if  $\ell \leq 2$ . Using  $\epsilon \leq \frac{1}{2}$  and  $\ell \geq 3$  we obtain

$$\text{dist}(s, t) = 1 \leq (\ell-1)(1-\epsilon).$$

If  $s$  and  $t$  are on different levels there is an edge of weight  $(\ell-1)(1-\epsilon)$  between  $s$  and  $t$  by construction. Thus we obtain in all cases

$$\text{cost}(G_\ell) = \text{dist}(s, t) \leq (\ell-1)(1-\epsilon).$$

To see the lower bound  $\text{cost}(G_\ell) \geq (\ell-1)(1-\epsilon)$ , we apply Lemma 26 to the unique point  $s \in V(G_\ell)$  with  $\phi_\ell(s) = 1$  and the unique point  $t \in V(G_\ell)$  with  $\phi_\ell(t) = \ell$ . This shows that  $\text{cost}(G_\ell) \geq \text{dist}(s, t) \geq (\ell-1)(1-\epsilon)$ . ◀

We show that complete linkage must reconstruct these components as clusters.

► **Lemma 28.** *Complete linkage must merge clusters on  $(V(P_k), \text{dist})$  in such a way that for all  $\ell < k$ , the clustering  $\mathcal{H}_{\ell(1-\epsilon)}$  consists exactly of the  $(\ell+1)$ -graphs that make up  $P_k$ .*

**Proof.** We prove the claim by induction. Complete linkage always starts with every point in a separate cluster. Since those are exactly the 1-graphs and any merge of two points costs at least  $(1-\epsilon)$ , the claim follows for  $\ell = 0$ . Suppose now that  $\mathcal{H}_{(\ell-1)(1-\epsilon)}$  consists exactly of the  $\ell$ -graphs of the instance. Consider two  $\ell$ -graphs  $G_\ell \neq G'_\ell$  contained in the current clustering. We compute the cost of merging  $G_\ell$  with  $G'_\ell$ . For this purpose we distinguish whether they are contained in the same  $(\ell+1)$ -graph or not.

**Case 1:** If they are contained in the same  $(\ell+1)$ -graph  $G_{\ell+1}$  merging  $G_\ell$  with  $G'_\ell$  results in  $G_{\ell+1}$ . We obtain by Lemma 27, that  $\text{cost}(G_{\ell+1}) = \ell(1-\epsilon)$ .

**Case 2:** If they are not contained in the same  $(\ell+1)$ -graph, we show that merging  $G_\ell$  with  $G'_\ell$  costs more than  $\ell(1-\epsilon)$ . We make the following observations.

1. The edges connecting  $x \in V(G_\ell)$  and  $y \in V(G'_\ell)$  with  $\phi_k(x) \neq \phi_k(y)$  are of weight  $\geq (\ell+1)(1-\epsilon)$ .
2. There exist  $s \in V(G_\ell)$  and  $t \in V(G'_\ell)$  with  $|\phi_k(s) - \phi_k(t)| \geq \ell-1$ .

The last observation follows from the fact that each of the graphs contains two points whose difference in levels is exactly  $\ell-1$ .

We prove that  $\text{dist}(s, t) > \ell(1-\epsilon)$  and therefore merging  $G_\ell$  with  $G'_\ell$  costs more than  $\ell(1-\epsilon)$ . Any shortest path connecting  $s$  and  $t$  in  $P_k$  must contain an edge  $\{u, w\}$  between a point  $u \in V(G_\ell)$  and a point  $w \in V(G'_\ell)$ . By above observation this edge is either of weight  $\geq (\ell+1)(1-\epsilon)$  or  $u$  and  $w$  are on the same level and the edge is of weight 1. In the first case we conclude

$$\text{dist}(s, t) \geq (\ell+1)(1-\epsilon) > \ell(1-\epsilon).$$



In the second case we obtain that

$$\begin{aligned}
\text{dist}(s, t) &= \text{dist}(s, u) + 1 + \text{dist}(w, t) \\
&\geq |\phi_k(s) - \phi_k(u)|(1 - \epsilon) + 1 + |\phi_k(w) - \phi_k(t)|(1 - \epsilon) \\
&= |\phi_k(s) - \phi_k(t)|(1 - \epsilon) + 1 \\
&\geq (\ell - 1)(1 - \epsilon) + 1 \\
&> \ell(1 - \epsilon).
\end{aligned}$$

We see that  $\mathcal{H}_{\ell(1-\epsilon)}$  must consist exactly of the  $(\ell + 1)$ -graphs of  $P_k$ .  $\blacktriangleleft$

Lemma 28 shows that  $\mathcal{H}_{(k-1)(1-\epsilon)}$  consists of all the  $k$ -graphs that make up  $P_k$ . There are exactly  $k$  of them, thus the  $k$ -clustering produced by complete linkage costs  $(k - 1)(1 - \epsilon)$ .

► **Corollary 29.** *However the tie-breaks are resolved, complete linkage computes a  $k$ -clustering on  $(V(P_k), \text{dist})$  with diameter  $(k - 1)(1 - \epsilon)$  while the optimal  $k$ -clustering has diameter 1.*

## B.2 Radius-Based Cost

We explain how to adjust the construction of the  $k$ -components for the radius. Let  $\epsilon \in (0, \frac{1}{2})$ . The definition of  $K_1$  does not change. As before a  $k$ -component is constructed from two copies  $K_{k-1}^{(0)}, K_{k-1}^{(1)}$  of the  $(k - 1)$ -component by taking the disjoint union of the corresponding graphs and increasing the level of each point in  $K_{k-1}^{(1)}$  by one. We complete  $G_k$  by adding edges between  $x \in V(G_{k-1}^{(0)})$  and  $y \in V(G_{k-1}^{(1)})$  if  $\phi_k(x) \neq \phi_k(y)$  and we assign this edge a weight of  $\lceil \frac{k}{2} \rceil (1 - \epsilon)$  if  $|\phi_k(x) - \phi_k(y)| \leq \lceil \frac{k}{2} \rceil - 1$  and otherwise a weight of  $|\phi_k(x) - \phi_k(y)|(1 - \epsilon)$ .

As before the instance  $P_k$  is constructed from  $k$ -copies  $K_k^{(1)}, \dots, K_k^{(k)}$  of the  $k$ -component  $K_k$ . We take the disjoint union of the corresponding  $k$ -graphs  $G_k^{(1)}, \dots, G_k^{(k)}$  and connect them by adding edges  $\{x, y\}$  of weight 1 for every two points  $x \in V(G_k^{(i)})$  and  $y \in V(G_k^{(j)})$  with  $\phi_k^{(i)}(x) = \phi_k^{(j)}(y)$ . We observe that Lemma 26 still holds on the adjusted instance. Also notice that the diameter of an  $\ell$ -graph is still upper bounded by  $(\ell - 1)(1 - \epsilon)$ .

► **Lemma 30.** *Let  $G_{2\ell}$  be any of the  $2\ell$ -graphs that constitute  $P_k$  for  $1 \leq \ell \leq \frac{k}{2}$ . It holds that  $\text{cost}(G_{2\ell}) = \ell(1 - \epsilon)$ . Furthermore let  $G'_{2\ell}$  be a second  $2\ell$ -graph which is not contained in the same  $2(\ell + 1)$ -graph as  $G_{2\ell}$ . Any cluster containing  $G_{2\ell}$  and  $G'_{2\ell}$  costs at least  $\ell(1 - \epsilon) + 1$ .*

**Proof.** We know that  $G_{2\ell}$  contains points  $s$  and  $t$  with  $|\phi_k(s) - \phi_k(t)| = 2\ell - 1$ . Thus for any  $x \in V(P_k)$  we have  $\max\{|\phi_k(s) - \phi_k(x)|, |\phi_k(t) - \phi_k(x)|\} \geq \ell$ . By Lemma 26 we know that  $\max\{\text{dist}(s, x), \text{dist}(t, x)\} \geq \ell(1 - \epsilon)$  and therefore  $\text{cost}(G_{2\ell}) \geq \ell(1 - \epsilon)$ .

To prove the upper bound suppose that  $G_{2\ell}$  covers the levels  $\lambda$  up to  $\lambda + 2\ell - 1$  in  $P_k$ . Consider the unique  $(\ell + 1)$ -graph  $H_{\ell+1}$  contained in  $G_{2\ell}$  covering the levels  $\lambda + \ell - 1$  to  $\lambda + 2\ell - 1$ . Let  $c$  be the unique point in  $H_{\ell+1}$  with level  $\lambda + \ell - 1$ . Remember that the diameter of  $H_{\ell+1}$  is at most  $\ell(1 - \epsilon)$ , so any point in  $H_{\ell+1}$  is at distance  $\leq \ell(1 - \epsilon)$  to  $c$ . Consider now a point  $x \in V(G_{2\ell}) \setminus V(H_{\ell+1})$ . We know that  $\phi_k(x) < \lambda + 2\ell - 1$ . Thus  $|\phi_k(x) - \phi_k(c)| \leq \ell - 1$ . By construction there exists an edge of weight at most  $\ell(1 - \epsilon)$  between  $x$  and  $c$  and thus  $\text{dist}(x, c) \leq \ell(1 - \epsilon)$ .

It is left to show that any cluster containing  $G_{2\ell}$  and  $G'_{2\ell}$  costs at least  $\ell(1 - \epsilon) + 1$ . Let  $y \in V(P_k)$  and let  $H_{2(\ell+1)}$  be the  $2(\ell + 1)$ -graph containing  $y$ . Assume without loss of generality that  $G_{2\ell}$  is not contained in  $H_{2(\ell+1)}$ . Let  $x \in V(G_{2\ell})$  be a point with  $|\phi_k(x) - \phi_k(y)| \geq \ell$ . We claim that  $\text{dist}(x, y) \geq (\ell - 1)(1 - \epsilon) + 1$ . A shortest path connecting  $x$  and  $y$  must contain an edge  $\{u, w\}$  with  $u \in V(P_k) \setminus V(H_{2(\ell+1)})$  and  $w \in V(H_{2(\ell+1)})$ . We

## 18:22 Upper and Lower Bounds for Complete Linkage in General Metric Spaces

know by construction that either  $\phi_k(u) = \phi_k(w)$ , or the edge weights at least  $(\ell + 2)(1 - \epsilon)$ . In the first case we use Lemma 26 and obtain

$$\begin{aligned} \text{dist}(x, y) &= \text{dist}(x, u) + \text{dist}(u, w) + \text{dist}(w, y) \\ &\geq |\phi_k(x) - \phi_k(u)|(1 - \epsilon) + 1 + |\phi_k(w) - \phi_k(y)|(1 - \epsilon) \\ &= |\phi_k(x) - \phi_k(y)|(1 - \epsilon) + 1 \\ &\geq \ell(1 - \epsilon) + 1 \end{aligned}$$

and in the second case we obtain

$$\text{dist}(x, y) \geq (\ell + 2)(1 - \epsilon) \geq \ell(1 - \epsilon) + 1. \quad \blacktriangleleft$$

This immediately leads the following results.

► **Corollary 31.** *Complete linkage must merge clusters on  $(V(P_k), \text{dist})$  in such a way that for all  $1 \leq \ell \leq \frac{k}{2}$ , the clustering  $\mathcal{H}_{\ell(1-\epsilon)}$  consists exactly of the  $2\ell$ -graphs that make up  $P_k$ .*

► **Corollary 32.** *However the tie-breaks are resolved, complete linkage computes a  $k$ -clustering on  $(V(P_k), \text{dist})$  with radius  $\frac{k}{2}(1 - \epsilon)$ , while the optimal  $k$ -clustering has radius 1.*

### C An Upper Bound for Radius-Based Cost

► **Corollary 11.** *For all  $i \in \mathbb{N}_+$  and  $x \geq 0$  it holds that  $|\mathcal{H}_{x+i}| \leq k + \frac{1}{2^i}(|\mathcal{H}_x| - k)$ .*

**Proof.** First, we consider what happens when we increase the cost by 1. We fix an arbitrary  $x' \geq 0$ . Lemma 10 shows that at most  $k$  clusters from  $\mathcal{H}_{x'}$  are left untouched, while the remaining  $|\mathcal{H}_{x'}| - k$  clusters have to be merged with at least one other cluster (thus at least halving the number of those clusters) to get to  $\mathcal{H}_{x'+1}$ . This yields a bound of

$$|\mathcal{H}_{x'+1}| \leq k + \frac{1}{2}(|\mathcal{H}_{x'}| - k).$$

Now, the case for general  $i \in \mathbb{N}$  follows by a straightforward induction. We have just shown that the claim is true for  $i = 1$ , where we set  $x' = x$ . For the induction step suppose that

$$|\mathcal{H}_{x+i-1}| \leq k + \frac{1}{2^{i-1}}(|\mathcal{H}_x| - k).$$

Substituting this into the inequality

$$|\mathcal{H}_{x+i}| \leq k + \frac{1}{2}(|\mathcal{H}_{x+i-1}| - k),$$

derived from the first part of our proof with  $x' = x + i - 1$ , yields




$$|\mathcal{H}_{x+i}| \leq k + \frac{k + \frac{1}{2^{i-1}}(|\mathcal{H}_x| - k) - k}{2} = k + \frac{1}{2^i}(|\mathcal{H}_x| - k)$$

as claimed. ◀

# On Two-Pass Streaming Algorithms for Maximum Bipartite Matching

Christian Konrad   

Department of Computer Science, University of Bristol, UK

Kheeran K. Naidu   

Department of Computer Science, University of Bristol, UK

---

## Abstract

We study two-pass streaming algorithms for Maximum Bipartite Matching (MBM). All known two-pass streaming algorithms for MBM operate in a similar fashion: They compute a maximal matching in the first pass and find 3-augmenting paths in the second in order to augment the matching found in the first pass. Our aim is to explore the limitations of this approach and to determine whether current techniques can be used to further improve the state-of-the-art algorithms. We give the following results:

We show that every two-pass streaming algorithm that solely computes a maximal matching in the first pass and outputs a  $(2/3 + \epsilon)$ -approximation requires  $n^{1+\Omega(\frac{1}{\log \log n})}$  space, for every  $\epsilon > 0$ , where  $n$  is the number of vertices of the input graph. This result is obtained by extending the Ruzsa-Szemerédi graph construction of [Goel et al., SODA'12] so as to ensure that the resulting graph has a close to perfect matching, the key property needed in our construction. This result may be of independent interest.

Furthermore, we combine the two main techniques, i.e., subsampling followed by the GREEDY matching algorithm [Konrad, MFCS'18] which gives a  $2 - \sqrt{2} \approx 0.5857$ -approximation, and the computation of *degree-bounded semi-matchings* [Esfandiari et al., ICDMW'16][Kale and Tirodkar, APPROX'17] which gives a  $\frac{1}{2} + \frac{1}{12} \approx 0.5833$ -approximation, and obtain a meta-algorithm that yields Konrad's and Esfandiari et al.'s algorithms as special cases. This unifies two strands of research. By optimizing parameters, we discover that Konrad's algorithm is optimal for the implied class of algorithms and, perhaps surprisingly, that there is a second optimal algorithm. We show that the analysis of our meta-algorithm is best possible. Our results imply that further improvements, if possible, require new techniques.

**2012 ACM Subject Classification** Information systems → Data streaming; Mathematics of computing → Matchings and factors; Theory of computation → Communication complexity

**Keywords and phrases** Data streaming, matchings, lower bounds

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.19

**Category** APPROX

**Funding** *Kheeran K. Naidu*: EPSRC Doctoral Training Studentship EP/T517872/1.

## 1 Introduction

In the *semi-streaming model* for processing large graphs, an  $n$ -vertex graph is presented to an algorithm as a sequence of its edges in arbitrary order. The algorithm makes one or few passes over the input stream and maintains a memory of size  $O(n \text{ polylog } n)$ .

The semi-streaming model has been extensively studied since its introduction by Feigenbaum et al. in 2004 [11], and various graph problems, including matchings, independent sets, spanning trees, graph sparsification, subgraph detection, and others are known to admit semi-streaming algorithms (see [23] for an excellent survey). Among these problems, the Maximum Matching problem and, in particular, its bipartite version, the Maximum Bipartite Matching (MBM) problem, have received the most attention (see, for example, [11, 22, 1, 20, 13, 16, 8, 15, 19, 12, 5, 10, 2, 4, 17]).



© Christian Konrad and Kheeran K. Naidu;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 19; pp. 19:1–19:18



Leibniz International Proceedings in Informatics

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

**Algorithm 1** GREEDY Matching.

---

**Input:** Graph  $G = (A, B, E)$

- 1:  $M \leftarrow \emptyset$
- 2: **for each** edge  $e \in E$  (arbitrary order)
- 3:     **if**  $M \cup \{e\}$  is a matching
- 4:          $M \leftarrow M \cup \{e\}$
- 5: **return**  $M$

---

**Algorithm 2** GREEDY <sub>$d$</sub>  Semi-Matching.

---

**Input:** Graph  $G = (A, B, E)$ , integer  $d$

- 1:  $S \leftarrow \emptyset$
- 2: **for each** edge  $ab \in E$  (arbitrary order)
- 3:     **if**  $\deg_S(a) = 0$  **and**  $\deg_S(b) < d$
- 4:          $S \leftarrow S \cup \{ab\}$
- 5: **return**  $S$

---

In this paper, we focus on MBM. The currently best one-pass semi-streaming algorithm for MBM is the GREEDY matching algorithm (depicted in Algorithm 1). GREEDY processes the edges of a graph in arbitrary order and inserts the current edge into an initially empty matching if possible. It produces a maximal matching, which is known to be at least half the size of a maximum matching, and constitutes a  $\frac{1}{2}$ -approximation semi-streaming algorithm for MBM. It is a long-standing open question whether GREEDY is optimal for the class of semi-streaming algorithms or whether an improved approximation ratio is possible. Progress has been made on the lower bound side ([13, 16, 17]), ruling out semi-streaming algorithms with approximation ratio better than  $\frac{1}{1+\ln 2} \approx 0.5906$  [17].

Konrad et al. [20] were the first to show that an approximation ratio better than  $\frac{1}{2}$  can be achieved if two passes over the input are allowed, and further successive improvements [15, 8, 19] led to a two-pass semi-streaming algorithm with an approximation factor of  $2 - \sqrt{2} \approx 0.58578$  [19] (see Table 1 for an overview of two-pass algorithms for MBM).

**Table 1** Two-pass semi-streaming algorithms for Maximum Bipartite Matching.

Approximation Factor	Reference	Comment
$\frac{1}{2} + 0.019$	Konrad et al. [20]	randomized
$\frac{1}{2} + \frac{1}{16} = 0.5625$	Kale and Tirodkar [15]	deterministic
$\frac{1}{2} + \frac{1}{12} \approx 0.5833$	Esfandiari et al. [8]	deterministic
$2 - \sqrt{2} \approx 0.5857$	Konrad [19]	randomized

All known two-pass streaming algorithms proceed in a similar fashion. In the first pass, they run GREEDY in order to compute a maximal matching  $M$ . In the second pass, they pursue different strategies to compute additional edges  $F$  that allow them to increase the size of  $M$ . Two techniques for computing the edge set  $F$  have been used:

1. **Subsampling and Greedy** [19] (see also [20]): Given a bipartite graph  $G = (A, B, E)$  and a first-pass maximal matching  $M$ , they first subsample the edges  $M$  with probability  $p$  and obtain a matching  $M' \subseteq M$ . Then, in the second pass, they compute GREEDY matchings  $M_L$  and  $M_R$  on subgraphs  $G_L = G[A(M') \cup \overline{B(M)}]$  and  $G_R = G[\overline{A(M)} \cup B(M')]$ , respectively, where  $A(M')$  are the matched  $A$ -vertices in  $M'$ ,  $\overline{B(M)}$  are the unmatched  $B$  vertices, and  $B(M')$  and  $\overline{A(M)}$  are defined similarly. It can be seen that if  $M$  is relatively small, then  $M' \cup M_L \cup M_R$  contains many disjoint 3-augmenting paths. Setting  $p = \sqrt{2} - 1$  yields the approximation factor  $2 - \sqrt{2}$ .
2. **Semi-matchings and Greedy <sub>$d$</sub>**  [15, 8]: Given a bipartite graph  $G = (A, B, E)$  and a first-pass maximal matching  $M$ , the second pass consists of finding *degree- $d$ -constrained semi-matchings*  $S_L$  and  $S_R$  on subgraphs  $G_L = G[A(M) \cup \overline{B(M)}]$  and  $G_R = G[\overline{A(M)} \cup B(M)]$ , respectively, using the algorithm GREEDY <sub>$d$</sub>  (as depicted in Algorithm 2). A degree- $d$ -constrained semi-matching in a bipartite graph is a subset of edges  $S \subseteq E$  such

that  $\deg_S(a) \leq 1$  and  $\deg_S(b) \leq d$ , for every  $a \in A$  and  $b \in B$  or vice versa<sup>1</sup>. Similar to the method above, it can be seen that if the matching  $M$  is relatively small,  $M \cup S_L \cup S_R$  contains many disjoint 3-augmenting paths. The setting  $d = 3$  yields the approximation factor  $\frac{1}{2} + \frac{1}{12}$ .

**Our Results.** In this paper, we explore the limitations of this approach and investigate whether current techniques can be used to further improve the state-of-the-art.

Our first result is a limitation result on the approximation factor achievable by algorithms that follow the scheme described above:

► **Theorem 1 (simplified).** *Every two-pass semi-streaming algorithm for MBM that solely runs GREEDY in the first pass has an approximation factor of at most  $\frac{2}{3}$ .*

Our result builds upon a result by Goel et al. [13] who proved that the lower bound of Theorem 1 applies to one-pass streaming algorithms. Their construction relies on the existence of dense *Ruzsa-Szemerédi* graphs with large induced matchings, i.e., bipartite  $2n$ -vertex graphs  $G = (A, B, E)$  with  $|A| = |B| = n$  whose edge sets can be partitioned into disjoint induced matchings such that each matching is of size at least  $(\frac{1}{2} - \delta)n$ , for some small  $\delta$ . Our construction requires similarly dense RS graphs with equally large matchings, however, in addition to these properties, our RS graphs must contain a *near-perfect* matching, i.e., a matching that matches all but a small constant fraction of the vertices. To this end, we augment the RS graph construction by Goel et al.: We show that, for each induced matching  $M$  in Goel et al.'s construction, we can add a matching  $M'$  to the construction without violating the induced matching property such that  $M \cup M'$  forms a near-perfect matching. We believe this result may be of independent interest.

Next, we combine the subsampling and semi-matching techniques and give a meta-algorithm that yields Konrad's and Esfandiari et al.'s algorithms as special cases, thereby unifying two strands of research. Our meta-algorithm is parameterised by a sampling probability  $0 < p \leq 1$  and an integral degree bound  $d \geq 1$ . First, as in the subsampling technique, the edges of the first-pass matching  $M$  are subsampled independently with probability  $p$ , which yields a subset  $M' \subseteq M$ . Next, as in the semi-matching technique, incomplete semi-matchings  $S_L$  and  $S_R$  with degree bounds  $d$  are computed, however, now in the subgraphs  $G'_L = G[A(M') \cup \overline{B(M)}]$  and  $G'_R = G[\overline{A(M)} \cup B(M')]$ . The algorithm then outputs the largest matching among the edges  $M \cup S_L \cup S_R$ .

As our second result, we establish the approximation factor of our meta-algorithm:

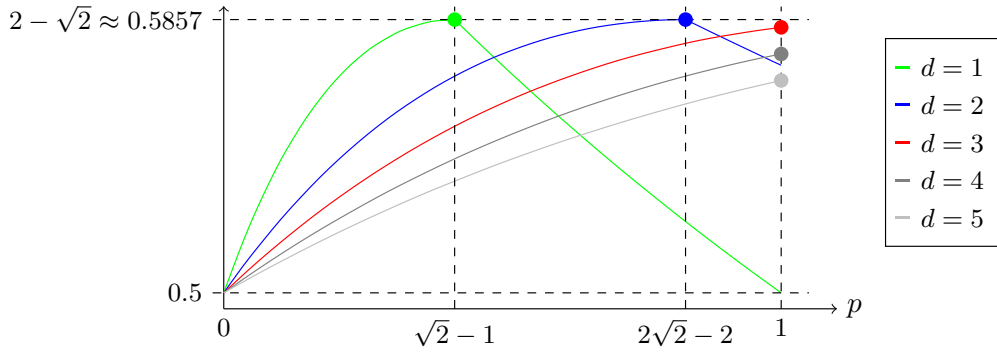
► **Theorem 2 (simplified).** *Combining the subsampling and semi-matching techniques yields a two-pass semi-streaming algorithm for MBM with approximation factor*

$$\begin{cases} \frac{1}{2} + (\frac{1}{d+p} - \frac{1}{2d}) \cdot p, & \text{if } p \leq d(\sqrt{2} - 1) \\ \frac{1}{2} + \frac{d-p}{6d+2p}, & \text{otherwise,} \end{cases}$$

(ignoring lower order terms) that succeeds with high probability.

Interestingly, two parameter settings maximize the approximation factor in Theorem 2, achieving the ratio  $2 - \sqrt{2}$  (see Figure 1). This is achieved by setting  $d = 1$  and  $p = \sqrt{2} - 1$  which recovers Konrad's algorithm, and by setting  $d = 2$  and  $p = 2\sqrt{2} - 2$  which gives a new algorithm. The setting  $d = 3$  and  $p = 1$  yields the slightly weaker bound  $\frac{1}{2} + \frac{1}{12} \approx 0.5833$  and recovers Esfandiari et al.'s algorithm.

<sup>1</sup> The usual definition of a semi-matching requires  $\deg_S(a) = 1$ , for every  $a \in A$  (e.g. [9, 21]). This property is not required here, and, for ease of notation, we stick to this term.



■ **Figure 1** Approximation factors for different settings of  $d$ .

We also show that the analysis of our meta-algorithm is tight, by giving instances on which our meta-algorithm does not perform better than the claimed bound (**Theorem 12**).

**Discussion.** Our results demonstrate that new techniques are needed in order to improve on the  $(2 - \sqrt{2})$  approximation factor. However, one may wonder whether  $2 - \sqrt{2}$  is the best approximation ratio achievable by the class of two-pass matching algorithms that solely computes a maximal matching in the first pass. As pointed out by Kapralov [17], his techniques for establishing the  $\frac{1}{1+\ln 2}$  lower bound for one-pass algorithms can probably also be applied to a construction by Huang et al. [14], which would then show that  $2 - \sqrt{2}$  is the best approximation factor achievable by one-pass semi-streaming algorithms for MBM. It is unclear whether a first-pass GREEDY matching could be embedded in the resulting construction without affecting its hardness, however, if possible, this would render Konrad’s algorithm optimal for the considered class of two-pass streaming algorithms.

**Further Related Work.** Besides two passes over the input, improvements over the GREEDY algorithm can also be obtained under the assumption that the input stream is in random order. Assadi and Behnezhad [2] recently showed that an approximation factor of  $\frac{2}{3} + \epsilon$  can be obtained, for some fixed small but constant  $\epsilon > 0$ , building on Bernstein’s breakthrough result [5], and improving on previous algorithms [5, 10, 19, 20]. In insertion-deletion streams, where previously inserted edges may be deleted again, space  $\tilde{\Theta}(n^{2-3\epsilon})$  is necessary [7] and sufficient [3, 6] for computing a  $n^\epsilon$ -approximation (see also [18]).

**Outline.** We first give notation and definitions in Section 2. Subsequently, we show in Section 3 that every two-pass semi-streaming algorithm that solely runs GREEDY in the first pass cannot have an approximation ratio of  $\frac{2}{3} + \epsilon$ , for any  $\epsilon > 0$ . Our main algorithmic result, i.e., the combination of subsampling and  $\text{GREEDY}_d$ , is presented in Section 4. Finally, we conclude in Section 5.

## 2 Preliminaries

Let  $G = (A, B, E)$  be a bipartite graph with  $V = A \cup B$  and  $|V| = n$ . For  $F \subseteq E$  and  $v \in V$ , we write  $\text{deg}_F(v)$  to denote the degree of vertex  $v$  in subgraph  $(A, B, F)$ . For any  $U \subseteq V$  and  $F \subseteq E$ ,  $U(F)$  denotes the set of vertices in  $U$  which are the endpoints of edges in  $F$ , and we denote its complement by  $\overline{U(F)} = U \setminus U(F)$ . For a subset of vertices  $U \subseteq V$ , we write  $G[U]$

for the subgraph of  $G$  induced by  $U$ . For any edges  $e, f \in E$ ,  $e$  is *incident* to  $f$  if they share an endpoint. We say that  $e$  and  $f$  are *vertex-disjoint* if  $e$  is not incident to  $f$ . Lastly, for any two sets  $X$  and  $Y$ , we define  $X \oplus Y := (X \setminus Y) \cup (Y \setminus X)$  as their symmetric difference.

A *matching* in  $G$  is a subset  $M \subseteq E$  of vertex-disjoint edges. It is *maximal* if every  $e \in E \setminus M$  is incident to an edge in  $M$ . We denote by  $\mu(G)$  the *matching number* of  $G$ , i.e., the cardinality of a largest matching. A *maximum matching* is one of size  $\mu(G)$ . Additionally,  $M$  is called an *induced matching* if the edge set of the subgraph of  $G$  induced by  $V(M)$  is exactly  $M$ .

**Wald's Equation.** We require the following well-known version of *Wald's Equation*:

► **Lemma 3.** *Let  $X_1, X_2, \dots$  be a sequence of non-negative random variables with  $\mathbb{E}[X_i] \leq \tau$ , for all  $i \leq T$ , and let  $T$  be a random stopping time for the sequence with  $\mathbb{E}[T] < \infty$ . Then:*

$$\mathbb{E}\left[\sum_{i=1}^T X_i\right] \leq \tau \cdot \mathbb{E}[T] .$$

### 3 Lower Bound

We now prove that every two-pass streaming algorithm for MBM with approximation factor  $\frac{2}{3} + \epsilon$ , for any  $\epsilon > 0$ , that solely runs GREEDY in the first pass requires space  $n^{1+\Omega(\frac{1}{\log \log n})}$ . To this end, we adapt the lower bound by Goel et al. [13], which we discuss first.

#### 3.1 Goel et al.'s Lower Bound for One-pass Algorithms

Goel et al.'s lower bound is proved in the *one-way two-party communication framework*. Two parties, denoted Alice and Bob, each hold subsets  $E_1$  and  $E_2$ , respectively, of the input graph's edges. Alice sends a single message to Bob who, upon receipt, outputs a large matching. Goel et al. showed that there is a distribution  $\lambda$  over input graphs so that every deterministic communication protocol with constant distributional error over  $\lambda$  and approximation factor  $\frac{2}{3} + \epsilon$ , for any  $\epsilon > 0$ , requires a message of length  $n^{1+\Omega(\frac{1}{\log \log n})}$ . A similar result then applies for randomized constant error protocols by Yao's Lemma [25], and the well-known connection between streaming algorithms and one-way communication protocols allows us to translate this lower bound to a lower bound on the space requirements of constant error one-pass streaming algorithms.

Goel et al.'s construction is based on the existence of a dense Ruzsa-Szemerédi graph:

► **Definition 4** (Ruzsa-Szemerédi Graph). *A bipartite graph  $G = (A, B, E)$  is an  $(r, t)$ -Ruzsa-Szemerédi graph (RS graph in short) if the edge set  $E$  can be partitioned into  $t$  disjoint matchings  $M_1, M_2, \dots, M_t$  such that, for every  $i$ , (1)  $|M_i| \geq r$ ; and (2)  $M_i$  is an induced matching in  $G$ .*

They give a construction for a family of  $((\frac{1}{2} - \delta)n, n^{\Omega(\frac{1}{\log \log n})})$ -RS graphs, for any small constant  $\delta > 0$ , on  $2n$  vertices (with  $|A| = |B| = n$ ) that we will extend further below.

Their hard input distribution  $\lambda$  for the two-party communication setting is displayed in Figure 2. Observe that the graphs  $G \sim \lambda$  are such that  $\mu(G) \geq \frac{3}{2}N$  since the matching  $M_X^* \cup M_Y^* \cup \widehat{M}_s$  is of this size.

Goel et al. prove the following hardness result:

► **Theorem 5.** *For any small  $\epsilon > 0$ , every deterministic  $(\frac{2}{3} + \epsilon)$ -approximation one-way two-party communication protocol with constant distributional error over  $\lambda$  requires a message of size  $n^{1+\Omega(\frac{1}{\log \log n})}$ , where  $n$  is the number of vertices in the input graph.*

1. Let  $G^{RS} = (A, B, E)$  be an  $(r, t)$ -RS graph with  $|A| = |B| = N$  and  $r = (\frac{1}{2} - \delta) \cdot N$ , for some  $\delta > 0$ , and  $t = N^{\Omega(\frac{1}{\log \log N})}$ .
  2. For every  $i \in [t]$ , let  $\widehat{M}_i \subseteq M_i$  be a uniform random subset of size  $(\frac{1}{2} - 2\delta) \cdot N$  and let  $E_1 = \cup_{i=1}^t \widehat{M}_i$ .
  3. Let  $X$  and  $Y$  each be disjoint sets of  $(\frac{1}{2} + \delta) \cdot N$  vertices, which are also disjoint from  $A \cup B$ . Choose uniformly at random a special index  $s \in [t]$ .
  4. Let  $M_X^*$  and  $M_Y^*$  be arbitrary perfect matchings between  $X$  and  $\overline{B(M_s)}$ , and  $Y$  and  $\overline{A(M_s)}$ , respectively. Then, let  $E_2 = M_X^* \cup M_Y^*$ .
  5. Finally,  $G = (A \cup X, B \cup Y, E_1 \cup E_2)$  which has  $n = (3 + 2\delta) \cdot N$  vertices.
- Alice is given edges  $E_1$  and Bob is given edges  $E_2$ .

■ Figure 2 Hard input distribution  $\lambda$ .

### 3.2 Our Lower Bound Construction

In the following, we extend Goel et al.’s lower bound to the two-pass situation where a GREEDY matching is computed in the first pass. To this end, we need to augment Alice and Bob’s inputs, as defined by distribution  $\lambda$ , by a maximal matching  $M$  in the input graph  $G \sim \lambda$ , which then results in a distribution  $\lambda^+$ . Observe that if we place the edges of  $M$  at the beginning of the input stream, then running GREEDY in the first pass recovers exactly the matching  $M$ . Hence, when abstracting the second pass as a two-party communication problem, both Alice and Bob already know the matching  $M$ . Our main argument then is as follows: We will show that any two-party protocol under distribution  $\lambda^+$  can also be used for solving the distribution  $\lambda$  with the same distributional error, message size, and similar approximation factor. The hardness of Theorem 5, therefore, carries over.

#### 3.2.1 Ruzsa-Szemerédi Graphs with Near-Perfect Matchings

Adding a maximal matching  $M$  to Alice’s and Bob’s input requires care since we need to ensure that the hardness of the construction is preserved. Our construction requires that the underlying RS graph contains a near-perfect matching, which is a property that is not guaranteed by Goel et al.’s RS graph construction.

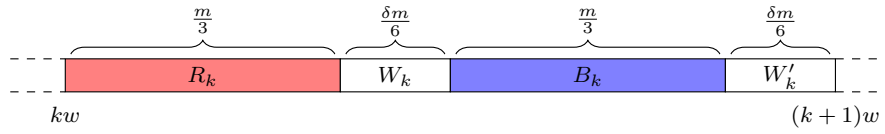
We therefore augment Goel et al.’s construction by complementing every induced matching,  $M_i$ , with a vertex-disjoint counterpart,  $M'_i$ , without destroying the RS graph properties. Then, since  $M_i$  and  $M'_i$  are vertex-disjoint,  $M_i \cup M'_i$  constitutes a matching, and, since both  $M_i$  and  $M'_i$  each already match nearly half of the vertices,  $M_i \cup M'_i$  constitutes a near-perfect matching in our family of RS graphs.

We will now present Goel et al.’s RS graph construction and then discuss how the additional matchings  $M'_i$  can be added to the construction.

#### Goel et al.’s Ruzsa-Szemerédi Graph Construction

For an integer  $m$ , let  $X = Y = [m^2]^m$  be the vertex sets of a bipartite graph, and let  $N = |X| = |Y| = m^{2m}$  denote their cardinalities. Every induced matching  $M_I$  of Goel et al.’s RS graph construction is indexed by a subset of coordinates  $I \subset [m]$  of size  $\frac{\delta m}{6}$ , for some small  $\delta > 0$ . Then, the edges  $M_I$  are defined by means of a colouring of the vertices  $X$  and  $Y$  (which depends on  $I$ ), that we discuss first.





■ **Figure 3** One group of the partitioned number line of natural numbers.

**Colouring the Vertex Sets.** Let  $w = \frac{(2+\delta)m}{3}$ . Then, define a partition of the natural numbers into groups of size  $w$  such that, for all  $k \in \mathbb{N}_0$ ,

$$\begin{aligned}
 R_k &= \left[ kw, kw + \frac{m}{3} \right) && \text{where } |R_k| = \frac{m}{3}, \\
 W_k &= \left[ kw + \frac{m}{3}, kw + \frac{m}{3} + \frac{\delta m}{6} \right) && \text{where } |W_k| = \frac{\delta m}{6}, \\
 B_k &= \left[ kw + \frac{m}{3} + \frac{\delta m}{6}, kw + \frac{2m}{3} + \frac{\delta m}{6} \right) && \text{where } |B_k| = \frac{m}{3}, \\
 W'_k &= \left[ kw + \frac{2m}{3} + \frac{\delta m}{6}, (k+1)w \right) && \text{where } |W'_k| = \frac{\delta m}{6}.
 \end{aligned}$$

See Figure 3 for an illustration.

Given  $I$ , let  $L_s = \{\vec{x} \in [m^2]^m : \sum_{i \in I} x_i = s\}$  represent a layer of vectors in  $[m^2]^m$  where the 1-norm of their subvectors<sup>2</sup> w.r.t.  $I$  is  $s$ , for all  $s \in \mathbb{N}_0$ . Next, colour the vectors in each  $L_s$  either red if  $s \in R_k$ , blue if  $s \in B_k$ , or white if  $s \in W_k \cup W'_k$ , for some  $k \in \mathbb{N}_0$ . Doing this gives the following coloured strips for any  $k \in \mathbb{N}_0$  (see Figure 4):

$$R(k) = \bigcup_{\forall s \in R_k} L_s, \quad W(k) = \bigcup_{\forall s \in W_k} L_s, \quad B(k) = \bigcup_{\forall s \in B_k} L_s \quad \text{and} \quad W'(k) = \bigcup_{\forall s \in W'_k} L_s.$$

Next, these strips are grouped together by colour, as follows:

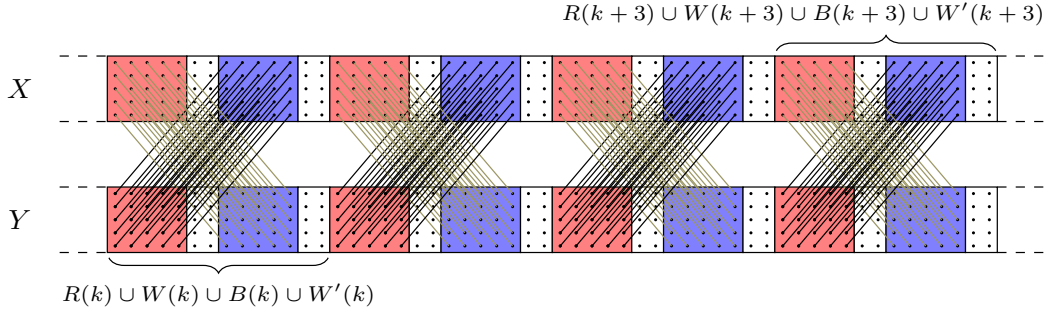
$$R = \bigcup_{\forall k \in \mathbb{N}_0} R(k), \quad W = \bigcup_{\forall k \in \mathbb{N}_0} W(k), \quad B = \bigcup_{\forall k \in \mathbb{N}_0} B(k) \quad \text{and} \quad W' = \bigcup_{\forall k \in \mathbb{N}_0} W'(k).$$

We now define the colours of the vertices  $X$  and  $Y$  as follows: A vertex  $\vec{z} \in X \cup Y$  is coloured red if  $\vec{z} \in R$ , blue if  $\vec{z} \in B$ , and white if  $\vec{z} \in W \cup W'$ . Let  $R^X = R \cap X$  and define  $B^X, W^X, W'^X, R^Y, B^Y, W^Y, W'^Y$  similarly.

**Definition of the Induced Matchings.** Goel et al. construct the edges of the induced matching  $M_I$  by pairing every blue vertex  $\vec{b} \in B^X$  with each coordinate greater than  $\frac{2}{\delta} + 1$  to a red vertex  $\vec{r} \in R^Y$ , such that  $\vec{r} = \vec{b} - (\frac{2}{\delta} + 1) \cdot \vec{1}_I$ , where  $\vec{1}_I$  is the characteristic vector of set  $I$ . See Figure 4 for an illustration.

Goel et al. show that  $M_I$  is large, i.e.,  $|M_I| \geq (\frac{1}{2} - \delta) \cdot N - o(N)$ . Observe that any two distinct indexing sets  $I$  and  $J$  produce their own vertex colourings and matchings  $M_I$  and  $M_J$ . They prove that, as long as the index sets  $I$  and  $J$  have a sufficiently small intersection (at most  $(\frac{5\delta}{12})(\frac{\delta m}{6})$ ),  $M_I$  and  $M_J$  are induced matchings w.r.t. to each other. Hence, they show the existence of a large family  $\mathcal{T}$ , with  $|\mathcal{T}| = N^{\Omega(\frac{1}{\log \log N})}$ , of subsets  $I \subset [m]$  whose pairwise intersections are of size at most  $(\frac{5\delta}{12})(\frac{\delta m}{6})$ . Then, the matchings of the RS graph are identified as the matchings  $M_I$ , for every  $I \in \mathcal{T}$ .

<sup>2</sup> A subvector in this context is the result of a trivial mapping of the vector to a lower dimensional subspace.



■ **Figure 4** Illustration of the vertex colouring and induced matchings for a fixed  $I$ . The black edges are  $M_I$  and the gold ones are  $M'_I$ .

### Extending Goel et al.'s Construction

For every indexing set  $I \in \mathcal{T}$  and respective matching  $M_I$  of Goel et al.'s construction, we symmetrically construct an additional matching  $M'_I$  by pairing every blue vertex in  $Y$  (instead of  $X$ ),  $\vec{b} \in B^Y$ , with each coordinate greater than  $\frac{2}{\delta} + 1$ , to a red vertex in  $X$ ,  $\vec{r} \in R^X$ , such that  $\vec{r} = \vec{b} - (\frac{2}{\delta} + 1) \cdot \vec{1}_I$ . See Figure 4 for an illustration.

We immediately see that, by virtue of being symmetrical,  $|M'_I| = |M_I| (\geq (\frac{1}{2} - \delta) \cdot N - o(N))$ .

Furthermore, by construction,  $M'_I$  and  $M_I$  are vertex-disjoint matchings, hence  $M_I \cup M'_I$  is a matching, and, taking their respective sizes into account,  $M_I \cup M'_I$  is a near-perfect matching as required. Since, for any distinct  $I, J \in \mathcal{T}$ ,  $M_I$  and  $M_J$  are induced matchings w.r.t. each other, the symmetrical nature of our additional matchings implies the same for  $M'_I$  and  $M'_J$ . However, showing that  $M_I$  and  $M'_J$  are induced with respect to each other is not immediately clear. Fortunately, Goel et al.'s proof already implicitly shows this, and, for completeness, we reproduce the decisive argument:

► **Lemma 6.** *Given two distinct sets of indices  $I$  and  $J$  such that  $|I \cap J| \leq (\frac{5\delta}{12})(\frac{\delta m}{6})$ , no edge in  $M_I$  is induced by  $M'_J$ , for any small enough  $\delta > 0$ .*

**Proof.** Let  $\vec{b} \in B^X$  be matched to  $\vec{r} \in R^Y$  by  $M_I$ , i.e.,  $\vec{b} - \vec{r} = (\frac{2}{\delta} + 1) \cdot \vec{1}_I$ . If the edge  $(\vec{b}, \vec{r})$  is induced by  $M'_J$ , then one endpoint is coloured blue and the other red in the colouring of  $X$  and  $Y$  with respect to  $J$ . Hence,  $\vec{b}$  and  $\vec{r}$  are separated by a single white strip (see Figure 4) and

$$|\sum_{j \in J} (\vec{b} - \vec{r})_j| \geq \frac{\delta m}{6}. \quad (1)$$

On the other hand,

$$|\sum_{j \in J} (\vec{b} - \vec{r})_j| = |\sum_{j \in J} ((\frac{2}{\delta} + 1) \cdot \vec{1}_I)_j| = (\frac{2}{\delta} + 1) \cdot |I \cap J| \leq (\frac{5}{6} + \frac{5\delta}{12})(\frac{\delta m}{6}),$$

which contradicts Equation 1 for small enough  $\delta$ . ◀

We thus obtain the following theorem:

► **Theorem 7.** *For any small enough constant  $\delta > 0$ , there exists a family of bipartite  $(r, t)$ -Ruzsa-Szemerédi graphs where  $|A| = |B| = N$ ,  $r = (\frac{1}{2} - \delta) \cdot N$ , and  $t = N^{\Omega(\frac{1}{\log \log N})}$  such that there are  $N^{\Omega(\frac{1}{\log \log N})}$  disjoint near-perfect matchings each of size exactly  $(1 - 2\delta) \cdot N$ .*

1. Let  $G^{RS}$  be an RS graph as in Theorem 7. Fix some induced matching  $M_i$  and let  $M_i \cup M'_i$  be its near-perfect matching of size  $(1 - 2\delta) \cdot N$ .
  2. Let  $F$  be an arbitrary set of  $2\delta N$  additional edges such that  $P = M_i \cup M'_i \cup F$  is a perfect matching in  $G^{RS}$ .
  3. Consider distribution  $\lambda$  constructed using RS graph  $G^{RS} \setminus (M_i \cup M'_i)$ .
  4. For every  $G = (V, E) \sim \lambda$ , let  $P_G = M_i \cup M'_i \cup (F \setminus E)$  (to avoid multi-edges) and add  $P_G$  to  $G$  to obtain the input graph  $G^+$ .
- The edges  $P \cup E_1$  are given to Alice and the edges  $P \cup E_2$  are given to Bob (recall that  $E_1$  and  $E_2$  are defined in distribution  $\lambda$ ).

■ **Figure 5** Hard input distribution  $\lambda^+$ .

### 3.2.2 Lower Bound Proof

Equipped with RS graphs with near-perfect matchings and input distribution  $\lambda$ , we now define our hard input distribution  $\lambda^+$ , see Figure 5.

We are now ready to prove our main lower bound theorem:

► **Theorem 8.** *For any  $\epsilon > 0$ , every deterministic  $(\frac{2}{3} + \epsilon)$ -approximation one-way communication protocol with constant distributional error over  $\lambda^+$  for MBM requires a message of size  $n^{1+\Omega(\frac{1}{\log \log n})}$ , where  $n$  is the number of vertices in the input graph.*

**Proof.** Let  $\gamma^+$  be a deterministic  $(\frac{2}{3} + \epsilon)$ -approximation protocol that solves distribution  $\lambda^+$  with constant distributional error. Given  $\gamma^+$ , we will now define a protocol  $\gamma$  that solves distribution  $\lambda$  with the same communication cost, same error, and approximation ratio strictly better than  $\frac{2}{3}$ . Invoking Theorem 5 then proves our result.

The protocol  $\gamma$  is easy to obtain: Observe that  $P$  in distribution  $\lambda^+$  is the same for every sampled input graph  $G^+ \sim \lambda^+$ . Hence, in protocol  $\gamma$ , Alice and Bob first make sure that the edges  $P$  are included in their inputs. This is achieved by Alice adding the edges  $P \setminus E_1 = P_G$  to her input, and Bob adding the edges  $P$  to his input. In doing so, Alice and Bob's input is equivalently distributed to choosing an input graph  $G^+$  from  $\lambda^+$ . Alice and Bob can, therefore, run protocol  $\gamma^+$  which produces an output matching  $M_{\text{out}}^+$ . Bob then outputs the largest matching  $M_{\text{out}}$  among the edges  $M_X^* \cup M_Y^* \cup (M_{\text{out}}^+ \setminus P_G)$  as the output of the protocol  $\gamma$ .

Next, we will argue that  $|M_{\text{out}}| \geq |M_{\text{out}}^+| - |F| = |M_{\text{out}}^+| - 2\delta N$ . We can construct a matching  $\tilde{M}$  of this size as follows: First, add every edge  $e \in M_{\text{out}}^+$  that is not contained in  $P$  to  $\tilde{M}$ . Second, for every edge  $e \in M_{\text{out}}^+ \cap (M_i \cup M'_i)$ , we insert the incident edge to  $e$  that is contained in  $M_X^* \cup M_Y^*$  into  $\tilde{M}$  (notice that these incident edges always exist except for edges from the special induced matching). This implies that  $|M_{\text{out}}| \geq |\tilde{M}| \geq |M_{\text{out}}^+| - |F|$ .

Recall that  $\mu(G) \geq \frac{2}{3}N$  and, since  $G$  is a subgraph of  $G^+$ ,  $\mu(G^+) \geq \mu(G)$ . This implies that  $N \leq \frac{2}{3}\mu(G^+)$ . Since  $\gamma^+$  is a  $(\frac{2}{3} + \epsilon)$ -approximation protocol, we have  $|M_{\text{out}}^+| \geq (\frac{2}{3} + \epsilon)\mu(G^+)$ , and thus:

$$|M_{\text{out}}| \geq |M_{\text{out}}^+| - 2\delta N \geq (\frac{2}{3} + \epsilon)\mu(G^+) - 2\delta \frac{2}{3}\mu(G^+) = (\frac{2}{3} + \epsilon - \frac{4}{3}\delta)\mu(G^+).$$

Hence, setting  $\delta < \frac{3}{4}\epsilon$  in distribution  $\lambda$  yields a protocol with approximation ratio strictly above  $\frac{2}{3}$ . This, however, implies that  $\gamma$  requires a message of length  $n^{1+\Omega(\frac{1}{\log \log n})}$  (Theorem 5), and since the message sent in  $\gamma$  and  $\gamma^+$  is equivalent, the result follows. ◀

Applying Yao's Lemma and the usual connection between streaming algorithms and one-way communication protocols, we obtain our main lower bound result:

■ **Algorithm 3** Finding Augmenting Paths.

**Input:** A stream of edges  $\pi$  of a bipartite graph  $G = (A, B, E)$ , a maximal matching  $M$  in  $G$ ,  $p \in (0, 1]$  and  $d \in \mathbb{N}^+$ .

- 1: Let  $M' \subseteq M$  be a random subset such that  $\forall e \in M, \Pr[e \in M'] = p$
- 2: Let  $G'_L = G[A(M') \cup \overline{B(M)}]$  and  $G'_R = G[\overline{A(M)} \cup B(M')]$
- 3: Denote by  $\pi_{G'_L}$  ( $\pi_{G'_R}$ ) the substream of  $\pi$  of edges of  $G'_L$  ( $G'_R$ , respectively)
- 4:  $S_L \leftarrow \text{GREEDY}_d(\pi_{G'_L})$  such that  $\deg_{S_L}(a) \leq 1$ , for every  $a \in A(M')$ , and  $\deg_{S_L}(b) \leq d$ , for every  $b \in \overline{B(M)}$
- 5:  $S_R \leftarrow \text{GREEDY}_d(\pi_{G'_R})$  such that  $\deg_{S_R}(b) \leq 1$ , for every  $b \in B(M')$ , and  $\deg_{S_R}(a) \leq d$ , for every  $a \in \overline{A(M)}$
- 6:  $\mathcal{P} \leftarrow \{ab', ab, a'b : ab' \in S_L, ab \in M', a'b \in S_R\}$
- 7: **return** A largest subset  $\mathcal{Q} \subseteq \mathcal{P}$  of vertex-disjoint paths.

► **Theorem 1.** For any  $\epsilon > 0$ , every (possibly randomised) two-pass streaming algorithm for MBM with approximation ratio  $\frac{2}{3} + \epsilon$  that solely computes a GREEDY matching in the first pass requires  $n^{1+\Omega(\frac{1}{\log \log n})}$  space, where  $n$  is the number of vertices in the graph.

## 4 Algorithm

In this section, we combine the subsampling approach as used by Konrad [19] and the semi-matching approach as used by Esfandiari et al. [8] and Kale and Tirodkar [15] in order to find many disjoint 3-augmenting paths, see Algorithm 3.

The input to Algorithm 3 is a stream of edges  $\pi$  of a bipartite graph  $G = (A, B, E)$ , a maximal matching  $M$  in  $G$  (e.g., computed in a first pass by GREEDY), a sampling probability  $p$ , and an integral degree bound  $d$ . First, each edge of  $M$  is included in  $M'$  with probability  $p$ . Then, while processing the stream, degree- $d$ -bounded semi-matchings  $S_L$  and  $S_R$  are computed using the algorithm  $\text{GREEDY}_d$  (see Algorithm 2 in Section 1). The algorithm then returns a largest subset of vertex-disjoint 3-augmenting paths  $\mathcal{Q}$ . We can thus obtain a matching of size  $|M| + |\mathcal{Q}|$ .

### 4.1 Analysis of Algorithm 3

The main task in analysing Algorithm 3 is to bound the sizes of  $S_L$  and  $S_R$  from below. A bound that holds in expectation for the case  $d = 1$  was previously proved by Konrad et al. [20], and a high probability result (for  $d = 1$ ) was later obtained by Konrad [19]. We also first give a bound that holds in expectation (Lemma 9), which is achieved by extending the original proof by Konrad et al. [20]. Our extension, however, is non-trivial as it requires a very different progress measure. Then, following Konrad [19], we obtain a high probability version in Lemma 10.

We also remark that Lemmas 9 and 10 are stated in a more general context, however, it is not hard to see that they capture the situation of the computations of  $S_L$  and  $S_R$  in subgraphs  $G'_L$  and  $G'_R$ , respectively.

► **Lemma 9.** Let  $G = (A, B, E)$  be a bipartite graph,  $\pi$  an arbitrarily ordered stream of its edges,  $p \in (0, 1]$ , and  $d \in \mathbb{N}^+$ . Let  $A' \subseteq A$  be a random subset such that  $\forall a \in A, \Pr[a \in A'] = p$ , and let  $d$  be the degree bound of the  $B$  vertices. Let  $H = G[A' \cup B]$  and denote by  $\pi_H$  the substream of  $\pi$  consisting of the edges in  $H$ . Then,

$$\mathbb{E}_{A'}[|\text{GREEDY}_d(\pi_H)|] \geq \frac{d}{d+p} \cdot p \cdot \mu(G).$$

**Proof.** Let  $M^*$  be a fixed maximum matching in  $G$  and let  $M_H^* := \{ab \in M^* : a \in A'\}$  be the subset of edges incident to  $A'$ .

**Game Setup.** Consider the following game: On selection of an edge by  $\text{GREEDY}_d(\pi_H)$ , the edge *attacks* the (at most two) incident edges of  $M_H^*$  and deals damage to them. Initially, the damage of every edge in  $M_H^*$  is 0, and the maximum damage of each such edge is 1. A damage below 1 means that the edge could still be selected by the algorithm. A damage equal to 1 implies that the edge can no longer be selected.

Denote by  $S_i$  the first  $i$  edges selected by  $\text{GREEDY}_d(\pi_H)$  and let  $ab$  be the  $(i + 1)^{\text{th}}$  edge selected. The way damage is dealt is as follows:

- If there is an edge  $a'b \in M_H^*$  such that  $a' \notin A(S_{i+1})$  then attack edge  $a'b$  by adding  $\frac{1}{d}$  damage to it;
- If there is an edge  $ab' \in M_H^*$  then attack edge  $ab'$  by adding  $1 - \frac{\deg_{S_i}(b')}{d}$  damage to it, maxing out the damage to 1.

Observe that the maximum damage an edge selected by  $\text{GREEDY}_d(\pi_H)$  can inflict is at most  $1 + \frac{1}{d}$  (applying both cases to the two incident optimal edges). Furthermore, observe that the maximum damage every edge in  $M_H^*$  receives is 1, and, indeed, at the end of the algorithm, every edge in  $M_H^*$  has damage 1.

**Applying Wald's Equation.** Denote by  $s$  the cardinality of the semi-matching computed by  $\text{GREEDY}_d(\pi_H)$  and let  $X_1, X_2, \dots, X_s$  be the sequence of edges selected. Define the random variable  $Y_i$  to be the damage dealt by edge  $X_i$ . Let  $T$  be the smallest  $i$  such that  $\sum_{j=1}^i Y_j = |M_H^*|$  holds. Observe that  $T$  is a random stopping time. To apply the version of Wald's Equation presented in Lemma 3, we need to show that  $\mathbb{E}[T]$  is finite and find a value  $\tau$  such that, for all  $i \leq T$ ,  $\mathbb{E}[Y_i] \leq \tau$  holds:

The expected stopping time  $\mathbb{E}[T]$  is finite since  $T \leq s$  always holds by the end of the algorithm, i.e., the total damage dealt is  $|M_H^*|$ . Finding  $\tau$  is less obvious. By definition, the damage  $Y_i$  dealt by any edge  $X_i$  is either 0,  $\frac{1}{d}, \dots, 1$  or  $1 + \frac{1}{d}$ . Hence, we obtain the following:

$$\mathbb{E}[Y_i] \leq \Pr[Y_i \leq 1] \cdot 1 + \underbrace{\Pr\left[Y_i = 1 + \frac{1}{d}\right]}_q \cdot \left(1 + \frac{1}{d}\right) = (1 - q) \cdot 1 + q \cdot \left(1 + \frac{1}{d}\right) = 1 + \frac{q}{d}.$$

It remains to bound  $\Pr[Y_i = 1 + \frac{1}{d}] (= q)$ . Let  $X_i = ab$ . Then, by definition of the game, the event  $Y_i = 1 + \frac{1}{d}$  only happens if there exists an edge  $a'b \in M_H^*$  such that  $a' \notin A(S_i)$ . In this case,  $ab$  inflicts a damage of 1 on edge  $a'b$ . However, observe that since  $a' \notin A(S_i)$ , the random choice as to whether  $a' \in A'$  and thus whether  $a'b \in M_H^*$  had not needed to occur yet (principle of deferred decision). Hence, we obtain:

$$\Pr[Y_i = 1 + \frac{1}{d}] \leq \Pr[a' \in A'] = p.$$

Having shown that  $\mathbb{E}[T]$  is finite and  $\mathbb{E}[Y_i] \leq 1 + \frac{p}{d}$  for all  $i \leq T$ , we can apply Wald's Equation (Lemma 3) and we obtain  $\mathbb{E}[\sum_{j=1}^T Y_j] \leq (1 + \frac{p}{d})\mathbb{E}[T]$ . Finally, since  $\mathbb{E}[\sum_{j=1}^T Y_j] = \mathbb{E}[|M_H^*|] = p \cdot \mu(G)$  and  $T \leq s = |\text{GREEDY}_d(\pi_H)|$ , it follows that

$$\mathbb{E}\left[\sum_{j=1}^T Y_j\right] = p \cdot \mu(G) \leq \left(1 + \frac{p}{d}\right) \cdot \mathbb{E}[T] \leq \left(1 + \frac{p}{d}\right) \cdot \mathbb{E}[|\text{GREEDY}_d(\pi_H)|],$$

which implies the result. ◀

## 19:12 On Two-Pass Streaming Algorithms for Maximum Bipartite Matching

Next, we follow the approach by Konrad [19] to strengthen Lemma 9 and obtain the following high probability result (see Appendix A for the proof):

► **Lemma 10.** *Let  $G = (A, B, E)$  be a bipartite graph,  $\pi$  be any arbitrary stream of its edges,  $p \in (0, 1]$  and  $d \in \mathbb{N}^+$ . Let  $A' \subseteq A$  be a random subset such that  $\forall a \in A, \Pr[a \in A'] = p$ , let  $d$  be the degree bound of the  $B$  vertices and let  $H = G[A' \cup B]$ . Then, the following holds with probability at least  $1 - 2\mu(G)^{-18}$ :*

$$|\text{GREEDY}_d(\pi_H)| \geq \frac{d}{d+p} \cdot p \cdot \mu(G) - o(\mu(G)).$$

Equipped with Lemma 10, we are now ready to bound the number of augmenting paths found by Algorithm 3.

► **Lemma 11.** *Suppose that  $|M| = (\frac{1}{2} + \epsilon)\mu(G)$ . Then, with probability  $1 - \mu(G)^{-16}$ , the number of vertex-disjoint 3-augmenting paths  $|\mathcal{Q}|$  found by Algorithm 3 is at least:*

$$|\mathcal{Q}| \geq \left(\frac{1-2\epsilon}{d+p} - \frac{1+2\epsilon}{2d}\right) \cdot p \cdot \mu(G) - o(\mu(G)).$$

**Proof.** Let  $M^*$  be a fixed maximum matching in  $G$ . In this proof, we will refer to the quantities used by Algorithm 3. First, using a Chernoff bound for independent Poisson trials, we see that  $|M'| = p \cdot |M| \pm O(\sqrt{|M| \ln |M|})$  with probability at least  $1 - |M|^{-C}$  for an arbitrarily large constant  $C$ .

Consider the subgraphs  $G_L = G[A(M) \cup \overline{B(M)}]$  and  $G_R = G[\overline{A(M)} \cup B(M)]$ .  $M \oplus M^*$  contains  $(\frac{1}{2} - \epsilon)\mu(G)$  vertex-disjoint augmenting paths where each path starts and ends with an edge in  $G_L \cup G_R$ . This implies that

$$\mu(G_L) + \mu(G_R) \geq 2\left(\frac{1}{2} - \epsilon\right)\mu(G) = (1 - 2\epsilon)\mu(G). \quad (2)$$

Following Konrad [19], we will argue next that

$$|\mathcal{P}| \geq |S_L| + |S_R| - |M'|. \quad (3)$$

Observe that there are  $|M'| - |S_L|$  vertices of  $|M'|$  that are not incident to an edge in  $S_L$ , and similarly,  $|M'| - |S_R|$  vertices of  $|M'|$  that are not incident to an edge in  $S_R$ . Hence, there are at least  $|M'| - (|M'| - |S_L|) - (|M'| - |S_R|) = |S_L| + |S_R| - |M'|$  edges of  $|M'|$  that are incident to both an edge from  $S_L$  and  $S_R$ . We thus obtain that there are at least  $|\mathcal{P}| \geq |S_L| + |S_R| - |M'|$  3-augmenting paths.

Next, Esfandiari et al. (Lemma 6 in [8]) consider a similar structure to  $\mathcal{P}$  and argue that there is at least a  $d$ -fraction of augmenting paths in  $\mathcal{P}$  that are vertex-disjoint, and, hence,

$$|\mathcal{Q}| \geq \frac{1}{d}|\mathcal{P}|. \quad (4)$$

Using Lemma 10 and Inequalities 2, 3, and 4, we obtain:

$$\begin{aligned} |\mathcal{Q}| &\geq \frac{1}{d}(|S_L| + |S_R| - |M'|) \\ &\geq \frac{1}{d}\left(\frac{d}{d+p} \cdot p \cdot (1 - 2\epsilon)\mu(G) - o(\mu(G)) - p \cdot \left(\frac{1}{2} + \epsilon\right)\mu(G) - O(\sqrt{\mu(G) \ln \mu(G)})\right) \\ &= \left(\frac{1-2\epsilon}{d+p} - \frac{1+2\epsilon}{2d}\right) \cdot p \cdot \mu(G) - o(\mu(G)). \end{aligned}$$

Using the union bound, the error of the algorithm is bounded by  $|M|^{-C} + 2\mu(G)^{-18} \leq \mu(G)^{-16}$ . ◀

1. Let  $A_{\text{in}} = \{a_{\text{in}}^1, a_{\text{in}}^2, \dots, a_{\text{in}}^N\}$ ,  $A_{\text{out}} = \{a_{\text{out}}^1, \dots, a_{\text{out}}^N\}$ ,  $B_{\text{in}} = \{b_{\text{in}}^1, \dots, b_{\text{in}}^N\}$ , and  $B_{\text{out}} = \{b_{\text{out}}^1, \dots, b_{\text{out}}^N\}$  be sets of vertices, for some integer  $N$ .
2. Let  $M = \{a_{\text{in}}^i b_{\text{in}}^i : 1 \leq i \leq N\}$  be a perfect matching between  $A_{\text{in}}$  and  $B_{\text{in}}$ . Let  $G_L = (A_{\text{in}}, B_{\text{out}}, E_L)$  be a semi-complete graph such that  $a_{\text{in}}^i b_{\text{out}}^j \in E_L \Leftrightarrow i \geq j$ , and let  $G_R = (A_{\text{out}}, B_{\text{in}}, E_R)$  be a semi-complete graph such that  $a_{\text{out}}^i b_{\text{in}}^j \in E_R \Leftrightarrow i \geq j$ .
3. Our bipartite hard instance graph is defined as  $G = (A_{\text{in}} \cup A_{\text{out}}, B_{\text{in}} \cup B_{\text{out}}, M \cup E_L \cup E_R)$  and has  $n = 4N$  vertices.
4. Finally, let  $\pi$  be a stream of its edges where the edges of  $M$  arrive first followed by the edges  $E_L$  and  $E_R$ . The edges in  $E_L$  are ordered so that  $a_{\text{in}}^i b_{\text{out}}^j$  arrives before  $a_{\text{in}}^{i'} b_{\text{out}}^{j'}$  only if  $i > i'$ , or  $i = i'$  and  $j < j'$ . Similarly, the edges in  $E_R$  are ordered so that  $a_{\text{out}}^i b_{\text{in}}^j$  arrives before  $a_{\text{out}}^{i'} b_{\text{in}}^{j'}$  only if  $i > i'$ , or  $i = i'$  and  $j < j'$ .

■ **Figure 6** Hard input instance  $G$  for Algorithm 3.

We are now ready to state our main algorithmic result:

► **Theorem 2.** *For every  $p \in (0, 1]$  and every integral  $d \geq 1$ , there is a two-pass semi-streaming algorithm for MBM with approximation factor*

$$\begin{cases} \frac{1}{2} + \left(\frac{1}{d+p} - \frac{1}{2d}\right) \cdot p - o(1), & \text{if } p \leq d(\sqrt{2} - 1) \\ \frac{1}{2} + \frac{d-p}{6d+2p} - o(1), & \text{otherwise,} \end{cases}$$

that succeeds with high probability (in  $\mu(G)$ , where  $G$  is the input graph). The settings ( $d = 1, p = \sqrt{2} - 1$ ) and ( $d = 2, p = 2(\sqrt{2} - 1)$ ) maximize the approximation factor to  $2 - \sqrt{2} - o(1)$ .

**Proof.** Let  $M$  be a maximal matching such that  $|M| = (\frac{1}{2} + \epsilon)\mu(G)$ , for some  $0 \leq \epsilon \leq \frac{1}{2}$  and some bipartite graph  $G = (A, B, E)$  with a stream  $\pi$  of its edges. Let  $\mathcal{Q}$  be the disjoint augmenting paths found by Algorithm 3 on input  $\pi, M, p$  and  $d$ . Then, augmenting  $M$  with  $\mathcal{Q}$  yields a matching of size  $|M| + |\mathcal{Q}|$ . By Lemma 11, the following inequality holds with high probability:

$$|M| + |\mathcal{Q}| \geq \left(\frac{1}{2} + \epsilon\right)\mu(G) + \left(\frac{1-2\epsilon}{d+p} - \frac{1+2\epsilon}{2d}\right) \cdot p \cdot \mu(G) - o(\mu(G)). \quad (5)$$

We distinguish two cases:

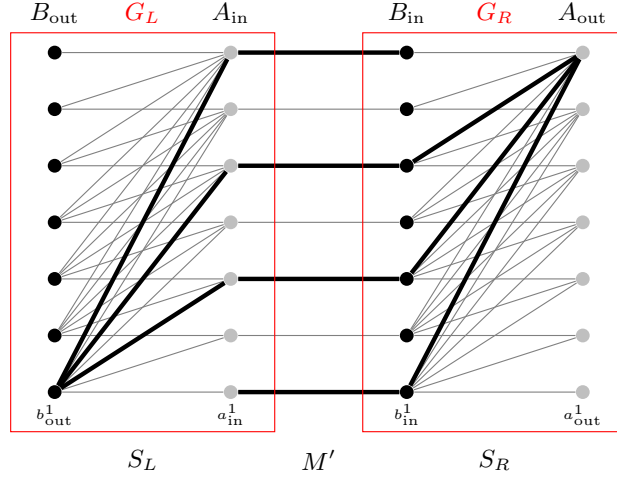
1. If  $p \leq d(\sqrt{2} - 1)$  then  $\epsilon = 0$  minimizes the RHS of Inequality 5, and we obtain the claimed bound by plugging the value  $\epsilon = 0$  into the inequality.
2. If  $p \geq d(\sqrt{2} - 1)$  (only possible if  $d \in \{1, 2\}$ ) then  $\epsilon = \frac{d-p}{6d+2p}$  minimizes the RHS of Inequality 5, and we obtain the claimed bound by plugging the value  $\epsilon = \frac{d-p}{6d+2p}$  into the inequality.

It can be seen that, for a fixed  $d$ , the maximum is obtained if  $p = \min\{d\sqrt{2} - d, 1\}$ , and the values  $d \in \{1, 2\}$  yield the claimed bound of  $2 - \sqrt{2} - o(1)$  (see Figure 1 in Section 1). ◀

## 4.2 Optimality of the Analysis

We will show now that our analysis of Algorithm 3 is best possible. To this end, we define a worst-case input graph  $G$  in Figure 6, and prove in Theorem 12 that Algorithm 3 does not perform better on  $G$  than predicted by our analysis. See Figure 7 for an illustration.

Observe that  $M$  is a maximal matching in  $G$ , and if we run GREEDY in the first pass on  $\pi$  then  $M$  would be returned. Let  $M_L^* = \{a_{\text{in}}^i b_{\text{out}}^i : 1 \leq i \leq N\}$  and  $M_R^* = \{a_{\text{out}}^i b_{\text{in}}^i : 1 \leq i \leq N\}$ . Then,  $M_L^*$  is a perfect matching in  $G_L$ ,  $M_R^*$  is a perfect matching in  $G_R$ , and  $M_L^* \cup M_R^*$  is a perfect matching in  $G$ .



■ **Figure 7** Algorithm 3 on a hard input instance with  $N = 7$ ,  $d = 3$  and  $p = 0.5$ .

► **Theorem 12.** *Algorithm 3 with parameters  $d \geq 1$  and  $0 < p \leq 1$  on input  $G$  received via stream  $\pi$  and maximal matching  $M$  finds at most*

$$\left( \left( \frac{1}{d+p} - \frac{1}{2d} \right) \cdot p + o(1) \right) \mu(G)$$

*augmenting paths with high probability. This renders our analysis of Algorithm 3 best possible when  $p \leq d\sqrt{2} - d$ .*

**Proof.** In this proof, we will refer to the quantities used by Algorithm 3, that is,  $M'$  (the edges of  $M$  subsampled with probability  $p$ ),  $S_L$  and  $S_R$ .

We will use the following claim in our proof:

▷ **Claim 13.** With high probability, for every pair  $i, j \in [N]$  with  $i \leq j$ , we have

$$|\{a_{\text{in}}^k b_{\text{in}}^k \in M' \mid i \leq k \leq j\}| = p \cdot (j - i) \pm o(N).$$

*Proof.* This claim is easy to prove. Indeed, for any fixed  $i, j \in [N]$  with  $i \leq j$ , the statement above follows directly from the Chernoff bound. Using the union bound over all pairs  $i, j \in [N]$ , the claim follows. ◁

From now on, we condition on the event that the statement in Claim 13 holds.

Let  $A'_{\text{in}} = A(M')$  and let  $B'_{\text{in}} = B(M')$ . We will first argue that, for two different vertices  $a_{\text{in}}^i, a_{\text{in}}^j \in A'_{\text{in}}$  with  $i < j$ , if  $a_{\text{in}}^i \in A(S_L)$  then  $a_{\text{in}}^j \in A(S_L)$  also holds. Indeed, suppose that this was not the case. Let  $b_{\text{out}}^k$  be the partner of  $a_{\text{in}}^i$  in  $S_L$ . Observe that the edges  $a_{\text{in}}^i b_{\text{out}}^k, a_{\text{in}}^j b_{\text{out}}^k \in E_L$ , and, in particular, the edge  $a_{\text{in}}^j b_{\text{out}}^k$  arrives before the edge  $a_{\text{in}}^i b_{\text{out}}^k$  in  $\pi$ . Hence, edge  $a_{\text{in}}^j b_{\text{out}}^k$  would have been selected, a contradiction. A similar argument holds for vertices  $b_{\text{out}}^i, b_{\text{out}}^j \in B_{\text{out}}$  with  $i > j$ ; if  $\deg_{S_L}(b_{\text{out}}^i) \geq 1$  then  $\deg_{S_L}(b_{\text{out}}^j) = d$ .

Let  $i_{\text{min}}$  be the smallest index such that  $a_{\text{in}}^{i_{\text{min}}} \in A(S_L)$ . We will now argue that  $i_{\text{min}} \geq \frac{pN}{p+d} - o(N)$ . Observe that the vertices  $A'_{\text{in}}$  are matched in order from the largest to smallest index, and each matched vertex in  $A'_{\text{in}}$  is matched only once. The vertices in  $B_{\text{out}}$  are matched from the smallest to largest index, and each matched vertex is matched  $d$  times (except possibly the last such matched vertex). Consider the last edge  $a_{\text{in}}^{i_{\text{min}}} b_{\text{out}}^q$  inserted into  $S_L$ . Then,  $q \leq i_{\text{min}}$ , and, thus,  $|B(S_L)| \leq i_{\text{min}}$ . By Claim 13 (applied with  $j = N$ ), we have



$|A(S_L)| \geq p \cdot (N - i_{\min}) - o(N)$  with high probability. Since  $|A(S_L)|$  is matched to  $B(S_L)$  in  $S_L$ , and each  $B$ -vertex is matched at most  $d$  times, we obtain  $|A(S_L)| \leq d \cdot |B(S_L)|$ , and, hence:

$$p \cdot (N - i_{\min}) - o(N) \leq |A(S_L)| \leq d \cdot |B(S_L)| \leq d \cdot i_{\min} ,$$

which implies  $i_{\min} \geq \frac{pN}{p+d} - o(N)$ .

Let  $i_{\max}$  be the largest index such that  $b_{\text{in}}^{i_{\max}} \in B(S_R)$ . Using a similar argument as above, we see that  $i_{\max} \leq \frac{dN}{p+d} + o(N)$ .

Let  $M'' = \{a_{\text{in}}^i b_{\text{in}}^i \in M' : i_{\min} \leq i \leq i_{\max}\}$  be the subset of augmentable edges, i.e., edges for which there exists a left wing in  $S_L$  and a right wing in  $S_R$ . Then, by Claim 13, we have

$$|M''| \leq p \cdot (i_{\max} - i_{\min}) + o(N) \leq \frac{p(d-p)N}{p+d} + o(N) .$$

All but constantly many vertices in  $A(M'')$  share the same neighbour in  $S_L$  with  $d-1$  other vertices of  $A(M'')$ . Hence, at most a  $d$ -fraction (plus up to the constantly many exceptions, which disappear in the  $o(N)$  term) of  $M''$  can be augmented simultaneously. Using  $N = \frac{1}{2}\mu(G)$ , we obtain the following bound on the number of edges that can be augmented simultaneously:

$$\frac{1}{d}|M''| \leq \frac{1}{d} \left( \frac{p(d-p)N}{p+d} + o(N) \right) = \left( \left( \frac{1}{d+p} - \frac{1}{2d} \right) \cdot p + o(1) \right) \mu(G) . \quad \blacktriangleleft$$

## 5 Conclusion

In this paper, we studied the class of two-pass semi-streaming algorithms for MBM that solely compute a GREEDY matching in the first pass. We showed that algorithms of this class cannot have an approximation ratio of  $\frac{2}{3} + \epsilon$ , for any  $\epsilon > 0$ . We also combined the two dominant techniques that have previously been used for designing such algorithms and discovered another algorithm that matches the state-of-the-art approximation factor of  $2 - \sqrt{2} \approx 0.58578$ .

We conclude with two open problems. First, we are particularly interested in whether there exists a one-pass semi-streaming algorithm that is able to augment a maximal matching so as to yield an approximation ratio above  $2 - \sqrt{2}$ . Second, is there a two-pass semi-streaming algorithm for MBM that improves on the approximation factor of  $2 - \sqrt{2}$  and operates differently in the first pass to the class of algorithms considered in this paper?

---

## References

- 1 Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II*, volume 6756 of *Lecture Notes in Computer Science*, pages 526–538. Springer, 2011. doi:10.1007/978-3-642-22012-8\_42.
- 2 Sepehr Assadi and Soheil Behnezhad. Beating two-thirds for random-order streaming matching. 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 19:1–19:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.ICALP.2021.19.

## 19:16 On Two-Pass Streaming Algorithms for Maximum Bipartite Matching

- 3 Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1345–1364. SIAM, 2016. doi:10.1137/1.9781611974331.ch93.
- 4 Sepehr Assadi, S. Cliff Liu, and Robert E. Tarjan. An auction algorithm for bipartite matching in streaming and massively parallel computation models. In Hung Viet Le and Valerie King, editors, *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, pages 165–171. SIAM, 2021. doi:10.1137/1.9781611976496.18.
- 5 Aaron Bernstein. Improved bounds for matching in random-order streams. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPICs*, pages 12:1–12:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ICALP.2020.12.
- 6 Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1326–1344. SIAM, 2016. doi:10.1137/1.9781611974331.ch92.
- 7 Jacques Dark and Christian Konrad. Optimal lower bounds for matching and vertex cover in dynamic graph streams. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPICs*, pages 30:1–30:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.CCC.2020.30.
- 8 Hossein Esfandiari, MohammadTaghi Hajiaghayi, and Morteza Monemizadeh. Finding large matchings in semi-streaming. In Carlotta Domeniconi, Francesco Gullo, Francesco Bonchi, Josep Domingo-Ferrer, Ricardo Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu, editors, *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain*, pages 608–614. IEEE Computer Society, 2016. doi:10.1109/ICDMW.2016.0092.
- 9 Jittat Fakcharoenphol, Bundit Laekhanukit, and Danupon Nanongkai. Faster algorithms for semi-matching problems. *ACM Trans. Algorithms*, 10(3), 2014. doi:10.1145/2601071.
- 10 Alireza Farhadi, Mohammad Taghi Hajiaghayi, Tung Mai, Anup Rao, and Ryan A. Rossi. Approximate maximum matching in random 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 1773–1785. SIAM, 2020. doi:10.1137/1.9781611975994.108.
- 11 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. doi:10.1007/978-3-540-27836-8\_46.
- 12 Buddhima Gamlath, Sagar Kale, Slobodan Mitrovic, and Ola Svensson. Weighted matchings via unweighted augmentations. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 491–500. ACM, 2019. doi:10.1145/3293611.3331603.
- 13 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In Yuval Rabani, editor, *Proceedings of the 23rd ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages pp. 468–485. SIAM, 2012. doi:10.1137/1.9781611973099.41.

- 14 Zhiyi Huang, Binghui Peng, Zhihao Gavin Tang, Runzhou Tao, Xiaowei Wu, and Yuhao Zhang. Tight competitive ratios of classic matching algorithms in the fully online model. 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 2875–2886. SIAM, 2019. doi:10.1137/1.9781611975482.178.
- 15 Sagar Kale and Sumedh Tirodkar. Maximum matching in two, three, and a few more passes over graph streams. In Klaus Jansen, José D. P. Rolim, David Williamson, and Santosh S. Vempala, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*, volume 81 of *LIPICs*, pages 15:1–15:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICs.APPROX-RANDOM.2017.15.
- 16 Michael Kapralov. Better bounds for matchings in the streaming model. 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 1679–1697. SIAM, 2013. doi:10.1137/1.9781611973105.121.
- 17 Michael Kapralov. Space lower bounds for approximating maximum matching in the edge arrival model. In Dániel Marx, editor, *Proceedings of the 32nd ACM-SIAM Symposium on Discrete Algorithms, (SODA)*, pages pp. 1874–1893. SIAM, 2021. doi:10.1137/1.9781611976465.112.
- 18 Christian Konrad. Maximum matching in turnstile streams. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, pages 840–852. Springer, 2015. doi:10.1007/978-3-662-48350-3\_70.
- 19 Christian Konrad. A simple augmentation method for matchings with applications to streaming algorithms. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, volume 117 of *LIPICs*, pages 74:1–74:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.MFCS.2018.74.
- 20 Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In Anupam Gupta, Klaus Jansen, José D. P. Rolim, and Rocco A. Servedio, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, volume 7408 of *Lecture Notes in Computer Science*, pages 231–242. Springer, 2012. doi:10.1007/978-3-642-32512-0\_20.
- 21 Christian Konrad and Adi Rosén. Approximating semi-matchings in streaming and in two-party communication. *ACM Trans. Algorithms*, 12(3), 2016. doi:10.1145/2898960.
- 22 Andrew McGregor. Finding graph matchings in data streams. In Chandra Chekuri, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, editors, *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th International Workshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, volume 3624 of *Lecture Notes in Computer Science*, pages 170–181. Springer, 2005. doi:10.1007/11538462\_15.
- 23 Andrew McGregor. Graph stream algorithms: a survey. *SIGMOD Rec.*, 43(1):9–20, 2014. doi:10.1145/2627692.2627694.
- 24 Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005. doi:10.1017/CB09780511813603.
- 25 Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th Symposium on Foundations of Computer Science (FOCS)*, pages pp. 222–227. IEEE Computer Society, 1977. doi:10.1109/SFCS.1977.24.

## A

 Strengthening Lemma 9

Following [19], we use tail inequalities for martingales to strengthen Lemma 9 and give a high probability result. The proof of Lemma 10 uses the *Azuma-Hoeffding's Inequality* [24, Theorem 12.4]:

► **Lemma 14** (Azuma-Hoeffding's Inequality). *Let  $Z_0, Z_1, \dots, Z_n$  be a martingale such that  $\forall k \geq 0, |Z_{k+1} - Z_k| \leq c_k$ . Then,  $\forall t \geq 0$  and any  $\lambda > 0$ ,*

$$\Pr[|Z_t - Z_0| \geq \lambda] \leq 2 \exp\left(\frac{-\lambda^2}{2 \sum_{k=0}^{t-1} c_k^2}\right).$$

► **Lemma 10.** *Let  $G = (A, B, E)$  be a bipartite graph,  $\pi$  be any arbitrary stream of its edges,  $p \in (0, 1]$  and  $d \in \mathbb{N}^+$ . Let  $A' \subseteq A$  be a random subset such that  $\forall a \in A, \Pr[a \in A'] = p$ , let  $d$  be the degree bound of the  $B$  vertices and let  $H = G[A' \cup B]$ . Then, the following holds with probability at least  $1 - 2\mu(G)^{-18}$ :*

$$|\text{GREEDY}_d(\pi_H)| \geq \frac{d}{d+p} \cdot p \cdot \mu(G) - o(\mu(G)).$$

**Proof.** Let  $X_1, X_2, \dots, X_s$  be the sequence of random variables representing the edges selected by  $\text{GREEDY}_d(\pi_H)$  with the source of randomness from the choice of  $A'$ . Define  $Y := |\text{GREEDY}_d(\pi_H)|$ . Then, we define the random variables  $Z_i := \mathbb{E}[Y | X_1, \dots, X_i]$  for all  $i = 0, \dots, s$  to be the corresponding Doob Martingale, and let  $Z_i = Z_{i-1}$ , for every  $i > s$ . Notice that  $Z_s = Y$  and  $Z_0 = \mathbb{E}[Y] \geq \frac{d}{d+p} \cdot p \cdot \mu(G)$  by Lemma 9. Now, we will show that any deviation of  $Y$  from its expectation,  $|Z_s - Z_0|$ , is small with high probability.

To that end, we first need to bound  $|Z_{i+1} - Z_i|$  for all  $i \geq 0$ . Notice that  $|Z_{i+1} - Z_i| = 0$  for all  $i \geq s$ . Next, we will argue that  $|Z_{i+1} - Z_i| \leq 1$  for all  $i < s$ . Indeed, for any fixed first  $i$  edges added to the semi-matching, any two different choices for  $X_{i+1}$  yield two potentially different semi-matchings  $S_1, S_2$ , respectively, such that  $S_1 \oplus S_2$  consists of at most one alternating path. Hence, the two semi-matchings differ by at most one edge, which proves the claim.

Then, we have that  $s = Y \leq d \cdot \mu(H) \leq d \cdot \mu(G)$  and it follows that  $|Z_{i+1} - Z_i| \leq 1$  for all  $i \leq d \cdot \mu(G)$  and  $|Z_{i+1} - Z_i| = 0$  for all  $i > d \cdot \mu(G)$ . Finally, by applying Azuma-Hoeffding's Inequality (see Lemma 14), we finalise the proof:

$$\Pr\left[|Z_s - Z_0| \geq 6\sqrt{d\mu(G) \ln \mu(G)}\right] \leq 2\mu(G)^{-18},$$

where  $|Z_s - Z_0| = |Y - \mathbb{E}[Y]|$ . ◀

# Approximation Algorithms for Demand Strip Packing

Waldo Gálvez  

Technische Universität München, Germany

Fabrizio Grandoni 

IDSIA, USI-SUPSI, Lugano, Switzerland

Afrouz Jabal Ameli 

IDSIA, USI-SUPSI, Lugano, Switzerland

Kamyar Khodamoradi  

Universität Würzburg, Germany

---

## Abstract

In the Demand Strip Packing problem (DSP), we are given a time interval and a collection of tasks, each characterized by a processing time and a demand for a given resource (such as electricity, computational power, etc.). A feasible solution consists of a schedule of the tasks within the mentioned time interval. Our goal is to minimize the peak resource consumption, i.e. the maximum total demand of tasks executed at any point in time.

It is known that DSP is NP-hard to approximate below a factor  $3/2$ , and standard techniques for related problems imply a (polynomial-time) 2-approximation. Our main result is a  $(5/3 + \varepsilon)$ -approximation algorithm for any constant  $\varepsilon > 0$ . We also achieve best-possible approximation factors for some relevant special cases.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Packing and covering problems; Theory of computation  $\rightarrow$  Scheduling algorithms

**Keywords and phrases** Strip Packing, Two-Dimensional Packing, Approximation Algorithms

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.20

**Category** APPROX

**Related Version** *Full Version:* <https://arxiv.org/pdf/2105.08577.pdf> [17]

**Funding** *Waldo Gálvez:* Supported by the European Research Council, Grant Agreement No. 691672, project APEG.

*Fabrizio Grandoni:* Partially supported by the SNF Excellence Grant 200020B\_182865.

*Afrouz Jabal Ameli:* Partially supported by the SNF Excellence Grant 200020B\_182865.

*Kamyar Khodamoradi:* Partially supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - Project number 399223600. This project was carried out in part when the author was a postdoctoral researcher at IDSIA, USI-SUPSI, Switzerland.

## 1 Introduction

Consider the following scenario: we are given a time interval and a collection of tasks, where each task is characterized by a processing time (no longer than the time interval) and a demand for a given resource. A feasible solution consists of a schedule of all the tasks within the mentioned time interval, and our goal is to minimize the peak resource consumption, i.e. the maximum total demand of tasks scheduled at any point in time. It is easy to imagine concrete applications of this scenario; for example, the considered resource might be electricity, bandwidth along a communication channel, or computational power.



© Waldo Gálvez, Fabrizio Grandoni, Afrouz Jabal Ameli, and Kamyar Khodamoradi; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 20; pp. 20:1–20:24



Leibniz International Proceedings in Informatics

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

The above scenario can be naturally formalized via the following DEMAND STRIP PACKING problem (DSP). We interpret the time interval as a path graph  $G = (V, E)$  with  $W$  edges, where each edge is interpreted as a time slot where we can start to process a task. Let  $\mathcal{I} = \{1, \dots, n\}$  be the set of tasks, where task  $i$  has integer *processing time* (or *width*)  $w(i) \in [1, W]$  and integer *demand* (or *height*)  $h(i) \geq 0$ . A feasible solution (or *schedule* of the tasks) consists of a subpath  $P(i)$  of  $G$  for each  $i \in \mathcal{I}$  containing precisely  $w(i)$  edges. Our goal is to minimize the peak resource consumption (or simply *peak*) which is defined as

$$\max_{e \in E} \sum_{i \in \mathcal{I}: e \in P(i)} h(i).$$

A problem closely related to DSP is the GEOMETRIC STRIP PACKING problem (GSP)<sup>1</sup>, which can be interpreted as a variant of DSP with an extra geometric packing constraint. Here we are given an axis-aligned half-strip of integer width  $W$  (and unbounded height) and a collection of open rectangles (or tasks), where each rectangle  $i$  has integer width  $w(i) \in [1, W]$  and integer height  $h(i) \geq 0$ . Our goal is to find an axis-aligned non-overlapping packing of all the rectangles within the strip that minimizes the peak height, i.e. the maximum height spanned by any rectangle. Notice that one can reinterpret DSP as a variant of GSP, where the processing time and demand of each task correspond to the width and height of a rectangle, resp. (this also motivated our notation). A critical difference w.r.t. GSP however is that DSP does not require to pack such rectangles geometrically<sup>2</sup>.

Obviously, a feasible solution to GSP induces a feasible solution to DSP of no larger peak. The converse is however not true (see Figure 1), and consequently it makes sense to design algorithms specifically for DSP. We remark that there are applications that are better formalized by GSP than by DSP. In particular, this happens when each task requires a contiguous and fixed portion of the considered resource. For example, we might need to allocate consecutive frequencies or memory locations to each task: changing this allocation over time might be problematic. Another natural application of GSP is cutting rectangular pieces from a roll of some raw material (e.g., paper, metal, or leather). However, for other applications, the geometric constraint in GSP does not seem to be necessary, and hence it makes sense to drop it (i.e., to rather consider DSP): this might lead to better solutions, possibly via simpler and/or more efficient algorithms. Consider for example the minimization of the peak energy consumption in smart-grids [31, 44, 39].

A straightforward reduction to the NP-complete PARTITION problem (similar to the one known for GSP, see also [43]) shows that DSP is NP-hard to approximate below a factor  $3/2$ . Constant approximation algorithms for DSP are given in [43, 45]. However, a better 2-approximation can be obtained by applying an algorithm by Steinberg [42] which was developed for GSP: the reason is that Steinberg uses area-based lower bounds that extend directly from GSP to DSP.

## 1.1 Our Results and Techniques

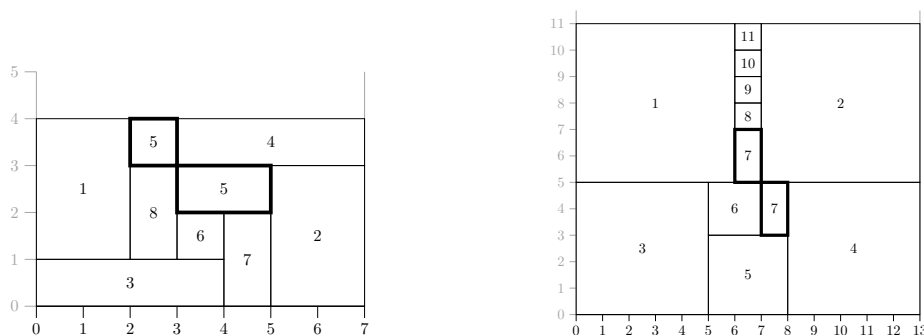
Our main result is as follows<sup>3</sup>.

► **Theorem 1.** *For any constant  $\varepsilon > 0$ , there is a polynomial-time deterministic  $(5/3 + \varepsilon)$ -approximation algorithm for DSP.*

<sup>1</sup> GSP is usually simply called STRIP PACKING in the literature. We added the word “geometric” to better highlight the differences between the two problems.

<sup>2</sup> Or, equivalently, we can split such rectangles into unit-width vertical slices, and then pack such slices geometrically so that slices of the same rectangle appear consecutively in a horizontal sense.

<sup>3</sup> The same result as in Theorem 1 was achieved independently in [15]; their approach is however substantially different from ours.



(a) DSP solution of peak 4 whose corresponding optimal GSP solution has peak 5.

(b) SQUARE-DSP solution of peak 11 whose corresponding optimal GSP solution has peak at least 12.

■ **Figure 1** Gap instances between DSP and GSP.

The above approximation ratio matches the best-known result for GSP from Harren et al. [23], achieved using dynamic programming based techniques to place almost all the rectangles except for a set of very small total area, followed by a careful and quite involved case distinction to pack these remaining rectangles. However, we remark that our algorithm is entirely different, and in particular it does not compute a geometric packing of tasks/rectangles. Furthermore, our analysis is substantially simpler. Notice that the result in [23] does *not* imply a  $(5/3 + \varepsilon)$ -approximation for DSP since some lower bounds used in their proofs do not hold necessarily for DSP.

We also achieve improved approximation algorithms for relevant special cases of DSP. We obtain a PTAS for the special case where the demand of each task is much lower than the optimal peak  $OPT$ . This captures applications where each job consumes a relatively small amount of the available resource (think about the electricity consumption of large-scale systems such as cities or countries).

► **Theorem 2.** *Given  $\varepsilon > 0$  small enough, there exists  $\delta > 0$  and a polynomial time algorithm such that, given an instance of DSP with optimal value  $OPT$  and consisting solely of tasks having height at most  $\delta \cdot OPT$ , it computes a  $(1 + O(\varepsilon))$ -approximate solution.*

Motivated by the special case of GSP and related problems where all rectangles are squares, we also study the special case of DSP where  $h(j) = w(j)$  for all tasks (the SQUARE-DSP problem). The  $3/2 - \varepsilon$  hardness of approximation extends to this case (see Appendix A), and we are still able to show that there is a gap between DSP and GSP (see Figure 1b and Section 5). However, in this case, we are able to provide an optimal  $3/2$ -approximation. We defer the proof of the following theorem to the full version of the paper [17].

► **Theorem 3.** *There is a deterministic polynomial-time  $3/2$ -approximation for SQUARE-DSP.*

At a high level, our approach is based on a classification of tasks into groups depending on their heights and widths. We carefully schedule some groups first, so that their *demand profile* has a convenient structure. Here, by demand profile we simply mean the total demand of the already scheduled tasks over each edge. The structure of the demand profile allows us to pack the remaining groups (intuitively, on top of such profile) in a convenient way. We critically exploit the fact that, differently from GSP, we only care about the total demand on each edge. This allows us to adapt techniques from Bin Packing or Makespan Minimization (see Lemmas 6 and 12).

## 1.2 Related Work

GSP generalizes famous problems such as Makespan Minimization on identical machines [12] (here all the rectangles have width 1 and  $W$  corresponds to the number of processors) or Bin Packing [13] (here all the rectangles have height 1 and the height of the solution corresponds to the number of bins). Consequently, it is known that for any  $\varepsilon > 0$ , there is no  $(3/2 - \varepsilon)$ -approximation for the problem unless  $P=NP$ . The first non-trivial approximation algorithm for GSP, with an approximation ratio of 3, was given by Baker, Coffman, and Rivest [5]. After a series of very technical and involved refinements [14, 41, 40, 42, 24], the current best approximation factor for the problem is  $(5/3 + \varepsilon)$  due to Harren et al. [23]. GSP has been also studied in the pseudopolynomial setting, i.e., when  $W = n^{O(1)}$  [30, 38, 1, 20, 25, 28, 27] and in the asymptotic setting, i.e. when the optimal value is assumed to be large [32, 29]. In both cases, approximation algorithms and almost matching lower bounds have been developed.

There is a very rich line of research on generalizations and variants of DSP such as online versions [34, 35], tasks with availability constraints or time windows [45, 44, 31], a mixture of preemptable and non-preemptable tasks [39] or generalized cost functions based on the demand at each edge [10, 35]. The variant of DSP with the extra feature of interrupting the tasks is known as STRIP PACKING WITH SLICING, for which there exists an FPTAS [3]; on the other hand, the case of DSP is still hard to approximate by a factor better than  $3/2$  as noted by Tang et al. [43].

Another problem closely related to DSP is PARALLEL JOB SCHEDULING. Here we are given a set of jobs and  $m$  machines, where each job is characterized by a processing time and a number of machines where the job must be processed simultaneously (these machines do not need to be contiguous), and the goal is to minimize the makespan. The same hardness of approximation applies in this case, but interestingly an almost tight  $(3/2 + \varepsilon)$ -approximation algorithm has been developed [26] and also a pseudopolynomial  $(1 + \varepsilon)$ -approximation is known [30]. See [16] for a comprehensive survey on the problem and its many variants.

It is also worth mentioning another case where the distinction between geometric and demand-based packing plays a substantial role: the UNSPLITTABLE FLOW ON A PATH problem (UFP) [4, 22, 7, 21] and the STORAGE ALLOCATION problem (SAP) [36, 37]. In both problems, we are given a path graph with edge capacities, and tasks specified by a subpath, a demand (or height), and a profit. In both problems, the goal is to select a maximum profit subset of tasks that can be *packed* while respecting edge capacities. For UFP, analogously to DSP, we require that the total demand of the selected tasks on each edge  $e$  is at most the capacity of  $e$ . For SAP, analogously to GSP, we interpret each task as a rectangle (with the width given by its number of edges) and, intuitively, we need to pack such rectangles non-overlappingly below the capacity profile. Notice that, differently from DSP and GSP, here the path associated with each task is fixed in the input. Furthermore, not all the tasks need to be packed.

Finally, in the DYNAMIC STORAGE ALLOCATION problem (DSA) the setting is analogous to SAP but, similarly to GSP, we are asked for an embedding of all the rectangles minimizing the peak height, i.e. the maximum height reached by any rectangle (in particular, there are no edge capacities). Notice that in DSA a lower bound is provided by the peak demand, i.e. the maximum over the edges  $e$  of the sum of the heights of rectangles whose path uses  $e$ . Buchsbaum et al. [9] studied in detail the relation between the optimal peak height and the peak demand, providing examples where these values differ by a constant factor. The authors also present a  $(2 + \varepsilon)$ -approximation for DSA that provides guarantees even when compared with the peak demand.



### 1.3 Organization

We start by introducing in Section 2 some useful definitions and known results. As a warm-up, in Section 3 we present a very simple 2-approximation for DSP that allows us to illustrate part of our ideas. Then in Section 4 we present our main result, namely a  $(5/3 + \varepsilon)$ -approximation for DSP. Finally, in Section 5 we provide details about the gap instances in Figure 1. The results for special cases of DSP (Theorems 2 and 3) can be found in Appendix B and the full version of this paper [17] respectively.

## 2 Preliminaries

Let  $e_1, \dots, e_W$  be the edges of  $G$  from left to right. Recall that a feasible solution or schedule  $P(\cdot)$  specifies a subpath  $P(i)$  of  $G$  of length  $w(i)$  for each task  $i$ . Sometimes it is convenient to consider a *partial* schedule  $P(\cdot)$  which specified the path of a subset  $\mathcal{I}'$  of tasks only (it is convenient to consider  $P(i)$  as an *empty path* for the remaining tasks). We call this a schedule of  $\mathcal{I}'$ .

Let us define, for a given subset  $\mathcal{I}'$  of tasks,  $h_{\max}(\mathcal{I}') := \max_{i \in \mathcal{I}'} h(i)$  and  $h(\mathcal{I}') := \sum_{i \in \mathcal{I}'} h(i)$ . We define analogously  $w_{\max}(\mathcal{I}')$  and  $w(\mathcal{I}')$  w.r.t. widths. Let also  $a(\mathcal{I}') := \sum_{i \in \mathcal{I}'} a(i)$ , where  $a(i) := h(i) \cdot w(i)$  corresponds to the *area* of task  $i$ . We will start by showing a couple of simple lower bounds for the optimal peak  $OPT$  that will be used extensively along this work.

► **Proposition 4.**  $OPT \geq \max\{h_{\max}(\mathcal{I}), \sum_{i \in \mathcal{I}: w(i) > W/2} h(i), a(\mathcal{I})/W\}$ .

**Proof.** Since the total demand of any edge used by the task of largest height is at least  $h_{\max}(\mathcal{I})$ , it holds that  $OPT \geq h_{\max}(\mathcal{I})$ . Also notice that, in any scheduling, the tasks of width larger than  $W/2$  use the edge  $e_{\lceil W/2 \rceil}$ , being then the total demand of this edge (and consequently  $OPT$ ) at least  $\sum_{i \in \mathcal{I}: w(i) > W/2} h(i)$ . Finally, the last bound follows from an averaging argument and the fact that the sum over the edges of the total demand on each edge is equal to  $a(\mathcal{I})$ . ◀

### 2.1 Demand Profile and Left-Pushing

Consider a schedule  $P(\cdot)$  of  $\mathcal{I}' \subseteq \mathcal{I}$ . We define the *demand profile*  $h(P)$  of  $P(\cdot)$  as the vector that stores for each edge  $e$  the total demand  $\sum_{i \in \mathcal{I}': e \in P(i)} h(i)$  of the tasks whose path contains  $e$  (if the path of  $i$  is not specified, then  $i$  does not contribute to the demand profile). Since  $W$  can be exponential in  $n$ , we need to store the demand profile in a more efficient way. This can be done by noticing that the number of times the total demand changes from an edge to the next one is at most  $2n$  (when a task starts or finishes). Hence we just need to store the edges where the demand profile changes value and the corresponding demand. In particular, we can efficiently store the demand profile. Furthermore, we can efficiently update it, e.g., when augmenting an existing schedule by specifying the path  $P(i)$  of one more task  $i$ , or when we modify the value of some  $P(i)$  by *shifting* tasks as we will discuss later.

Given a schedule  $P(\cdot)$  of  $\mathcal{I}' \subseteq \mathcal{I}$  and  $i \in \mathcal{I}'$ , a *left-shifting* of  $i$  in  $P(\cdot)$  is the operation of replacing  $P(i)$  with the path  $P'(i)$  of length  $w(i)$  that starts one edge to the left of  $P(i)$ . Clearly, this operation is allowed only if  $P(i)$  does not start at the leftmost edge of  $G$ . Consider a schedule  $P(\cdot)$  with peak  $\pi$ , and let  $\pi' \geq \pi$ . A  $\pi'$ -*left-pushing* of  $P(\cdot)$  is the operation of iteratively performing left-shiftings in any order until it is not possible to continue while guaranteeing that the peak is always at most  $\pi'$ . We will critically use left-pushings in our algorithms. Notice that a left-pushing can be computed in polynomial time (see Appendix B for some more details).

Intuitively, left-pushing accumulates the demand over the first edges while inducing a non-increasing demand profile to the right. For a node  $t^*$  of the path and a value  $Q \geq 0$ , we will say that a (possibly partial) schedule  $P(\cdot)$  is  $(Q, t^*)$ -sorted if the corresponding demand on the edges to the left of  $t^*$  is at least  $Q$  and on the edges to the right of  $t^*$  the demand profile is non-increasing (see Figure 2); if  $t^*$  is the leftmost node we just say that the schedule is sorted. Our algorithms will first schedule some tasks and then perform a left-pushing. After that, it will be possible to schedule the remaining tasks in a convenient way thanks to the properties of the resulting demand profile.

## 2.2 Container-based Scheduling

Similar to recent work on related rectangle packing problems (e.g., [18, 6]), we will exploit a *container-based* scheduling approach. A *container*  $C$  can be interpreted as an artificial task, with its own width  $w(C)$  (i.e. a number of edges) and height  $h(C)$ . Furthermore, it is classified as *vertical* or *horizontal*, with a meaning which is explained later. The containers are scheduled as usual tasks in a DSP instance (in particular by defining a path  $P(C)$  for each container  $C$ ), with the goal of minimizing the peak  $\pi$ . We also define a packing of tasks into containers  $C$  respecting the following constraints: if  $C$  is vertical, the tasks  $\mathcal{I}(C)$  packed into  $C$  must have height at most  $h(C)$  and total width at most  $w(C)$ ; if  $C$  is horizontal, tasks  $\mathcal{I}(C)$  must have width at most  $w(C)$  and total height at most  $h(C)$ . Intuitively, the tasks packed into a vertical (resp., horizontal) container induce a geometric packing of the rectangles associated with each task into the rectangle corresponding to the container, where the task rectangles are packed non-overlappingly one next to the other (resp., one on top of the other). Any such packing and scheduling of containers naturally induces a schedule of the tasks: if  $C$  is horizontal, tasks  $\mathcal{I}(C)$  are all scheduled starting on the leftmost edge of  $P(C)$ . Otherwise, tasks  $\mathcal{I}(C)$  are scheduled one after the other starting at the leftmost edge of  $P(C)$ . It is hopefully clear to the reader that the demand profile of such a schedule of the tasks is dominated by the demand profile of the containers' schedule. In particular, if the latter has peak  $\pi$ , then the corresponding schedule of the tasks has a no larger peak.

The general strategy is then as follows: we first show that there exists a convenient packing of tasks into a constant number of containers and that there exists a scheduling  $P^*(\cdot)$  of these containers with a small peak  $\pi$ . We also require that these containers are *guessable*, meaning that we can guess their sizes by exploring a polynomial number of options. Once we guessed the correct set of containers, a  $\pi$ -left-pushing of  $P^*(\cdot)$  can be computed by brute force (since they are constantly many tasks). Finally, we pack tasks into containers, inducing a schedule of the tasks with peak  $\pi$ .

This final step can be performed (almost completely) via a reduction to the GENERALIZED ASSIGNMENT problem (GAP). Recall that in GAP we are given a set of  $k$  bins, where each bin  $j$  has an associated capacity  $C_j \geq 0$ , and a set of  $n$  items. For each item  $i$  and bin  $j$ , the input specifies a size  $s_{ij} \geq 0$  and a profit  $p_{ij} \geq 0$  of item  $i$  w.r.t. bin  $j$ . A feasible solution assigns each item to some bin so that the total size of the items assigned to each bin  $j$  is at most  $C_j$ . Our goal is to maximize the total profit associated with this assignment. GAP admits a PTAS in the case of a constant number of bins (see Section E.2 in [19]).

► **Lemma 5.** *For any constant  $\varepsilon' > 0$ , given a set of tasks  $\mathcal{I}'$  that can be packed into a given set of containers of constant cardinality, there is a polynomial-time algorithm to pack  $\mathcal{I}'' \subseteq \mathcal{I}'$  with  $a(\mathcal{I}'') \geq (1 - \varepsilon')a(\mathcal{I}')$  into the mentioned containers.*

**Proof.** We define a GAP instance as follows: we create one bin per container, where the capacity of the bin is equal to the width of the container if it is vertical or the height of the container if it is horizontal. For each task  $i$  we define an item that has uniform profit equal

to its area  $a(i)$  over all the bins. Given a task  $i$  and a vertical (resp., horizontal) container  $j$ , the size  $s_{ij}$  of  $i$  into bin  $j$  is set to  $w(i)$  (resp.,  $h(i)$ ) if task  $i$  can be packed into container  $j$  according to the mentioned rules. Otherwise we set  $s_{ij} = +\infty$ . The claim follows by applying the aforementioned PTAS for GAP with parameter  $\varepsilon'$ . ◀

Notice that the above lemma allows us to pack all the tasks but a subset of small total area, hence we need to schedule somehow such *leftover* tasks. This is not necessarily a trivial task; indeed, such tasks, though of small area, might have large height, and hence scheduling them on top of the rest might substantially increase the peak. To circumvent this issue we will identify special containers reserved for tasks of large height where we will be able to pack all such tasks with no leftovers. We will then apply the PTAS from Lemma 5 only to the remaining tasks and containers.

### 3 A Simple 2-Approximation for DSP

In order to introduce part of our ideas, in this section, we present a simple 2-approximation for DSP. As mentioned before, a 2-approximation can also be achieved via Steinberg's algorithm [42], however, that algorithm is substantially more complex (not surprisingly since it computes a *geometric packing* of tasks interpreted as rectangles like in GSP).

The following lemma exploits a modification of Next-Fit-Decreasing [13], the well-known approximation algorithm for Bin Packing.

► **Lemma 6.** *Let  $P(\cdot)$  be a sorted schedule of  $\mathcal{I}' \subseteq \mathcal{I}$  with peak at most  $\pi$ . For  $\mathcal{I}'' := \mathcal{I} \setminus \mathcal{I}'$ , assume that:*

- $\pi \geq h_{\max}(\mathcal{I}'') + \max\{a(\mathcal{I})/W, h_{\max}(\mathcal{I}'')\}$ ,
- $w_{\max}(\mathcal{I}'') \leq W/2$ , and
- $(W - w_{\max}(\mathcal{I}''))(\pi - h_{\max}(\mathcal{I}'')) + w_{\max}(\mathcal{I}'') \cdot h_{\max}(\mathcal{I}'') \geq a(\mathcal{I})$ .

*Then it is possible to compute in polynomial time a schedule of  $\mathcal{I}$  having peak at most  $\pi$ .*

**Proof.** By slightly abusing notation, we will next use an edge label  $e$  also to denote the position  $i$  of  $e$  in the sequence  $e_1, \dots, e_W$  of edges from left to right. We do not modify the schedule of  $\mathcal{I}'$  and schedule the remaining tasks  $\mathcal{I}''$  as follows. Let us fix an arbitrary order for tasks in  $\mathcal{I}''$ , and let us initially define  $e^{check}$  to be the leftmost edge. We scan completely the list of tasks  $\mathcal{I}''$  and, if the current task  $i$  can be scheduled starting on edge  $e^{check}$  while maintaining a peak of at most  $\pi$ , we do that and remove  $i$  from  $\mathcal{I}''$ ; otherwise, we keep  $i$  in  $\mathcal{I}''$  and try with the next task. Once we consider the final task, we update  $e^{check}$  to be the leftmost edge to the right of the current  $e^{check}$  whose demand is different from the demand of the current  $e^{check}$ . We iterate the procedure on the new  $e^{check}$  until all tasks are scheduled or we identify a task  $i$  which cannot be scheduled.

First, notice that we update  $e^{check}$  at most  $|\mathcal{I}'|$  times, and after each time, we iterate through at most  $|\mathcal{I}''|$  tasks, so the running time is polynomial in the size of the input. It is also not difficult to see that with this procedure, the demand profile from  $e^{check}$  to its right is always non-increasing (restricted to these edges, scheduling a task is equivalent to summing up two non-decreasing profiles), and none of the remaining tasks can fit in any one of the edges to the left of  $e^{check}$  (as we actually tried to place them there but it was not possible). Notice also that, if this procedure manages to schedule all the tasks, then the claimed peak is automatically achieved. So we will assume by contradiction that this is not the case.

Let  $i$  be a task that could not be scheduled. This could only happen due to  $i$  being too wide for the current edge  $e^{check}$  where it should be scheduled (and hence for any subsequent edge). This implies that  $e^{check} > W - w(i)$  and hence there are more than  $W - w(i)$  edges

## 20:8 Approximation Algorithms for Demand Strip Packing

having demand larger than  $\pi - h(i)$ . Thus the total area of the scheduled tasks plus task  $i$  is strictly larger than

$$A(h(i), w(i)) := (W - w(i)) \cdot (\pi - h(i)) + h(i) \cdot w(i).$$

This expression is decreasing both as a function of  $h(i)$  and as a function of  $w(i)$ . Indeed,

$$\frac{\partial}{\partial h(i)} A(h(i), w(i)) = 2w(i) - W \leq 0, \text{ and } \frac{\partial}{\partial w(i)} A(h(i), w(i)) = 2h(i) - \pi \leq 0,$$

where we used the fact that, by assumption,  $w(i) \leq \frac{W}{2}$  and  $\pi \geq 2h_{\max}(\mathcal{I}'') \geq 2h(i)$ . We conclude that

$$A(h(i), w(i)) \geq A(h_{\max}(\mathcal{I}''), w_{\max}(\mathcal{I}'')) = (W - w_{\max}(\mathcal{I}''))(\pi - h_{\max}(\mathcal{I}'')) + w_{\max}(\mathcal{I}'') \cdot h_{\max}(\mathcal{I}'') \geq a(\mathcal{I}),$$

where in the last inequality we used the third assumption. This is a contradiction since a subset of tasks would have area strictly larger than the total area  $a(\mathcal{I})$ . ◀

We are now ready to provide a simple 2-approximation.

► **Corollary 7.** *There exists a deterministic 2-approximation for DSP.*

**Proof.** Let  $\mathcal{I}$  be an instance of DSP. We will first schedule the tasks  $\mathcal{I}'$  having width larger than  $W/2$  starting on the leftmost edge. Let  $\mathcal{I}'' := \mathcal{I} \setminus \mathcal{I}'$ . This partial schedule is sorted and has peak  $\sum_{i \in \mathcal{I}'} h(i) \leq M := \max\{h_{\max}(\mathcal{I}), \sum_{i \in \mathcal{I}'} h(i), a(\mathcal{I})/W\}$ . Recall that, by Proposition 4,  $M \leq OPT$ . Define  $\pi = 2M$ , and observe that

$$(\pi - h_{\max}(\mathcal{I}''))(W - w_{\max}(\mathcal{I}'')) + h_{\max}(\mathcal{I}'') \cdot w_{\max}(\mathcal{I}'') \geq M \cdot (W/2) + M \cdot (W/2) = M \cdot W \geq a(\mathcal{I}).$$

Thus we can apply Lemma 6 with parameter  $\pi = 2M$ . This provides a schedule with peak at most  $2M \leq 2OPT$ . ◀

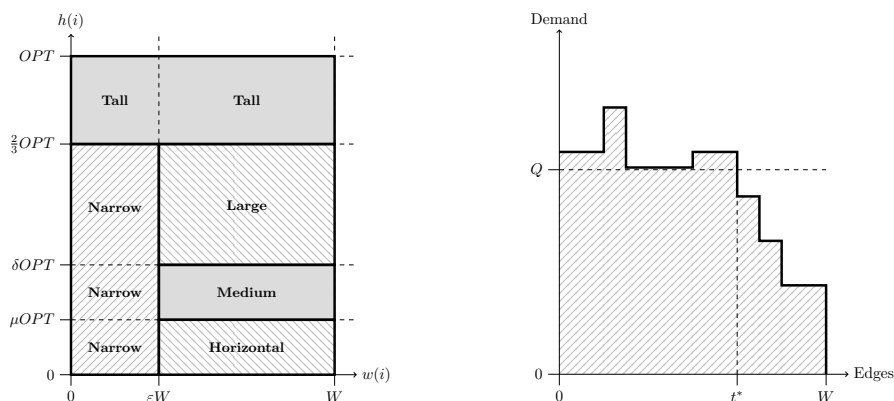
In the following sections, we will extend the approach in the above 2-approximation as follows. We will first compute a feasible solution of some given peak that includes all the tasks having height larger than some threshold and width larger than some threshold. Then we will left-push this schedule to add some structure to the demand profile. Finally, we schedule the remaining tasks by means of a generalization of Lemma 6 (Lemma 12) which considers  $(Q, t^*)$ -sorted schedules (rather than just sorted ones).

### 4 A $(5/3 + \varepsilon)$ -Approximation for DSP

In this section we will prove Theorem 1. In order to attain the claimed result, we will provide first some useful definitions and preprocessing lemmas.

Let us assume that the optimal value  $OPT$  is known to the algorithm (this assumption can be dropped by approximately guessing this value, introducing an extra  $(1 + \varepsilon)$  factor in the approximation). We start by classifying the tasks in the instance according to their widths and heights (see Figure 2). Let  $\mu, \delta, \mu < \delta \leq \varepsilon$ , be two constant parameters to be fixed later. We say that a task  $i$  is:

- *tall* if  $h(i) > \frac{2}{3}OPT$ ,
- *large* if  $h(i) \in (\delta OPT, \frac{2}{3}OPT]$  and  $w(i) > \varepsilon W$ ,
- *horizontal* if  $h(i) \leq \mu OPT$  and  $w(i) > \varepsilon W$ ,
- *narrow* if  $h(i) \leq \frac{2}{3}OPT$  and  $w(i) \leq \varepsilon W$ , or
- *medium* if  $h(i) \in (\mu OPT, \delta OPT]$  and  $w(i) > \varepsilon W$ .



■ **Figure 2** (Left) Classification of the tasks according to Section 4. (Right) Representation of a  $(Q, t^*)$ -sorted demand profile.

We denote by  $\mathcal{T}, \mathcal{L}, \mathcal{H}, \mathcal{N}$  and  $\mathcal{M}$  the sets of tall, large, horizontal, narrow and medium tasks respectively. Recall that the general idea of our approach is to show that we can schedule almost every task in a constant number of containers, which in turn can be enumerated in polynomial time by brute force. In the rest of this section, we basically show the existence of such a solution for most of the previously described classes. More precisely, we argue that the optimal schedule can be rearranged so as to define constantly many containers to pack all the tasks in  $\mathcal{T} \cup \mathcal{L}$  while increasing the peak of the solution by at most an additive  $\frac{2}{3}OPT$  term (see Lemma 9). We then show that a subset of the tasks in  $\mathcal{H}$  can be scheduled in  $\mathcal{O}(1)$  containers such that the left-over from this class has a very small area compared  $W \cdot OPT$  (see Lemma 10). Then, it is not difficult to show that the left-over from  $\mathcal{H}$  can be placed in a single container with a very small height. We also prove that all the tasks in  $\mathcal{M}$  can be packed in a single container of small height by showing how to choose the parameters  $\mu$  and  $\delta$  (see Lemma 8). This leads in conclusion to a container-based scheduling for  $\mathcal{I} \setminus \mathcal{N}$  of peak  $(\frac{5}{3} + 7\varepsilon)OPT$  (see Lemma 11). Finally, we present a method to schedule every task in  $\mathcal{N}$  on top of the previous solution without increasing its peak by using a greedy approach (see Lemma 12).

We first deal with the medium tasks. As the following lemma states, it is possible to choose  $\mu$  and  $\delta$  in such a way that the two parameters differ by a large factor and the total height of medium tasks is small.

► **Lemma 8.** *Given a polynomial-time computable function  $f : (0, 1) \rightarrow (0, 1)$ , with  $f(x) < x$ , and any constant  $\varepsilon \in (0, 1)$ , we can compute in polynomial time a set  $\Delta$  of  $\frac{2}{\varepsilon^2}$  many positive real numbers upper bounded by  $\varepsilon$ , such that there is at least one number  $\delta \in \Delta$  so that, by choosing  $\mu = f(\delta)$ , one has  $a(\mathcal{M}) \leq \varepsilon^2 \cdot OPT \cdot W$  (hence  $h(\mathcal{M}) \leq \varepsilon OPT$ ).*

**Proof.** Let  $y_1 = \varepsilon$  and, for each  $j \in \{1, \dots, |\Delta|\}$ , define  $y_{j+1} = f(y_j)$ . For each  $j \leq |\Delta|$ , let  $I_j = \{i \in \mathcal{I} : h(i) \in [y_{j+1}, y_j]\}$ . Note that  $y_j$ 's are decreasing since  $f(x) < x$ . Observe that  $I_{j'}$  is disjoint from  $I_{j''}$  for every  $j' \neq j''$ , and the total area of tasks in  $\bigcup I_j$  is at most  $W \cdot OPT$ . Thus, there exists a value  $\bar{j}$  such that the total area of the tasks in  $I_{\bar{j}}$  is at most  $\frac{2OPT \cdot W}{|\Delta|} = \varepsilon^2 \cdot OPT \cdot W$ . Choosing  $\delta = y_{\bar{j}}$  and  $\mu = y_{\bar{j}+1}$  verifies all the conditions of the lemma as in that case  $\mathcal{M} \subseteq I_{\bar{j}}$ . Notice that, since every task in  $\mathcal{M}$  has width at least  $\varepsilon W$ , we have that  $h(\mathcal{M}) \leq \varepsilon OPT$ . ◀

Function  $f$  will be given later. From now on, we will assume that  $\mu$  and  $\delta$  are chosen according to Lemma 8. Notice that this implies that  $\mu, \delta = O_\varepsilon(1)$ . The rest of this section is organized as follows. In Section 4.1 we define a container-based scheduling of  $\mathcal{T} \cup \mathcal{L}$ . In Section 4.2 we extend this in order to include also  $\mathcal{H}$ . In Section 4.3 we schedule the remaining tasks and prove Theorem 1.

#### 4.1 Containers for Tall and Large Tasks

In this section, we define a packing of tall and large tasks into a constant number of guessable containers. This packing can be computed exactly, i.e. with no leftovers (in particular, we will not use Lemma 5 to compute such packing). To that aim, we will exploit the following structural result.

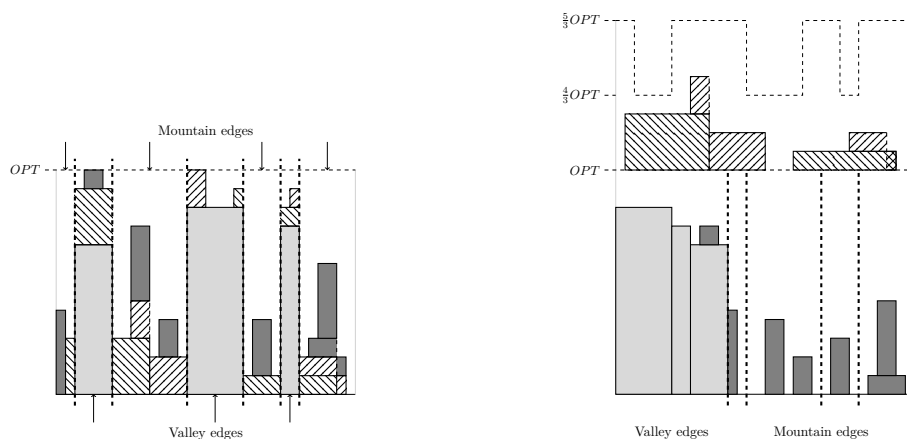
► **Lemma 9.** *Let  $P(\cdot)$  be an optimal schedule of  $\mathcal{I}$  (hence with peak  $OPT$ ). There exists a packing  $P'(\cdot)$  with peak at most  $\frac{5}{3}OPT$  satisfying that all the tall tasks are scheduled one after the other starting on the leftmost edge in non-increasing order of height.*

**Proof.** Let  $\mathcal{T}$  be the tall tasks (having height larger than  $\frac{2}{3}OPT$ ). Notice that the paths of these tasks in  $P(\cdot)$  need to be edge disjoint. Let us classify the edges into *valley edges* if some task in  $\mathcal{T}$  uses that edge in  $P(\cdot)$  and *mountain edges* otherwise (see also Figure 3a). We let  $\mathcal{I}_{mnt}$  be the (*mountain*) tasks whose path in  $P(\cdot)$  consists solely of mountain edges,  $\mathcal{I}_{vll}$  be the (*valley*) tasks whose path in  $P(\cdot)$  consists solely of valley edges (notice that this set includes  $\mathcal{T}$ ), and  $\mathcal{I}_{crs} := \mathcal{I} \setminus (\mathcal{I}_{vll} \cup \mathcal{I}_{mnt})$  the remaining (*crossing*) tasks.

We next define a modified partial schedule  $P'(\cdot)$  of  $\mathcal{I}_{vll} \cup \mathcal{I}_{mnt}$  as follows. Let us reorder the edges of the path (and the tasks accordingly) so that valley edges appear to the left and mountain edges appear to the right in the path (maintaining their relative order). Furthermore, we rearrange the valley edges so that tasks in  $\mathcal{T}$  are scheduled from left to right in non-increasing order of height. Observe that by construction  $\mathcal{I}_{mnt}$  are scheduled on  $W - w(\mathcal{T})$  edges (i.e. the total number of mountain edges). Since we temporarily removed crossing tasks, this induces a feasible packing of  $\mathcal{I}_{vll} \cup \mathcal{I}_{mnt}$ . The resulting packing  $P'(\cdot)$  clearly has a peak of at most  $OPT$ .

Consider next the schedule  $P(\cdot)$  restricted to  $\mathcal{I}_{crs}$ . We claim that this schedule has peak at most  $\frac{2}{3}OPT$ . Indeed, notice first that the demand of valley edges is at most  $\frac{1}{3}OPT$ . Consider next a mountain edge  $e$ . Let  $e_\ell$  be the rightmost valley edge to the left of  $e$  (if any), and define  $e_r$  symmetrically to the right of  $e$ . Any task in  $\mathcal{I}_{crs}$  using  $e$  must also use  $e_\ell$  or  $e_r$  (or both). Hence the total demand on  $e$  is at most the total demand on  $e_\ell$  plus the total demand on  $e_r$ , thus at most  $\frac{2}{3}OPT$ . The claim follows by combining the schedule of  $\mathcal{I}_{crs}$  (taken from  $P(\cdot)$ ) with the above schedule  $P'(\cdot)$  of  $\mathcal{I}_{vll} \cup \mathcal{I}_{mnt}$  (see also Figure 3b). ◀

We will next assume that tall tasks are scheduled as in the above lemma. By increasing the peak by  $\varepsilon OPT$  (up to  $(\frac{5}{3} + \varepsilon)OPT$ ), one can define a set of  $O_\varepsilon(1)$  (tall) containers where such tasks can be packed (respecting the mentioned order and with no leftovers). Consider the demand profile of tall tasks in the considered schedule, and round it to the next multiple of  $\varepsilon OPT$ . Consider the tasks  $\mathcal{T}_k$  corresponding to the value  $k \cdot \varepsilon OPT$  in the rounded profile. Notice that these tasks are scheduled consecutively along some path  $P_k$ . We create a vertical container  $C_k$  of height  $k \cdot \varepsilon OPT$  and width  $|E(P_k)|$ , pack  $\mathcal{T}_k$  into  $C_k$ , and schedule  $C_k$  on  $P_k$ . Clearly, we need to create at most  $1/\varepsilon$  containers. Notice also that the dimensions of these containers can be guessed in polynomial time since there is a constant number of options for the height, and the widths correspond to the total width of a subsequence of tall tasks in the considered ordering by non-increasing height (breaking ties arbitrarily).



(a) A scheduling of peak  $OPT$ . Light gray rectangles correspond to tasks in  $\mathcal{T}$ , dark gray ones represent mountain and valley tasks, and dashed ones the crossing tasks.

(b) Structured solution having peak at most  $\frac{5}{3}OPT$ , where tasks in  $\mathcal{T}$  are placed one next to the other, starting at the leftmost edge and sorted non-increasingly by height.

■ **Figure 3** Depiction of the proof of Lemma 9.

It remains to consider large tasks. Since they are at most  $\frac{1}{\varepsilon\delta} = O_\varepsilon(1)$  many, it is sufficient to define a distinct (large) container for each one of them and pack the large tasks accordingly. We schedule the large containers exactly as in the solution guaranteed by Lemma 9. Clearly tall and large containers can be scheduled together with a peak of at most  $(\frac{5}{3} + \varepsilon)OPT$  by the above construction.

## 4.2 Containers for Horizontal Tasks

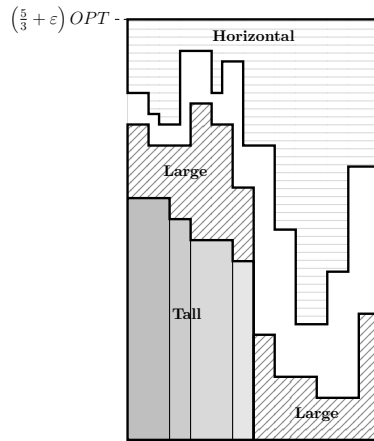
In this section, we define a packing of horizontal tasks into a constant number of guessable containers. These containers can be scheduled together with the tall and large containers with small enough peaks. This will induce a convenient schedule of non-narrow tasks.

Let us focus on the schedule of tall and large containers with a peak of at most  $(\frac{5}{3} + \varepsilon)OPT$  from the previous section. Consider now the demand profile of such container schedule. Since the demand profile of tall containers has at most  $1/\varepsilon = O_\varepsilon(1)$  jumps, and the demand profile of large containers has at most  $2/(\varepsilon\delta) = O_\varepsilon(1)$  jumps, then the overall demand profile has  $O_\varepsilon(1)$  jumps.

Assume next that horizontal tasks are scheduled as in Lemma 9: notice that such tasks can be scheduled with the tall and large containers without increasing the peak. This implies that the demand profile of horizontal tasks is upper bounded (on each coordinate) by the difference between  $(\frac{5}{3} + \varepsilon)OPT$  and the demand profile of tall and large containers (see Figure 4). Under these conditions, it is possible to build containers for horizontal tasks, using the standard linear grouping technique, as the following lemma shows.

► **Lemma 10.** *Suppose there exists a schedule of  $\mathcal{H}$  such that its demand profile is upper bounded (vectorially) by a demand profile  $D$  with  $O_\varepsilon(1)$  jumps. Then there exists a container packing for  $\mathcal{H}$  into  $O_\varepsilon(1)$  horizontal guessable containers with demand profile upper bounded by  $D$  plus  $4\varepsilon OPT$  on each coordinate.*

**Proof.** Let us assume by now that horizontal tasks are *horizontally sliced*, meaning that each task  $i$ , having height  $h(i)$  and width  $w(i)$ , is replaced by  $h(i)$  sibling slices, which are tasks of width  $w(i)$  and height 1. The schedule of the slices is the same as for the corresponding task.



■ **Figure 4** The demand profile of tall and large tasks in the schedule obtained from Lemma 9 has  $O_\epsilon(1)$  jumps, bounding the profile of (sliced) horizontal tasks (on top).

In order to reduce the possible number of distinct slice widths to a constant, we will use the technique of linear grouping while increasing the final peak by at most  $2\epsilon OPT$ . We start by considering all the slices in a pile, one on top of the other and sorted non-increasingly by width from bottom to top (and putting sibling slices consecutively). Since these slices have a width at least  $\epsilon W$  and total area at most  $OPT \cdot W$ , the pile has total height at most  $\frac{1}{\epsilon} OPT$ . Starting from the bottom, we partition the pile into groups  $G_1, \dots, G_q$  of height exactly  $\epsilon OPT$  (except possibly for  $G_q$  which may have a smaller height). We remove from the solution the slices in  $G_1$  and any slice in  $G_2$  which used to have a sibling slice in  $G_1$ , and temporarily remove the corresponding tasks. Observe that we are removing the tasks whose slices are fully contained in  $G_1$  plus at most one extra task. In particular, the total height of the removed tasks is at most  $(\epsilon + \mu)OPT \leq 2\epsilon OPT$  (here we use  $\mu \leq \epsilon$ ).

Next, we round up the widths of the remaining slices as follows: for  $i = 2, \dots, q$ , the width of slices (still) in  $G_i$  are rounded to the smallest width of any slice originally in  $G_{i-1}$ . We call this set of slices the rounded slices, and next focus on packing them. Notice that rounded slices have at most  $1/\epsilon^2$  distinct widths. This also induces a matching between each rounded slice  $a$  and a distinct original slice  $b$ , so that  $w(b) \geq w(a)$ . In particular, we can schedule each such  $a$  starting on the first edge of  $P(b)$  without increasing the overall peak.

Now we left-shift the horizontal slices in the solution as much as possible while still obtaining a schedule whose demand profile is upper bounded by  $D$ . Let  $e$  be the starting edge of some slice  $S$  at the end of the process. Notice that one of the following cases holds: (1)  $D$  increases on edge  $e$  (including as a special case when  $e$  is the leftmost edge of  $G$ ) or (2) the edge  $f$  to the left of  $e$  is the ending edge of some other slice  $S'$ . Indeed otherwise it would be possible to left-shift  $S$  while respecting all the constraints. This implies that the possible positions for the starting edge of any slice can be obtained by considering the  $O_\epsilon(1)$  edges where  $D$  increases and then adding the total width of a few slices. Notice that there are at most  $1/\epsilon^2$  such widths, and we can sum up at most  $1/\epsilon$  of them (since horizontal slices have widths at least  $\epsilon W$ ). Altogether, the number of possible starting edges for the slices is  $O_\epsilon(1)$ .

Consider the leftmost possible such edge  $e$  and all the rounded slices  $S_e$  starting on  $e$  in this left-shifted schedule. We partition  $S_e$  by width  $w$ , and for each such width  $w$  and corresponding set of slices  $S_{e,w}$ , we construct a horizontal container of width  $w$  and height  $h(S_{e,w})$  where we pack  $S_{e,w}$ . We repeat this procedure for each possible starting edge  $e$ ,



obtaining in the end  $K \leq O_\varepsilon(1)$  containers in total where we packed all the rounded slices. Notice that the width of each container is the width of some rounded slice, which in turn is the width of some task. Hence the widths of the containers are guessable in the usual sense.

We next turn the above packing of rounded slices into a feasible packing of tasks (into the same containers). First of all, we repack the rounded slices as follows. We consider all the slices  $S_w$  of a given width  $w$  in any order where sibling slices appear consecutively, and all the containers  $C_w$  of that width in any order. We pack each slice  $s \in S_w$  in the first container  $C \in C_w$  where  $s$  still fits. Notice that  $h(S_w) = h(C_w)$  by construction, hence we repack all rounded slices this way. Next we consider the tasks  $i$  whose slices are all contained in the *same* container  $C$ , and pack  $i$  into  $C$ . By construction this packing is feasible. We add the tasks which are not packed this way to the set of removed tasks defined earlier. We round up the heights of the containers to the next multiple of  $\frac{\varepsilon}{K}OPT$ , hence making such heights guessable. This way the peak increases at most by  $\varepsilon OPT$ .

Consider the set of removed tasks. Recall that the tasks removed in the initial rounding phase have total height at most  $2\varepsilon OPT$ . In the following packing phase, we remove at most one task per container, hence these removed tasks have height at most  $K \cdot \mu \cdot OPT \leq \varepsilon OPT$ . Here we assume that  $\mu \leq \varepsilon/K$ : this can be achieved by choosing  $f(x) = x/K$  in Lemma 8. Hence the removed tasks altogether have total height at most  $3\varepsilon OPT$ : we pack these tasks in one extra (guessable) horizontal container of width  $W$  and height  $3\varepsilon OPT$ . ◀

From the above construction it is possible to derive a schedule of non-narrow tasks of small enough peak.

► **Lemma 11.** *It is possible to compute in polynomial-time a feasible schedule of  $\mathcal{I} \setminus \mathcal{N}$  with peak at most  $(\frac{5}{3} + 7\varepsilon) OPT$ .*

**Proof.** Consider the guessable containers for tall, large, and horizontal tasks and the corresponding schedule as described before. This schedule has a peak of at most  $(\frac{5}{3} + \varepsilon)OPT + 4\varepsilon OPT$ . By Lemma 8 the medium tasks fit into a horizontal container of width  $W$  and height  $\varepsilon OPT$ . Altogether this leads to a packing of  $\mathcal{L} \cup \mathcal{T} \cup \mathcal{H} \cup \mathcal{M} = \mathcal{I} \setminus \mathcal{N}$  into  $O_\varepsilon(1)$  guessable containers that can be scheduled with peak at most  $(\frac{5}{3} + 6\varepsilon)OPT$ .

It is easy to pack  $\mathcal{L} \cup \mathcal{T} \cup \mathcal{M}$  into the corresponding containers. For  $\mathcal{H}$  we apply Lemma 5 with  $\varepsilon' = \varepsilon^2$  to assign the horizontal tasks to them, obtaining a set of horizontal unplaced tasks of an area at most  $\varepsilon^2 \cdot W \cdot OPT$  (hence of total height at most  $\varepsilon OPT$ ). The latter tasks can be placed into an extra horizontal container of height  $\varepsilon OPT$  and width  $W$ . The resulting set of containers can be scheduled with a peak at most  $(\frac{5}{3} + 7\varepsilon)OPT$ , and such a schedule can be efficiently computed as already discussed. ◀

### 4.3 Scheduling Narrow Tasks

At this point it just remains to schedule the narrow tasks. For this goal we need the following generalization of Lemma 6 that considers  $(Q, t^*)$ -sorted partial schedules. Recall that for a node  $t^*$  of the path and a value  $Q \geq 0$ , a schedule is  $(Q, t^*)$ -sorted if the demand to the left of  $t^*$  is at least  $Q$ , and to the right of  $t^*$  the demand profile is non-increasing.

► **Lemma 12.** *Let  $\mathcal{I}$  be an instance of DSP and  $\alpha > 0$ . Suppose we are given a  $((1 + \alpha)OPT, t^*)$ -sorted schedule of  $\mathcal{I}' \subseteq \mathcal{I}$ . Let  $\mathcal{I}'' := \mathcal{I} \setminus \mathcal{I}'$  and assume that:*

- *The peak of the schedule is at most  $\pi$ , with  $\pi \geq (1 + \alpha)OPT + h_{\max}(\mathcal{I}'')$ , and*
- *$w_{\max}(\mathcal{I}'') \leq \frac{\alpha}{2(\alpha+1)}W$ .*

*Then it is possible to compute in polynomial time a schedule of  $\mathcal{I}$  with peak at most  $\pi$ .*

## 20:14 Approximation Algorithms for Demand Strip Packing

**Proof.** By overloading notation, let  $t^*$  also denote the number of edges to the left of  $t^*$ . Notice first that  $W - t^* \geq 2w_{\max}(\mathcal{I}'')$ . Indeed otherwise, since the input schedule is  $((1 + \alpha)OPT, t^*)$ -sorted, the total area of the tasks in  $\mathcal{I}'$  would be at least

$$t^* \cdot (1 + \alpha)OPT > (W - 2w_{\max}(\mathcal{I}''))(1 + \alpha)OPT \geq W \cdot OPT$$

which is not possible. Roughly speaking, to prove the desired claim, we will apply Lemma 6 to the demand profile induced by the edges to the right of  $t^*$ . In more detail, we consider a new instance defined by a path with  $\tilde{W} = W - t^*$  edges and a set of tasks  $\tilde{\mathcal{I}}$  consisting of  $\tilde{\mathcal{I}}'' := \mathcal{I}''$  plus a set  $\tilde{\mathcal{I}}'$  of  $\tilde{W}$  tasks having width 1 and, for each edge  $e$  to the right of  $t^*$ , height equal to the total demand on edge  $e$  in the original schedule for  $\mathcal{I}'$ . To see that the required hypotheses are satisfied, notice that by scheduling the tasks in  $\tilde{\mathcal{I}}'$  one next to the other sorted non-increasingly by height we obtain a sorted partial schedule of a peak at most  $\pi$ , where

$$\pi \geq h_{\max}(\mathcal{I}'') + (1 + \alpha)OPT \geq h_{\max}(\tilde{\mathcal{I}}'') + \max\{a(\tilde{\mathcal{I}})/\tilde{W}, h_{\max}(\tilde{\mathcal{I}}'')\}.$$

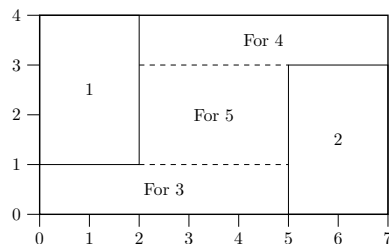
The last inequality above holds since  $a(\tilde{\mathcal{I}}) \leq OPT \cdot W - a(\mathcal{I}') \leq OPT(W - (1 + \alpha)t^*)$ . Finally, we notice that  $w_{\max}(\tilde{\mathcal{I}}'') \leq \tilde{W}/2$ , and

$$\begin{aligned} & (\tilde{W} - w_{\max}(\tilde{\mathcal{I}}''))(\pi - h_{\max}(\tilde{\mathcal{I}}'')) + w_{\max}(\tilde{\mathcal{I}}'') \cdot h_{\max}(\tilde{\mathcal{I}}'') \\ \geq & (\tilde{W} - w_{\max}(\tilde{\mathcal{I}}''))(1 + \alpha)OPT \\ = & OPT(W - t^*(1 + \alpha)) + \alpha W \cdot OPT - w_{\max}(\mathcal{I}'')(1 + \alpha)OPT \\ \geq & a(\tilde{\mathcal{I}}) + \alpha W \cdot OPT - \frac{\alpha}{2}W \cdot OPT \geq a(\tilde{\mathcal{I}}). \end{aligned}$$

Hence all the conditions of Lemma 6 apply. Given that the demand profile of the partial schedule for  $\tilde{\mathcal{I}}$  is the same as the demand profile induced by the edges to the right of  $t^*$  in the original schedule for  $\mathcal{I}'$ , we can schedule  $\mathcal{I}''$  on top of the input schedule without exceeding the peak  $\pi$ .  $\blacktriangleleft$

We now have all the ingredients to prove Theorem 1.

**Proof of Theorem 1.** Consider the schedule of  $\mathcal{I} \setminus \mathcal{N}$  with peak at most  $\pi := (\frac{5}{3} + 7\varepsilon)OPT$  provided by Lemma 11. We perform a  $\pi$ -left-pushing of this schedule, however without left-shifting any tall task. Let us prove that this partial schedule of  $\mathcal{I}' := \mathcal{I} \setminus \mathcal{N}$  satisfies all the required properties of Lemma 12 with parameter  $\pi$ . First of all, there exists a node  $t^*$  for which (1) every edge to the left of  $t^*$  (if any) has demand larger than  $(1 + 7\varepsilon)OPT$ , and (2) the demand profile to the right of  $t^*$  is non-increasing. Indeed, if (1) does not hold, then there exists an edge having demand less than  $(1 + 7\varepsilon)OPT$  and the following edge has demand larger than  $(1 + 7\varepsilon)OPT$ . But this means that some task which is not tall can be left-shifted (as there can be only one tall task per edge); similarly, if (2) does not hold, there is a pair of contiguous edges to the right of  $t^*$  where the demand strictly increases from left to right. But since the tall tasks are sorted non-increasingly by height, this implies that there exists a task that is not tall that can be left-shifted. In conclusion, the solution is  $((1 + 7\varepsilon)OPT, t^*)$ -sorted and also  $w_{\max}(\mathcal{N}) \leq \varepsilon W \leq \frac{7\varepsilon}{2(1+7\varepsilon)}W$  for  $\varepsilon$  small enough. Since  $\pi \geq (1 + 7\varepsilon)OPT + h_{\max}(\mathcal{N})$ , by Lemma 12 we obtain a feasible schedule of peak at most  $\pi$ . The claim follows by scaling  $\varepsilon$  appropriately.  $\blacktriangleleft$



■ **Figure 5** If we assume by contradiction that some optimal solution of height 4 for the GSP instance described in Lemma 13 exists, it must have this structure.

## 5 Comparison between DSP and GSP

In this Section we provide instances where a gap between the optimal values they achieve interpreted as DSP and GSP instances can be observed. First we discuss the general case and then the case of SQUARE-DSP. It is worth noticing that, for the general case, an analogous proof can be derived from the results in [8].

► **Lemma 13.** *There exists an instance of DSP with optimal peak 4 such that the corresponding GSP instance has optimal peak 5.*

**Proof.** Consider the following DSP instance  $\mathcal{I}$ , where  $W = 7$  and the set of tasks consists of the following eight elements (see Figure 1a for a depiction):

- Two tasks of width 2 and height 3 (tasks 1 and 2 in the figure),
- Two tasks of width 4 and height 1 (tasks 3 and 4 in the figure),
- One task of width 3 and height 1 (task 5 in the figure),
- One task of width 1 and height 1 (task 6 in the figure), and
- Two tasks of width 1 and height 2 (tasks 7 and 8 in the figure).

As it is possible to see in Figure 1a, the optimal solution has peak 4 (since  $OPT \geq a(\mathcal{I})/W = 4$ ). We will show now that there is no solution for the corresponding GSP instance of height 4, which would conclude the proof.

Suppose by contradiction that there exists a solution to the corresponding GSP instance of height 4. Let us imagine for the sake of presentation that we draw a grid of unit-size cells over the rectangular region  $[0, 7] \times [0, 4]$ , defining four rows of height 1 and seven columns of width 1. First of all, notice that in any feasible packing of the rectangles into the region, rectangles 1 and 2 cannot be touching the top (resp. bottom) boundary of the region at the same time. If that is the case, then the rectangles 3 and 4 do not fit in the region as they cannot be placed in the same row and none of them fits in the rows which are partially occupied by rectangles 1 and 2. So let us assume w.l.o.g. that rectangle 1 touches the top boundary and rectangle 2 touches the bottom boundary. Since they both partially occupy the middle rows of the region, rectangles 3 and 4 must be placed one touching the bottom boundary and the other touching the top boundary. This implies that rectangle 5 has to be placed in one of the middle rows (in the other rows there is just one cell free), forcing us to place rectangles 1 and 2 one touching the left boundary and the other touching the right boundary (see Figure 5). Suppose rectangle 5 is assigned to the second row from bottom to top (the other case being symmetric). Then in the two topmost rows we have to pack two rectangles of height 2 plus a rectangle of width 4 which is not possible as their total width is larger than the space left due to rectangle 1. This contradicts the fact that there is a feasible solution for the GSP instance  $\mathcal{I}$  of height 4. ◀

## 20:16 Approximation Algorithms for Demand Strip Packing

Now we will prove that even for the case of square tasks, the optimum packing for the two problems of SQUARE-DSP and SQUARE-GSP can exhibit a gap.

► **Lemma 14.** *There exists an instance of SQUARE-DSP such that the optimal schedule has peak 11 but every feasible solution for the corresponding SQUARE-GSP instance has height at least 12.*

**Proof.** Consider a SQUARE-DSP with  $W = 13$  and containing the following set  $\mathcal{I}$  of tasks (see Figure 1b for a depiction):

- Two tasks of height/width 6 (tasks 1 and 2 in the figure),
- Two tasks of height/width 5 (tasks 3 and 4 in the figure),
- One task of height/width 3 (task 5 in the figure),
- Two tasks of height/width 2 (tasks 6 and 7 in the figure), and
- Four tasks of height/width 1 (tasks 8, 9, 10 and 11 in the figure).

Since  $a(\mathcal{I}) = 11 \cdot 13$ , we have that  $OPT \geq 11$ . Figure 1b shows that the optimal peak is at most 11 and hence it is exactly 11.

Assume by contradiction that there exists a feasible packing for the corresponding SQUARE-GSP instance of height at most 11. Consider  $\mathcal{K}$  to be the region  $[0, 0] \times [13, 11]$  in the plane, and let  $(x_i, y_i)$  be the coordinate of the bottom-left corner of task  $i$  in the solution. Notice that  $\mathcal{K}$  must be completely filled with tasks.

We can assume that  $x_1 \leq x_2$ , and since tasks 1 and 2 have height 6 and the height of  $\mathcal{K}$  is 11, it must hold that  $x_1 \leq x_2 + 6$ . Hence, w.l.o.g. there are two cases to consider:

- $x_1 = 0$  and  $x_2 = 6$ :

In this case the region  $[12, y_2] \times [13, y_2 + 6]$  can only contain squares of size 1, and they cannot fill the region completely, so this case cannot happen.

- $x_1 = 0$  and  $x_2 = 7$ :

We show that  $y_1, y_2 \in \{0, 5\}$ ; Assume that  $y_1 \notin \{0, 5\}$ . Then tasks 2, 3 and 4 must be packed inside the region  $[6, 0] \times [13, 11]$  since they can not be packed above or below task 1. Since  $a(j_2) + a(j_3) + a(j_4) > 77$ , this is not possible, hence proving the claim.

Note that if  $y_1 = y_2$  then, similarly to the previous case, the area in  $[6, y_1] \times [7, y_1 + 6]$  can only contain tasks of size 1 and they cannot fill this region completely. So we can assume that  $(x_1, y_1) = (0, 0)$  and  $(x_2, y_2) = (7, 5)$ .

Now every remaining rectangle is either packed in  $[6, 0] \times [13, 5]$  or in  $[0, 6] \times [7, 11]$ . However, among tasks 3, 4 and 5, it is not possible to place two of them in one of the previously mentioned rectangular region together, contradicting the existence of a feasible solution of height 11. ◀

---

### References

- 1 Anna Adamaszek, Tomasz Kociumaka, Marcin Pilipczuk, and Michal Pilipczuk. Hardness of approximation for strip packing. *ACM Trans. Comput. Theory*, 9(3):14:1–14:7, 2017. doi:10.1145/3092026.
- 2 Anna Adamaszek and Andreas Wiese. A quasi-ptas for the two-dimensional geometric knapsack problem. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1491–1505. SIAM, 2015. doi:10.1137/1.9781611973730.98.
- 3 Soroush Alamdari, Therese C. Biedl, Timothy M. Chan, Elyot Grant, Krishnam Raju Jampani, Srinivasan Keshav, Anna Lubiw, and Vinayak Pathak. Smart-grid electricity allocation via strip packing with slicing. In *13th International Symposium on Algorithms and Data Structures (WADS)*, volume 8037, pages 25–36. Springer, 2013. doi:10.1007/978-3-642-40104-6\_3.

- 4 Aris Anagnostopoulos, Fabrizio Grandoni, Stefano Leonardi, and Andreas Wiese. A mazing  $(2 + \varepsilon)$ -approximation for unsplittable flow on a path. *ACM Transactions on Algorithms*, 14(4):55:1–55:23, 2018. doi:10.1145/3242769.
- 5 Brenda S. Baker, Edward G. Coffman Jr., and Ronald L. Rivest. Orthogonal packings in two dimensions. *SIAM Journal on Computing*, 9(4):846–855, 1980. doi:10.1137/0209064.
- 6 Nikhil Bansal, Alberto Caprara, Klaus Jansen, Lars Prädell, and Maxim Sviridenko. A structural lemma in 2-dimensional packing, and its implications on approximability. In *Algorithms and Computation, 20th International Symposium (ISAAC)*, volume 5878, pages 77–86. Springer, 2009. doi:10.1007/978-3-642-10631-6\_10.
- 7 Jatin Batra, Naveen Garg, Amit Kumar, Tobias Mömke, and Andreas Wiese. New approximation schemes for unsplittable flow on a path. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 47–58. SIAM, 2015. doi:10.1137/1.9781611973730.5.
- 8 Iwo Blazek, Maciej Drozdowski, Frédéric Guinand, and Xavier Schepler. On contiguous and non-contiguous parallel task scheduling. *J. Sched.*, 18(5):487–495, 2015. doi:10.1007/s10951-015-0427-z.
- 9 Adam L. Buchsbaum, Howard J. Karloff, Claire Kenyon, Nick Reingold, and Mikkel Thorup. OPT versus LOAD in dynamic storage allocation. *SIAM Journal on Computing*, 33(3):632–646, 2004. doi:10.1137/S0097539703423941.
- 10 Mihai Burcea, Wing-Kai Hon, Hsiang-Hsuan Liu, Prudence W. H. Wong, and David K. Y. Yau. Scheduling for electricity cost in a smart grid. *Journal of Scheduling*, 19(6):687–699, 2016. doi:10.1007/s10951-015-0447-8.
- 11 Gruia Călinescu, Amit Chakrabarti, Howard J. Karloff, and Yuval Rabani. An improved approximation algorithm for resource allocation. *ACM Transactions on Algorithms*, 7(4):48:1–48:7, 2011. doi:10.1145/2000807.2000816.
- 12 Edward Grady. Coffman and John L. Bruno. *Computer and job-shop scheduling theory / edited by E. G. Coffman, Jr. ; coauthors, J. L. Bruno ... [et al.]*. Wiley New York, 1976.
- 13 Edward G. Coffman Jr., János Csirik, Gábor Galambos, Silvano Martello, and Daniele Vigo. *Bin Packing Approximation Algorithms: Survey and Classification*, pages 455–531. Springer New York, 2013. doi:10.1007/978-1-4419-7997-1\_35.
- 14 Edward G. Coffman Jr., M. R. Garey, David S. Johnson, and Robert Endre Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9(4):808–826, 1980. doi:10.1137/0209062.
- 15 Max A. Deppert, Klaus Jansen, Arindam Khan, Malin Rau, and Malte Tutas. Peak demand minimization via sliced strip packing. *CoRR*, abs/2105.07219, 2021. URL: <https://arxiv.org/abs/2105.07219>.
- 16 Pierre-François Dutot, Grégory Mounié, and Denis Trystram. Scheduling parallel tasks approximation algorithms. In Joseph Y.-T. Leung, editor, *Handbook of Scheduling - Algorithms, Models, and Performance Analysis*. Chapman and Hall/CRC, 2004.
- 17 Waldo Gálvez, Fabrizio Grandoni, Afrouz Jabal Ameli, and Kamyar Khodamoradi. Approximation algorithms for demand strip packing. *CoRR*, abs/2105.08577, 2021. arXiv:2105.08577.
- 18 Waldo Gálvez, Fabrizio Grandoni, Sandy Heydrich, Salvatore Ingala, Arindam Khan, and Andreas Wiese. Approximating geometric knapsack via L-packings. In *58th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 260–271. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.32.
- 19 Waldo Gálvez, Fabrizio Grandoni, Sandy Heydrich, Salvatore Ingala, Arindam Khan, and Andreas Wiese. Approximating geometric knapsack via L-packings. *CoRR*, abs/1711.07710, 2017. URL: <http://arxiv.org/abs/1711.07710>.
- 20 Waldo Gálvez, Fabrizio Grandoni, Salvatore Ingala, and Arindam Khan. Improved pseudo-polynomial-time approximation for strip packing. In *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 65,

- pages 9:1–9:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.FSTTCS.2016.9.
- 21 Fabrizio Grandoni, Salvatore Ingala, and Sumedha Uniyal. Improved approximation algorithms for unsplittable flow on a path with time windows. In *Approximation and Online Algorithms - 13th International Workshop, (WAOA)*, volume 9499, pages 13–24. Springer, 2015. doi:10.1007/978-3-319-28684-6\_2.
  - 22 Fabrizio Grandoni, Tobias Mömke, Andreas Wiese, and Hang Zhou. A  $(5/3 + \epsilon)$ -approximation for unsplittable flow on a path: placing small tasks into boxes. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 607–619. ACM, 2018. doi:10.1145/3188745.3188894.
  - 23 Rolf Harren, Klaus Jansen, Lars Prädell, and Rob van Stee. A  $(5/3 + \epsilon)$ -approximation for strip packing. *Computational Geometry*, 47(2):248–267, 2014. doi:10.1016/j.comgeo.2013.08.008.
  - 24 Rolf Harren and Rob van Stee. Improved absolute approximation ratios for two-dimensional packing problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop (APPROX)*, volume 5687, pages 177–189. Springer, 2009. doi:10.1007/978-3-642-03685-9\_14.
  - 25 Sören Henning, Klaus Jansen, Malin Rau, and Lars Schmarje. Complexity and inapproximability results for parallel task scheduling and strip packing. *Theory of Computing Systems*, 64(1):120–140, 2020. doi:10.1007/s00224-019-09910-6.
  - 26 Klaus Jansen. A  $(3/2 + \epsilon)$  approximation algorithm for scheduling moldable and non-moldable parallel tasks. In *24th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 224–235. ACM, 2012. doi:10.1145/2312005.2312048.
  - 27 Klaus Jansen and Malin Rau. Closing the gap for pseudo-polynomial strip packing. In *27th Annual European Symposium on Algorithms (ESA)*, volume 144, pages 62:1–62:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ESA.2019.62.
  - 28 Klaus Jansen and Malin Rau. Improved approximation for two dimensional strip packing with polynomial bounded width. *Theoretical Computer Science*, 789:34–49, 2019. doi:10.1016/j.tcs.2019.04.002.
  - 29 Klaus Jansen and Roberto Solis-Oba. Rectangle packing with one-dimensional resource augmentation. *Discrete Optimization*, 6(3):310–323, 2009. doi:10.1016/j.disopt.2009.04.001.
  - 30 Klaus Jansen and Ralf Thöle. Approximation algorithms for scheduling parallel jobs. *SIAM Journal on Computing*, 39(8):3571–3615, 2010. doi:10.1137/080736491.
  - 31 Mohammad M. Karbasioun, Gennady Shaikhet, Evangelos Kranakis, and Ioannis Lambadaris. Power strip packing of malleable demands in smart grid. In *Proceedings of IEEE International Conference on Communications, (ICC)*, pages 4261–4265. IEEE, 2013. doi:10.1109/ICC.2013.6655233.
  - 32 Claire Kenyon and Eric Rémila. A near-optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operations Research*, 25(4):645–656, 2000. doi:10.1287/moor.25.4.645.12118.
  - 33 Joseph Y.-T. Leung, Tommy W. Tam, C. S. Wong, Gilbert H. Young, and Francis Y. L. Chin. Packing squares into a square. *Journal of Parallel and Distributed Computing*, 10(3):271–275, 1990. doi:10.1016/0743-7315(90)90019-L.
  - 34 Fu-Hong Liu, Hsiang-Hsuan Liu, and Prudence W. H. Wong. Greedy is optimal for online restricted assignment and smart grid scheduling for unit size jobs. In *Approximation and Online Algorithms - 17th International Workshop (WAOA)*, volume 11926, pages 217–231. Springer, 2019. doi:10.1007/978-3-030-39479-0\_15.
  - 35 Fu-Hong Liu, Hsiang-Hsuan Liu, and Prudence W. H. Wong. Non-preemptive scheduling in a smart grid model and its implications on machine minimization. *Algorithmica*, 82(12):3415–3457, 2020. doi:10.1007/s00453-020-00733-3.

- 36 Tobias Mömke and Andreas Wiese. A  $(2 + \varepsilon)$ -approximation algorithm for the storage allocation problem. In *Automata, Languages, and Programming - 42nd International Colloquium (ICALP)*, volume 9134, pages 973–984. Springer, 2015. doi:10.1007/978-3-662-47672-7\_79.
- 37 Tobias Mömke and Andreas Wiese. Breaking the barrier of 2 for the storage allocation problem. In *47th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 168, pages 86:1–86:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.86.
- 38 Giorgi Nadiradze and Andreas Wiese. On approximating strip packing with a better ratio than  $3/2$ . In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1491–1510. SIAM, 2016. doi:10.1137/1.9781611974331.ch102.
- 39 Anshu Ranjan, Pramod P. Khargonekar, and Sartaj Sahni. Offline first fit scheduling in smart grids. In *2015 IEEE Symposium on Computers and Communication (ISCC)*, pages 758–763. IEEE Computer Society, 2015. doi:10.1109/ISCC.2015.7405605.
- 40 Ingo Schiermeyer. Reverse-fit: A 2-optimal algorithm for packing rectangles. In Jan van Leeuwen, editor, *Algorithms (ESA) - Second Annual European Symposium*, volume 855, pages 290–299. Springer, 1994. doi:10.1007/BFb0049416.
- 41 Daniel Dominic Sleator. A 2.5 times optimal algorithm for packing in two dimensions. *Information Processing Letters*, 10(1):37–40, 1980. doi:10.1016/0020-0190(80)90121-0.
- 42 A. Steinberg. A strip-packing algorithm with absolute performance bound 2. *SIAM Journal on Computing*, 26(2):401–409, 1997. doi:10.1137/S0097539793255801.
- 43 Shaojie Tang, Qiuyuan Huang, Xiang-Yang Li, and Dapeng Wu. Smoothing the energy consumption: Peak demand reduction in smart grid. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, pages 1133–1141. IEEE, 2013. doi:10.1109/INFCOM.2013.6566904.
- 44 Sean Yaw and Brendan Mumey. Scheduling non-preemptible jobs to minimize peak demand. *Algorithms*, 10(4):122, 2017. doi:10.3390/a10040122.
- 45 Sean Yaw, Brendan Mumey, Erin McDonald, and Jennifer Lemke. Peak demand scheduling in the smart grid. In *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 770–775. IEEE, 2014. doi:10.1109/SmartGridComm.2014.7007741.

## A Hardness of Approximation for Square-DSP

For several rectangle packing problems it is usually the case that they are NP-hard (or APX-hard) even when restricted to instances consisting solely of squares [33]. This holds also for DSP, as the following theorem shows.

► **Theorem 15.** *For any  $\varepsilon > 0$ , there exists no polynomial-time  $(3/2 - \varepsilon)$ -approximation algorithm for SQUARE-DSP unless  $NP = P$ .*

In order to prove this result, we show a gap-producing reduction from the NP-complete BALANCED PARTITION problem, formally defined as follows.

► **Definition 16** (BALANCED PARTITION). *In an instance of the BALANCED PARTITION problem, we are given a set of  $2n$  positive integers  $A = \{a_1, a_2, \dots, a_{2n}\}$ . The goal is to decide whether there exists a partitioning of  $A$  into  $A_1$  and  $A_2$  such that  $|A_1| = |A_2| = n$ , and the sum of the numbers in each of the two sets is equal to a target value  $B = \left(\sum_{j=1}^{2n} a_j\right) / 2$ .*

We start first by proving that the BALANCED PARTITION problem is NP-complete. This result is folklore by now but, for the sake of completeness, we bring a complete proof.

► **Theorem 17.** *BALANCED PARTITION is NP-complete.*

**Proof.** We will reduce the PARTITION problem to the balanced variant in polynomial time. Given an instance  $\mathcal{I}$  of the PARTITION problem with  $n$  numbers  $a_1, a_2, \dots, a_n$ , we construct an instance  $\mathcal{I}'_k$  of the BALANCED PARTITION problem for each  $k \in \{1, 2, \dots, \lfloor n/2 \rfloor\}$ . Let  $C$  be  $(\sum_{j=1}^n a_j) + 1$ . For each  $k$ , the instance  $\mathcal{I}'_k$  is defined as follows. Let  $\mathcal{I}'_k$  have all the initial numbers,  $a_1, a_2, \dots, a_n$ . Add the set **dummy** of  $n - 2k + 2$  extra numbers where **dummy** =  $\{\alpha, \beta_1, \beta_2, \dots, \beta_{n-2k+1}\}$  in which  $\alpha = (n - 2k + 1)C$  and  $\beta_i = C$  for each  $i \in [n - k + 1]$ . We claim that  $\mathcal{I}$  is a *Yes* instance of the PARTITION problem if and only if at least one  $\mathcal{I}'_k$  is a *Yes* instance of the BALANCED PARTITION problem.

**Completeness.** Assume that  $\mathcal{I}$  is a *Yes* instance of the PARTITION problem. Let  $S_1$  and  $S_2$  be the two sets of equal sum, say  $B$ . These sets may not necessarily have the same cardinality. With sum renumbering, assume that  $S_1 = \{a_1, a_2, \dots, a_k\}$  and  $S_2 = \{a_{k+1}, a_{k+2}, \dots, a_n\}$  for some  $k \in [\lfloor n/2 \rfloor]$ . It is easy to see that the instance  $\mathcal{I}'_k$  of the BALANCED PARTITION problem is a *Yes* instance, since we can make the sets  $S'_1 = \{a_1, a_2, \dots, a_k, \beta_1, \beta_2, \dots, \beta_{n-2k+1}\}$  and  $S'_2 = \{a_{k+1}, a_{k+2}, \dots, a_n, \alpha\}$  both with the sum  $B + (n - 2k + 1)C$  and cardinality  $n - k + 1$ .

**Soundness.** Now assume that  $\mathcal{I}$  is a *No* instance of the PARTITION problem. We claim that no  $\mathcal{I}'_k$  can be a *Yes* instance of the BALANCED PARTITION problem either. For the sake of contradiction, assume  $\mathcal{I}'_k$  is a *Yes* instance with two partitions  $S'_1$  and  $S'_2$  of the same sum and cardinality. Note that if no  $\beta_i$  is placed in the same set as  $\alpha$ , we reach a contradiction since we then can find two sets  $S_1 = \{a_1, a_2, \dots, a_k\}$  and  $S_2 = \{a_{k+1}, a_{k+2}, \dots, a_n\}$  of the same sum of the original PARTITION instance  $\mathcal{I}$ . So with some renumbering, we can assume we have  $S'_1 = \{a_1, a_2, \dots, a_{k'}, \alpha, \beta_1, \beta_2, \dots, \beta_\ell\}$  and  $S'_2 = \{a_{k'+1}, a_{k'+2}, \dots, a_n, \beta_{\ell+1}, \beta_{\ell+2}, \dots, \beta_{n-2k+1}\}$  for some  $k'$  and  $\ell$  in which:

$$\sum_{j=1}^{k'} a_j + (n - 2k + 1)C + \ell \cdot C = \sum_{j=k'+1}^n a'_j + (n - 2k + 1 - \ell)C.$$

This implies that

$$2\ell \cdot C = \sum_{j=k'+1}^n a'_j - \sum_{j=1}^{k'} a_j \leq \sum_{j=k'+1}^n a'_j < C,$$

which is a contradiction. ◀

**Proof of Theorem 15.** Assume an instance  $\mathcal{I}$  of the BALANCED PARTITION problem is given. Based in this instance, we define an instance  $\mathcal{I}'$  of SQUARE-DSP. Let  $a_{max}$  denote the maximum value among the integers in  $A$ . Define  $C$  as  $1/\varepsilon \cdot \sum_{j=1}^{2n} a_j$ , where  $\varepsilon$  is chosen such that  $1/\varepsilon$  is a large but constant integer. Note that  $C > 1/\varepsilon \cdot a_{max}$ . Let  $\mathcal{I}'$  have  $2n$  tasks, where each task  $i$  has width and height  $C + a_i$  for  $i \in [2n]$ . Our goal is to schedule the  $2n$  tasks into a path of  $W = n \cdot C + B$  edges while minimizing the peak. Based on the hardness of the BALANCED PARTITION problem, we show that it is hard to distinguish between the case where an schedule with peak  $2C(1 + \varepsilon)$  exists and the case where the minimum peak is larger than  $3(C + 1) - \varepsilon$ .

**Completeness.** Assume that  $\mathcal{I}$  is a *Yes* instance, meaning that a partitioning  $A = A_1 \dot{\cup} A_2$  exists that satisfies the cardinality and sum constraints. Define two shelves of squares,  $S_i = \{j | a_j \in A_i\}$  for  $i = 1, 2$ , and schedule them starting at the leftmost edge. The width of each shelf is equal to  $n \cdot C + B$  and the peak is at most  $2 \cdot (C + a_{max}) < 2C(1 + \varepsilon)$ .



**Soundness.** Now, consider a No instance  $\mathcal{I}$ . We claim that, for the corresponding SQUARE-DSP instance, no schedule with peak smaller than or equal to  $3(C+1) - \varepsilon$  exists. For the sake of contradiction, assume that it is the case. Since the size of each task is at least  $C+1$ , it means that in the optimal solution for  $\mathcal{I}'$ , no three tasks use the same edge. Also, the total width of the tasks is  $2W = 2(nC + b)$ , so no edge can have less than two tasks. This allows us to split the tasks  $S$  into two sets  $S_1$  and  $S_2$ . We start at the leftmost edge and pick one of the two tasks placed on this edge arbitrarily and put in  $S_1$ . Since every edge has exactly two tasks, immediately to the right of this task at least one another task must start. We put it in  $S_1$  as well and proceed until we reach the rightmost edge, breaking ties arbitrarily along the way. We set  $S_2 = S \setminus S_1$ . It remains to show that each set has exactly  $n$  tasks. Assume otherwise; let  $S_1$  be composed of tasks  $s_1, \dots, s_{n+k}$ , and  $S_2$  be the tasks  $s_{n+k+1}, \dots, s_{2n}$  for some  $k$ ,  $1 \leq k \leq n$ . Since the tasks are placed one next to the other in each shelf, we have that  $(n+k)C + \sum_{j=1}^{n+k} a_j = (n-k)C + \sum_{j'=n+k+1}^{2n} a_{j'}$ . Therefore  $\sum_{j'=n+k+1}^{2n} a_{j'} \geq 2k \cdot C$ , which is a contradiction for any value of  $k > 0$  by our choice of  $C$ .

As a result, assuming that  $\text{NP} \neq \text{P}$ , no polynomial-time algorithm can approximate the SQUARE-DSP problem within a factor of  $\frac{3(C+1)-\varepsilon}{2C(1+\varepsilon)} = 3/2 - \varepsilon'$ , for some  $\varepsilon' = O(\varepsilon)$ . ◀

## B A PTAS for DSP with short tasks

In this section we will prove Theorem 2 restated below.

► **Theorem 2.** *Given  $\varepsilon > 0$  small enough, there exists  $\delta > 0$  and a polynomial time algorithm such that, given an instance of DSP with optimal value  $OPT$  and consisting solely of tasks having height at most  $\delta \cdot OPT$ , it computes a  $(1 + O(\varepsilon))$ -approximate solution.*

Before proving the result in detail we provide a couple of required technical lemmas regarding the computation of  $\pi$ -left-pushing of a given schedule  $P(\cdot)$ . First of all, we prove that such a solution can be indeed computed efficiently.

► **Lemma 18.** *Given a feasible schedule  $P(\cdot)$  with peak  $\pi$  for an instance  $\mathcal{I}$ , one can compute a  $\pi'$ -left-pushing of  $P(\cdot)$ , with  $\pi' \geq \pi$ , in polynomial time.*

**Proof.** Let  $1, \dots, n$  be the tasks sorted according to their starting edge in  $P(\cdot)$  from left to right. Let  $S_i$  be the starting edge of task  $i$ . First, inductively, we compute a  $\pi'$ -left-pushing of  $\mathcal{I} \setminus \{n\}$  and do not left-shift task  $n$ . Since we only left-shifted the tasks, the demand on the edges from  $S_n$  to  $e_W$  cannot increase. Thus, we reach a feasible solution such that its peak does not exceed  $\pi'$ . Now we compute the starting time of task  $n$ ,  $s^*$ , if we left-shift this task as much as possible. Note that  $s^*$  can only be either the leftmost edge or some edge  $e$  such that some previous task finishes next to the left of  $e$ , as otherwise at least one more unit of left-shifting is possible for task  $n$ . Now, using this fact, we have at most  $n$  possibilities for  $s^*$  and we can compute this value in polynomial time. Note that if we call the obtained schedule as  $P'(\cdot)$ , then  $P'(\cdot)$  is indeed a  $\pi'$ -left-pushing of  $P(\cdot)$ . ◀

The following lemma summarizes the useful properties we can get when computing a left-pushing.

► **Lemma 19.** *Given a feasible schedule  $P(\cdot)$  with peak  $\pi$  for an instance  $\mathcal{I}$ , the  $\pi'$ -left-pushing of  $P(\cdot)$  for  $\pi' \geq \pi$ , let us say  $P'(\cdot)$ , satisfies the following properties:*

1. *There exists a node  $t^*$  such that  $P'(\cdot)$  is  $(\pi' - h_{\max}(\mathcal{I}), t^*)$ -sorted, and*
2. *every  $i \in \mathcal{I}$  has a starting edge in  $\mathcal{E}'$  of the form  $\sum_{j \in \mathcal{I}'} w(j)$  for some  $\mathcal{I}' \subseteq \mathcal{I} \setminus \{i\}$  (0 if  $\mathcal{I}'$  is empty).*

**Proof.** We now show a proof of the two properties:

1. Suppose that there exists a node  $k$  such that the demand on the edge to the left of  $k$  is smaller than  $\pi' - h_{\max}(\mathcal{I})$  and the demand on the edge to the right of  $k$  is larger than  $\pi' - h_{\max}(\mathcal{I})$ . This implies that some task starts at the edge to the right of  $k$ , but then it is possible to left-shift this task without surpassing the threshold of  $\pi'$  which is a contradiction. At this point we know that there exists  $k'$  such that every edge to the left of  $k'$  has demand larger than  $\pi' - h_{\max}(\mathcal{I})$ , and let  $k^*$  be the rightmost such node. Similarly to the previous case, if after  $k^*$  there exists a node  $k$  such that the load to the left of  $k$  is smaller than the demand to the right of  $k$ , then again there must exist a task starting to the right of  $k$  and, since their demands are at most  $\pi' - h_{\max}(\mathcal{I})$ , left-shifting such task does not violate the threshold of  $\pi'$  which is a contradiction.

2. Suppose there exists a task not satisfying the claim, and let  $i$  be the leftmost such task in  $P'(\cdot)$ . It is easy to see that  $i$  cannot start at the leftmost edge and also that the demand on the edge just to the left of  $P'(i)$  is larger than  $\pi' - h(i)$  as otherwise a left-shifting of  $i$  is possible. Due to  $i$  being the leftmost task, no task  $i'$  can finish just to the left of  $P'(i)$ , as otherwise the number of edges before  $P'(i)$  would be the sum of some widths in  $\mathcal{I}$  plus  $w(i')$ , thus fulfilling the claim for  $i$ . This implies that every task using the edge just to the left of  $P'(i)$  must also use edge  $P'(i)$ . But then the total demand just to the left of  $P'(i)$  would be at most the total demand on  $P'(i)$  minus  $h(i)$ , which is at most  $\pi' - h(i)$ . ◀

We can now proceed with the proof of Lemma 2, where at some point in the proof we will make use of the following concentration bound which was proved in [11].

► **Lemma 20** (Calinescu et al. [11]). *Let  $X_1, X_2, \dots, X_n$  be independent random variables and let  $0 \leq \beta_1, \beta_2, \dots, \beta_n \leq 1$  be real numbers, where for each  $i = 1, 2, \dots, n$ ,  $X_i = \beta_i$  with probability  $p_i$  and  $X_i = 0$  otherwise. Let  $X = \sum_{i=1}^n X_i$  and  $\mu = \mathbb{E}[X]$ . Then*

1. *The variance of  $X$ ,  $\sigma^2(X)$ , is at most  $\mu$ , and*
2. *For any  $0 < \lambda < \sqrt{\mu}$ ,  $\mathbb{P}[X > \mu + \lambda\sqrt{\mu}] < e^{-\frac{\lambda^2}{2}(1-\lambda/\sqrt{\mu})}$ .*

**Proof of Theorem 2.** Let  $\delta > 0$  be a constant that we will specify later. We will partition the tasks into two sets according to their widths: we will say that a task  $i$  is *horizontal* if  $w(i) > \delta \cdot W$  and otherwise we will say it is *narrow*. Consider by now only the horizontal tasks in  $\mathcal{I}$ , and assume that the value  $OPT$  is known. Thanks to Lemma 19, by computing an  $OPT$ -left-pushing of the optimal solution, we know there exists a set  $\mathcal{E}_H \subseteq E$  that can be computed in polynomial time such that the starting edge of every task belongs to  $\mathcal{E}_H$ . Indeed, edges in  $\mathcal{E}_H$  correspond to the sum of widths of some horizontal tasks, implying that the number of widths in the sum must be at most  $\frac{1}{\delta}$ . Hence, all the possible starting edges are of the form  $\sum_{i \in \mathcal{I}'} w(i)$  where  $|\mathcal{I}'| \leq \frac{1}{\delta}$ . The set  $\mathcal{E}_H$  consisting of these edges has size at most  $n^{1/\varepsilon-1}$  and can clearly be computed in polynomial time.

With the following integer program we can compute a feasible solution corresponding to a  $OPT$ -left-pushing of some scheduling for these tasks. We define a variable  $x_{i,k}$  for each task  $i$  and starting edge  $k$  in the previously computed set  $\mathcal{E}_H$  (if task  $i$  cannot be scheduled starting at edge  $k$  this variable is not considered):

$$\begin{array}{ll}
 \min & \lambda \\
 \text{s.t.} & \sum_{k \in \mathcal{E}_H} x_{i,k} = 1 \quad \forall i \text{ horizontal} \\
 & \sum_{i \text{ hor.}} \sum_{k' \in \mathcal{E}_H(i,q)} h(i) \cdot x_{i,k'} \leq OPT \quad \forall q \in \mathcal{E}_H \\
 & x_{i,k} \in \{0, 1\} \quad \forall i \text{ horizontal, } k \in \mathcal{E}_H,
 \end{array}$$

where, given  $i \in \mathcal{I}$  and  $q \in \{1, \dots, W\}$ ,  $\mathcal{E}_H(i, e)$  is the set of edges  $k \in \mathcal{E}_H$  such that, if  $i$  has  $k$  as starting edge, then it uses edge  $e$ . In other words, the second family of constraints is ensuring that the total demand of the constructed solution is at most  $OPT$  in every edge (which can be done with polynomially many constraints thanks to the size of  $\mathcal{E}_H$ ).

We will consider the canonical linear relaxation of the formulation, and let  $\vec{x}$  be an optimal solution to this LP (which can be computed in polynomial time). In order to derive a feasible solution we will use *Randomized Rounding with Alterations*, a technique previously used in similar settings for Packing and Scheduling problems [11, 36, 2]. In a first stage, for each task  $i$ , we will sample one starting edge  $k$  according to the probability distribution induced by  $\{x_{i,k}\}_{k \in \mathcal{E}_H}$ . Now, in a second stage, we scan the starting edges  $k$  from left to right, and the sampled tasks  $i$  starting at node  $k$  according to the sample in any order, and we add  $i$  to the current solution as long as the obtained peak is no more than  $(1 + \varepsilon)OPT$ . Observe that this is a dependent rounding where each task  $i$  is finally scheduled in the solution with marginal probability at most  $x_{i,k}$ .

Suppose we are applying the previous procedure, and let  $k$  be a fixed edge in that order. Let  $\tilde{X}_{i,k} \in \{0, 1\}$  be equal to 1 if and only if  $i$  is scheduled starting at edge  $k$  in the first stage, and similarly we define  $\tilde{Y}_{i,k}$  to be 1 if and only if  $i$  is scheduled starting at edge  $k$  in the second stage. Notice that  $\tilde{Y}_{i,k} \leq \tilde{X}_{i,k}$  deterministically. By stochastic domination, we have that

$$\mathbb{P} \left[ \sum_{i \text{ hor.}} \sum_{k \in \mathcal{E}_H(i, q)} \tilde{Y}_{i,k} \cdot h(i) > (1 + \varepsilon)OPT \right] \leq \mathbb{P} \left[ \sum_{i \text{ hor.}} \sum_{k \in \mathcal{E}_H(i, q)} \tilde{X}_{i,k} \cdot h(i) > (1 + \varepsilon)OPT \right].$$

To upper bound the latter quantity we will consider two cases:

- If  $\mu \leq \frac{3}{4\delta}$ , then we can use Chebyshev's inequality for the variable  $Z := \sum_{i \text{ hor.}} \frac{\tilde{X}_{i,k} \cdot h(i)}{\delta OPT}$  (notice that thanks to Lemma 20 it holds that  $\sigma(Z) \leq \sqrt{\mu}$ ), from where we obtain that

$$\begin{aligned} \mathbb{P} \left[ \sum_{i \text{ hor.}} \sum_{k \in \mathcal{E}_H(i, q)} \tilde{X}_{i,k} \cdot h(i) > (1 + \varepsilon)OPT \right] &= \mathbb{P} \left[ Z > \frac{1 + \varepsilon}{\delta} \right] \\ &\leq \mathbb{P} \left[ |Z - \mu| > \left( \frac{1 + \varepsilon}{\delta} - \frac{3}{4\delta} \right) \cdot \frac{\sigma(Z)}{\sqrt{\mu}} \right] \\ &\leq \frac{16\mu\delta^2}{(1 + 4\varepsilon)^2} \leq \varepsilon \end{aligned}$$

for  $\delta \leq \frac{\varepsilon}{4}$ .

- If  $\mu > \frac{3}{4\delta}$ , we first set  $\lambda = \frac{1 + \varepsilon - \mu\delta}{\delta\sqrt{\mu}}$  so that  $\mu + \lambda\sqrt{\mu} = \frac{1 + \varepsilon}{\delta}$ . Notice that  $\mu = \sum_{i \text{ hor.}} \frac{x_{i,k} \cdot h(i)}{\delta OPT} \leq \frac{1}{\delta}$  due to the constraints in the LP.

Now, it is not difficult to see that  $\lambda$  is decreasing as a function of  $\mu$ , implying that  $\lambda \geq \frac{1 + 4\varepsilon}{\sqrt{12\delta}}$ . Furthermore, we have that  $1 - \frac{\lambda}{\sqrt{\mu}} = 2 - \frac{1 + \varepsilon}{\delta\mu} \geq \frac{2}{3}$ , and thus also  $\lambda < \sqrt{\mu}$ . Now we can use Lemma 20 applied to the variables  $\{X_{i,k}h(i)/(\delta OPT)\}_{i \text{ hor.}}$  and their sum  $Z$  and obtain

$$\begin{aligned} \mathbb{P} \left[ \sum_{i \text{ hor.}} \sum_{k \in \mathcal{E}_H(i, q)} \tilde{X}_{i,k} \cdot h(i) > (1 + \varepsilon)OPT \right] &= \mathbb{P} [Z > \mu + \lambda\sqrt{\mu}] \\ &< e^{-\frac{\lambda^2}{2}(1 - \lambda/\sqrt{\mu})} \\ &< e^{-\frac{2}{9} \frac{(1 + 4\varepsilon)^2}{12\delta^2}} \leq \varepsilon \end{aligned}$$

for  $\delta \leq \frac{(1 + 4\varepsilon)^2}{54} \ln \frac{1}{\varepsilon}$ .

## 20:24 Approximation Algorithms for Demand Strip Packing

This implies that we get a solution with peak at most  $(1 + \varepsilon)OPT$  and the probability that a task is not scheduled is at most  $\varepsilon$ . As a consequence, in expectation the total area of tasks that were not placed is at most  $\varepsilon W \cdot OPT$ , and hence using Markov's inequality we get that the probability that these tasks have area larger than  $2\varepsilon W \cdot OPT$  is at most  $\frac{1}{2}$ . Thus, if the area of these tasks is at most  $2\varepsilon W \cdot OPT$  and since their heights are at most  $\delta \cdot OPT$ , we can place them into a rectangular region of height  $4\varepsilon OPT$  and width  $W$  using Corollary 7. If the area guarantee is not satisfied then we repeat the whole process to ensure it as, in expectation, a constant number of times only is required.

Now we will include the set  $\mathcal{N}$  of narrow tasks into the solution by applying Lemma 12 with parameter  $\pi = (1 + 5\varepsilon)OPT$ . Consider a  $\pi$ -left-pushing of the solution. Thanks to Lemma 19, there exists a node  $t^*$  such that the obtained schedule is  $(\pi, t^*)$ -sorted. Furthermore, it is not difficult to see that  $\pi \geq (1 + 4\varepsilon)\frac{a(\mathcal{I})}{W} + h_{\max}(\mathcal{I}'')$  and  $w_{\max}(\mathcal{N}) \leq \varepsilon W \leq \frac{4\varepsilon}{2(4\varepsilon+1)}$  for  $\varepsilon \leq 1/4$ , hence satisfying the requirements of the lemma. This way, we obtain a feasible scheduling with peak at most  $(1 + 5\varepsilon)OPT$ .

Finally, in order to avoid knowing the value of  $OPT$ , we can approximately guess it using any constant approximation (such as Corollary 7) and define a constant number of candidates. ◀

# Peak Demand Minimization via Sliced Strip Packing

Max A. Deppert  

Universität Kiel, Germany

Klaus Jansen  

Universität Kiel, Germany

Arindam Khan  

Department of Computer Science and Automation, Indian Institute of Science, Bengaluru, India

Malin Rau  

Universität Hamburg, Germany

Malte Tutas  

Universität Kiel, Germany

---

## Abstract

---

We study the Nonpreemptive Peak Demand Minimization (NPDM) problem, where we are given a set of jobs, specified by their processing times and energy requirements. The goal is to schedule all jobs within a fixed time period such that the peak load (the maximum total energy requirement at any time) is minimized. This problem has recently received significant attention due to its relevance in smart-grids. Theoretically, the problem is related to the classical strip packing problem (SP). In SP, a given set of axis-aligned rectangles must be packed into a fixed-width strip, such that the height of the strip is minimized. NPDM can be modeled as strip packing with slicing and stacking constraint: each rectangle may be cut vertically into multiple slices and the slices may be packed into the strip as individual pieces. The stacking constraint forbids solutions where two slices of the same rectangle are intersected by the same vertical line. Nonpreemption enforces the slices to be placed in contiguous horizontal locations (but may be placed at different vertical locations).

We obtain a  $(5/3 + \epsilon)$ -approximation algorithm for the problem. We also provide an asymptotic efficient polynomial-time approximation scheme (AEPTAS) which generates a schedule for almost all jobs with energy consumption  $(1 + \epsilon)\text{OPT}$ . The remaining jobs fit into a thin container of height 1. The previous best result for NPDM was a 2.7 approximation based on FFDH [41]. One of our key ideas is providing several new lower bounds on the optimal solution of a geometric packing, which could be useful in other related problems. These lower bounds help us to obtain approximative solutions based on Steinberg's algorithm in many cases. In addition, we show how to split schedules generated by the AEPTAS into few segments and to rearrange the corresponding jobs to insert the thin container mentioned above.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Approximation algorithms analysis

**Keywords and phrases** scheduling, peak demand minimization, approximation

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.21

**Category** APPROX

**Related Version** *Full Version:* <https://arxiv.org/abs/2105.07219>

**Funding** *Max A. Deppert:* Research supported by German Research Foundation (DFG) project JA 612/25-1.

*Klaus Jansen:* Research supported by German Research Foundation (DFG) project JA 612/25-1.

*Malin Rau:* Research supported by the French research program ENERGUMEN ANR-18-CE25-0008.



© Max A. Deppert, Klaus Jansen, Arindam Khan, Malin Rau, and Malte Tutas;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 21; pp. 21:1–21:24



Leibniz International Proceedings in Informatics

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

## 1 Introduction

Recent years have seen a substantial increase in the demand of electricity, due to rapid urbanization, economic growth and new modes of electrical energy consumption (e.g., electric cars). Traditionally, electricity generation, transmission, and distribution relied on building infrastructure to support the peak load, when the demand for electricity is maximum. However, the peak is rarely achieved and thus more demands can be accommodated using the inherent flexibility of scheduling of certain jobs. E.g., the energy requirements for HVAC units, electric vehicles, washers and dryers, water heaters, etc. can be met with a flexible scheduling of these appliances. Smart-grids [48, 32, 45] are next-generation cyber-physical systems that couple digital communication systems on top of the existing grid infrastructure for such efficient utilization of power, e.g., by shifting users' demand to off-peak hours in order to reduce peak load.

Future smart-grids are expected to obtain demand requirements for a time period and schedule the jobs such that the peak demand is minimized. Recently, this problem has received considerable attention [3, 37, 40, 42, 12]. Each job can also be modeled as a rectangle, with desired power demand as height and required running time as width. This gives a geometric optimization problem where the goal is to pack the slices of the rectangles into a strip of width as the time period. The goal is to minimize the maximum height of the packing. There is another additional *stacking constraint* requiring that no vertical line may intersect two slices from the same rectangle.

In this paper, we study this problem known as Nonpreemptive Peak Demand Minimization (NPDM). Formally, we are given a set of jobs  $\mathcal{J}$ . Each job  $j \in \mathcal{J}$  has a processing time  $p(j) \in \mathbb{N}$  (also called width) and an energy requirement  $e(j) \in \mathbb{N}$  (also called height). Furthermore, we are given a deadline  $D \in \mathbb{N}$ . All the jobs are available from the time 0 and have to be finished before the deadline  $D$ . A schedule  $\sigma$  of the jobs  $\mathcal{J}$  assigns each job a starting time  $\sigma(j) \in \mathbb{N}$  such that it is finished before the deadline, i.e.,  $c(j) := \sigma(j) + p(j) \leq D$ . The total energy consumption at a time  $\tau \in \{0, \dots, D\}$  is given by  $e(\tau) := \sum_{j \in \mathcal{J}, \sigma(j) \leq \tau < \sigma(j) + p(j)} e(j)$ . The objective is to minimize the peak of energy consumption, i.e., minimize  $T_\sigma := \max_{\tau \in \{0, \dots, D-1\}} e(\tau)$ .

NPDM can be viewed as a variant of strip packing problem, where we are allowed to slice the rectangles vertically and the slices must be packed in contiguous horizontal positions (but may be placed at different vertical positions). In the classical strip packing problem, we are given a set of rectangles as well as a bounded-width strip and the objective is to find a non-overlapping, axis-aligned packing of all rectangles into the strip so as to minimize the height of the packing. A simple reduction from the PARTITION problem shows a lower bound of  $3/2$  for polynomial-time approximation for the problem. In 1980, Baker et al. [4] first gave a 3-approximation algorithm. Later Coffman et al. [31] introduced two simple shelf-based algorithms: Next Fit Decreasing Height (NFDH), First Fit Decreasing Height (FFDH), with approximation ratios as 3 and 2.7, respectively. Sleator [46] gave a 2.5-approximation. Thereafter, Steinberg [47] and Schiermeyer [44] independently improved the approximation ratio to 2. Afterwards, Harren and van Stee [21] obtained a 1.936-approximation. The present best approximation is  $(5/3 + \varepsilon)$ , due to Harren et al. [20].

Alamdari et al. [3] studied a variant where we allow preemption of jobs, also known as two-dimensional strip packing with slicing and stacking constraints (2SP-SSC), or preemptive offline cost optimal scheduling problem (P-OCOSP) [41]. They showed this variant to be NP-hard and obtained an FPTAS. They also studied several shelf-based algorithms and provide a practical polynomial time algorithm that allows only one preemption per job. Ranjan et al. [42] have proposed a practical  $4/3$ -approximation algorithm for this problem.

For NPDM, Tang et al. [48] first proposed a 7-approximation algorithm. Yaw et al. [49] showed that NPDM is NP-hard to approximate within a factor better than  $3/2$ . They have given a 4-approximation for a special case when all jobs require the same execution time. Ranjan et al. [40], have proposed a 3-approximation algorithms for NPDM. They [41] also proposed an FFDH-based 2.7-approximation algorithm for a mixed variant where some jobs can be preempted and some can not be preempted.

**Our Contributions.** We obtain improved approximation algorithms for NPDM.<sup>1</sup> Note that the optimal solutions of sliced strip packing/NPDM and strip packing can be quite different. In fact, in [9] an example with a ratio  $5/4$  is presented. Thus, the techniques from strip packing do not always translate directly to our problem. We exploit the property that, due to slicing, we can separately guess regions (*profile*) for packing of jobs with large energy demand (*tall* jobs) and jobs with large time requirements (*wide* jobs). We show that we can remove a small amount of jobs with large energy demand so that we can approximately guess the optimal profile of jobs with large processing time so that their starting positions come from a set containing a constant number of values. This helps us to show the existence of a structured solution that we can pack near-optimally using linear programs. This shows the existence of an asymptotic efficient polynomial-time approximation scheme (AEPTAS):

► **Theorem 1.** *For any  $\varepsilon > 0$ , there is an algorithm that schedules all jobs such that the peak load is bounded by  $(1 + \varepsilon)\text{OPT} + e_{\max}$ , where  $e_{\max}$  denotes the maximal energy demand among the given jobs. The time complexity of this algorithm is bounded by  $\mathcal{O}(n \log(n)) + 1/\varepsilon^{1/\varepsilon^{\mathcal{O}(1/\varepsilon)}}$ .*

In fact, we show a slightly stronger result here, providing a schedule for almost all jobs  $\mathcal{J} \setminus \mathcal{C}$  with peak energy demand bounded by  $(1 + \varepsilon)\text{OPT}$  plus a schedule for the remaining jobs  $\mathcal{C}$  with peak energy demand  $e_{\max}$  and schedule length  $\lambda D$  for a sufficiently small  $\lambda \in [0, 1]$ .

Using the AEPTAS and Steinberg’s algorithm[47], we obtain our main result:

► **Theorem 2.** *For any  $\varepsilon > 0$ , there is a polynomial-time  $(5/3 + \varepsilon)$ -approximation algorithm for NPDM.*

Previously, in strip packing (and related problems) the lower bound on the optimal packing height is given based on the height of the tallest job or the total area of all jobs [47, 44]. One of our main technical contributions is to show several additional lower bounds on the optimal load. These bounds may be helpful in other related geometric problems. In fact, these can be helpful to simplify some of the analyses of previous algorithms. Using these lower bounds, we show, intuitively, that if there is a large amount of energy consuming jobs (or time consuming jobs) we can obtain a good packing using Steinberg’s algorithm. Otherwise, we start with the packing from AEPTAS and modify the packing to obtain a packing within  $(5/3 + \varepsilon)$ -factor of the optimal. This repacking utilizes novel insights about the structure of the packing that precedes it, leading to a less granular approach when repacking.

**Related Work.** Strip packing has also been studied under asymptotic approximation. The seminal work of Kenyon and Rémila [34] provided an APTAS with an additive term  $\mathcal{O}(e_{\max}/\varepsilon^2)$ , where  $e_{\max}$  is the height of the tallest rectangle. The latter additive term was subsequently improved to  $e_{\max}$  by Jansen and Solis-Oba [28]. Pseudo-polynomial time algorithm for strip packing has received recent attention [39, 18, 2, 22]. Finally, Jansen and

<sup>1</sup> The same result as in Theorem 1 was achieved independently in [15]; their approach is however substantially different from ours.

Rau [27] gave an almost tight pseudo-polynomial time  $(5/4 + \varepsilon)$ -approximation algorithm. Recently, Galvez et al. [14] gave a tight  $(3/2 + \varepsilon)$ -approximation algorithm for a special case when all rectangles are skewed (each has either width or height  $\leq \delta D$ , where  $\delta \in (0, 1]$  is a small constant).

A related problem is non-contiguous multiple organization packing [10], where the width of each rectangle represents a demand for a number of concurrent processors. This is similar to sliced strip packing, however, the slices need to be horizontally aligned to satisfy concurrency. Several important scheduling problems are related, such as multiple strip packing [27], malleable task scheduling [23], parallel task scheduling [29], moldable task scheduling [24, 25], etc.

Several geometric packing problems are well-studied in combinatorial optimization. In two-dimensional bin packing, we are given a set of rectangles and the goal is to pack all rectangles into the minimum number of unit square bins. This well-studied problem [5, 26] is known to admit no APTAS [6], unless  $P=NP$ , and the present best approximation ratio is 1.406 [7]. Another related problem is two-dimensional geometric knapsack [30, 17], where each rectangle has an associated profit and we wish to pack a maximum profit subset of rectangles in a given square knapsack. The present best approximation ratio for the problem is 1.89 [16]. These problems are also studied under guillotine cuts [8, 36, 35] where all jobs can be cut out by a recursive sequence of end-to-end cuts. There are several other important related problems such as maximum independent set of rectangles [1], unsplittable flow on a path [19], storage allocation problem [38], etc. We refer the readers to [13] for a survey.

The following result from [47] will be a crucial subroutine in our algorithms.

► **Theorem 3** (Steinberg's Algorithm [47]). *Steinberg's algorithm packs a set of rectangular objects  $\mathcal{R}$  into a rectangular container of height  $a$  and width  $b$  in polynomial time, if and only if the following inequalities hold:*

$$e_{\max} \leq a, \quad p_{\max} \leq b, \quad 2 \sum_{r \in \mathcal{R}} e(r)p(r) \leq ab - (2e_{\max} - a)_+(2p_{\max} - b)_+, \quad (\text{Steinberg Cond.})$$

where  $x_+ = \max\{x, 0\}$ ,  $p_{\max}$  is the maximal width of a rectangle, and  $e_{\max}$  is the maximal height of a rectangle,  $e(r)$  represents the height of a rectangle and  $p(r)$  represents the width of a rectangle.

**General Approach.** The general idea of our  $(5/3 + \varepsilon)$ -approximation algorithm is as follows: If the jobs that are large in at least one of the two dimensions have a sufficiently large total amount of work, a  $(5/3 + \varepsilon)$ -approximation can be achieved by placing these jobs in a structured manner and using Steinberg's algorithms to place the residual jobs. We describe two of these cases in Section 2.

Otherwise, we know that the total amount of work of these jobs not too large. In this case, we find a schedule  $\sigma_1$  that schedules almost all the jobs using an energy demand of at most  $(1 + \mathcal{O}(\gamma))\text{OPT}$ . The residual jobs are contained in an extra schedule  $\sigma_2$  of length  $\gamma D$  and peak energy demand bounded by  $T$ . These schedules are generated using the algorithm described Section 4 in the proof of Theorem 10. Note that we have to choose  $\gamma \in \mathcal{O}_\varepsilon(1)$  small enough, in order to meet the requirements for the next step.

Given these schedules, we find a rescheduling argument, where we rearrange the schedule  $\sigma_1$  such that we can add the schedule  $\sigma_2$  while increasing the peak energy demand by at most  $(2/3)T$ . This repacking argument is described in Section 3 in the proof of Theorem 7.



**Notation.** For an instance  $I = (\mathcal{J}, D)$ , we denote by  $\text{OPT}(I)$  (or just  $\text{OPT}$ ) the optimal energy consumption peak. For some set of jobs  $\mathcal{J}$  we define  $\text{work}(\mathcal{J}) = \sum_{i \in \mathcal{J}} p(i)e(i)$ , the total processing time as  $p(\mathcal{J}) = \sum_{i \in \mathcal{J}} p(i)$ , as well as the total energy demand as  $e(\mathcal{J}) = \sum_{i \in \mathcal{J}} e(i)$ . With the additional notation of  $\mathcal{J}_{P(i)} = \{i \in \mathcal{J} \mid P(i)\}$  as a restriction of  $\mathcal{J}$  using the predicate  $P$ . E.g., we may express the energy demand of jobs of  $\mathcal{J}$  which have a processing time of at least  $D/2$  by  $e(\mathcal{J}_{p(i) \geq D/2})$ . Furthermore, given a set of jobs  $\mathcal{J}$ , we denote  $p_{\max}(\mathcal{J}) := \max_{i \in \mathcal{J}} p(i)$  and  $e_{\max}(\mathcal{J}) := \max_{i \in \mathcal{J}} e(i)$  and write  $e_{\max}$  and  $p_{\max}$  if the set of jobs is clear from the context. We say a job  $i$  that is placed at  $\sigma(i)$  overlaps a point in time  $\tau$  if and only if  $\sigma(i) \leq \tau < \sigma(i) + p(i)$ . The set  $\mathcal{J}(\tau)$  denotes the set of jobs that overlap the point in time  $\tau$ . Additionally, we introduce segments  $S$  of the schedule which refer to time intervals and container  $C$  which can be seen as sub schedules. The starting point of a time interval  $S$  will be denoted by  $\sigma(S)$  and its endpoint as  $c(S)$ . On the other hand, a container  $C$  has a length (time), which is denoted as  $p(C)$ , and a bound on the energy demand  $e(C)$ . If these containers are scheduled, they get a start point  $\sigma(C)$ , which is added to the start point of any job scheduled in  $C$ .

## 2 Cases solved with Steinberg's algorithm

In this section, we first bound the peak energy demand from below and then use Steinberg's algorithm to handle some cases. Two obvious lower bounds are the energy demand of the most energy demanding job  $e_{\max}$  and the bound given by the total amount of work of the jobs, i.e.,  $\text{OPT} \geq \max\{e_{\max}, \text{work}(\mathcal{J})/D\}$ . Another simple lower bound is the total energy demand of jobs longer than  $D/2$ , since they have to be scheduled in parallel. This gives us the first lower bound on  $\text{OPT}$  and we call it  $T_1 := \max\{e_{\max}, \text{work}(\mathcal{J})/D, e(\mathcal{J}_{p(i) > D/2})\}$ . In the following, we will present three more complex lower bounds. The next bound is related to the items with a large energy demand. We denote this lower bound as

$$T_2 := \min\{T \mid p(\mathcal{J}_{e(i) \geq T/3}) + p(\mathcal{J}_{e(i) \geq 2T/3}) \leq 2D \wedge p(\mathcal{J}_{e(i) \geq T/2}) \leq D\}.$$

The next two lower bounds depend on the ratio of long jobs and jobs with large energy demand. For a given  $k \in [n]$ , we define  $\mathcal{J}_k$  to be the set of the  $k$  jobs with the largest energy demand in  $\mathcal{J}$  and  $\mathcal{J}'_k$  to be the set of the  $k$  jobs with the largest energy demand in  $\mathcal{J} \setminus \mathcal{J}_{p(i) > D/2}$ . Let  $i_k$  and  $i'_k$  be the jobs with the smallest energy demand in  $\mathcal{J}_k$  and  $\mathcal{J}'_k$ , respectively. We define:

$$T_{3,a} := \max\{\min\{e(i_k) + e(\mathcal{J}_{p(i) > D - p(\mathcal{J}_k)/2} \setminus \mathcal{J}_k), 2e(i_k)\} \mid k \in \{1, \dots, n\}, p(\mathcal{J}_k) \leq D\},$$

$$T_{3,b} := \max\{\min\{e(i'_k) + e(\mathcal{J}_{p(i) > D - p(\mathcal{J}'_k)/2}), 2e(i'_k)\} \mid k \in \{1, \dots, n\}, p(\mathcal{J}'_k) \leq D\},$$

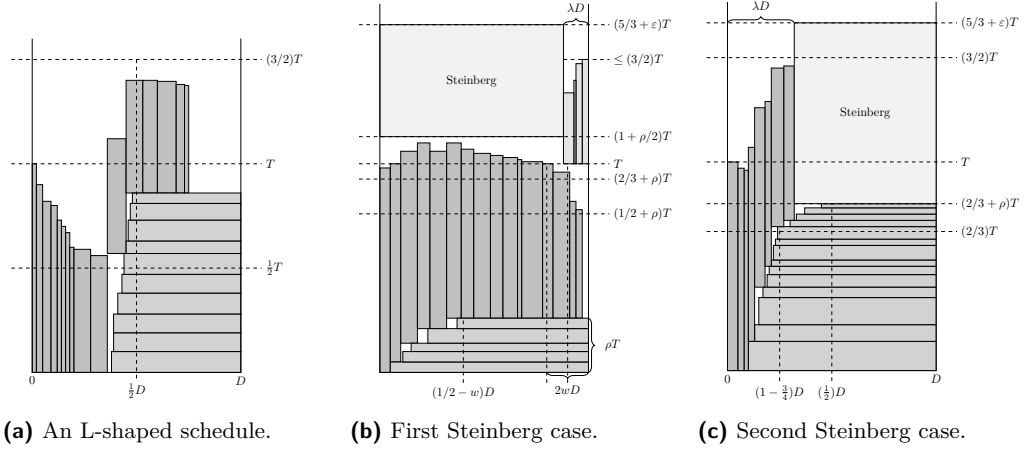
and  $T_3 = \max\{T_{3,a}, T_{3,b}\}$ .

Finally, define  $\mathcal{J}_k$  as the set of the  $k$  jobs with largest energy demand,  $\mathcal{J}_{D,k} := \mathcal{J}_{p(i) > (\max\{D - p(\mathcal{J}_k), D/2\})} \setminus \mathcal{J}_k$ . Let  $i_k$  be the job with the smallest energy demand in  $\mathcal{J}_k$ , then define

$$T_4 := \max\{\min\{2e(i_k), e(i_k) + e(\mathcal{J}_{D,k})/2\} \mid k \in \{1, \dots, n\}, p(\mathcal{J}_k) \leq D\}.$$

► **Theorem 4.** *Given an instance  $I = (\mathcal{J}, D)$ , the value  $T := \max\{T_1, T_2, T_3, T_4\}$  is a lower bound on  $\text{OPT}(I)$ . The value of  $T$  can be found in  $\mathcal{O}(n \log(n))$ .*

This lower bound on  $\text{OPT}$  helps us to identify two cases, that can be solved by scheduling jobs that are large in at least one of the two dimensions in a sorted manner, while the other jobs are scheduled using Steinberg's algorithm. We prove this and the following two theorems in the appendix.



■ **Figure 1** Subfigure 1a shows one possible L-shaped schedule, where  $\mathcal{J}_{\text{seq}}$  contains all the jobs with energy demand larger than  $T/2$  and  $\mathcal{J}_D$  contains all the jobs with processing time larger than  $D/2$ . Subfigure 1b shows a schedule in the case that  $p(\mathcal{J}_{e(i) > (2/3)T}) \geq (1 - w)D$ . Subfigure 1c shows a schedule in the case that  $e(\mathcal{J}_{p(i) > (3/4)D}) \geq (2/3)T$ .

▶ **Theorem 5 (First Steinberg Case).** Let  $T := \max\{T_1, T_2, T_3, T_4\}$  be the lower bound on OPT as defined above. If  $p(\mathcal{J}_{e(j) > (2/3)T}) \geq (1 - (3/4)\varepsilon)D$ , there is a polynomial time algorithm to place all jobs inside a schedule with peak energy demand at most  $(5/3 + \varepsilon)T$ .

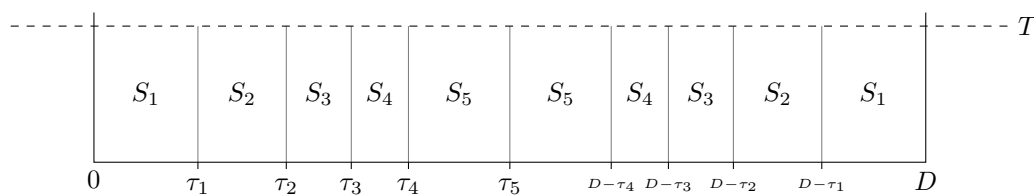
▶ **Theorem 6 (Second Steinberg Case).** Let  $T$  be the lower bound on OPT defined as above. If  $e(\mathcal{J}_{p(i) \geq (3/4)D}) > (2/3)T$ , then there is a polynomial time algorithm that places all the jobs inside the area  $[0, D] \times [0, (5/3)T]$ .

### 3

 $(5/3 + \varepsilon)$ -Approximation

This section inspects schedules generated by the AEPTAS from Theorem 10, more closely. The AEPTAS generates a schedule that fits almost all jobs into an amount of work of peak energy demand  $T \leq (1 + \varepsilon)\text{OPT}$ . Left out of the schedule is a set of jobs that has a very small total processing time, where each job can have an energy demand up to OPT. As such, this set of jobs can be fit into a strip of energy demand OPT and processing time  $\gamma D$  for some  $\gamma > 0$ . Since we aim to generate a schedule of peak energy demand  $(5/3 + \varepsilon)\text{OPT}$ , it does not suffice to simply place this set atop the generated schedule, as this would result in a peak energy demand of  $2\text{OPT}$ . Instead, we must find some area in the generated schedule, inside of which we can remove jobs such that an energy demand of  $\text{OPT}/3$  for a processing time of  $\lambda D$  is empty, where  $\lambda \in [0, 1]$  is a small constant depending on  $\varepsilon$ . Once we have achieved this, and placed the jobs removed by this procedure in a way that does not intersect this strip, we can then place the strip of energy demand OPT at exactly that place, resulting in a schedule of peak energy demand  $(5/3 + \varepsilon)\text{OPT}$ . If none of the previously mentioned cases (as in Theorem 5 and 6, that can be solved using Steinberg's algorithm) apply, then the following Theorem combined with Theorem 10 proves Theorem 2.

▶ **Theorem 7.** Let  $\varepsilon \in (0, 1/3]$ ,  $\varepsilon' \leq (3/5)\varepsilon$  and  $\gamma \leq (3/40)\varepsilon$ . Given an instance  $I$  with  $e(\mathcal{J}_{p(j) > (3/4)D}) \leq (2/3)T'$  and  $p(\mathcal{J}_{e(j) > (2/3)T'}) \leq (1 - (3/4)\varepsilon)D$ , for  $T' = \max\{T_1, T_2, T_3, T_4\}$  and a schedule  $\sigma$  (e.g. generated by the APTAS) where almost all jobs are placed such that the peak energy demand is  $T \leq (1 + \varepsilon')\text{OPT}$ , and the residual jobs inside an additional box  $C_\gamma$  of energy demand  $T$  and processing time  $\gamma D$ , we can find a restructured schedule that places all the jobs up to a schedule with peak energy demand of at most  $(5/3 + \varepsilon)\text{OPT}$ .



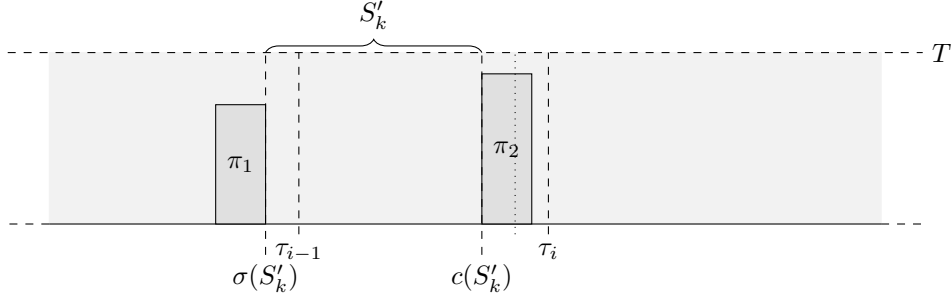
■ **Figure 2** Splitting the given schedule into segments at time points  $\tau_1 = \frac{D}{8}$ ,  $\tau_2 = \frac{(15-24\gamma)D}{64}$ ,  $\tau_3 = \frac{(9+11\gamma)D}{32}$ ,  $\tau_4 = \frac{(3+2\gamma)D}{8}$  and  $\tau_5 = \frac{D}{2}$ .

**Proof.** From the schedule  $\sigma$ , we will generate a new schedule  $\sigma'$ . Some jobs will be shifted to new starting positions  $\sigma'$ . Other jobs  $j$  that are not mentioned in this proof keep their original starting positions, i.e.,  $\sigma'(j) = \sigma(j)$ .

If the schedule contains a job  $j$  with processing time  $p(j) \in [\gamma, (1-2\gamma)D]$  and energy demand  $e(j) \in [(1/3)T, (2/3)T]$ , we proceed as follows to make room for  $C_\gamma$  by shifting job  $j$ : Since  $p(j) \leq (1-2\gamma)D$  it holds that  $\max\{\sigma(j), D - (\sigma(j) + p(j))\} \geq \gamma D$ . Let us, w.l.o.g., assume that  $\sigma(j) \leq D - (\sigma(j) + p(j))$ , otherwise we mirror the schedule at  $D/2$ . We shift the job  $j$  completely to the right (by at least  $\gamma D$ ) such that it is positioned at  $\sigma'(j) := D - p(j)$ . This increases the peak energy demand to at most  $(5/3)T$ . Now the schedule between  $\sigma(j)$  and  $\sigma(j) + \gamma D$  has an energy demand of at most  $(2/3)T$ . We place the box  $C_\gamma$  at  $\sigma(j)$ . Since the box has an energy demand of at most  $T$ , the resulting schedule still has a peak energy demand of at most  $(5/3)T \leq (5/3 + \varepsilon)\text{OPT}$ .

If such a job does not exist, we search for segments that are not overlapped by jobs with energy demand larger than  $(2/3)T$ . We split the schedule into segments at the times  $\tau_1 = \frac{D}{8}$ ,  $\tau_2 = \frac{(15-24\gamma)D}{64}$ ,  $\tau_3 = \frac{(9+11\gamma)D}{32}$ ,  $\tau_4 = \frac{(3+2\gamma)D}{8}$  and  $\tau_5 = \frac{D}{2}$  as well at  $\tau'_i = D - \tau_i$  for  $i \in \{1, 2, 3, 4\}$  and set  $\tau_0 = 0$ . We number the resulting segments in increasing order from 0 to  $D/2$  and from  $D$  to  $D/2$  such that similar segments on both sides get the same number, see Figure 2. For  $k \in \{1, 2, 3, 4, 5\}$ , we denote by  $\sigma(S_k)$  ( $= \tau_{k-1}$ ) the start-time of a segment, by  $c(S_k)$  ( $= \tau_k$ ) the end-time of the segment and by  $p(S_k)$  ( $= \tau_k - \tau_{k-1}$ ) the processing time of the segment  $S_k$ . Since  $p(\mathcal{J}_{e(j) > (2/3)T}) \leq (1 - (3/4)\varepsilon)D$ , we know, by pigeonhole principle, that in one of these segments a total time of at least  $(3/4)\varepsilon D/10 \geq (3/40)\varepsilon D \geq \gamma D$  is not overlapped by these jobs.

Let  $S_{l,k_1}$  be the earliest such strip, and  $S_{r,k_2}$  the latest (they might be the same) such that  $k_1, k_2 \in \{1, 2, 3, 4, 5\}$  represent the index of the strips  $S_1, \dots, S_5$ . In the next step, we modify the start- and end-times of  $S_{l,k_1}$  such that it starts at the end of a job with energy demand at least  $(2/3)T$  or at 0. We denote the shifted start times as  $\sigma'(\cdot)$  and the shifted completion time as  $c'(\cdot)$ . If the start-time of  $S_{l,k_1}$ , i.e.  $\tau_{k-1}$  intersects a job  $j$  with  $e(j) \geq (2/3)T$ , we define  $\sigma'(S_{l,k_1}) := \sigma(j) + p(j) \leq c(S_{l,k_1}) - \gamma D$ . Otherwise if  $k_1 \neq 1$ , we find the last job  $j$  ending before  $\sigma(S_{l,k_1})$  with  $e(j) \geq (2/3)T$  and define  $\sigma'(S_{l,k_1}) := \sigma(j) + p(j) \geq \sigma(S_{l,k_1}) - \gamma D$  and shift the end-time of  $S_{l,k_1}$  by the same amount. Note that, since  $S_{l,k_1}$  is the first strip with at least  $\gamma D$  time not occupied by jobs with energy demand larger than  $(2/3)T$ , the starting time of  $S_{l,k_1}$  is reduced by at most  $\gamma D$ , while the processing time of the segment is not increased. Finally, if the end-time of  $S_{l,k_1}$  intersects a job  $j$  that has an energy demand larger than  $(2/3)T$ , we reduce it to  $c'(S_{l,k_1}) := \sigma(j)$  and call the modified segment  $S'_{l,k_1}$ . These modifications never decrease the total time that is not overlapped by jobs with energy demand larger than  $(2/3)T$  in  $S_{l,k_1}$ . We do the same but mirrored for  $S_{r,k_2}$  resulting in a modified segment  $S'_{r,k_2}$ . For an illustration of this procedure, see Figure 3. If  $D - c(S'_{r,k_2}) \leq \sigma(S'_{l,k_1})$ , we mirror the schedule such that  $\sigma'(j) = D - c(j)$ . We denote by  $S'_k$



■ **Figure 3** An illustration of the border shifting procedure. The original borders are indicated by  $\tau_{i-1}$  and  $\tau_i$ . As  $\tau_{i-1}$  is not intersected by a job with energy demand larger than  $(2/3)T$  we shift  $\sigma(S'_k)$  to an earlier point in time, such that the job with energy demand larger than  $(2/3)T$   $\pi_1$  ends at the exact same time. We then shift  $c(S'_k)$  by the same amount. The shifted  $c(S'_k)$  may intersect a job with energy demand larger than  $(2/3)T$   $\pi_2$ , indicated by the dotted line, and in this case we shift the border further such that  $c(S'_k) = \sigma(\pi_2)$  holds.

the segment in  $\{S'_{l,k_1}, S'_{r,k_2}\}$  that appears first in this new schedule, where  $k = \min\{k_1, k_2\}$  represents the original number of the chosen segment. As a consequence of this mirroring if  $k \geq 2$ , we ensured there exists a job  $j$  with  $e(j) > (2/3)T$  and  $c(S'_k) \leq \sigma(j) \leq D - \sigma(S'_k)$ . Additionally, we know about the start and endpoints of this segment that  $\tau_{k-1} - \gamma D \leq \sigma(S'_k)$  and  $p(S'_k) \leq \tau_k - \tau_{k-1}$ .

We aim to remove jobs from  $S'_k$ , such that the peak energy consumption reached inside  $S'_k$  is bounded by  $(2/3)T$ . We categorize the jobs to be removed in three classes; first the set of jobs  $\mathcal{J}_{\text{cont}}$  that are wholly contained in  $S'_k$  due to the earlier shifting and have an energy demand less than  $(2/3)T$ , second the set of jobs that have an energy demand larger than  $(2/3)T$ , and finally the set of jobs intersecting one of the time points  $\sigma(S'_k)$  or  $c(S'_k)$ . First, we remove  $\mathcal{J}_{\text{cont}}$  from the segment and schedule them inside a container that has an energy demand of at most  $(2/3)T$  and length at most  $3p(S'_k)$ .

► **Lemma 8.** *The jobs  $\mathcal{J}_{\text{cont}}$  can be scheduled inside a container  $C_{\text{cont}}$  of energy demand  $(2/3)T$  and processing time  $3p(S'_k) \leq D/2$ .*

**Proof.** First note that  $\text{work}(\mathcal{J}_{\text{cont}}) \leq p(S'_k)T$ , since the peak energy demand in  $\sigma$  is bounded by  $T$ . We place these jobs using Steinberg's algorithm. Recall that this procedure allows us to place a set of rectangles  $\mathcal{R}$  into a container of size  $a \cdot b$  as long as the following conditions are met:  $p_{\max}(\mathcal{J}) \leq a$ ,  $e_{\max}(\mathcal{J}) \leq b$ ,  $2 \cdot \text{work}(\mathcal{J}) \leq (ab - (2e_{\max}(\mathcal{J}) - T)_+)(2p_{\max}(\mathcal{J}) - D)_+$ . Setting our values for  $b = 3p(S'_k)$  and  $a = (2/3)T$  yields the desired property. Clearly no job wholly contained in a segment of processing time  $p(S'_k)$  can have a processing time greater than  $p(S'_k)$ . Furthermore, the maximum energy demand of any job in  $\mathcal{J}_{\text{cont}}$  is  $(2/3)T$ . Finally, we have:

$$\begin{aligned}
 2 \cdot \text{work}(\mathcal{J}_{\text{cont}}) &\leq 3p(S'_k) \cdot (2/3)T - (2p(S'_k) - 3p(S'_k))_+ \cdot (2(2/3)T - (2/3)T)_+ \\
 &= (ab - (2e_{\max}(\mathcal{J}) - b)_+(2p_{\max}(\mathcal{J}) - a)_+) \quad \blacktriangleleft
 \end{aligned}$$

In the next step, we consider the jobs with energy demand larger than  $(2/3)T$ . By construction of the strip, we know that the total processing time of jobs with energy demands larger than  $(2/3)T$  is bounded by  $p(S'_k) - \gamma D$ . We remove all these jobs from the strip and combine them with the extra container  $C_\gamma$  of energy demand at most  $T$  and processing time  $\gamma D$  to a new container called  $C_{\text{tall}}$ . It has an energy demand of at most  $T$  and a processing time of at most  $p(S'_k)$ .

After this step, the only jobs remaining inside the area of  $S'_k$  are the jobs that overlap the borders of  $S'_k$ . If the peak energy demand in  $S'_k$  is lower than  $(2/3)T$ , we place the container  $C_{\text{tall}}$  inside the strip  $S'_k$  as well as the container  $C_{\text{cont}}$  right of  $D/2$  and are done. Otherwise, we have to remove jobs that overlap the borders of the strip  $S'_k$  until the peak energy demand in  $S'_k$  is bounded by  $(2/3)T$ . The jobs we choose to remove are dependent on the position of the strip. The following lemma helps to see how these jobs can be shifted without increasing the peak energy demand of the schedule too much.

► **Lemma 9.** *Consider a schedule  $\sigma$  with peak energy demand bounded by  $T$  and a time  $\bar{\tau}$ , as well as a subset of jobs  $\mathcal{J}_{\text{Move}} \subseteq \mathcal{J}(\bar{\tau})$  with  $e(\mathcal{J}_{\text{Move}}) \leq a \cdot T$  for some  $a \in [0, 1]$ . Let  $\tau$  be the smallest value  $\sigma(j)$  for  $j \in \mathcal{J}_{\text{Move}}$ . Consider the schedule  $\sigma'$ , where all the jobs in  $\mathcal{J}_{\text{Move}}$  are delayed such that they end at  $D$ , i. e.,  $\sigma'(j) = D - p(j)$  for all  $\mathcal{J}_{\text{Move}}$  and  $\sigma'(j) = \sigma(j)$  for all other jobs.*

*In the schedule  $\sigma'$  before of  $D/2 + \tau/2$ , the peak energy demand is bounded by  $T$ , while after of  $D/2 + \tau/2$  the peak energy demand is bounded by  $(1 + a)T$ .*

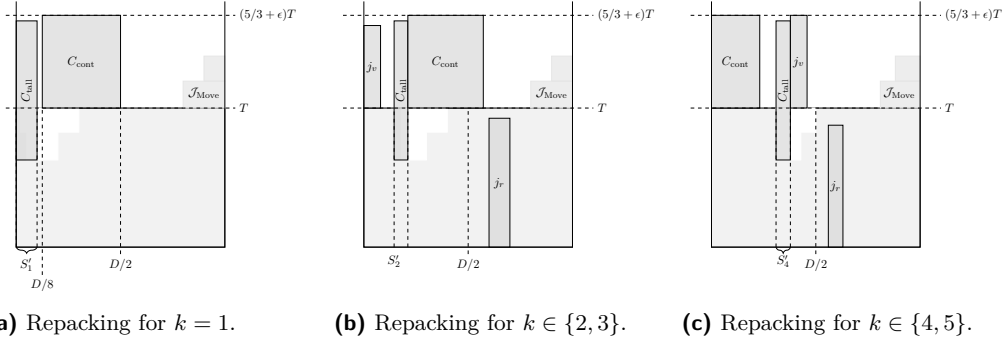
**Proof.** If the energy demand of the schedule  $\sigma'$  is larger than  $T$  at a position  $\tau'$ , it has to be because one of the jobs in  $\mathcal{J}_{\text{Move}}$  overlaps it. Hence, that peak energy demand is bounded by  $(1 + a)T$ , since we shifted jobs with total energy demand bounded by  $aT$ . Let  $j$  be one of the shifted jobs. If  $\sigma'(j) > D/2 + \tau/2$ , the energy demand of the schedule before of  $D/2 + \tau/2$  cannot be influenced by this job. Therefore, assume that  $D - p(j) = \sigma'(j) \leq D/2 + \tau/2$ . As a consequence,  $p(j) \geq D/2 - \tau/2$ . Since  $\sigma(j) \leq \tau$  it holds that  $\sigma(j) + p(j) \geq D/2 + \tau/2$ . Thus before time  $D/2 + \tau/2$ , the job  $j$  overlaps its previous positions and cannot increase the peak energy demand above  $T$ . ◀

We choose which of the overlapping jobs to shift depending if  $k = 1$  or  $k \neq 1$ . Remember that none of the borders of  $S'_k$  overlap a job that has an energy demand larger than  $(2/3)T$ , and assume for the following, that there is a point inside  $S'_k$  where the total energy demand of overlapping jobs is larger than  $(2/3)T$ .

**Case 1:  $k = 1$ .** Consider the time  $\tau = c(S'_1) \leq D/8$  and the set of jobs  $\mathcal{J}(\tau)$  that are intersected by this line. We know that the total energy demand of jobs with processing time greater than  $(3/4)D$  is bounded by  $(2/3)T$ . Let  $\mathcal{J}_{\text{Move}}$  be the set of jobs generated as follows: Greedily take the jobs with the largest energy demand from  $\mathcal{J}(\tau) \setminus \mathcal{J}_{p(j) > (3/4)D}$ , until either all the jobs from  $\mathcal{J}(\tau) \setminus \mathcal{J}_{p(j) > (3/4)D}$  are contained in  $\mathcal{J}_{\text{Move}}$  or  $e(\mathcal{J}_{\text{Move}}) \in [(1/3)T, (2/3)T]$ . In this process, we never exceed  $(2/3)T$  since, if there is a job with energy demand larger than  $(1/3)T$  in  $\mathcal{J}(\tau) \setminus \mathcal{J}_{p(j) > (3/4)D}$ , we choose it first and immediately stop. We delay the jobs in  $\mathcal{J}_{\text{Move}}$  to new start positions  $\sigma'$  such that for each job  $j \in \mathcal{J}_{\text{Move}}$  we have that  $c'(j) := \sigma'(j) + p(j) = D$ . Note that  $\sigma'(j) \geq (1/4)D$  for each  $j \in \mathcal{J}_{\text{Move}}$  and therefore no longer overlaps  $c(S'_1)$ . Furthermore, we know by Lemma 9 that before  $D/2$  the peak energy demand is bounded by  $T$ , while after  $D/2$  the peak energy demand is bounded by  $(5/3)T$ . Furthermore, the peak energy demand inside  $S'_1$  is bounded by  $(2/3)T$ .

Since  $S'_1$  has a processing time of at most  $D/8$ , we know by Lemma 8 that  $\mathcal{J}_{\text{cont}}$  can be placed inside a container  $C_{\text{cont}}$  with energy demand at most  $(2/3)T$  and processing time bounded by  $3D/8$ . Therefore, we can schedule this container at  $D/8$  and know that it is finished before  $D/2$ . Finally, we schedule the container  $C_{\text{tall}}$  at  $\sigma'(C_{\text{tall}}) = 0$ . The peak energy demand of the resulting schedule is bounded by  $(5/3)T$ . See Figure 4a for the repacking procedure.

## 21:10 Peak Demand Minimization via Sliced Strip Packing



■ **Figure 4** Illustration of the steps in the proof of Theorem 7. Note that the set  $\mathcal{J}_{\text{Move}}$  is delayed such that the jobs end at  $D$ . The containers  $C_{\text{tall}}$  and  $C_{\text{cont}}$  are placed such that they do not intersect. For 4b and 4c, the jobs  $j_v$  are placed in the same manner, and the job  $j_r$  is denoted.

**Case 2:  $k \neq 1$ .** In this case, the borders of the considered strip can be overlapped from both sides. Furthermore, we know that the left border of  $S'_k$  is right of  $D/8 - \gamma D \geq \gamma D$ .

Consider the largest total energy demand of jobs that are intersected by any vertical line through  $S'_k$  and denote this energy demand as  $T_{S'_k}$ . Since there is a job  $j_l$  with energy demand larger than  $(2/3)T$  with  $\sigma(j_l) + p(j_l) = \sigma(S'_k)$ , we know that the total energy demand of jobs intersecting  $\sigma(S'_k)$  can be at most  $(1/3)T$ . Next, consider the closest job  $j_r$  that starts after  $c(S'_k)$  and has an energy demand larger than  $(2/3)T$ . By the choice of  $S'_k$ , we know that such a job must exist and that  $\sigma(j_r) \leq D - \sigma(S'_k)$ , by the choice of  $S'_k$  out of  $S'_{k_1,l}$  and  $S'_{k_2,r}$ . Furthermore, we know that the total energy demand of jobs intersecting the vertical line at  $\sigma(j_r)$  is bounded by  $(1/3)T$ .

Hence the jobs that overlap the vertical line at  $\sigma(j_r)$  and the jobs that overlap the vertical line at  $\sigma(S'_k)$  add a total energy demand of at most  $(2/3)T$  to  $T_{S'_k}$ . Let us now consider the jobs  $\mathcal{J}_M$  that overlap the time  $c(S'_k)$  but neither the time  $\sigma(S'_k)$  nor the time  $\sigma(j_r)$ . Each of them has a processing time of at most  $D - \sigma(S'_k) - \sigma(j_r) \leq D - 2\sigma(S'_k)$ . Hence when delaying their start points such that  $\sigma'(j) = D - p(j)$ , they no longer overlap the time  $c(S'_k)$  since  $p(S'_k) \leq \sigma(S'_k)$  for each  $k \in \{2, 3, 4, 5\}$ .

We greedily take jobs from  $\mathcal{J}_M$  that have the earliest starting point until we have all jobs from  $\mathcal{J}_M$  or we have a total energy demand of at least  $(1/3)T$ . If the total energy demand of the chosen jobs is larger than  $(2/3)T$ , the last job  $j_v$  has an energy demand of at least  $(1/3)T$ . Since it has a processing time lower than  $(1 - 2\gamma)D$ , it has to have a processing time of at most  $\gamma D$ . We remove this job and place it later, while we shift all the others to new positions  $\sigma'$  such that  $\sigma'(j) = D - p(j)$  for each of the taken jobs  $j$ . We call the set of shifted jobs  $\mathcal{J}_{\text{Move}}$ .

Furthermore, since  $\mathcal{J}_{\text{Move}}$  has a total energy demand of at most  $(2/3)T$  and a starting point right of  $\sigma(S'_k)$ , we know by Lemma 9 that the peak energy demand right of  $D/2 + \sigma(S'_k)/2$  is bounded by  $(5/3)T$  while left of  $D/2 + \sigma(S'_k)/2$  it is bounded by  $T$ .

Let  $\sigma(j_l)$  be the starting time of the last taken job. Before  $\sigma(j_l)$  (in  $S'_k$ ) there is no longer a job from  $\mathcal{J}_M$ , and, hence inside in the strip  $S'_k$  that is left of  $\sigma(j_l)$ , the peak energy demand is bounded by  $(2/3)T$ . On the other hand, after  $\sigma(j_l)$  (in  $S'_k$ ) we either have removed jobs with total energy demand at least  $(1/3)T$ , or all the jobs from  $\mathcal{J}_M$  and hence the schedule there can have a total energy demand of at most  $(2/3)T$  as well. Therefore, we can place the container  $C_{\text{tall}}$  inside  $S'_k$  without increasing the energy demand above  $(5/3)T$ .

The container  $C_{\text{cont}}$  and the job  $j_v$  remain to be placed. For  $k \in \{2, 3\}$ , we set  $C_{\text{cont}}$  at  $\sigma'(C_{\text{cont}}) = c(S'_k)$  and the job  $\sigma'(j_v) = 0$ , while for  $k \in \{4, 5\}$ , we set  $\sigma'(C_{\text{cont}}) = 0$  and  $\sigma'(j_v) = c(S'_k)$ . We will now see, for each segment, that the peak energy demand of  $(5/3)T$  is not exceeded by this new schedule.

First note that  $p(j_v) \leq \gamma D \leq D/8 - \gamma D$  and hence does not intersect  $S'_2$ , when scheduled at  $\sigma'(j_v) = 0$ . Similarly, it is more narrow than  $S'_5$  and  $\sigma(S'_5)/2$ , and hence fits right of  $S'_4$  and  $S'_5$  without increasing the schedule more than  $(5/3)T$ .

Let us now check the conditions for  $C_{\text{cont}}$ : For  $k \in \{2, 3\}$ , we have to ensure that  $c(S'_k) + p(C_{\text{cont}})/2 \leq D/2 + \sigma(S'_k)$ , while for  $k \in \{4, 5\}$  we have to prove that  $p(C_{\text{cont}}) \leq \sigma(S'_k)$ . It holds that  $p(S'_2) \leq \frac{(15-24\gamma)D}{64} - \frac{D}{8} = \frac{(7-24\gamma)D}{64}$  and hence  $p(C_{\text{cont}}) \leq 3 \left( \frac{(7-24\gamma)D}{64} \right)$ . Therefore,  $c(S'_2) + p(C_{\text{cont}}) \leq \frac{D}{2} + \frac{\sigma(S'_2)}{2}$ . Furthermore,  $p(S'_3) \leq \frac{(9+14\gamma)D}{32} - \frac{(15-24\gamma)D}{64} = \left( \frac{3}{64} + \frac{52\gamma}{32} \right) D$  and hence  $p(C_{\text{cont}}) \leq 3 \left( \frac{3}{64} + \frac{52\gamma}{32} \right) D$ . Therefore,  $c(S'_3) + p(C_{\text{cont}}) \leq \frac{(9+14\gamma)D}{32} + 3 \left( \frac{3}{64} + \frac{52\gamma}{32} \right) D = \frac{(27+184\gamma)D}{64}$ , while  $D/2 + \sigma(S'_3)/2 \geq D/2 + \left( \frac{15-24\gamma}{64} - \gamma \right) D/2 = \left( \frac{79}{128} - \frac{11}{8}\gamma \right) D$ . As a consequence,  $c(S'_3) + p(C_{\text{cont}}) \leq D/2 + \sigma(S'_3)/2$ , since  $\gamma \leq 1/40 \leq 25/392$ .

Finally, we have  $p(S'_4) \leq \frac{(3+2\gamma)D}{8} - \frac{(9+14\gamma)D}{32} = \frac{(3-6\gamma)D}{32}$ . Hence,  $p(C_{\text{cont}}) \leq 3 \left( \frac{(3-6\gamma)D}{32} \right) = \frac{(9+14\gamma)D}{32} - \gamma D \leq \sigma(S'_4)$ . While it holds that  $p(S'_5) \leq \frac{D}{2} - \frac{(3+2\gamma)D}{8} = \frac{(1+2\gamma)D}{8}$ . Therefore,  $p(C_{\text{cont}}) \leq 3 \cdot \frac{(1+2\gamma)D}{8} = \frac{(3+6\gamma)D}{8} = \frac{(1+2\gamma)D}{8} - \gamma D \leq \sigma(S'_5)$ .

For a visual representation of this repacking procedure see Figure 4. In all the cases the generated schedule has a height of at most  $(5/3)T \leq (5/3)(1 + \varepsilon')\text{OPT} \leq (5/3 + \varepsilon)\text{OPT}$ . ◀

## 4 AEPTAS for NPDM

In this section, we will prove the following theorem.

► **Theorem 10.** *Let  $\varepsilon > 0$ . There is an algorithm that places almost all jobs such that the peak energy demand is bounded by  $T' := (1 + \mathcal{O}(\varepsilon))\text{OPT}$ . For the residual jobs, we can choose one of the following containers for them to be placed in:  $C_1$  with processing time  $\varepsilon D$  and energy demand  $T'$  or a container  $C_2$  with processing time  $D$  and energy demand  $e_{\max}$ . The time complexity of this algorithm is bounded by  $\mathcal{O}(n \log(n)/\varepsilon) + 1/\varepsilon^{1/\varepsilon^{\mathcal{O}(1/\varepsilon)}}$ .*

The statement, in fact, gives two variants of the algorithm. The first variant where all residual jobs are placed in  $C_1$  is used in our  $5/3 + \varepsilon$  approximation algorithm, where the second variant with all residual jobs in  $C_2$  can be used to obtain the AEPTAS by setting  $\sigma(C_2) = 0$ . The described algorithm follows the dual-approximation framework. We describe an algorithm that given a bound on the schedule peak energy demand  $T$  computes a schedule with peak energy demand  $(1 + \mathcal{O}(\varepsilon))T' + e_{\max}$  or decides correctly that there is no schedule with peak energy demand at most  $T'$ . This algorithm then can be called in binary search fashion with values  $T$  between  $T' = \max\{T_1, T_2, T_3, T_4\}$  and  $\max\{2\text{work}(\mathcal{J})/D, 2e_{\max}\}$ , using only multiples of  $\varepsilon T'$ . Note that if  $e_{\max} \leq \mathcal{O}(\varepsilon^3 T')$ , we can use the algorithm in [11] to find an  $(1 + \varepsilon)\text{OPT} + \mathcal{O}(\log(1/\varepsilon)/\varepsilon \cdot \varepsilon^3 T') = (1 + \mathcal{O}(\varepsilon))\text{OPT}$  approximation. Hence we can assume that  $e_{\max} > \mathcal{O}(\varepsilon^3 T')$ .

### Classification of Jobs

Given two values  $\delta$  and  $\mu$  with  $\mu < \delta$ , we partition the jobs into five sets: large, horizontal, vertical, small, and medium sized jobs. We define  $\mathcal{J}_{\text{large}} := \{i \in \mathcal{J} | e(i) \geq \delta T', p(i) > \delta D\}$ ,  $\mathcal{J}_{\text{hor}} := \{i \in \mathcal{J} | e(i) < \mu T', p(i) > \delta D\}$ ,  $\mathcal{J}_{\text{ver}} := \{i \in \mathcal{J} | e(i) \geq \delta T', p(i) < \mu D\}$ ,  $\mathcal{J}_{\text{small}} := \{i \in \mathcal{J} | e(i) < \mu T', p(i) < \mu D\}$ , and  $\mathcal{J}_{\text{medium}} := \mathcal{J} \setminus (\mathcal{J}_{\text{large}} \cup \mathcal{J}_{\text{hor}} \cup \mathcal{J}_{\text{ver}} \cup \mathcal{J}_{\text{small}})$ .

► **Lemma 11.** *In  $\mathcal{O}(n + 1/\varepsilon^2)$  operations it is possible to find values  $\geq \varepsilon^{\mathcal{O}(1/\varepsilon^2)}$  for  $\delta$  and  $\mu$  such that  $\text{work}(\mathcal{J}_{\text{medium}}) \leq (\varepsilon^2/4)DT$  and  $\mu \leq c\varepsilon^5\delta$  for any given constant  $c$ .*

**Proof.** Consider the sequence  $\rho_0 := \varepsilon^5/4$ ,  $\rho_{i+1} := c\rho_i\varepsilon^3$ . Due to the pigeonhole principle, there exists an  $i \in \{0, \dots, 8/\varepsilon^2\}$  such that when defining  $\delta := \sigma_i$  and  $\mu := \sigma_{i+1}$  the total amount of work of the medium sized jobs is bounded by  $(\varepsilon^2/4)DT$ , because each job appears only in two possible sets of medium jobs. We have  $\delta \geq \mu \geq \varepsilon^{\mathcal{O}(1/\varepsilon^2)}$ . ◀

## 21:12 Peak Demand Minimization via Sliced Strip Packing

► **Lemma 12.** [43] *We can round the energy demands  $e(i)$  of the vertical and large jobs to multiples  $k_i \varepsilon \delta T$  with  $k_i \in \{1/\varepsilon, \dots, 1/\varepsilon \delta\}$  such that the number of different demands is bounded by  $O(1/\varepsilon^2 \log(1/\delta))$ . This rounding increases the optimal energy demand by at most  $2\varepsilon T$*

### Profile for vertical jobs

In the following we will dismiss the medium jobs from the schedule. Given an optimal schedule, we partition the schedule into  $1/\gamma$  segments of processing time  $\gamma D$ , for a constant  $\gamma \in \mathcal{O}_\varepsilon(1)$ . Given a schedule of jobs  $J$ , we define profile of  $J$  to be  $\{(x, y) | y = \sum_{j \in J | \sigma(j) \leq x \leq \sigma(j) + p(j)} e(j), 0 \leq x \leq D, \}$ . Energy demand of profile of jobs  $J$  at time  $t$  is  $\mathcal{E}_J(t) := \sum_{j \in J | \sigma(j) \leq t \leq \sigma(j) + p(j)} e(j)$ . Now consider the profile of large and horizontal jobs. Let  $\tilde{J} := \mathcal{J}_{large} \cup \mathcal{J}_{hor}$ . We search for the segments where the maximal energy demand of the profile of large and horizontal jobs and the minimal energy demand of this profile differs more than  $\varepsilon T$ , i.e., if in segment  $S := (t_a, t_b)$ ,  $|\max_{t \in S} \mathcal{E}_{\tilde{J}}(t) - \min_{t \in S} \mathcal{E}_{\tilde{J}}(t)| \geq \varepsilon T$ , then we remove all vertical and small jobs from these segments fractionally, i.e., we slice jobs, which are cut by the borders of the segment.

▷ **Claim 1.** Let  $\mathcal{J}_{rem}$  be the set of removed vertical and small jobs. Then  $\text{work}(\mathcal{J}_{rem})$  is bounded by  $\mathcal{O}(\gamma/\varepsilon \delta) \cdot D \cdot T$ .

*Proof.* Note that the energy demand of the profile of horizontal or large jobs only changes, when horizontal or large jobs end or start. The large and horizontal jobs have a total energy demand of at most  $T/\delta$  since they have a processing time of at least  $\delta D$  and the total area of the schedule is bounded by  $T \cdot D$ . Hence there can be at most  $2(T/\delta)/\varepsilon T = \mathcal{O}(1/\varepsilon \delta)$  segments, where the energy demand of the profile changes more than  $\varepsilon T$ . As a result, the total area of the removed vertical jobs can be bounded by  $\mathcal{O}(1/\varepsilon \delta) \cdot (\gamma D \cdot T)$ . ◁

▷ **Claim 2 (Size of  $\gamma$ ).** In the case of container  $C_1$ , we can choose  $\gamma \in \mathcal{O}(\varepsilon \delta \lambda)$  such that we can schedule the removed vertical jobs fractionally inside a container  $C_{1,1/4}$  of processing time  $p(C_1)/4$  and energy demand  $e(C_1)$ . Otherwise, we can choose  $\gamma \in \mathcal{O}(\varepsilon^4 \delta)$  such that we can schedule the removed vertical jobs fractionally inside a container  $C_{2,1/4}$  of processing time  $p(C_2)/4$  and energy demand  $e(C_2)$ .

*Proof.* Let  $k \in \{1, 2\}$  depending on the chosen container. First we place all the jobs  $\mathcal{J}_{rem,tall}$ , i.e., jobs in  $\mathcal{J}_{rem}$  with energy demand larger than  $e(C_{k,1/4})/2$  next to each other. The total processing time of these jobs is bounded by  $2 \cdot \text{work}(\mathcal{J}_{rem,tall})/e(C_{k,1/4})$ . Next, we place the residual jobs  $\mathcal{J}_{rem,res} := \mathcal{J}_{rem} \setminus \mathcal{J}_{rem,tall}$ , which have an energy demand of at most  $e(C_{k,1/4})/2$ . We take slices of processing time 1 of the jobs and place them on top of each other until the energy demand  $e(C_{k,1/4})/2$  is reached. Since each job has an energy demand of at most  $e(C_{k,1/4})/2$  the energy demand  $e(C_{k,1/4})$  is not exceeded. The total processing time of this schedule is bounded by  $2\text{work}(\mathcal{J}_{rem,res})/e(C_{k,1/4}) + 1 \leq \mathcal{O}(\gamma/(\varepsilon \delta) \cdot D \cdot T)/e(C_{k,1/4})$ . Hence, for  $C_{1,1/4}$  the total processing time is bounded by  $\mathcal{O}(\gamma/(\varepsilon \delta) \cdot D \cdot T)/T = \mathcal{O}(\gamma/(\varepsilon \delta))D$ . Hence, when choosing  $\gamma \in \mathcal{O}(\lambda \varepsilon \delta)$  for a suitable constant, the total processing time of this schedule is bounded by  $p(C_1)/4$ . Otherwise, for container  $C_{2,1/4}$  the total processing time is bounded by  $\mathcal{O}((\gamma/(\varepsilon \delta) \cdot D \cdot T)/\varepsilon^3 T) = \mathcal{O}(\gamma/\varepsilon^4 \delta)D$ . Hence, when choosing  $\gamma \in \mathcal{O}(\varepsilon^4 \delta)$  for a suitable constant, the total processing time of this schedule is bounded by  $p(C_2)/4$ . ◁

### Algorithm to place the vertical, small, and medium jobs

In the algorithm, we first round the energy demands of the vertical jobs to at most  $\mathcal{O}(1/\varepsilon^2 \cdot \log(1/\delta)) = (1/\varepsilon)^{\mathcal{O}(1)}$  sizes using Lemma 12 (geometric rounding).



Afterward, we guess for each of the  $1/\gamma$  segments the energy demand reserved for the vertical and small jobs rounding up to the next multiple of  $\varepsilon T$ , adding at most one more  $\varepsilon T$  to the energy demand of the schedule. There are at most  $\mathcal{O}((1/\varepsilon)^{1/\gamma})$  possible guesses. Furthermore, we introduce one segment  $\top$  of energy demand  $\lceil e(C_k)/(\varepsilon T) \rceil \cdot \varepsilon T$  and processing time  $p(C_k)/4$  ( $k \in \{1, 2\}$ ) for the set of removed vertical jobs. Let  $S_{ver}$  be the set of all introduced segments, and for each  $s \in S_{ver}$  let  $e_{s,ver}$  be the energy demand reserved for vertical and small jobs. Note that for each  $s \in S_{ver}$  there exists an  $i \in \{0, \dots, 1/\varepsilon + 3\}$  such that  $e_{s,ver} = i\varepsilon T$ . Furthermore, let  $S_{ver,e}$  be the set of segments that have exactly energy demand  $e$  and let  $p(S_{ver,e})$  be their total processing time.

To place the vertical jobs into the segments  $S_{ver}$ , we use a configuration LP. Let  $C = \{a_\eta : \eta | \eta \in \{e(j) | j \in \mathcal{J}_{ver}\}\}$  be a configuration for vertical jobs, where  $a_\eta$  denotes the multiplicity with which the energy demand  $\eta$  is contained in  $C$ . We denote by  $e(C) := \sum_{\eta \in \{e(j) | j \in \mathcal{J}_{ver}\}} a_\eta \cdot \eta$  the energy total demand of  $C$ , and by  $\mathcal{C}_e$  the set of configurations with energy demand at most  $e$ . Furthermore, for a given configuration  $C$  we denote by  $a_\eta(C)$  the number of jobs contained in  $C$  that have an energy demand of  $\eta$ . Since each vertical job has a energy demand of at least  $\delta T$ , there are at most  $(1/\varepsilon)^{\mathcal{O}(1/\delta)}$  different configurations. Consider the following linear program:

$$\sum_{C \in \mathcal{C}_{i\varepsilon T}} x_{C,i} = p(S_{ver,i\varepsilon T}) \quad \forall i \in \{1, \dots, 1/\varepsilon + 3\} \quad (1)$$

$$\sum_{s \in S} \sum_{C \in \mathcal{C}_{e_{s,ver}}} a_\eta(C) x_{C,s} = \sum_{j \in \mathcal{J}_{ver}, e(j)=\eta} p(j) \quad \forall \eta \in \{e(j) | j \in \mathcal{J}_{ver}\} \quad (2)$$

$$x_{C,i} \geq 0 \quad \forall C \in \mathcal{C}, i \in \{1, \dots, 1/\varepsilon\} \quad (3)$$

The variable  $x_{C,i}$  represents the processing-time of configuration  $C$  inside segments  $s \in S_{ver}$  with reserved energy capacity  $e_{s,ver} = i\varepsilon T$ . The first equation ensures that the total processing-time assigned to configurations inside segments with a certain energy capacity does not exceed the total processing time of these segments. The second equation ensures that each job is fully scheduled. More precisely, that the total processing time of jobs with a certain energy demand is covered by the configurations. A basic solution has at most  $(1/\varepsilon + |\{e(j) | j \in \mathcal{J}_{ver}\}| + 1) = (1/\varepsilon)^3$  nonzero components. We can solve the above linear program by guessing the set of non zero components and then solving the resulting LP in  $((1/\varepsilon)^{\mathcal{O}(1/\delta)})^{(1/\varepsilon)^3}$  time.

To place the vertical jobs, we first fill them greedily inside the configurations (slicing when the corresponding configuration slot is full) and afterwards place the configurations inside the schedule, slicing the jobs at the segment borders. For each nonzero component we have one configuration that contains at most  $2/\delta$  fractionally placed vertical jobs on top of each other, which have a total energy demand of at most  $2T'$ . Additionally, for each segment we have the same amount of fractional jobs. Hence total area of fractionally placed jobs can be bounded by  $\mu D \cdot 2T \cdot ((1/\varepsilon)^3 + 1/\gamma)$ . If we choose  $C_1$  this can be bounded by  $\mathcal{O}(\mu/(\lambda\varepsilon\delta))DT \leq \lambda DT/8$ , since  $\mu = c\varepsilon\delta\lambda^2$  and otherwise by  $\in \mathcal{O}(\mu D \cdot T/(\varepsilon^5\delta)) \leq DT/8$ , since  $\mu = c\varepsilon^5\delta$  for a suitable small constant  $c$ . We remove the fractionally placed jobs  $\mathcal{J}_{frac}$ .

Next, we place the small jobs inside the empty area that can appear above each configuration for vertical jobs. Note that there are at most  $((1/\varepsilon)^{\mathcal{O}(1)} + 1/\gamma)$  configurations and the free area inside these configurations has at least the size of the total area of the small jobs. As a consequence, we have at most  $2/\gamma$  rectangular areas to place the small jobs, which have a total area, which is at least the size of the small jobs. We use the NFDH algorithm to place these jobs inside the boxes until no other job fits inside.

## 21:14 Peak Demand Minimization via Sliced Strip Packing

Assume we could not place all the small jobs inside these boxes. When considering the area of free energy in each box, there are three parts that contribute to it. First, each box can have a free strip at its end, which has a processing time of at most  $\mu D$ . The total area of free energy contributed by this strip is bounded by  $(2/\gamma)\mu D \cdot 2T$ . Second, each box can have a free strip of energy demand at most  $\mu T$  on the top because otherwise, another line of jobs would have fitted inside this box. Since there are no boxes on top of each other, we can bound the total area of free energy inside this strip by  $\mu T \cdot D$ . Finally, there can be free energy between the shelves of the jobs generated by the NFDH algorithm. This total free energy is bounded by the energy demand of the tallest job times the processing time of the widest box, i.e.,  $\mu T \cdot \gamma D$ . Hence the total area of free energy inside the boxes is bounded by  $5\mu D \cdot T \cdot \gamma + D \cdot \mu T$ . Since  $\gamma \in \mathcal{O}(1/\varepsilon^5\delta)$  and we have chosen  $\mu \leq c\delta\varepsilon^6$  for a suitable small constant  $c \in \mathbb{Q}$ , the total work of the remaining small jobs  $\mathcal{J}_{small,res}$ , which could not be placed is bounded by  $\varepsilon TD$ .

We place the residual small jobs  $\mathcal{J}_{small,res}$  on top of the schedule using NFDH. This adds an energy demand of at most  $2\varepsilon T$  to the schedule. Next we place the medium jobs. We start all the medium jobs, that have a processing time larger than  $p(C_k)/4$  with Steinberg's algorithm inside a box of energy demand at most  $\mathcal{O}(\varepsilon)T$  and processing time  $D$ . This is possible since they have processing time in  $(\varepsilon D, D]$  and therefore each has an energy demand of at most  $\mathcal{O}(\varepsilon)T$  because their total work is bounded by  $(\varepsilon^2/4)DT$ . The residual jobs (that might have a processing time larger than  $\varepsilon T$ ) are placed inside the first half of the container using Steinberg's algorithm. The later half of the container is filled with the extra box for vertical jobs defined for the LP and the fractionally scheduled jobs. The extra box has a width of at most  $p(C_k)/4$ . Since the fractionally placed vertical jobs  $\mathcal{J}_{frac}$  have an area of at most  $\varepsilon(C_k) \cdot p(C_k)/8$  and each has a width of at most  $\mu D < p(C_k)/8$ , we can use Steinberg's algorithm to place them inside the last quarter of the container  $C_k$ .

### Placement of horizontal jobs

In this section, we first reduce the number of possible starting points for horizontal jobs and then use a linear program to place the jobs in the schedule.

First step: use geometric grouping to reduce the number of processing times of horizontal jobs. At a loss of at most  $2\varepsilon T$  in the approximation ratio, we can reduce the number of processing times of horizontal jobs to  $\mathcal{O}(\log(1/\delta)/\varepsilon)$  using geometric grouping (see [33, Theorem 2] by Karmarkar and Karp). These rounded jobs can be placed fractionally instead of the original jobs and an extra box of energy demand at most  $\mathcal{O}(\varepsilon)T$ . In this fractional packing, the horizontal jobs are sliced along the axis of the processing-time, i.e., different fractions of a job might have different starting points, but a fraction that is started, will not be interrupted and require the same amount of energy during its procession. We denote the rounded processing-time of a job  $j$  as  $p'(j)$ .

In the next step, we will reduce the number of starting points of the large and fractionally placed horizontal jobs without exceeding the given profile. Remember, we know the profile of large and horizontal jobs with precision  $\varepsilon T$  for the segments of processing time  $\gamma D$ .

▷ **Claim 3.** Without loss in the approximation ratio, we can reduce the number of different starting points of rounded horizontal and large jobs to  $(1/\varepsilon)^{\mathcal{O}(1/\varepsilon)}$ .

*Proof.* Consider the large and horizontal jobs starting in the first segment. Since this segment has a processing time of  $\gamma D \leq \delta D$ , there can be no job ending in this segment. Hence this segment is maximally filled at the point  $\gamma D$ . We can shift the start point of each job in this segment to 0 and we will not change the maximal energy demand of this segment.

Now consider a job  $i \in \mathcal{J}_{hor} \cup \mathcal{J}_{large}$  starting in the second segment. If there is no horizontal or large job ending before the start of  $i$ , we can shift the start point of  $i$  to  $\gamma D$  without changing the maximal filling energy demand in this segment. However, if there is a job  $j \in \mathcal{J}_{hor} \cup \mathcal{J}_{large}$  ending before  $i$  in this segment, we can not shift this job to  $\gamma D$  since then  $i$  and  $j$  overlap, which they did not before. This could change the maximal energy demand of the profile in this segment. Nevertheless, if  $j$  is the last job ending before  $i$ , we can shift  $i$  to the left, such that  $i$  starts at the endpoint of  $j$ .

We iterate this shifting with all segments and all jobs in  $\mathcal{J}_{hor} \cup \mathcal{J}_{large}$ . As a result, all jobs start either at a multiple of  $\gamma D$ , or they start at an endpoint of an other job in  $\mathcal{J}_{hor} \cup \mathcal{J}_{large}$ . Therefore, we can describe the set of possible starting points for jobs in  $\mathcal{J}_{hor} \cup \mathcal{J}_{large}$  as  $S_{hor,large} := \{l\gamma D + \sum_{j=1}^{1/\delta} p(i_j) \mid l \in \{0, 1, \dots, 1/\gamma\}, i_j \in \mathcal{J}_{hor} \cup \mathcal{J}_{large} \forall j \in \{1, \dots, 1/\delta_w\}\}$ . It holds that  $|S_{hor,large}| \leq (1/\gamma) \cdot (\log(1/\delta)/\epsilon)^{1/\delta} = (1/\epsilon)^{(1/\epsilon)^{\mathcal{O}(1/\epsilon)}}$ .  $\triangleleft$

$\triangleright$  **Claim 4.** At a loss of at most  $\mathcal{O}(\epsilon T)$  in the approximation ratio, we can reduce the number of *used* starting points for rounded horizontal jobs to  $\mathcal{O}(1/\epsilon\delta)$ .

*Proof.* We partition the set of horizontal jobs by their processing time into  $\mathcal{O}(\log(1/\delta))$  sets  $\mathcal{J}_{hor}^l := \{i \in \mathcal{J}_{hor} \mid D/2^l < p(i) \leq D/2^{l-1}\}$ . For each of these sets, we will reduce the number of starting positions to  $2^l/\epsilon^2$ . We partition the schedule into  $2^l$  segments of processing time  $D/2^l$ . Each job from the set  $\mathcal{J}_{hor}^l$  has a processing time larger than  $D/2^l$  and hence it starts in an other segment as it ends. We consider for each segment all the horizontal jobs of the set  $\mathcal{J}_{hor}^l$  ending in this segment and sort them by increasing starting position. Let  $e_{l,i}$  be the energy demand of the stack of jobs in  $\mathcal{J}_{hor}^l$  ending in the  $i$ -th segment. We partition the stack into  $1/\epsilon$  layers of energy demand  $\epsilon e_{l,i}$  and slice the horizontal jobs overlapping the layer borders. We remove all the jobs in the bottom most layer and shift the jobs from the layers above to the left, such that they start at the latest original start position from the layer below. We repeat this procedure for each segment. By this shift, we reduce the total number of starting positions from jobs from the set  $\mathcal{J}_{hor}^l$  to  $2^l/\epsilon$ . The total energy demand of the jobs we removed is bounded by  $\epsilon e(\mathcal{J}_{hor}^l)$ . Since these jobs have a processing time of at most  $D/2^{l-1}$ , we can schedule  $2^{l-1}$  of these jobs after one another (horizontally), without violating the deadline. Hence, when scheduling these jobs fractionally, we add at most  $\epsilon e(\mathcal{J}_{hor}^l)/2^{l-1}$  to the schedule. Note that since all the jobs in set  $\mathcal{J}_{hor}^l$  have a processing time of at least  $D/2^l$ , it holds that  $\sum_{l=1}^{\lceil \log(1/\delta) \rceil} e(\mathcal{J}_{hor}^l)/2^l \leq T$  and, hence, we add at most  $\sum_{l=1}^{\lceil \log(1/\delta) \rceil} \epsilon e(\mathcal{J}_{hor}^l)/2^{l-1} \leq 2\epsilon T$  to the energy demand of the schedule, when scheduling the removed horizontal jobs. The total number of starting positions is bounded by  $\sum_{l=1}^{\lceil \log(1/\delta) \rceil} 2^l/\epsilon = (2^{\lceil \log(1/\delta) \rceil + 1} - 1)/\epsilon \in \mathcal{O}(1/\delta_w \epsilon)$ .  $\triangleleft$

### Algorithm to place horizontal and large jobs

To place the jobs in  $\mathcal{J}_{hor} \cup \mathcal{J}_{large}$ , we first guess the starting positions of the large jobs  $\mathcal{J}_{large}$  in  $\mathcal{O}(|S_{hor,large}|^{|\mathcal{J}_{large}|}) = (1/\epsilon)^{(1/\epsilon)^{\mathcal{O}(1/\epsilon)}}$ . Note that this guess affects the energy demand that is left for horizontal jobs. Next we guess which  $\mathcal{O}(1/\epsilon\delta)$  starting points in  $S_{hor,large}$  will be used after the shifting due to Claim 4. There are at most  $|S_{hor,large}|^{\mathcal{O}(1/\epsilon\delta)} = (1/\epsilon)^{(1/\epsilon)^{\mathcal{O}(1/\epsilon)}}$  possible guesses total. We call the set of guessed starting points  $\bar{S}_{h,l}$ . For each starting point in  $\bar{S}_{h,l}$ , we calculate the residual total energy demand, that is left after the guess for the large jobs. For a given  $s \in \bar{S}_{h,l}$  let  $e_{s,hor}$  be this residual total energy demand.

Consider the following linear program for horizontal jobs:

$$\begin{aligned}
\sum_{\rho \in \{p'(j) | j \in \mathcal{J}_{hor}\}} \sum_{\substack{s' \in \bar{S}_{h,l} \\ s' \leq s < s' + \rho}} x_{\rho, s'} &\leq e_{s, hor} && \forall s \in \bar{S}_{h,l} \\
\sum_{s \in \bar{S}_{h,l}} x_{\rho, s} &= \sum_{j \in \mathcal{J}_{hor}, p'(j) = \rho} e(j) && \forall \rho \in \{p'(j) | j \in \mathcal{J}_{hor}\} \\
x_{\rho, s} &\geq 0 && \forall s \in \bar{S}_{h,l}, \rho \in \{p'(j) | j \in \mathcal{J}_{hor}\}
\end{aligned}$$

The variable  $x_{\rho, s}$  denotes the total energy demand of jobs with rounded processing time  $\rho$  starting at  $s$ . The first equation ensures that the energy capacity at a starting time  $s$  is not exceeded by the jobs starting at or overlapping  $s$ . The second equation ensures that the total energy requirement of jobs with rounded processing time  $\rho$  is covered by energy demand of jobs with this processing time started in the schedule.

A basic solution to this linear program has at most  $|\bar{S}_{h,l}| + |\mathcal{J}_{hor}| = \mathcal{O}(1/\varepsilon\delta)$  non zero components. We can guess the non zero components in at most  $(|\bar{S}_{h,l}| \cdot |\mathcal{J}_{hor}|)^{|\bar{S}_{h,l}| + |\mathcal{J}_{hor}|} = (1/\varepsilon)^{\mathcal{O}(1/\varepsilon)}$ . Furthermore, we can guess their value with precision  $\mu T$  in at most  $(1/\mu)^{|\bar{S}_{h,l}| + |\mathcal{J}_{hor}|} = (1/\varepsilon)^{\mathcal{O}(1/\varepsilon)}$  guesses. Scheduling all the horizontal jobs integral and the error due to the precision add at most  $2\mu T \cdot (|\bar{S}_{h,l}| + |\mathcal{J}_{hor}|)$  to the peak energy demand. Note that  $2\mu T \cdot (|\bar{S}_{h,l}| + |\mathcal{J}_{hor}|) \leq \mathcal{O}(\varepsilon)T'$  since  $\mu \leq \mathcal{O}(\varepsilon^2\delta)$ .

After this step, we either have scheduled all given jobs or have decided that it is not possible for the given guess of  $T$  and the profile. If it is not possible for any profile, we have to increase  $T$ . If we have found a schedule, we try the next smaller value for  $T$ . Each of the steps has increased the peak energy demand by at most  $\mathcal{O}(\varepsilon)T$  above  $T$ . Besides of the job classification and rounding, each step of the algorithm is bounded by  $(1/\varepsilon)^{\mathcal{O}(1/\varepsilon)}$ . Therefore, the described algorithms fulfills the claims of Theorem 10.

## 5 Conclusion

In this paper, we presented an AEPTAS with additive term  $e_{\max}$  as well as a  $(5/3 + \varepsilon)$ -approximation for Nonpreemptive Peak Demand Minimization (NPDM). Since the lower bound for approximation algorithms for this problem is known to be  $3/2$ , this leaves a small gap between the lower bound and the approximation guarantee. Closing this gap is an interesting open question for further research, especially since for the related strip packing problem the same gap is yet to be resolved.

---

## References

- 1 Anna Adamaszek, Sarel Har-Peled, and Andreas Wiese. Approximation schemes for independent set and sparse subsets of polygons. *J. ACM*, 66(4):29:1–29:40, 2019.
- 2 Anna Adamaszek, Tomasz Kociumaka, Marcin Pilipczuk, and Michal Pilipczuk. Hardness of approximation for strip packing. *ACM Transactions on Computation Theory*, 9(3):14:1–14:7, 2017. doi:10.1145/3092026.
- 3 Soroush Alamdari, Therese Biedl, Timothy M Chan, Elyot Grant, Krishnam Raju Jampani, Srinivasan Keshav, Anna Lubiw, and Vinayak Pathak. Smart-grid electricity allocation via strip packing with slicing. In *Workshop on Algorithms and Data Structures*, pages 25–36. Springer, 2013.
- 4 Brenda S. Baker, Edward G. Coffman Jr., and Ronald L. Rivest. Orthogonal packings in two dimensions. *SIAM J. Comput.*, 9(4):846–855, 1980. doi:10.1137/0209064.

- 5 Nikhil Bansal, Alberto Caprara, and Maxim Sviridenko. A new approximation method for set covering problems, with applications to multidimensional bin packing. *SIAM J. Comput.*, 39(4):1256–1278, 2009. doi:10.1137/080736831.
- 6 Nikhil Bansal, José R. Correa, Claire Kenyon, and Maxim Sviridenko. Bin packing in multiple dimensions: Inapproximability results and approximation schemes. *Math. Oper. Res.*, 31(1):31–49, 2006. doi:10.1287/moor.1050.0168.
- 7 Nikhil Bansal and Arindam Khan. Improved approximation algorithm for two-dimensional bin packing. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 13–25. SIAM, 2014. doi:10.1137/1.9781611973402.2.
- 8 Nikhil Bansal, Andrea Lodi, and Maxim Sviridenko. A tale of two dimensional bin packing. In *FOCS*, pages 657–666, 2005.
- 9 Iwo Błądek, Maciej Drozdowski, Frédéric Guinand, and Xavier Schepler. On contiguous and non-contiguous parallel task scheduling. *Journal of Scheduling*, 18(5):487–495, 2015.
- 10 Marin Bougeret, Pierre François Dutot, Klaus Jansen, Christina Otte, and Denis Trystram. Approximating the non-contiguous multiple organization packing problem. In *IFIP International Conference on Theoretical Computer Science*, pages 316–327. Springer, 2010.
- 11 Marin Bougeret, Pierre-François Dutot, Klaus Jansen, Christina Robenek, and Denis Trystram. Approximation algorithms for multiple strip packing and scheduling parallel jobs in platforms. *Discret. Math. Algorithms Appl.*, 3(4):553–586, 2011. doi:10.1142/S1793830911001413.
- 12 Nilotpal Chakraborty, Arijit Mondal, and Samrat Mondal. Efficient scheduling of nonpreemptive appliances for peak load optimization in smart grid. *IEEE Transactions on Industrial Informatics*, 14(8):3447–3458, 2017.
- 13 Henrik I. Christensen, Arindam Khan, Sebastian Pokutta, and Prasad Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79, 2017.
- 14 Waldo Gálvez, Fabrizio Grandoni, Afrouz Jabal Ameli, Klaus Jansen, Arindam Khan, and Malin Rau. A tight  $(3/2+\epsilon)$  approximation for skewed strip packing. In Jaroslaw Byrka and Raghu Meka, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*, volume 176 of *LIPICs*, pages 44:1–44:18, 2020.
- 15 Waldo Gálvez, Fabrizio Grandoni, Afrouz Jabal Ameli, and Kamyar Khodamoradi. Approximation algorithms for demand strip packing. *CoRR*, abs/2105.08577, 2021. arXiv:2105.08577.
- 16 Waldo Gálvez, Fabrizio Grandoni, Sandy Heydrich, Salvatore Ingala, Arindam Khan, and Andreas Wiese. Approximating geometric knapsack via l-packings. In *FOCS*, pages 260–271, 2017.
- 17 Waldo Gálvez, Fabrizio Grandoni, Arindam Khan, Diego Ramirez-Romero, and Andreas Wiese. Improved approximation algorithms for 2-dimensional knapsack: Packing into multiple l-shapes, spirals and more. In *SoCG*, pages 39:1–39:17, 2021.
- 18 Waldo Gálvez, Fabrizio Grandoni, Salvatore Ingala, and Arindam Khan. Improved pseudo-polynomial-time approximation for strip packing. In *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 65, pages 9:1–9:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.FSTTCS.2016.9.
- 19 Fabrizio Grandoni, Tobias Mömke, Andreas Wiese, and Hang Zhou. A  $(5/3 + \epsilon)$ -approximation for unsplittable flow on a path: placing small tasks into boxes. In *STOC*, pages 607–619, 2018.
- 20 Rolf Harren, Klaus Jansen, Lars Prädell, and Rob van Stee. A  $(5/3 + \epsilon)$ -approximation for 2d strip packing. In Andreas Brieden, Zafer-Korcan Görgülü, Tino Krug, Erik Kropat, Silja Meyer-Nieberg, Goran Mihelcic, and Stefan Wolfgang Pickl, editors, *11th Cologne-Twente Workshop on Graphs and Combinatorial Optimization, Munich, Germany, May 29-31, 2012. Extended Abstracts*, pages 139–142, 2012.

- 21 Rolf Harren and Rob van Stee. Improved absolute approximation ratios for two-dimensional packing problems. In Irit Dinur, Klaus Jansen, Joseph Naor, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop, APPROX 2009, and 13th International Workshop, RANDOM 2009, Berkeley, CA, USA, August 21-23, 2009. Proceedings*, volume 5687 of *Lecture Notes in Computer Science*, pages 177–189. Springer, 2009. doi:10.1007/978-3-642-03685-9\_14.
- 22 Sören Henning, Klaus Jansen, Malin Rau, and Lars Schmarje. Complexity and inapproximability results for parallel task scheduling and strip packing. *Theory of Computing Systems*, 64(1):120–140, 2020. doi:10.1007/s00224-019-09910-6.
- 23 Klaus Jansen. Scheduling malleable parallel tasks: An asymptotic fully polynomial time approximation scheme. *Algorithmica*, 39(1):59–81, 2004.
- 24 Klaus Jansen. A  $(3/2 + \epsilon)$  approximation algorithm for scheduling moldable and non-moldable parallel tasks. In *Proceedings of the twenty-fourth annual ACM symposium on Parallelism in algorithms and architectures*, pages 224–235, 2012.
- 25 Klaus Jansen and Felix Land. Scheduling monotone moldable jobs in linear time. In *2018 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2018, Vancouver, BC, Canada, May 21-25, 2018*, pages 172–181. IEEE Computer Society, 2018. doi:10.1109/IPDPS.2018.00027.
- 26 Klaus Jansen and Lars Prädel. A new asymptotic approximation algorithm for 3-dimensional strip packing. In *40th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, volume 8327, pages 327–338. Springer, 2014. doi:10.1007/978-3-319-04298-5\_29.
- 27 Klaus Jansen and Malin Rau. Closing the gap for pseudo-polynomial strip packing. In *ESA*, volume 144, pages 62:1–62:14, 2019.
- 28 Klaus Jansen and Roberto Solis-Oba. Rectangle packing with one-dimensional resource augmentation. *Discret. Optim.*, 6(3):310–323, 2009. doi:10.1016/j.disopt.2009.04.001.
- 29 Klaus Jansen and Ralf Thöle. Approximation algorithms for scheduling parallel jobs. *SIAM J. Comput.*, 39(8):3571–3615, 2010. doi:10.1137/080736491.
- 30 Klaus Jansen and Guochuan Zhang. Maximizing the total profit of rectangles packed into a rectangle. *Algorithmica*, 47(3):323–342, 2007. doi:10.1007/s00453-006-0194-5.
- 31 Edward G. Coffman Jr., M. R. Garey, David S. Johnson, and Robert Endre Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM J. Comput.*, 9(4):808–826, 1980. doi:10.1137/0209062.
- 32 Mohammad M Karbasioun, Gennady Shaikhet, Ioannis Lambadaris, and Evangelos Kranakis. Asymptotically optimal scheduling of random malleable demands in smart grid. *Discrete Mathematics, Algorithms and Applications*, 10(02):1850025, 2018.
- 33 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, Chicago, Illinois, USA, 3-5 November 1982*, pages 312–320. IEEE Computer Society, 1982.
- 34 Claire Kenyon and Eric Rémila. A near-optimal solution to a two-dimensional cutting stock problem. *Math. Oper. Res.*, 25(4):645–656, 2000. doi:10.1287/moor.25.4.645.12118.
- 35 Arindam Khan, Arnab Maiti, Amatya Sharma, and Andreas Wiese. On guillotine separable packings for the two-dimensional geometric knapsack problem. In *SoCG*, pages 48:1–48:17, 2021.
- 36 Arindam Khan and Madhusudhan Reddy Pittu. On guillotine separability of squares and rectangles. In *APPROX*, pages 47:1–47:22, 2020.
- 37 Fu-Hong Liu, Hsiang-Hsuan Liu, and Prudence WH Wong. Optimal nonpreemptive scheduling in a smart grid model. In *27th International Symposium on Algorithms and Computation (ISAAC 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.
- 38 Tobias Mömke and Andreas Wiese. Breaking the barrier of 2 for the storage allocation problem. In *ICALP*, pages 86:1–86:19, 2020.

- 39 Giorgi Nadiradze and Andreas Wiese. On approximating strip packing with a better ratio than  $3/2$ . In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1491–1510. SIAM, 2016. doi:10.1137/1.9781611974331.ch102.
- 40 Anshu Ranjan, Pramod Khargonekar, and Sartaj Sahni. Offline preemptive scheduling of power demands to minimize peak power in smart grids. In *2014 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2014.
- 41 Anshu Ranjan, Pramod Khargonekar, and Sartaj Sahni. Offline first fit scheduling in smart grids. In *2015 IEEE Symposium on Computers and Communication (ISCC)*, pages 758–763. IEEE, 2015.
- 42 Anshu Ranjan, Pramod Khargonekar, and Sartaj Sahni. Smart grid power scheduling via bottom left decreasing height packing. In *2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 1128–1133. IEEE, 2016.
- 43 Malin Rau. *Useful Structures and How to Find Them: Hardness and Approximation Results for Various Variants of the Parallel Task Scheduling Problem*. dissertation, Kiel University, Kiel, Germany, 2019.
- 44 Ingo Schiermeyer. Reverse-fit: A 2-optimal algorithm for packing rectangles. In Jan van Leeuwen, editor, *Algorithms - ESA '94, Second Annual European Symposium, Utrecht, The Netherlands, September 26-28, 1994, Proceedings*, volume 855 of *Lecture Notes in Computer Science*, pages 290–299. Springer, 1994. doi:10.1007/BFb0049416.
- 45 Pierluigi Siano. Demand response and smart grids—a survey. *Renewable and sustainable energy reviews*, 30:461–478, 2014.
- 46 Daniel Dominic Sleator. A 2.5 times optimal algorithm for packing in two dimensions. *Inf. Process. Lett.*, 10(1):37–40, 1980. doi:10.1016/0020-0190(80)90121-0.
- 47 A. Steinberg. A strip-packing algorithm with absolute performance bound 2. *SIAM J. Comput.*, 26(2):401–409, 1997. doi:10.1137/S0097539793255801.
- 48 Shaojie Tang, Qiuyuan Huang, Xiang-Yang Li, and Dapeng Wu. Smoothing the energy consumption: Peak demand reduction in smart grid. In *32nd IEEE International Conference on Computer Communications (INFOCOM)*, pages 1133–1141. IEEE, 2013. doi:10.1109/INFOCOM.2013.6566904.
- 49 Sean Yaw, Brendan Mumey, Erin McDonald, and Jennifer Lemke. Peak demand scheduling in the smart grid. In *2014 IEEE international conference on smart grid communications (SmartGridComm)*, pages 770–775. IEEE, 2014.

## A Proof of Theorem 4

**Proof:**  $T_2 \leq \text{OPT}(I)$ .

► **Lemma 13.** *It holds that  $p(\mathcal{J}_{e(i) > (1/3)\text{OPT}}) + p(\mathcal{J}_{e(i) > (2/3)\text{OPT}}) \leq 2D$  and  $p(\mathcal{J}_{e(i) > (1/2)\text{OPT}}) \leq D$ .*

**Proof.** Note that jobs with energy demand larger than  $(1/3)\text{OPT}$  cannot intersect the same vertical line as jobs with energy demand larger than  $(2/3)\text{OPT}$  in an optimal schedule. Furthermore, each vertical line through an optimal schedule can intersect at most two jobs with energy demand larger than  $(1/3)\text{OPT}$ . Moreover, no vertical line can intersect two jobs from the set  $\mathcal{J}_{e(i) > (1/2)\text{OPT}}$ . The claim is a consequence. ◀

► **Corollary 14.** *The smallest value  $T$  such that  $p(\mathcal{J}_{e(i) > (1/3)T}) + p(\mathcal{J}_{e(i) > (2/3)T}) \leq 2D$  and  $p(\mathcal{J}_{e(i) > T/2}) \leq D$  is a lower bound for  $\text{OPT}$ .*

**Proof.** By Lemma 13, we know that  $p(\mathcal{J}_{e(i) > (1/3)\text{OPT}}) + p(\mathcal{J}_{e(i) > (2/3)\text{OPT}}) \leq 2D$  and obviously we have  $p(\mathcal{J}_{e(i) > \text{OPT}/2}) \leq D$ . Therefore, the smallest value such that  $p(\mathcal{J}_{e(i) > (1/3)T}) + p(\mathcal{J}_{e(i) > (2/3)T}) \leq 2D$  and  $p(\mathcal{J}_{e(i) > T/2}) \leq D$  has to be a lower bound on  $\text{OPT}$ . ◀

## 21:20 Peak Demand Minimization via Sliced Strip Packing

Note that we can find this smallest value in  $\mathcal{O}(n \log n)$  by starting with  $T = T_1$  and as long as  $p(\mathcal{J}_{e(i) \geq 1/3T}) + p(\mathcal{J}_{e(i) \geq 2/3T}) > 2D$  or  $p(\mathcal{J}_{e(i) > T/2}) > D$  update  $T$  as follows: For  $l \in [0, 1]$ , denote by  $e_l$  the energy demand of the smallest job in  $\mathcal{J}_{e(i) > l \cdot T}$  and set  $T := \min\{3e_{1/3}, (3/2)e_{2/3}, 2e_{1/2}\}$ . This iteratively excludes one job from one of the three sets. We denote this lower bound as

$$T_2 := \min\{T \mid p(\mathcal{J}_{e(i) \geq T/3}) + p(\mathcal{J}_{e(i) \geq 2T/3}) \leq 2D \wedge p(\mathcal{J}_{e(i) \geq T/2}) \leq D\}.$$

► **Lemma 15.** *Let  $w' \in [0, 1/2)$ . Then*

$$p(\mathcal{J}_{e(i) > (2/3)T_2}) > (1 - w')D \Rightarrow p(\mathcal{J}_{e(i) \in ((1/3)T_2, (2/3)T_2]}) \leq 2w'D.$$

**Proof.** We know that  $p(\mathcal{J}_{e(i) > (1/3)T_2}) + p(\mathcal{J}_{e(i) > (2/3)T_2}) \leq 2D$ . Because  $\mathcal{J}_{e(i) > (2/3)T_2} \subseteq \mathcal{J}_{e(i) > (1/3)T_2}$  and  $p(\mathcal{J}_{e(i) > (2/3)T_2}) > (1 - w')D$ , it holds that  $p(\mathcal{J}_{e(i) \in ((1/3)T_2, (2/3)T_2]}) \leq 2w'D$ . ◀

**Proof:  $T_3 \leq \text{OPT}(I)$ .** Next, we obtain a bound based on a set of jobs that do not overlap vertically in a given optimal schedule.

► **Lemma 16.** *Consider an optimal schedule and let  $\mathcal{J}_{\text{seq}}$  be a set of jobs such that no pair of jobs  $i, i' \in \mathcal{J}_{\text{seq}}$  overlaps vertically, i.e.,  $\sigma(i) + p(i) \leq \sigma(i')$  or  $\sigma(i') + p(i') \leq \sigma(i)$ . Furthermore, define  $\mathcal{J}_w := \mathcal{J}_{p(i) > (D - p(\mathcal{J}_{\text{seq}})/2)} \setminus \mathcal{J}_{\text{seq}}$ . Then there exists a vertical line through the schedule that intersects a job in  $\mathcal{J}_{\text{seq}}$  and all the jobs in  $\mathcal{J}_w$ .*

**Proof.** First note that  $(D - p(\mathcal{J}_{\text{seq}})/2) \geq D/2$ . Consider the vertical strip between  $p(\mathcal{J}_{\text{seq}})/2$  and  $(D - p(\mathcal{J}_{\text{seq}}))/2$ . Each job in  $\mathcal{J}_w$  completely overlaps this strip. Furthermore, either the strip itself contains a job in  $\mathcal{J}_{\text{seq}}$ , in which case the claim is trivially true, or on each position on both sides of the strip there is a job from  $\mathcal{J}_{\text{seq}}$ . Assume the latter case. Since the jobs in  $\mathcal{J}_w$  have a time demand strictly larger than  $(D - p(\mathcal{J}_{\text{seq}})/2)$ , there exists an  $\sigma > 0$  such that the vertical line at  $(D - p(\mathcal{J}_{\text{seq}})/2) + \sigma$  as well is overlapped by all the jobs in this set. Since this line intersects also a job from the set  $\mathcal{J}_{\text{seq}}$ , the claim follows. ◀

► **Corollary 17.** *Let  $\mathcal{J}_{\text{seq}}$  be a set of jobs such that  $p(\mathcal{J}_{\text{seq}}) \leq D$  and consider  $\mathcal{J}_w := \mathcal{J}_{p(i) > D - p(\mathcal{J}_{\text{seq}})/2} \setminus \mathcal{J}_{\text{seq}}$ . Furthermore let  $i_\perp \in \mathcal{J}_{\text{seq}}$  be the job with the smallest energy demand. Then it holds that  $\min\{e(i_\perp) + e(\mathcal{J}_w), 2e(i_\perp)\} \leq \text{OPT}$ .*

**Proof.** Consider an optimal solution. If two jobs from the set  $\mathcal{J}_{\text{seq}}$  intersect the same vertical line,  $2e(i_\perp)$  is obviously a lower bound on OPT. On the other hand, if in any optimal schedule there does not exist a pair of jobs from  $\mathcal{J}_{\text{seq}}$  that overlap the same vertical line, we know by Lemma 16 that there exists a job in  $\mathcal{J}_{\text{seq}}$  that overlaps with all the jobs in  $\mathcal{J}_w$  and therefore  $\text{OPT} \geq e(i_\perp) + e(\mathcal{J}_w)$  in this case. ◀

From Corollary 17, we derive a lower bound on OPT. For a given  $k \in [n]$ , we define  $\mathcal{J}_k$  to be the set of the  $k$  jobs with the largest energy demand in  $\mathcal{J}$  and  $\mathcal{J}'_k$  to be the set of the  $k$  jobs with the largest energy demand in  $\mathcal{J} \setminus \mathcal{J}_{p(i) > D/2}$ . Let  $i_k$  and  $i'_k$  be the jobs with the smallest energy demand in  $\mathcal{J}_k$  and  $\mathcal{J}'_k$ , respectively. We define:

$$T_{3,a} := \max\{\min\{e(i_k) + e(\mathcal{J}_{p(i) > D - p(\mathcal{J}_k)/2} \setminus \mathcal{J}_k), 2e(i_k)\} \mid k \in \{1, \dots, n\}, p(\mathcal{J}_k) \leq D\},$$

$$T_{3,b} := \max\{\min\{e(i'_k) + e(\mathcal{J}_{p(i) > D - p(\mathcal{J}'_k)/2}), 2e(i'_k)\} \mid k \in \{1, \dots, n\}, p(\mathcal{J}'_k) \leq D\},$$

and finally  $T_3 = \max\{T_{3,a}, T_{3,b}\}$ . Note that  $\mathcal{J}'_k$  and  $\mathcal{J}_{p(i) > D - p(\mathcal{J}'_k)/2}$  are disjoint, since  $\mathcal{J}'_k$  contains only jobs with processing time at most  $D/2$  and  $\mathcal{J}_{p(i) > D - p(\mathcal{J}'_k)/2}$  contains only jobs with processing time larger than  $D/2$ , and hence, by Corollary 17,  $T_3$  is a lower bound on OPT. For this lower bound, we prove the following property.



► **Lemma 18.** *Let  $T = \max\{T_1, T_2, T_3\}$ ,  $w \in (0, 1/2)$  and  $h \in (1/2, 1]$  as well as  $\mathcal{J}_h := \mathcal{J}_{e(i) \geq hT}$  and  $\mathcal{J}_w := \mathcal{J}_{p(i) > (1/2+w/2)D} \setminus \mathcal{J}_h$ . It holds that*

$$p(\mathcal{J}_h) \geq (1-w)D \Rightarrow e(\mathcal{J}_w) \leq (1-h)T.$$

**Proof.** Since  $T \geq T_2$ , it holds that  $p(\mathcal{J}_h) \leq D$ . By construction of  $T_3$  for each job  $j \in \mathcal{J}_h$ , it holds that  $e(j) + e(\mathcal{J}_{p(i) > D - p(\mathcal{J}_{e(i) \geq e(j)})/2} \setminus \mathcal{J}_{e(i) \geq e(j)}) \leq T_3$ , because  $2e(j) > T_3$  (and  $T_3 \geq \min\{2e(j), e(j) + e(\mathcal{J}_{p(i) > D - p(\mathcal{J}_{e(i) \geq e(j)})/2} \setminus \mathcal{J}_{e(i) \geq e(j)})\}$ ). Furthermore, note that  $\mathcal{J}_h = \mathcal{J}_{e(i) \geq e(j)}$  for the job  $j$  with the smallest energy demand in  $\mathcal{J}_h$ .

Therefore, if  $p(\mathcal{J}_h) \geq (1-w)D$ , it holds that  $\mathcal{J}_{p(i) > D - (1-w)D/2} \subseteq \mathcal{J}_{p(i) > D - p(\mathcal{J}_h)/2}$  and hence,

$$\begin{aligned} hT + e(\mathcal{J}_w) &= hT + e(\mathcal{J}_{p(i) > D - (1-w)D/2} \setminus \mathcal{J}_h) \\ &\leq e(j) + e(\mathcal{J}_{p(i) > D - p(\mathcal{J}_h)/2} \setminus \mathcal{J}_h) \leq T_3 \leq T. \end{aligned} \quad \blacktriangleleft$$

► **Lemma 19.** *Let  $T = \max\{T_1, T_2, T_3\}$ ,  $w \in (1/2, 1]$  and  $h \in (1/2, 1]$  as well as  $\mathcal{J}_w := \mathcal{J}_{p(i) \geq wD}$  and  $\mathcal{J}_h := \mathcal{J}_{e(i) > hT} \setminus \mathcal{J}_{p(i) > D/2}$ . It holds that*

$$e(\mathcal{J}_w) > (1-h)T \Rightarrow p(\mathcal{J}_h) \leq 2(1-w)D.$$

**Proof.** Let  $e(\mathcal{J}_w) > (1-h)T$ . Since for each job in  $\mathcal{J}_h$  it holds that  $e(i) > T/2 \geq T_{3,b}/2$ , by definition of  $T_{3,b}$ , for each  $j \in \mathcal{J}_h$  it holds that  $e(j) + e(\mathcal{J}_{p(i) > D - p(\mathcal{J}_{e(i) \geq e(j)})/2}) \leq T$ . Therefore for the smallest job  $j \in \mathcal{J}_h$ , it holds that  $e(j) + e(\mathcal{J}_{p(i) > D - p(\mathcal{J}_h)/2}) \leq T$ .

For contradiction assume that  $p(\mathcal{J}_h) > 2(1-w)D$ . Note that in this case  $D - p(\mathcal{J}_h)/2 < wD$  and hence  $e(\mathcal{J}_{p(i) > D - p(\mathcal{J}_h)/2}) \geq e(\mathcal{J}_D) > (1-h)T$ . As a consequence  $e(j) + e(\mathcal{J}_{p(i) > D - p(\mathcal{J}_h)/2}) > hT + (1-h)T = T$ , a contradiction.  $\blacktriangleleft$

**Proof:  $T_4 \leq \text{OPT}(I)$ .**

► **Lemma 20.** *Consider an optimal schedule and let  $\mathcal{J}_{\text{seq}} \subseteq \mathcal{J}$  be a set of jobs such that no pair of jobs  $j, j' \in \mathcal{J}_{\text{seq}}$  overlaps vertically, i.e.,  $\sigma(j) + p(j) \leq \sigma(j')$  or  $\sigma(j') + p(j') \leq \sigma(j)$ . Let  $\mathcal{J}_D \subseteq \mathcal{J}_{p(j) > (\max\{D - p(\mathcal{J}_{\text{seq}}), D/2\})} \setminus \mathcal{J}_{\text{seq}}$ . Then there exists a vertical line through the schedule that intersects a job in  $\mathcal{J}_{\text{seq}}$  and a subset  $\mathcal{J}_{W'} \subseteq \mathcal{J}_D$  with  $e(\mathcal{J}'_D) \geq e(\mathcal{J}_D)/2$ .*

**Proof.** First, we consider the trivial cases. If a job from  $\mathcal{J}_{\text{seq}}$  overlaps the vertical line at  $D/2$  the claim is trivially true, since all the jobs from  $\mathcal{J}_D$  overlap  $D/2$ . On the other hand, if all the jobs in  $\mathcal{J}_{\text{seq}}$  are left or right of  $D/2$ , it holds that  $p(\mathcal{J}_{\text{seq}}) \leq D/2$  and one of the jobs has a distance of at most  $D/2 - p(\mathcal{J}_{\text{seq}})$  from  $D/2$ . This job has to be overlapped by all the jobs from  $\mathcal{J}_D$  since they have a width larger than  $D - p(\mathcal{J}_{\text{seq}})$ .

Otherwise, consider the vertical line  $L_l$  through the right border of the rightmost job from  $\mathcal{J}_{\text{seq}}$  that is left of  $D/2$  and the vertical line  $L_r$  through the left border of the leftmost job from  $\mathcal{J}_{\text{seq}}$  that is right of  $D/2$ . Note that  $L_l$  and  $L_r$  have a distance of at most  $(D - p(\mathcal{J}_{\text{seq}}))$ . Consider the set  $\mathcal{J}_{D,l} \subseteq \mathcal{J}_D$  that is intersected by the vertical line  $L_l$ . Note that the residual jobs in  $\mathcal{J}_{D,r} := \mathcal{J}_D \setminus \mathcal{J}_{D,l}$  all overlap the vertical line at  $L_l + (D - p(\mathcal{J}_{\text{seq}})) \geq L_r$  and hence  $L_r$  as well. Since  $\mathcal{J}_{D,r} \cup \mathcal{J}_{D,l} = \mathcal{J}_D$ , one of the two sets has an energy demand of at least  $e(\mathcal{J}_D)/2$ . Finally, note that there exists a small enough  $\sigma > 0$  such that  $L_l - \sigma$  and  $L_r + \sigma$  overlap the same set of wide jobs as  $L_l$  and  $L_r$  as well as the corresponding job in  $\mathcal{J}_{\text{seq}}$ .  $\blacktriangleleft$

► **Corollary 21.** *Let  $\mathcal{J}_{\text{seq}}$  be a set of jobs such that  $p(\mathcal{J}_{\text{seq}}) \leq D$  and consider  $\mathcal{J}_D := \mathcal{J}_{p(i) > (\max\{D - p(\mathcal{J}_{\text{seq}}), D/2\})} \setminus \mathcal{J}_{\text{seq}}$ . Furthermore let  $i_\perp \in \mathcal{J}_{\text{seq}}$  be the job with the smallest energy demand. Then it holds that  $\min\{e(i_\perp) + e(\mathcal{J}_D)/2, 2e(i_\perp)\} \leq \text{OPT}$ .*

**Proof.** Consider an optimal solution. If two jobs from the set  $\mathcal{J}_{\text{seq}}$  intersect the same vertical line,  $2e(i_{\perp})$  is obviously a lower bound on OPT. Otherwise, if in any optimal schedule there does not exist a pair of jobs from  $\mathcal{J}_{\text{seq}}$  that overlap the same vertical line, we know by Lemma 20 that there exists a job in  $\mathcal{J}_{\text{seq}}$  that overlaps with a set  $\mathcal{J}'_D \subseteq \mathcal{J}_D$  such that  $e(\mathcal{J}'_D) \geq e(\mathcal{J}_D)/2$  and therefore  $\text{OPT} \geq e(i_{\perp}) + e(\mathcal{J}_D)/2$  in this case.  $\blacktriangleleft$

Define  $\mathcal{J}_k$  as the set of the  $k$  jobs with largest energy demand. Furthermore, define  $\mathcal{J}_{D,k} := \mathcal{J}_{p(i) > (\max\{D-p(\mathcal{J}_k), D/2\})} \setminus \mathcal{J}_k$ . Let  $i_k$  be the job with the smallest energy demand in  $\mathcal{J}_k$ . We define the value  $T_4$ , which by Corollary 21 is a lower bound for OPT as follows:

$$T_4 := \max\{\min\{2e(i_k), e(i_k) + e(\mathcal{J}_{D,k})/2\} \mid k \in \{1, \dots, n\}, p(\mathcal{J}_k) \leq D\}.$$

Given two disjoint sets of jobs  $\mathcal{J}_{\text{seq}}$  and  $\mathcal{J}_D$ , we say they are placed *L-shaped*, if the jobs  $i \in \mathcal{J}_D$  are placed such that  $\sigma(i) + p(i) = D$ , while the jobs in  $\mathcal{J}_{\text{seq}}$  are sorted by energy demand and placed left-aligned most demanding to the left, see Figure 1a.

► **Lemma 22.** *Let  $T = \max\{T_1, T_2, T_3, T_4\}$ . If we place  $\mathcal{J}_{\text{seq}} := \mathcal{J}_{e(i) > T/2}$  and  $\mathcal{J}_D := \mathcal{J}_{p(i) > D/2} \setminus \mathcal{J}_{\text{seq}}$  L-shaped, the schedule has a height of at most  $T + e(\mathcal{J}_D)/2 \leq (3/2)\text{OPT}$ .*

**Proof.** Consider a vertical line  $L$  through the generated schedule. If  $L$  does not intersect a job from  $\mathcal{J}_{\text{seq}}$ , the intersected jobs have a height of at most  $e(\mathcal{J}_D) \leq T$ . Otherwise, let  $i_L \in \mathcal{J}_{\text{seq}}$  and  $\mathcal{J}_{W,L} \subseteq \mathcal{J}_D$  be the jobs intersected by  $L$  and define  $\mathcal{J}_{\text{seq},L} := \mathcal{J}_{e(i) \geq e(i_L)}$ . Note that by definition of the schedule, it holds that  $\mathcal{J}_{W,L} \subseteq \mathcal{J}_{p(i) > (\max\{D-p(\mathcal{J}_{\text{seq},L}), D/2\})} \setminus \mathcal{J}_{\text{seq},L}$ . Since  $e(i_L) > T_4/2$ , it holds that  $T_4 \geq e(i_L) + e(\mathcal{J}_{W,L})/2$ , by definition of  $T_4$ . As a consequence  $e(i_L) + e(\mathcal{J}_{W,L}) \leq T + e(\mathcal{J}_{W,L})/2 \leq T + e(\mathcal{J}_D)/2 \leq (3/2)\text{OPT}$ .  $\blacktriangleleft$

## B Proof of Theorem 5 (First Steinberg Case)

**Proof.** We place jobs that are very time consuming or very energy demanding in an ordered fashion, while the residual jobs will be placed using Steinberg's Algorithm, see Figure 1b. We define  $\mathcal{J}_D := \mathcal{J}_{p(j) > (1/2+w)D} \setminus \mathcal{J}_{e(j) > T/2}$  to be the set of jobs with large processing times excluding jobs with large energy demands. We place each job  $j \in \mathcal{J}_D$  such that  $\sigma(j) = D - p(j)$ . All the jobs in  $\mathcal{J}_{e(j) > T/2}$  are sorted by energy demand and placed left aligned, most demanding first inside the schedule area. Let  $\rho := e(\mathcal{J}_D)/T$  and let  $e_{(1-2w)D}$  denote the energy demand of the job in  $\mathcal{J}_{e(j) > T/2}$  at position  $(1-2w)D$ . Then  $e_{(1-2w)D} \geq (2/3)T$ . By Lemma 18 and the choice of  $T$ , we know that  $e_{(1-2w)D} + e(\mathcal{J}_D) \leq T \leq \text{OPT}$  and hence  $\rho \leq (1/3)$ . Let  $L$  be a vertical line though the schedule, that is at or strictly left of  $(1/2-w)D$  and intersects a job from  $\mathcal{J}_{e(j) > T/2}$  and all the jobs from  $\mathcal{J}_D$ . By Lemma 22 at and left of  $L$  the peak energy demand of the schedule is bounded by  $(1 + \rho/2)T$ . On the other hand, right of  $L$  the energy demand of the schedule does not increase compared to  $L$ . As a consequence, the peak energy demand in the current schedule is bounded by  $(1 + \rho/2)T \leq (7/6)T$ . Furthermore, we know that right of  $(1-2w)D$  the schedule has a peak energy demand of at most  $T$ . Consider the set of jobs  $\mathcal{J}_{e(j) \in ((1/3)T, (1/2)T]}$ . By Lemma 15 we know  $p(\mathcal{J}_{e(j) \in ((1/3)T, (1/2)T]}) \leq 2w \cdot D$ , since  $\mathcal{J}_{e(j) > (2/3)T} \geq (1-w)D$ . Now we consider two cases.

**Case A.** If  $\varepsilon \leq \rho/2$ , we place all the jobs in  $\mathcal{J}_M := \mathcal{J}_{e(j) \in ((1/3)T, (1/2)T]}$  right-aligned next to each other inside the strip. Since they have an energy demand of at most  $(1/2)T$  and right of  $(1-2w)D$  the schedule has a peak energy demand of at most  $T$ , the peak energy demand of  $(5/3)T$  is not exceeded after adding these jobs. Define  $\lambda := p(\mathcal{J}_M)/D$ . Now at each point on the x-axis between 0 and  $a := (1-\lambda)D$  the schedule has an energy demand of at most

$(1 + \rho/2)T$ , and, therefore, we can use an energy demand of  $b := (2/3 - \rho/2 + \varepsilon)T$  to place the residual jobs. Let  $\mathcal{J}_{res}$  denote the set of residual jobs that still have to be placed. Note that each job in  $\mathcal{J}_{res}$  has an energy demand of at most  $(1/3)T$  and a processing time of at most  $(1/2 + w)D$  and the total area of these jobs can be bound by

$$\begin{aligned} \text{work}(\mathcal{J}_{res}) &\leq DT - (2/3)T \cdot (1 - w)D - \rho T \cdot (1/2 + w)D - (1/3)T \cdot \lambda D \\ &= (1/3 + (2/3)w - \rho(1/2 + w) - \lambda/3)DT, \end{aligned}$$

and hence  $2\text{work}(\mathcal{J}_{res}) \leq (2/3 + (4/3)w - \rho(1 + 2w) - (2/3)\lambda)DT$ . On the other hand, it holds that

$$\begin{aligned} &ab - (2p_{\max} - a)_+(2e_{\max} - b)_+ \\ &= (2/3 - \rho/2 + \varepsilon)T(1 - \lambda)D \\ &\quad - ((2(1/2 + w) - (1 - \lambda))D)_+((2(1/3) - (2/3 - \rho/2 + \varepsilon))T)_+ \\ &= (2/3 - \rho/2 + \varepsilon - (2/3)\lambda + (\rho/2 - \varepsilon)\lambda - (2w + \lambda)_+(\rho/2 - \varepsilon)_+)DT \\ &= (2/3 + \varepsilon(1 + 2w) - (1/2 + w)\rho - (2/3)\lambda)DT, \end{aligned}$$

since  $\rho/2 - \varepsilon \geq 0$ . Hence Steinberg's condition is fulfilled if  $(4/3)w - \rho(w + 1/2) \leq \varepsilon(1 + 2w)$ , which is true since  $w \leq (3/4)\varepsilon$ .

**Case B.** On the other hand, if  $\rho/2 < \varepsilon$ , it holds that  $(2/3 + \varepsilon - \rho/2)/2 \geq 1/3$ , and we consider the set  $\mathcal{J}_M := \mathcal{J}_{e(j) \in ((2/3 + \varepsilon - \rho/2)/2)T, (1/2)T]}$ , instead of the set  $\mathcal{J}_{e(j) \in ((1/3)T, (1/2)T]}$ , and place it right-aligned. Again, we define  $\lambda := p(\mathcal{J}_M)$ . Now, each job in  $\mathcal{J}_{res}$  has an energy demand of at most  $(1/3 + \varepsilon/2 - \rho/4)T$  and a processing time of at most  $(1/2 + w)D$ . The total area of these jobs can be bounded by

$$\begin{aligned} \text{work}(\mathcal{J}_{res}) &\leq DT - (2/3)T \cdot (1 - w)D - \rho T \cdot (1/2 + w)D - (1/3 + \varepsilon/2 - \rho/4)T \cdot \lambda D \\ &= (1/3 + (2/3)w - \rho(1/2 + w) - \lambda(1/3 + \varepsilon/2 - \rho/4))DT, \end{aligned}$$

and hence  $2\text{work}(\mathcal{J}_{res}) \leq (2/3 + 4/3w - \rho(1 + 2w) - (2/3 + \varepsilon - \rho/2)\lambda)DT$ . On the other hand, it holds that

$$\begin{aligned} &ab - (2p_{\max} - a)_+(2e_{\max} - b)_+ \\ &= (2/3 - \rho/2 + \varepsilon)T(1 - \lambda)D - (2(1/2 + w)D - (1 - \lambda)D)_+(2(1/3 + \varepsilon/2 - \rho/4)T \\ &\quad - (2/3 - \rho/2 + \varepsilon)T)_+ \\ &= (2/3 + \varepsilon - \rho/2 - (2/3 - \rho/2 + \varepsilon)\lambda)DT, \end{aligned}$$

Hence, Steinberg's condition is fulfilled if  $(4/3 - 2\rho)w - \rho/2 \leq \varepsilon$ , which is true since  $w \leq (3/4)\varepsilon$ .

Therefore, in both cases we use Steinberg's algorithm to place the jobs  $\mathcal{J}_{res}$  inside a rectangular container  $C$  of height  $(2/3 + \varepsilon - \rho)T$  and width  $(1 - \lambda)D$ , which in turn is positioned at  $\sigma(C) = 0$ . ◀

## C Proof of Theorem 6 (Second Steinberg Case)

**Proof.** In the first step, we place all the jobs in  $\mathcal{J}_D := \mathcal{J}_{p(j) > D/2}$  and  $\mathcal{J}_{seq} := \mathcal{J}_{e(j) > T/2} \setminus \mathcal{J}_D$  L-shaped. By Lemma 22 the resulting schedule has a peak energy demand of at most  $(3/2)\text{OPT}$ . Let  $e(\mathcal{J}_D) := (2/3 + \rho)T$  and  $p(\mathcal{J}_{seq}) := \lambda D$ . By Lemma 19, we know that, since  $e(\mathcal{J}_{p(j) \geq (3/4)D}) > (2/3)T$ , that  $\lambda D \leq 2(D - (3/4)D) = D/2$ .

## 21:24 Peak Demand Minimization via Sliced Strip Packing

The total amount of work  $\text{work}(\mathcal{J}_{res})$  of the residual jobs is bounded by

$$\text{work}(\mathcal{J}_{res}) \leq DT - (3/4)D \cdot (2/3)T - (1/2)D \cdot \rho T - \lambda D \cdot (1/2)T = (1/2 - \rho/2 - \lambda/2)DT.$$

On the other hand, there is a rectangular area with time  $a := (1 - \lambda)D$  and energy  $b := ((5/3) - (2/3 + \rho))T = (1 - \rho)T \geq (1/2)T$  where we can place the residual jobs. We will place the residual jobs into this area using Steinberg's algorithm. This is possible if the Steinberg's condition  $2\text{work}(\mathcal{J}_{res}) \leq ab - (2 \cdot p_{\max}(\mathcal{J}_{res}) - a)_+(2 \cdot e_{\max}(\mathcal{J}_{res}) - b)_+$  is fulfilled and each job fits inside the schedule area. Since  $p_{\max}(\mathcal{J}_{res}) \leq D/2 \leq a$  and  $e_{\max}(\mathcal{J}_{res}) \leq T/2 < b$ , it holds that

$$\begin{aligned} & ab - (2 \cdot p_{\max}(\mathcal{J}_{res}) - a)_+(2 \cdot e_{\max}(\mathcal{J}_{res}) - b)_+ \\ &= (1 - \lambda)D \cdot (1 - \rho)T - (D - (1 - \lambda)D)_+(T - (1 - \rho)T)_+ \\ &= (1 - \lambda - \rho)D \cdot T \\ &= 2(1/2 - \rho/2 - \lambda/2)DT \geq 2\text{work}(\mathcal{J}_{res}). \end{aligned}$$

The condition is fulfilled, and we can use the free rectangular area to place the residual jobs. ◀

# Tight Approximation Algorithms For Geometric Bin Packing with Skewed Items

Arindam Khan ✉

Department of Computer Science and Automation, Indian Institute of Science, Bengaluru, India

Eklavya Sharma ✉

Department of Computer Science and Automation, Indian Institute of Science, Bengaluru, India

---

## Abstract

In the *Two-dimensional Bin Packing (2BP)* problem, we are given a set of rectangles of height and width at most one and our goal is to find an axis-aligned nonoverlapping packing of these rectangles into the minimum number of unit square bins. The problem admits no APTAS and the current best approximation ratio is 1.406 by Bansal and Khan [SODA'14]. A well-studied variant of the problem is *Guillotine Two-dimensional Bin Packing (G2BP)*, where all rectangles must be packed in such a way that every rectangle in the packing can be obtained by recursively applying a sequence of end-to-end axis-parallel cuts, also called *guillotine cuts*. Bansal, Lodi, and Sviridenko [FOCS'05] obtained an APTAS for this problem. Let  $\lambda$  be the smallest constant such that for every set  $I$  of items, the number of bins in the optimal solution to G2BP for  $I$  is upper bounded by  $\lambda \text{opt}(I) + c$ , where  $\text{opt}(I)$  is the number of bins in the optimal solution to 2BP for  $I$  and  $c$  is a constant. It is known that  $4/3 \leq \lambda \leq 1.692$ . Bansal and Khan [SODA'14] conjectured that  $\lambda = 4/3$ . The conjecture, if true, will imply a  $(4/3 + \varepsilon)$ -approximation algorithm for 2BP. According to convention, for a given constant  $\delta > 0$ , a rectangle is *large* if both its height and width are at least  $\delta$ , and otherwise it is called *skewed*. We make progress towards the conjecture by showing  $\lambda = 4/3$  for *skewed instance*, i.e., when all input rectangles are skewed. Even for this case, the previous best upper bound on  $\lambda$  was roughly 1.692. We also give an APTAS for 2BP for skewed instance, though general 2BP does not admit an APTAS.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Packing and covering problems

**Keywords and phrases** Geometric bin packing, guillotine separability, approximation algorithms

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.22

**Category** APPROX

**Related Version** *ArXiv*: <https://arxiv.org/abs/2105.02827>

## 1 Introduction

Two-dimensional Bin Packing (2BP) is a well-studied problem in combinatorial optimization. It finds numerous applications in logistics, databases, and cutting stock. In 2BP, we are given a set of  $n$  rectangular items and square bins of side length 1. The  $i^{\text{th}}$  item is characterized by its width  $w(i) \in (0, 1]$  and height  $h(i) \in (0, 1]$ . Our goal is to find an axis-aligned nonoverlapping packing of these items into the minimum number of square bins of side length 1. There are two well-studied variants: (i) where the items cannot be rotated, and (ii) they can be rotated by 90 degrees.

As is conventional in bin packing, we focus on asymptotic approximation algorithms. For any optimization problem, the asymptotic approximation ratio (AAR) of algorithm  $\mathcal{A}$  is defined as  $\lim_{m \rightarrow \infty} \sup_{I: \text{opt}(I)=m} (\mathcal{A}(I)/\text{opt}(I))$ , where  $\text{opt}(I)$  is the optimal objective value and  $\mathcal{A}(I)$  is the objective value of the solution output by algorithm  $\mathcal{A}$ , respectively, on input  $I$ . Intuitively, AAR captures the algorithm's behavior when  $\text{opt}(I)$  is large. We call a



© Arindam Khan and Eklavya Sharma;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 22; pp. 22:1–22:23



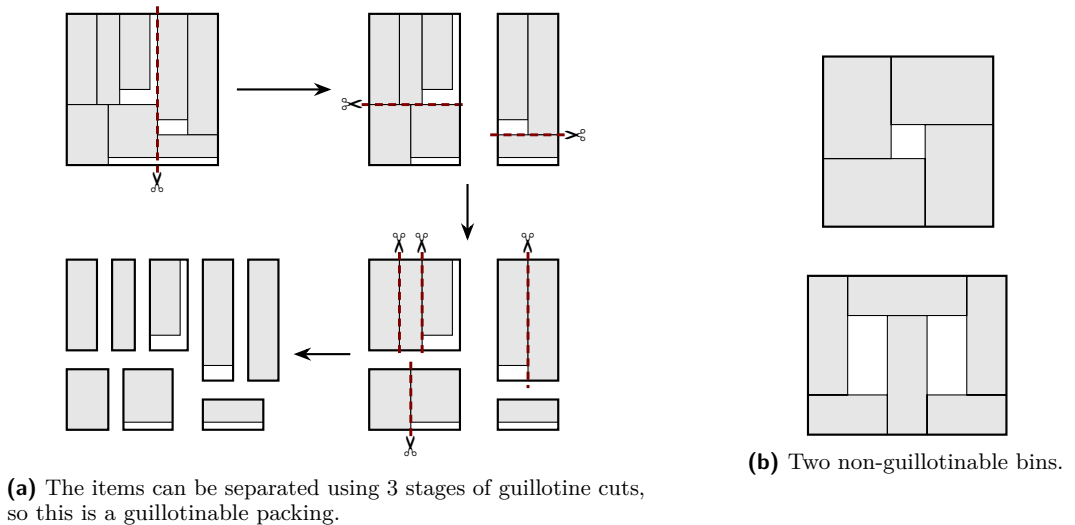
Leibniz International Proceedings in Informatics

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

bin packing algorithm  $\alpha$ -asymptotic-approximate iff its AAR is at most  $\alpha$ . An Asymptotic Polynomial-Time Approximation Scheme (APTAS) is an algorithm that accepts a parameter  $\varepsilon$  and has AAR of  $(1 + \varepsilon)$ .

2BP is a generalization of classical 1-D bin packing problem [24, 15]. However, unlike 1-D bin packing, 2BP does not admit an APTAS unless  $P=NP$  [5]. In 1982, Chung, Garey, and Johnson [13] gave an approximation algorithm with AAR 2.125 for 2BP. Caprara [9] improved the AAR to  $T_\infty (\approx 1.691)$ . After a series of works [4, 25, 7], the AAR was gradually improved to  $1 + \ln(1.5) + \varepsilon \approx 1.405$  (for both the rotational and non-rotational versions of 2BP). The best-known lower bounds on the AAR for 2BP are  $1 + 1/3792$  and  $1 + 1/2196$  [11], for the versions with and without rotations, respectively.

In the context of geometric packing, guillotine cuts are well-studied and heavily used in practice [41]. The notions of *guillotine cuts* and *k-stage packing* were introduced by Gilmore and Gomory in their seminal paper [22] on the cutting stock problem. In *k-stage packing*, the items can be separated from each other using *k* stages of axis-parallel end-to-end cuts, also called guillotine cuts, where in each stage, either all cuts are vertical or all cuts are horizontal. In each stage, each rectangular region obtained in the previous stage is considered separately and can be cut again using guillotine cuts. Note that in the cutting process we change the orientation (vertical or horizontal) of the cuts  $k - 1$  times. 2-stage packing, also called *shelf packing*, has been studied extensively. A packing is called *guillotinable* iff it is a *k-stage packing* for some integer *k*. See Figure 1 for examples. Caprara et al. [10] gave an APTAS for 2-stage 2BP. Bansal et al. [8] showed an APTAS for guillotine 2BP.



■ **Figure 1** Examples of guillotinable and non-guillotinable packing.

The presence of an APTAS for guillotine 2BP raises an important question: can the optimal solution to guillotine 2BP be used as a good approximate solution to 2BP? Formally, let  $\text{opt}(I)$  and  $\text{opt}_g(I)$  be the minimum number of bins and the minimum number of guillotinable bins, respectively, needed to pack items  $I$ . Let  $\lambda$  be the smallest constant such that for some constant  $c$  and for every set  $I$  of items, we get  $\text{opt}_g(I) \leq \lambda \text{opt}(I) + c$ . Then  $\lambda$  is called the Asymptotic Price of Guillotinability (APoG). It is easy to show that

$\text{APoG} \geq 4/3$ <sup>1</sup>. Bansal and Khan [7] conjectured that  $\text{APoG} = 4/3$ . If true, this would imply a  $(4/3 + \varepsilon)$ -asymptotic-approximation algorithm for 2BP [8]. However, the present upper bound on  $\text{APoG}$  is only  $T_\infty$  ( $\approx 1.691$ ), due to Caprara’s HDH algorithm [9] for 2BP, which produces a 2-stage packing.

APTASes are known for some special cases for 2BP, such as when all items are squares [5] or when all rectangles are small in both dimensions [14]. Another important class is *skewed* rectangles. We say that a rectangle is  $\delta$ -large if, for some constant  $\delta > 0$ , its width and height are more than  $\delta$ ; otherwise, the rectangle is  $\delta$ -skewed. We just say that a rectangle is large or skewed when  $\delta$  is clear from the context. An instance of 2BP is skewed if all the rectangles in the input are skewed. Skewed instances are important in geometric packing (see Section 1.1). This special case is practically relevant [18]: e.g., in scheduling, it captures scenarios where no job can consume a significant amount of a shared resource (energy, memory space, etc.) for a significant amount of time. Even for skewed instance for 2BP, the best known AAR is 1.406 [7]. Also, for skewed instance, the best known upper bound on  $\text{APoG}$  is  $T_\infty \approx 1.691$ .

## 1.1 Related Works

Multidimensional packing problems are fundamental in combinatorial optimization [12]. Vector packing (VP) is another variant of bin packing, where the input is a set of vectors in  $[0, 1]^d$  and the goal is to partition the vectors into the minimum number of parts (bins) such that in each part, the sum of vectors is at most 1 in every coordinate. The present best approximation algorithm attains an AAR of  $(0.807 + \ln(d + 1))$  [6] and there is a matching  $\Omega(\ln d)$ -hardness [37]. Generalized multidimensional packing [33, 32] generalizes both geometric and vector packing.

In two-dimensional strip packing (2SP) [14, 40], we are given a set of rectangles and a bounded width strip. The goal is to obtain an axis-aligned nonoverlapping packing of all rectangles such that the height of the packing is minimized. The best-known approximation ratio for 2SP is  $5/3 + \varepsilon$  [23] and it is NP-hard to obtain better than  $3/2$ -approximation. However, there exist APTASes for the problem [28, 26]. In two-dimensional knapsack (2GK) [27], the rectangles have associated profits and our goal is to pack the maximum profit subset into a unit square knapsack. The present best polynomial-time (resp. pseudopolynomial-time) approximation ratio for 2GK is 1.809 [20] (resp.  $4/3$  [21]). These geometric packing problems have also been studied for  $d$ -dimensions ( $d \geq 2$ ) [39].

2SP and 2GK are also well-studied under guillotine packing. Seiden and Woeginger [38] gave an APTAS for guillotine 2SP. Khan et al. [29] recently gave a pseudopolynomial-time approximation scheme for guillotine 2GK. Recently, guillotine cuts [35] have received attention due to their connection with the maximum independent set of rectangles (MISR) problem [2]. In MISR, we are given a set of possibly overlapping rectangles and the goal is to find the maximum cardinality set of rectangles so that there is no pairwise overlap. It was noted in [30, 1] that for any set of  $n$  non-overlapping axis-parallel rectangles, if there is a guillotine cutting sequence separating  $\alpha n$  of them, then it implies a  $1/\alpha$ -approximation for MISR.

Skewed instance is an important special case in these problems. In some problems, such as MISR and 2GK, if all items are  $\delta$ -large then we can solve them exactly in polynomial time. So, the inherent difficulty of these problems lies in skewed instances. For VP, hard instances are again skewed, e.g., Bansal, Eliáš and Khan [6] showed that hard instances for 2-D VP

<sup>1</sup> Consider a set  $I$  of items containing  $2m$  rectangles of width 0.6 and height 0.4 and  $2m$  rectangles of width 0.4 and height 0.6. Then  $\text{opt}(I) = m$  and  $\text{opt}_g(I) = \lceil 4m/3 \rceil$ .

(for a class of algorithms called *rounding based algorithms*) are skewed instances, where one dimension is  $1 - \varepsilon$  and the other dimension is  $\varepsilon$ . Galvez et al. [18] recently studied strip packing when all items are skewed. For skewed instances, they showed  $(3/2 - \varepsilon)$  hardness of approximation and a matching  $(3/2 + \varepsilon)$ -approximation algorithm. For 2GK, when the height of each item is at most  $\varepsilon^3$ , a  $(1 - 72\varepsilon)$ -approximation algorithm is known [17].

## 1.2 Our Contributions

We study 2BP for the special case of  $\delta$ -skewed rectangles, where  $\delta \in (0, 1/2]$  is a constant.

First, we make progress towards the conjecture [7] that  $\text{APoG} = 4/3$ . Even for skewed rectangles, we only knew  $4/3 \leq \text{APoG} \leq T_\infty (\approx 1.691)$ . We resolve the conjecture for skewed rectangles, by giving lower and upper bounds of roughly  $4/3$  when  $\delta$  is a small constant.

Specifically, we give an algorithm for 2BP, called  $\text{skewed4Pack}_\varepsilon$ , that takes a parameter  $\varepsilon \in (0, 1)$  as input. For a set  $I$  of  $\delta$ -skewed rectangles, we show that when  $\delta$  and  $\varepsilon$  are close to 0,  $\text{skewed4Pack}_\varepsilon(I)$  outputs a 4-stage packing of  $I$  into roughly  $4 \text{opt}(I)/3 + O(1)$  bins.

► **Theorem 1.** *Let  $I$  be a set of  $\delta$ -skewed items, where  $\delta \in (0, 1/2]$ . Then  $\text{skewed4Pack}_\varepsilon(I)$  outputs a 4-stage packing of  $I$  in time  $O((1/\varepsilon)^{O(1/\varepsilon)} + n \log n)$ . Furthermore, the number of bins used is at most  $(4/3)(1 + 8\delta)(1 + 7\varepsilon) \text{opt}(I) + (8/\varepsilon^2) + 30$ .*

The lower bound of  $4/3$  on APoG can be extended to skewed items. We formally prove this in Appendix D. Hence, our bounds are tight for skewed items. Our result indicates that to improve the bounds on APoG in the general case, we should focus on  $\delta$ -large items.

Our other main result is an APTAS for 2BP for skewed items. Formally, we give an algorithm for 2BP, called  $\text{skewedCPack}$ , and show that for every constant  $\varepsilon \in (0, 1)$ , there exists a constant  $\delta \in (0, \varepsilon)$  such that the algorithm has an AAR of  $1 + \varepsilon$  when all items in the input are  $\delta$ -skewed rectangles.  $\text{skewedCPack}$  can be extended to the rotational version of 2BP. The best-known AAR for 2BP is  $1 + \ln(1.5) + \varepsilon$ . Our result indicates that to improve upon algorithms for 2BP, one should focus on  $\delta$ -large items.

In Section 3, we describe the  $\text{skewed4Pack}$  algorithm and prove Theorem 1. In Section 4, we describe the  $\text{skewedCPack}$  algorithm and prove that it has an AAR of  $1 + \varepsilon$ .

## 2 Preliminaries

Let  $[n] := \{1, 2, \dots, n\}$ , for  $n \in \mathbb{N}$ . For a rectangle  $i$ , its area  $a(i) := w(i)h(i)$ . For a set  $I$  of rectangles, let  $a(I) := \sum_{i \in I} a(i)$ . An *axis-aligned packing* of an item  $i$  in a bin is specified by a pair  $(x(i), y(i))$ , where  $x(i), y(i) \in [0, 1]$ , so that  $i$  is placed in the region  $[x(i), x(i) + w(i)] \times [y(i), y(i) + h(i)]$ . A packing of rectangles in a bin is called *nonoverlapping* iff for any two distinct items  $i$  and  $j$ , the rectangles  $(x(i), x(i) + w(i)) \times (y(i), y(i) + h(i))$  and  $(x(j), x(j) + w(j)) \times (y(j), y(j) + h(j))$  are disjoint. Equivalently, items may only intersect at their boundaries.

**Next-Fit Decreasing Height (NFDH).** NFDH [14] is a simple algorithm for 2SP and 2BP. We will use the following results on NFDH (cf. Appendix B in [31]).

► **Lemma 2.** *Let  $I$  be a set of items where each item  $i$  has  $w(i) \leq \delta_W$  and  $h(i) \leq \delta_H$ . NFDH can pack  $I$  into a bin of width  $W$  and height  $H$  if  $a(I) \leq (W - \delta_W)(H - \delta_H)$ .*

► **Lemma 3.** *NFDH uses less than  $(2a(I) + 1)/(1 - \delta)$  bins to pack  $I$  when  $h(i) \leq \delta$  for each item  $i$  and less than  $2a(I)/(1 - \delta) + 3$  bins when  $w(i) \leq \delta$  for each item  $i$ .*



If we swap the coordinate axes in NFDH, we get the Next-Fit Decreasing Width (NFDW) algorithm. Analogs of the above results hold for NFDW.

**Slicing Items.** We will consider variants of 2BP where some items can be *sliced*. Formally, slicing a rectangular item  $i$  using a horizontal cut is the operation of replacing  $i$  by two items  $i_1$  and  $i_2$  such that  $w(i) = w(i_1) = w(i_2)$  and  $h(i) = h(i_1) + h(i_2)$ . Slicing using vertical cut is defined analogously. Allowing some items to be sliced may reduce the number of bins required to pack them.

Variants of 2SP where items can be sliced using vertical cuts find applications in resource allocation problems [3, 16, 19]. Many packing algorithms [28, 25, 8] solve the sliceable version of the problem as a subroutine.

### 3 Guillotunable Packing of Skewed Rectangles

An item is called  $(\delta_W, \delta_H)$ -skewed iff its width is at most  $\delta_W$  or its height is at most  $\delta_H$ . In this section, we consider the problem of obtaining tight upper and lower bounds on APoG for  $(\delta_W, \delta_H)$ -skewed items. We will describe the `skewed4Pack` algorithm and prove Theorem 1.

#### 3.1 Packing With Slicing

Before describing `skewed4Pack`, let us first look at a closely-related variant of this problem, called the *sliceable 2D bin packing problem*, denoted as S2BP. In this problem, we are given two sets of rectangular items,  $\widetilde{W}$  and  $\widetilde{H}$ , where items in  $\widetilde{W}$  have width more than  $1/2$ , and items in  $\widetilde{H}$  have height more than  $1/2$ .  $\widetilde{W}$  is called the set of wide items and  $\widetilde{H}$  is called the set of tall items. We are allowed to *slice* items in  $\widetilde{W}$  using horizontal cuts and slice items in  $\widetilde{H}$  using vertical cuts, and our task is to pack  $\widetilde{W} \cup \widetilde{H}$  into the minimum number of bins without rotating the items.

We first describe a  $4/3$ -asymptotic-approximation algorithm for S2BP, called `greedyPack`, that outputs a 2-stage packing. Later, we will use `greedyPack` to design `skewed4Pack`.

We assume that the bin is a square of side length 1. Since we can slice items, we allow items in  $\widetilde{W}$  to have height more than 1 and items in  $\widetilde{H}$  to have width more than 1.

For  $X \subseteq \widetilde{W}$ ,  $Y \subseteq \widetilde{H}$ , let  $\text{hsum}(X) := \sum_{i \in X} h(i)$ ;  $\text{wsum}(Y) := \sum_{i \in Y} w(i)$ ;  $\text{wmax}(X) := \max_{i \in X} w(i)$  if  $X \neq \emptyset$ , and 0 if  $X = \emptyset$ ;  $\text{hmax}(Y) := \max_{i \in Y} h(i)$  if  $Y \neq \emptyset$ , and 0 if  $Y = \emptyset$ .

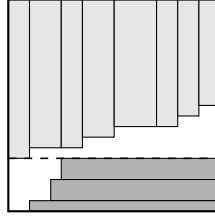
In the algorithm `greedyPack`( $\widetilde{W}, \widetilde{H}$ ), we first sort items  $\widetilde{W}$  in decreasing order of width and sort items  $\widetilde{H}$  in decreasing order of height. Suppose  $\text{hsum}(\widetilde{W}) \geq \text{wsum}(\widetilde{H})$ . Let  $X$  be the largest prefix of  $\widetilde{W}$  of total height at most 1, i.e., if  $\text{hsum}(\widetilde{W}) > 1$ , then  $X$  is a prefix of  $\widetilde{W}$  such that  $\text{hsum}(X) = 1$  (slice items if needed), and  $X = \widetilde{W}$  otherwise. Pack  $X$  into a bin such that the items touch the right edge of the bin. Then we pack the largest possible prefix of  $\widetilde{H}$  into the empty rectangular region of width  $1 - \text{wmax}(X)$  in the left side of the bin. We call this a type-1 bin. See Figure 2a for an example. If  $\text{hsum}(\widetilde{W}) < \text{wsum}(\widetilde{H})$ , we proceed analogously in a coordinate-swapped way, i.e., we first pack tall items in the bin and then pack wide items in the remaining space. Call this bin a type-2 bin. We pack the rest of the items into bins in the same way.

▷ **Claim 4.** `greedyPack`( $\widetilde{W}, \widetilde{H}$ ) outputs a 2-stage packing of  $\widetilde{W} \cup \widetilde{H}$  in  $O(m + |\widetilde{W}| \log |\widetilde{W}| + |\widetilde{H}| \log |\widetilde{H}|)$  time, where  $m$  is the number of bins used. It slices items in  $\widetilde{W}$  by making at most  $m - 1$  horizontal cuts and slices items in  $\widetilde{H}$  by making at most  $m - 1$  vertical cuts.

Since items in  $\widetilde{W}$  have width more than  $1/2$ , no two items can be placed side-by-side. Hence,  $\lceil \text{hsum}(\widetilde{W}) \rceil = \text{opt}(\widetilde{W}) \leq \text{opt}(\widetilde{W} \cup \widetilde{H})$ . Similarly,  $\lceil \text{wsum}(\widetilde{H}) \rceil \leq \text{opt}(\widetilde{W} \cup \widetilde{H})$ . So, if all bins have the same type, `greedyPack` uses  $\max(\lceil \text{hsum}(\widetilde{W}) \rceil, \lceil \text{wsum}(\widetilde{H}) \rceil) = \text{opt}(\widetilde{W} \cup \widetilde{H})$  bins. We will now focus on the case where some bins have type 1 and some have type 2.



(a) A type-1 bin produced by `greedyPack`. Wide items are packed on the right. Tall items are packed on the left.



(b) A type-2 bin produced by `greedyPack`. Tall items are packed above. Wide items are packed below.



(c) A type-1 bin in the packing of  $\widehat{I}$  computed by `skewed4Pack` (see Section 3.2). The packing contains 5 tall containers in 2 tall shelves and 18 wide containers in 8 wide shelves.

► **Definition 5.** In a type-1 bin, let  $X$  and  $Y$  be wide and tall items, respectively. The bin is called full iff  $\text{hsum}(X) = 1$  and  $\text{wsum}(Y) = 1 - \text{wmax}(X)$ . Similarly define full for type 2.

We first show that full bins pack items of large total area, and then show that if some bins have type 1 and some have type 2, then there are at most 2 non-full bins. This helps us upper-bound the number of bins used by `greedyPack`( $\widetilde{W}, \widetilde{H}$ ) in terms of  $a(\widetilde{W} \cup \widetilde{H})$ .

► **Lemma 6.** Let there be  $m_1$  type-1 full bins, containing items  $J_1$ . Then  $m_1 \leq 4a(J_1)/3 + 1/3$ .

**Proof.** In the  $j^{\text{th}}$  full bin of type 1, let  $X_j$  be the items from  $\widetilde{W}$  and  $Y_j$  be the items from  $\widetilde{H}$ . Let  $\ell_j := \text{wmax}(X_j)$  if  $j \leq m_1$  and  $\ell_{m_1+1} := 1/2$ . Since all items have their larger dimension more than  $1/2$ ,  $\ell_j \geq 1/2$  and  $\text{hmax}(Y_j) > 1/2$ , for any  $j \in [m_1]$ .

$a(X_j) \geq \ell_{j+1}$ , since  $X_j$  has height 1 and width at least  $\ell_{j+1}$ .  $a(Y_j) \geq (1 - \ell_j)/2$ , since  $Y_j$  has width  $1 - \ell_j$  and height more than  $1/2$ . So,  $a(J_1) = \sum_{j=1}^{m_1} (a(X_j) + a(Y_j)) \geq \sum_{j=1}^{m_1} (\ell_{j+1} + (1 - \ell_j)/2) \geq \sum_{j=1}^{m_1} ((\ell_{j+1}/2) + (1/4) + (1/2) - (\ell_j/2)) \geq (3m_1 - 1/4)$ . In the above inequalities, we used  $\ell_{j+1} \geq 1/2$  and  $\ell_1 \leq 1$ .

Therefore,  $m_1 \leq 4a(J_1)/3 + 1/3$ . ◀

An analog of Lemma 6 can be proven for type-2 bins. Lemma 6 implies that very few full bins can have items of total area significantly less than  $3/4$ .

Suppose `greedyPack`( $\widetilde{W}, \widetilde{H}$ ) uses  $m$  bins. After  $j$  bins are packed, let  $A_j$  be the height of the remaining items in  $\widetilde{W}$  and  $B_j$  be the width of the remaining items in  $\widetilde{H}$ . Let  $t_j$  be the type of the  $j^{\text{th}}$  bin (1 for type-1 bin, 2 for type-2 bin). So  $t_j = 1 \iff A_{j-1} \geq B_{j-1}$ .

We first show that  $|A_{j-1} - B_{j-1}| \leq 1 \implies |A_j - B_j| \leq 1$ , i.e., once  $|\text{hsum}(\widetilde{W}) - \text{wsum}(\widetilde{H})|$  becomes at most 1 during `greedyPack`, it continues to stay at most 1. Next, we show that  $t_j \neq t_{j+1} \implies |A_{j-1} - B_{j-1}| \leq 1$ , i.e., if all bins don't have the same type, then  $|\text{hsum}(\widetilde{W}) - \text{wsum}(\widetilde{H})|$  eventually becomes at most 1 during `greedyPack`. In the first non-full bin, we use up all the wide items or all the tall items. We will show that the remaining items have total height or total width at most 1, so we have at most 2 non-full bins.

In the  $j^{\text{th}}$  bin, let  $a_j$  be the height of items from  $\widetilde{W}$  and  $b_j$  be the width of items from  $\widetilde{H}$ . Hence, for all  $j \in [m]$ ,  $A_{j-1} = A_j + a_j$  and  $B_{j-1} = B_j + b_j$ .

► **Lemma 7.**  $|A_{j-1} - B_{j-1}| \leq 1 \implies |A_j - B_j| \leq 1$ .

**Proof.** W.l.o.g., assume  $A_{j-1} \geq B_{j-1}$ . So,  $t_j = 1$ . Suppose  $a_j < b_j$ . Then  $a_j < 1$ , so we used up  $\widetilde{W}$  in the  $j^{\text{th}}$  bin. Therefore,  $A_j = 0 \implies A_{j-1} = a_j < b_j \leq b_j + B_j = B_{j-1}$ , which contradicts. Hence,  $a_j \geq b_j$ . As  $0 \leq (A_{j-1} - B_{j-1}), (a_j - b_j) \leq 1$ , we get  $A_j - B_j = (A_{j-1} - B_{j-1}) - (a_j - b_j) \in [-1, 1]$ . ◀

► **Lemma 8.**  $t_j \neq t_{j+1} \implies |A_{j-1} - B_{j-1}| \leq 1$ .

**Proof.** W.l.o.g., assume  $t_j = 1$  and  $t_{j+1} = 2$ . Then

$$A_{j-1} \geq B_{j-1} \text{ and } A_j < B_j \implies B_{j-1} \leq A_{j-1} < B_{j-1} + a_j - b_j \implies A_{j-1} - B_{j-1} \in [0, 1]. \blacktriangleleft$$

► **Lemma 9.** *If all bins don't have the same type, then there can be at most 2 non-full bins.*

**Proof.** Let there be  $p$  full bins. Assume w.l.o.g. that in the  $(p+1)^{\text{th}}$  bin, we used up all items from  $\widetilde{W}$  but not  $\widetilde{H}$ . Hence,  $A_{p+1} = 0$  and  $\forall i \geq p+2, t_i = 2$ . Since all bins don't have the same type,  $\exists k \leq p+1$  such that  $t_k = 1$  and  $t_{k+1} = 2$ . By Lemmas 7 and 8, we get  $|A_{p+1} - B_{p+1}| \leq 1$ , implying  $B_{p+1} \leq 1$ . Hence, the  $(p+1)^{\text{th}}$  bin will use up all tall items, implying at most 2 non-full bins. ◀

► **Theorem 10.** *The number of bins  $m$  used by `greedyPack` is at most  $\max(\lceil \text{hsum}(\widetilde{W}) \rceil, \lceil \text{wsum}(\widetilde{H}) \rceil, \frac{4}{3}a(\widetilde{W} \cup \widetilde{H}) + \frac{8}{3})$ .*

**Proof.** If all bins have the same type, then  $m \leq \max(\lceil \text{hsum}(\widetilde{W}) \rceil, \lceil \text{wsum}(\widetilde{H}) \rceil)$ .

Let there be  $m_1$  (resp.  $m_2$ ) full bins of type 1 (resp. type 2) and let  $J_1$  (resp.  $J_2$ ) be the items inside those bins. Then by Lemma 6, we get  $m_1 \leq 4a(J_1)/3 + 1/3$  and  $m_2 \leq 4a(J_2)/3 + 1/3$ . Hence,  $m_1 + m_2 \leq 4a(\widetilde{W} \cup \widetilde{H})/3 + 2/3$ . If all bins don't have the same type, then by Lemma 9, there can be at most 2 non-full bins, so `greedyPack`( $\widetilde{W}, \widetilde{H}$ ) uses at most  $4a(\widetilde{W} \cup \widetilde{H})/3 + 8/3$  bins. ◀

### 3.2 The `skewed4Pack` Algorithm

We now return to the 2BP problem. `skewed4Pack` is an algorithm for 2BP takes as input a set  $I$  of rectangular items and a parameter  $\varepsilon \in (0, 1)$  where  $\varepsilon^{-1} \in \mathbb{Z}$ . It outputs a 4-stage bin packing of  $I$ . `skewed4Pack` has the following outline:

- A. Use linear grouping [15, 28] to round up the width or height of each item in  $I$ . This gives us a new instance  $\widehat{I}$ .
- B. Pack  $\widehat{I}$  into  $1/\varepsilon^2 + 1$  shelves, after *slicing* some items. A shelf is a rectangular region with width or height more than  $1/2$  and is fully packed, i.e., the area of items in a shelf equals the area of the shelf. If we treat each shelf as an item, we get a new instance  $\widetilde{I}$ .
- C. Compute a packing of  $\widetilde{I}$  into bins, after possibly slicing some items, using `greedyPack`.
- D. Repack most items of  $\widehat{I}$  without slicing into the shelves. We will prove that the remaining items have very small area, so they can be packed separately using NFDH.

**A. Item Classification and Rounding.** Define  $W := \{i \in I : h(i) \leq \delta_H\}$  and  $H := I - W$ . Items in  $W$  are called *wide* and items in  $H$  are called *tall*. Let  $W^{(L)} := \{i \in W : w(i) > \varepsilon\}$  and  $W^{(S)} := W - W^{(L)}$ . Similarly, let  $H^{(L)} := \{i \in H : h(i) > \varepsilon\}$  and  $H^{(S)} := H - H^{(L)}$ .

We will use *linear grouping* [15, 28] to round up the widths of  $W^{(L)}$  and the heights of  $H^{(L)}$  to get items  $\widehat{W}^{(L)}$  and  $\widehat{H}^{(L)}$ , respectively. By Claim 27 in Appendix A, items in  $\widehat{W}^{(L)}$  have at most  $1/\varepsilon^2$  distinct widths and items in  $\widehat{H}^{(L)}$  have at most  $1/\varepsilon^2$  distinct heights.

Let  $\widehat{W} := \widehat{W}^{(L)} \cup W^{(S)}$ ,  $\widehat{H} := \widehat{H}^{(L)} \cup H^{(S)}$ ,  $\widehat{I} := \widehat{W} \cup \widehat{H}$ . For  $\widehat{X} \subseteq \widehat{I}$ , let  $\text{fopt}(\widehat{X})$  be the minimum number of bins needed to pack  $\widehat{X}$  when items in  $\widehat{X} \cap \widehat{W}^{(L)}$  can be sliced using horizontal cuts, items in  $\widehat{X} \cap \widehat{H}^{(L)}$  can be sliced using vertical cuts, and items in  $\widehat{X} \cap (W^{(S)} \cup H^{(S)})$  can be sliced both vertically and horizontally. The following lemma follows from Lemma 28 in Appendix A.

► **Lemma 11.**  $\text{fopt}(\widehat{I}) < (1 + \varepsilon) \text{opt}(I) + 2$ .

**B. Creating Shelves.** We will use ideas from Kenyon and Rémila’s 2SP algorithm [28] to pack  $\widehat{I}$  into *shelves*. Roughly, we solve a linear program to compute an optimal strip packing of  $\widehat{W}$ , where the packing is 3-stage. The first stage of cuts gives us shelves and the second stage gives us containers. From each shelf, we trim off space that doesn’t belong to any container. See Section 3.3 for details. Let  $\widetilde{W}$  be the shelves thus obtained. Analogously, we can pack items  $\widehat{H}$  into shelves  $\widetilde{H}$ . Shelves in  $\widetilde{W}$  are called *wide shelves* and shelves in  $\widetilde{H}$  are called *tall shelves*. Let  $\widetilde{I} := \widetilde{W} \cup \widetilde{H}$ . We can interpret each shelf in  $\widetilde{I}$  as a rectangular item. We allow slicing  $\widetilde{W}$  and  $\widetilde{H}$  using horizontal cuts and vertical cuts, respectively. In Section 3.3, we prove the following facts.

► **Lemma 12.**  *$\widetilde{I}$  has the following properties: (a)  $|\widetilde{W}| \leq 1 + 1/\varepsilon^2$  and  $|\widetilde{H}| \leq 1 + 1/\varepsilon^2$ ; (b) Each item in  $\widetilde{W}$  has width more than  $1/2$  and each item in  $\widetilde{H}$  has height more than  $1/2$ ; (c)  $a(\widetilde{I}) = a(\widehat{I})$ ; (d)  $\max(\lceil \text{hsum}(\widetilde{W}) \rceil, \lceil \text{wsum}(\widetilde{H}) \rceil) \leq \text{fopt}(\widehat{I})$ .*

**C. Packing Shelves into Bins.** So far, we have packed  $\widehat{I}$  into shelves  $\widetilde{W}$  and  $\widetilde{H}$ . We will now use  $\text{greedyPack}(\widetilde{W}, \widetilde{H})$  to pack the shelves into bins. By Claim 4, we get a 2-stage packing of  $\widetilde{W} \cup \widetilde{H}$  into  $m$  bins, where we make at most  $m - 1$  horizontal cuts in  $\widetilde{W}$  and at most  $m - 1$  vertical cuts in  $\widetilde{H}$ . The horizontal cuts (resp. vertical cuts) increase the number of wide shelves (resp. tall shelves) from at most  $1 + 1/\varepsilon^2$  to at most  $m + 1/\varepsilon^2$ . By Theorem 10, Lemma 12(d) and Lemma 12(c), we get  $m \leq \max(\lceil \text{hsum}(\widetilde{W}) \rceil, \lceil \text{wsum}(\widetilde{H}) \rceil, \frac{4}{3}a(\widetilde{I}) + \frac{8}{3}) \leq \frac{4}{3}\text{fopt}(\widehat{I}) + \frac{8}{3}$ .

**D. Packing Items into Containers.** So far, we have a packing of shelves into  $m$  bins, where the shelves contain slices of items  $\widehat{I}$ . We will now repack a large subset of  $\widehat{I}$  into the shelves without slicing  $\widehat{I}$ . See Figure 2c for an example output. We do this using a standard greedy algorithm. See Appendix B for details of the algorithm and proof of the following lemma.

► **Lemma 13.** *Let  $P$  be a packing of  $\widetilde{I}$  into  $m$  bins, where we made at most  $m - 1$  horizontal cuts in wide shelves and at most  $m - 1$  vertical cuts in tall shelves. Then we can (without slicing) pack a large subset of  $\widehat{I}$  into the shelves in  $P$  such that the unpacked items (also called discarded items) from  $\widetilde{W}$  have total area less than  $\varepsilon \text{hsum}(\widetilde{W}) + \delta_H(1 + \varepsilon)(m + 1/\varepsilon^2)$ , and the unpacked items from  $\widetilde{H}$  have area less than  $\varepsilon \text{wsum}(\widetilde{H}) + \delta_W(1 + \varepsilon)(m + 1/\varepsilon^2)$ .*

We pack wide discarded items into new bins using NFDH and pack tall discarded items into new bins using NFDW. Finally, we prove the performance guarantee of  $\text{skewed4Pack}_\varepsilon(I)$ .

► **Lemma 14.** *Let  $I$  be a set of  $(\delta_W, \delta_H)$ -skewed items. Then  $\text{skewed4Pack}_\varepsilon(I)$  outputs a 4-stage packing of  $I$  in time  $O((1/\varepsilon)^{O(1/\varepsilon)} + n \log n)$  and uses less than  $\alpha(1 + \varepsilon)\text{opt}(I) + 2\beta$  bins, where  $\Delta := \frac{1}{2} \left( \frac{\delta_H}{1 - \delta_H} + \frac{\delta_W}{1 - \delta_W} \right)$ ,  $\alpha := \frac{4}{3}(1 + 4\Delta)(1 + 3\varepsilon)$ ,  $\beta := \frac{2\Delta(1 + \varepsilon)}{\varepsilon^2} + \frac{10}{3} + \frac{19\Delta}{3} + \frac{16\Delta\varepsilon}{3}$ .*

**Proof.** The discarded items are packed using NFDH or NFDW, which output a 2-stage packing. Since  $\text{greedyPack}$  outputs a 2-stage packing of the shelves and the packing of items into the shelves is a 2-stage packing, the bin packing of non-discarded items is a 4-stage packing. The time taken by  $\text{skewed4Pack}$  is at most  $O((1/\varepsilon)^{O(1/\varepsilon)} + n \log n)$ .

Suppose  $\text{greedyPack}$  uses at most  $m$  bins. By Theorem 10,  $m \leq 4\text{fopt}(\widehat{I})/3 + 8/3$ . Let  $W^d$  and  $H^d$  be the items discarded from  $W$  and  $H$ , respectively. By Lemma 13 and Lemma 12(d),  $a(W^d) < \varepsilon \text{fopt}(\widehat{I}) + \delta_H(1 + \varepsilon)(m + 1/\varepsilon^2)$  and  $a(H^d) < \varepsilon \text{fopt}(\widehat{I}) + \delta_W(1 + \varepsilon)(m + 1/\varepsilon^2)$ .

By Lemmas 3 and 11, the number of bins used by `skewed4Pack $_{\varepsilon}$ (I)` is less than

$$\begin{aligned} & m + \frac{2a(W^d) + 1}{1 - \delta_H} + \frac{2a(H^d) + 1}{1 - \delta_W} \\ & \leq (1 + 4\Delta(1 + \varepsilon))m + 4\varepsilon(1 + \Delta) \text{fopt}(\widehat{I}) + 2(1 + \Delta) + 4\Delta(1 + \varepsilon)/\varepsilon^2 \\ & \leq \alpha \text{fopt}(\widehat{I}) + 2(\beta - 1) < \alpha(1 + \varepsilon) \text{opt}(I) + 2\beta. \end{aligned} \quad \blacktriangleleft$$

Now we conclude with the proof of Theorem 1.

**Proof of Theorem 1.** This is a simple corollary of Lemma 14, where  $\delta \leq 1/2$  gives us  $\Delta \leq 2\delta$ ,  $\alpha(1 + \varepsilon) \leq (4/3)(1 + 8\delta)(1 + 7\varepsilon)$ , and  $\beta \leq 4/\varepsilon^2 + 15$ .  $\blacktriangleleft$

### 3.3 Creating Shelves

Here we will describe how to obtain shelves  $\widetilde{W}$  and  $\widetilde{H}$  from items  $\widehat{W}$  and  $\widehat{H}$ , respectively. Let  $\text{opt}_{\text{SP}}(\widehat{W})$  denote the optimal strip packing of  $\widehat{W}$  where items in  $\widehat{W}$  can be sliced using horizontal cuts. Then  $\text{fopt}(\widehat{W}) = \lceil \text{opt}_{\text{SP}}(\widehat{W}) \rceil$ . Hence, we will now try to compute a near-optimal strip packing of  $\widehat{W}$ .

Define a horizontal configuration  $S$  as a tuple  $(S_0, S_1, S_2, \dots)$  of  $1/\varepsilon^2 + 1$  non-negative integers, where  $S_0 \in \{0, 1\}$  and  $\sum_{j=1}^{1/\varepsilon^2} S_j w_j \leq 1$ . For any horizontal line at height  $y$  in a strip packing of  $\widehat{W}$ , the multiset of items intersecting the line corresponds to a configuration.  $S_0$  indicates whether the line intersects items from  $W^{(S)}$ , and  $S_j$  is the number of items from  $\widehat{W}_j^{(L)}$  that the line intersects. Let  $\mathcal{S}$  be the set of all horizontal configurations. Let  $N := |\mathcal{S}|$ .

To obtain an optimal packing, we need to determine the height of each configuration. This can be done with the following linear program.

$$\begin{aligned} & \min_{x \in \mathbb{R}^N} \sum_{S \in \mathcal{S}} x_S \\ & \text{where } \sum_{S \in \mathcal{S}} S_j x_S = h(\widehat{W}_j^{(L)}) \quad \forall j \in [1/\varepsilon^2] \\ & \text{and } \sum_{S: S_0=1} \left( 1 - \sum_{j=1}^{1/\varepsilon^2} S_j w_j \right) x_S = a(W^{(S)}) \\ & \text{and } x_S \geq 0 \quad \forall S \in \mathcal{S} \end{aligned}$$

Let  $x^*$  be an optimal extreme-point solution to the above LP. This gives us a packing where the strip is divided into rectangular regions called *shelves* that are stacked on top of each other. Each shelf has a configuration  $S$  associated with it and has height  $h(S) := x_S^*$  and contains  $S_j$  *containers* of width  $w_j$ . Containers of width  $w_j$  only contain items from  $\widehat{W}_j^{(L)}$ , and we call them *type- $j$*  containers. If  $S_0 = 1$ ,  $S$  also contains a container of width  $1 - \sum_{j=1}^{1/\varepsilon^2} S_j w_j$  that contains small items. We call this container a *type-0* container. Each container is fully filled with items. Let  $w(S)$  denote the width of shelf  $S$ , i.e., the sum of widths of all containers in  $S$ . Note that if  $S_0 = 1$ , then  $w(S) = 1$ . Otherwise,  $w(S) = \sum_{j=1}^{1/\varepsilon^2} S_j w_j$ .

► **Lemma 15.**  $x^*$  contains at most  $1/\varepsilon^2 + 1$  positive entries.

**Proof sketch.** Follows by applying Rank Lemma<sup>2</sup> to the linear program.  $\blacktriangleleft$

<sup>2</sup> Rank Lemma: the number of non-zero variables in an extreme-point solution to a linear program is at most the number of non-trivial constraints [34, Lemma 2.1.4].

► **Lemma 16.**  $x_S^* > 0 \implies w(S) > 1/2$ .

**Proof.** Suppose  $w(S) \leq 1/2$ . Then we could have split  $S$  into two parts by making a horizontal cut in the middle and packed the parts side-by-side, reducing the height of the strip by  $x_S^*/2$ . But that would contradict the fact that  $x^*$  is optimal. ◀

Treat each shelf  $S$  as an item of width  $w(S)$  and height  $h(S)$ . Allow each such item to be sliced using horizontal cuts. This gives us a new set  $\widetilde{W}$  of items such that  $\widetilde{W}$  can be packed inside  $\widetilde{W}$ . By applying an analogous approach to  $\widetilde{H}$ , we get a new set  $\widetilde{H}$  of items. Let  $\widetilde{I} := \widetilde{W} \cup \widetilde{H}$ . We call the shelves of  $\widetilde{W}$  *wide shelves* and the shelves of  $\widetilde{H}$  *tall shelves*. The containers in wide shelves are called *wide containers* and the containers in tall shelves are called *tall containers*.

► **Lemma 12.**  $\widetilde{I}$  has the following properties: (a)  $|\widetilde{W}| \leq 1 + 1/\varepsilon^2$  and  $|\widetilde{H}| \leq 1 + 1/\varepsilon^2$ ; (b) Each item in  $\widetilde{W}$  has width more than  $1/2$  and each item in  $\widetilde{H}$  has height more than  $1/2$ ; (c)  $a(\widetilde{I}) = a(\widehat{I})$ ; (d)  $\max(\lceil \text{hsum}(\widetilde{W}) \rceil, \lceil \text{wsum}(\widetilde{H}) \rceil) \leq \text{fopt}(\widehat{I})$ .

**Proof.** Lemma 15 implies (a) and Lemma 16 implies (b).  $a(\widehat{I}) = a(\widetilde{I})$  as the shelves are tightly packed. Since  $x^*$  is an optimal solution to the linear program,  $\lceil \text{hsum}(\widetilde{W}) \rceil = \lceil \sum_{S \in \mathcal{S}} x_S^* \rceil = \lceil \text{opt}_{\text{SP}}(\widehat{W}) \rceil = \text{fopt}(\widehat{W}) \leq \text{fopt}(\widehat{I})$ . Similarly,  $\lceil \text{wsum}(\widetilde{H}) \rceil = \text{fopt}(\widehat{H}) \leq \text{fopt}(\widehat{I})$ . ◀

## 4 Almost-Optimal Bin Packing of Skewed Rectangles

In this section, we will describe the algorithm `skewedCPack`. `skewedCPack` takes as input a set  $I$  of items and a parameter  $\varepsilon \in (0, 1/2]$ , where  $\varepsilon^{-1} \in \mathbb{Z}$ . We will prove that `skewedCPack` has AAR  $1 + 20\varepsilon$  when  $\delta$  is sufficiently small. `skewedCPack` works roughly as follows:

1. Invoke the subroutine `round`( $I$ ) (cf. Section 4.1). `round`( $I$ ) returns a pair  $(\widetilde{I}, I_{\text{med}})$ . Here  $I_{\text{med}}$ , called the set of *medium items*, has low total area, so we can pack it in a small number of bins.  $\widetilde{I}$ , called the set of *rounded items*, is obtained by rounding up the width or height of each item in  $I - I_{\text{med}}$ , so that  $\widetilde{I}$  has properties that help us pack it easily.
2. Compute the optimal *fractional compartmental* bin packing of  $\widetilde{I}$  (we will define *fractional* and *compartmental* later).
3. Use this packing of  $\widetilde{I}$  to obtain a packing of  $I$  that uses slightly more number of bins.

To bound the AAR of `skewedCPack`, we will prove a structural theorem (Section 4.2), which says that the optimal fractional compartmental packing of  $\widetilde{I}$  uses close to  $\text{opt}(I)$  bins.

### 4.1 Classifying and Rounding Items

We now describe the algorithm `round` and show that its output satisfies important properties.

First, we will find a set  $I_{\text{med}} \subseteq I$  and positive constants  $\varepsilon_1$  and  $\varepsilon_2$  such that  $a(I_{\text{med}}) \leq \varepsilon a(I)$ ,  $\varepsilon_2 \ll \varepsilon_1$ , and  $I - I_{\text{med}}$  is  $(\varepsilon_2, \varepsilon_1]$ -free, i.e., no item in  $I - I_{\text{med}}$  has its width or height in the interval  $(\varepsilon_2, \varepsilon_1]$ . Then we can remove  $I_{\text{med}}$  from  $I$  and pack it separately into a small number of bins using NFDH. We will see that the  $(\varepsilon_2, \varepsilon_1]$ -freeness of  $I - I_{\text{med}}$  will help us pack  $I - I_{\text{med}}$  efficiently. Specifically, we require  $\varepsilon_1 \leq \varepsilon$ ,  $\varepsilon_1^{-1} \in \mathbb{Z}$ , and  $\varepsilon_2 = f(\varepsilon_1)$ , where  $f(x) := \varepsilon x / (104(1 + 1/(\varepsilon x))^{2/x-2})$ . We explain this choice of  $f$  in Section 4.3.4. Intuitively, such an  $f$  ensures that  $\varepsilon_2 \ll \varepsilon_1$  and  $\varepsilon_2^{-1} \in \mathbb{Z}$ . For `skewedCPack` to work, we require  $\delta \leq \varepsilon_2$ . Finding such an  $I_{\text{med}}$  and  $\varepsilon_1$  is a standard technique [25, 7], so we defer the details to Appendix C.1.

Next, we classify the items in  $I - I_{\text{med}}$  into three disjoint classes:

- Wide items:  $W := \{i \in I : w(i) > \varepsilon_1 \text{ and } h(i) \leq \varepsilon_2\}$ .
- Tall items:  $H := \{i \in I : w(i) \leq \varepsilon_2 \text{ and } h(i) > \varepsilon_1\}$ .
- Small items:  $S := \{i \in I : w(i) \leq \varepsilon_2 \text{ and } h(i) \leq \varepsilon_2\}$ .

We will now use *linear grouping* [15, 28] to round up the widths of items  $W$  and the heights of items  $H$  to get items  $\widetilde{W}$  and  $\widetilde{H}$ , respectively. By Claim 27 in Appendix A, items in  $\widetilde{W}$  have at most  $1/(\varepsilon\varepsilon_1)$  distinct widths and items in  $\widetilde{H}$  have at most  $1/(\varepsilon\varepsilon_1)$  distinct heights. Let  $\widetilde{I} := \widetilde{W} \cup \widetilde{H} \cup S$ .

► **Definition 17** (Fractional packing). *Suppose we are allowed to slice wide items in  $\widetilde{I}$  using horizontal cuts, slice tall items in  $\widetilde{I}$  using vertical cuts and slice small items in  $\widetilde{I}$  using both horizontal and vertical cuts. For any  $\widetilde{X} \subseteq \widetilde{I}$ , a bin packing of the slices of  $\widetilde{X}$  is called a fractional packing of  $\widetilde{X}$ . The optimal fractional packing of  $\widetilde{X}$  is denoted by  $\text{fopt}(\widetilde{X})$ .*

► **Lemma 18.**  $\text{fopt}(\widetilde{I}) < (1 + \varepsilon) \text{opt}(I) + 2$ .

**Proof.** Directly follows from Lemma 28 in Appendix A. ◀

## 4.2 Structural Theorem

We will now define compartmental packing and prove the structural theorem, which says that the number of bins in the optimal fractional compartmental packing of  $\widetilde{I}$  is roughly equal to  $\text{fopt}(\widetilde{I})$ .

We first show how to *discretize* a packing, i.e., we show that given a fractional packing of items in a bin, we can remove a small fraction of tall and small items and shift the remaining items leftwards so that the left and right edges of each wide item belong to a constant-sized set  $\mathcal{T}$ , where  $|\mathcal{T}| \leq (1 + 1/\varepsilon\varepsilon_1)^{2/\varepsilon_1 - 2}$ . Next, we define *compartmental* packing and show how to convert a discretized packing to a compartmental packing.

For any rectangle  $i$  packed in a bin, let  $x_1(i)$  and  $x_2(i)$  denote the  $x$ -coordinates of its left and right edges, respectively, and let  $y_1(i)$  and  $y_2(i)$  denote the  $y$ -coordinates of its bottom and top edges, respectively. Let  $R$  be the set of distinct widths of items in  $\widetilde{W}$ . Given the way we rounded items,  $|R| \leq 1/\varepsilon\varepsilon_1$ . Recall that  $\varepsilon_1 \leq \varepsilon \leq 1/2$  and  $\varepsilon_1^{-1}, \varepsilon^{-1} \in \mathbb{Z}$ .

► **Theorem 19.** *Given a fractional packing of items  $\widetilde{J} \subseteq \widetilde{I}$  into a bin, we can remove tall and small items of total area less than  $\varepsilon$  and shift some of the remaining items to the left such that for every wide item  $i$ , we get  $x_1(i), x_2(i) \in \mathcal{T}$ .*

**Proof.** For wide items  $u$  and  $v$  in the bin, we say that  $u \prec v$  iff the right edge of  $u$  is to the left of the left edge of  $v$ . Formally  $u \prec v \iff x_2(u) \leq x_1(v)$ . We call  $u$  a *predecessor* of  $v$ . A sequence  $[i_1, i_2, \dots, i_k]$  such that  $i_1 \prec i_2 \prec \dots \prec i_k$  is called a *chain* ending at  $i_k$ . For a wide item  $i$ , define  $\text{level}(i)$  as the number of items in the longest chain ending at  $i$ . Formally,  $\text{level}(i) := 1$  if  $i$  has no predecessors, and  $(1 + \max_{j \prec i} \text{level}(j))$  otherwise. Let  $W_j$  be the items at level  $j$ , i.e.,  $W_j := \{i : \text{level}(i) = j\}$ . Note that the level of an item can be at most  $1/\varepsilon_1 - 1$ , since each wide item has width more than  $\varepsilon_1$ .

We will describe an algorithm for discretization. But first, we need to introduce two recursively-defined set families  $(S_1, S_2, \dots)$  and  $(T_0, T_1, \dots)$ . Let  $T_0 := \{0\}$  and  $t_0 := 1$ . For any  $j > 0$ , define  $t_j := (1 + 1/\varepsilon\varepsilon_1)^{2j}$ ,  $\delta_j := \varepsilon\varepsilon_1/t_{j-1}$ ,  $S_j := T_{j-1} \cup \{k\delta_j : k \in \mathbb{Z}, 0 \leq k < 1/\delta_j\}$ ,  $T_j := \{x + y : x \in S_j, y \in R \cup \{0\}\}$ . Note that  $\forall j > 0$ , we have  $T_{j-1} \subseteq S_j \subseteq T_j$  and  $\delta_j^{-1} \in \mathbb{Z}$ . Define  $\mathcal{T} := T_{1/\varepsilon_1 - 1}$ .

Our discretization algorithm proceeds in stages, where in the  $j^{\text{th}}$  stage, we apply two transformations to the items in the bin, called *strip-removal* and *compaction*.

**Strip-removal:** For each  $x \in T_{j-1}$ , consider a strip of width  $\delta_j$  and height 1 in the bin whose left edge has coordinate  $x$ . Discard the slices of tall and small items inside the strips.

**Compaction:** Move all tall and small items as much towards the left as possible (imagine a gravitational force acting leftwards on the tall and small items) while keeping the wide items fixed. Then move each wide item  $i \in W_j$  leftwards till  $x_1(i) \in S_j$ .

## 22:12 Geometric Bin Packing with Skewed Items

Observe that the algorithm maintains the following invariant: *after  $k$  stages, for each  $j \in [k]$ , each item  $i \in W_j$  has  $x_1(i) \in S_j$  (and hence  $x_2(i) \in T_j$ ). This ensures that after the algorithm ends,  $x_1(i), x_2(i) \in \mathcal{T}$ . All that remains to prove is that the total area of items discarded during strip-removal is at most  $\varepsilon$  and that compaction is always possible.*

► **Lemma 20.** *For all  $j \geq 0$ ,  $|T_j| \leq t_j$ .*

**Proof by induction.**  $|T_0| = t_0 = 1$ , so the base case holds. Now assume  $|T_{j-1}| \leq t_{j-1}$ . Then

$$|T_j| \leq (|R| + 1)|S_j| \leq \left(\frac{1}{\varepsilon\varepsilon_1} + 1\right) \left(|T_{j-1}| + \frac{1}{\delta_j}\right) \leq \left(\frac{1}{\varepsilon\varepsilon_1} + 1\right)^2 t_{j-1} = t_j. \quad \blacktriangleleft$$

Therefore,  $|\mathcal{T}| \leq t_{1/\varepsilon_1 - 1} = (1 + 1/\varepsilon\varepsilon_1)^{2/\varepsilon_1 - 2}$ .

► **Lemma 21.** *Discarded items (across all stages) have total area less than  $\varepsilon$ .*

**Proof.** In the  $j^{\text{th}}$  stage, we create  $|T_{j-1}|$  strips, and each strip has total area at most  $\delta_j$ . Therefore, the area discarded in the  $j^{\text{th}}$  stage is at most  $|T_{j-1}|\delta_j \leq t_{j-1}\delta_j = \varepsilon\varepsilon_1$ . Since there can be at most  $1/\varepsilon_1 - 1$  stages, we discard a total area of less than  $\varepsilon$  across all stages. ◀

► **Lemma 22.** *Compaction always succeeds, i.e., in the  $j^{\text{th}}$  stage, while moving item  $i \in W_j$  leftwards, no other item will block its movement.*

**Proof.** Let  $i \in W_j$ . Let  $z$  be the  $x$ -coordinate of the left edge of the strip immediately to the left of item  $i$ , i.e.,  $z := \max(\{x \in T_{j-1} : x \leq x_1(i)\})$ . For any wide item  $i'$ , we have  $x_2(i') \leq x_1(i) \iff i' \prec i \iff \text{level}(i') \leq j - 1$ . By our invariant, we get  $\text{level}(i') \leq j - 1 \implies x_2(i') \in T_{j-1} \implies x_2(i') \leq z$ . Therefore, for every wide item  $i'$ ,  $x_2(i') \notin (z, x_1(i)]$ .

In the  $j^{\text{th}}$  strip-removal, we cleared the strip  $[z, z + \delta_j] \times [0, 1]$ . If  $x_1(i) \in [z, z + \delta_j]$ , then  $i$  can freely move to  $z$ , and  $z \in T_{j-1} \subseteq S_j$ . Since no wide item has its right edge in  $(z, x_1(i)]$ , if  $x_1(i) > z + \delta_j$ , all the tall and small items whose left edge lies in  $[z + \delta_j, x_1(i)]$  will move leftwards by at least  $\delta_j$  during compaction. Hence, there would be an empty space of width at least  $\delta_j$  to the left of item  $i$ . Therefore, we can move  $i$  leftwards to make  $x_1(i)$  a multiple of  $\delta_j$ , and then  $x_1(i)$  would belong to  $S_j$ . ◀

Hence, compaction always succeeds and we get  $x_1(i), x_2(i) \in \mathcal{T}$  for each wide item  $i$ . ◀

► **Definition 23 (Compartmental packing).** *Consider a packing of some items into a bin. A compartment  $C$  is defined as a rectangular region in the bin satisfying the following properties:*

- $x_1(C), x_2(C) \in \mathcal{T}$ .
- $y_1(C), y_2(C)$  are multiples of  $\varepsilon_{\text{cont}} := \varepsilon\varepsilon_1/6|\mathcal{T}|$ .
- $C$  does not contain both wide items and tall items.
- If  $C$  contains tall items, then  $x_1(C)$  and  $x_2(C)$  are consecutive values in  $\mathcal{T}$ .

*If a compartment contains a wide item, it is called a wide compartment. Otherwise it is called a tall compartment. A packing of items into a bin is called compartmental iff there is a set of non-overlapping compartments in the bin such that each wide or tall item lies completely inside some compartment, and there are at most  $n_W := 3(1/\varepsilon_1 - 1)|\mathcal{T}| + 1$  wide compartments and at most  $n_H := (1/\varepsilon_1 - 1)|\mathcal{T}|$  tall compartments in the bin. A packing of items into multiple bins is called compartmental iff each bin is compartmental.*



Note that small items can be packed both inside and outside compartments.

The following lemma states that a discretized packing can be converted to a compartmental packing. It can be proved using standard techniques (e.g., Section 3.2.3 in [36]). See Appendix G.2 in [31] for a formal proof.

► **Lemma 24.** *If  $x_1(i), x_2(i) \in \mathcal{T}$  for each wide item  $i$  in a bin, then by removing wide and small items of area less than  $\varepsilon$ , we can get a compartmental packing of the remaining items.*

► **Theorem 25.** *For a set  $\tilde{I}$  of  $\delta$ -skewed rounded items, define  $\text{fcopt}(\tilde{I})$  as the number of bins in the optimal fractional compartmental packing<sup>3</sup> of  $\tilde{I}$ . Then  $\text{fcopt}(\tilde{I}) < (1 + 4\varepsilon) \text{fopt}(\tilde{I}) + 2$ .*

**Proof.** Consider a fractional packing of  $\tilde{I}$  into  $m := \text{fopt}(\tilde{I})$  bins. From each bin, we can discard items of area at most  $2\varepsilon$  and get a compartmental packing of the remaining items by Theorem 19 and Lemma 24.

Let  $X$  be the set of wide and small discarded items and let  $Y$  be the set of tall discarded items. For each item  $i \in X$ , if  $w(i) \leq 1/2$ , slice it using a horizontal cut in the middle and place the pieces horizontally next to each other to get a new item of width  $2w(i)$  and height  $h(i)/2$ . Repeat until  $w(i) > 1/2$ . Now pack the items in bins by stacking them one-over-the-other so that for each item  $i \in X$ ,  $x_1(i) = 0$ . This will require less than  $2a(X) + 1$  bins, and the packing will be compartmental.

Similarly, we can get a compartmental packing of  $Y$  into  $2a(Y) + 1$  bins. Since  $a(X \cup Y) < 2\varepsilon m$ , we will require less than  $4\varepsilon m + 2$  bins. Therefore, the total number of compartmental bins used to pack  $\tilde{I}$  is less than  $(1 + 4\varepsilon)m + 2$ . ◀

### 4.3 Packing Algorithm

We now describe the `skewedCPack` algorithm for packing a set  $I$  of  $n$   $\delta$ -skewed items.

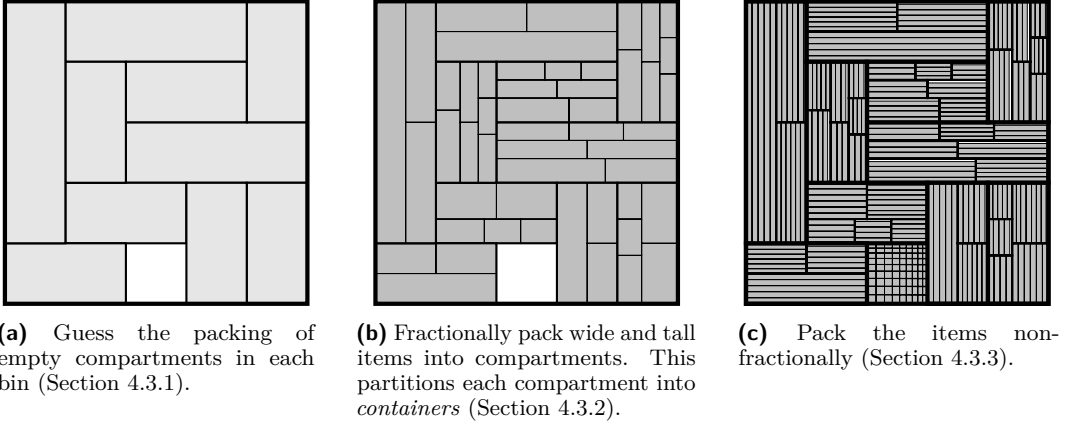
1. **Classifying and Rounding Items** (see Section 4.1): Compute  $(\tilde{I}, I_{\text{med}}) := \text{round}(I)$ . Recall that  $I_{\text{med}}$ , called the set of *medium items*, has low total area, and  $\tilde{I}$ , called the set of *rounded items*, is obtained by rounding up the width or height of each item in  $I - I_{\text{med}}$ .
2. **Enumerating Packing of Compartments:** Compute all possible packings of empty compartments into at most  $n$  bins.
3. **Fractionally Packing Items into Compartments:** For each packing  $P$  of empty compartments, fractionally pack  $\tilde{I}$  into  $P$  using a linear program.
4. **Converting a Fractional Packing to a Non-Fractional Packing:** Discard a small set  $D \subseteq \tilde{I}$  of items and use an extreme-point solution to the linear program to non-fractionally pack  $\tilde{I} - D$  into  $P$ .
5. Pack  $I_{\text{med}} \cup D$  into bins using NFDH.

See Figure 3 for a visual overview of `skewedCPack`. We describe steps 2, 3 and 4 in Sections 4.3.1–4.3.3, respectively. In Section 4.3.4, we bound the AAR of `skewedCPack`.

#### 4.3.1 Enumerating Packing of Compartments

We will now describe a subroutine, called `iterPackings`( $\tilde{I}$ ), that outputs all packings of empty compartments into at least  $\lceil a(\tilde{I}) \rceil$  bins and at most  $n$  bins. A packing of empty compartments in a bin is called a *configuration*. We will first enumerate all configurations and then output multisets of configurations of cardinality ranging from  $\lceil a(\tilde{I}) \rceil$  to  $n$ .

<sup>3</sup> A *fractional compartmental packing* of  $\tilde{I}$  is a fractional packing of  $\tilde{I}$  that is also compartmental.



■ **Figure 3** Major steps of `skewedCPack` after rounding  $I$ .

There can be at most  $n_W := 3(1/\varepsilon_1 - 1)|\mathcal{T}| + 1$  wide compartments in a bin. Each wide compartment can have  $(1/\varepsilon_{\text{cont}})^2$   $y$ -coordinates of the top and bottom edges and at most  $|\mathcal{T}|^2/2$   $x$ -coordinates of the left and right edges, where  $\varepsilon_{\text{cont}} := \varepsilon\varepsilon_1/6|\mathcal{T}|$ . The rest of the space is for tall compartments. Therefore, the number of configurations is at most

$$n_C := ((1/\varepsilon_{\text{cont}})^2|\mathcal{T}|^2/2)^{n_W} \leq \left(\frac{3|\mathcal{T}|^2}{\varepsilon\varepsilon_1}\right)^{6|\mathcal{T}|/\varepsilon_1} \leq \left(1 + \frac{1}{\varepsilon\varepsilon_1}\right)^{\left(1 + \frac{1}{\varepsilon\varepsilon_1}\right)^{2/\varepsilon_1+1}}.$$

Since each configuration can have at most  $n$  bins, the number of combinations of configurations is at most  $(n+1)^{n_C}$ . Therefore, we can output all possible bin packings of empty compartments in  $O(n^{n_C})$  time. This completes the description of `iterPackings`.

### 4.3.2 Fractionally Packing Items into Compartments

For each bin packing  $P$  of empty compartments, we will try to fractionally pack the items into the bins. To do this, we will create a feasibility linear program, called  $\text{FP}(\tilde{I}, P)$ , that is feasible iff wide and tall items in  $\tilde{I}$  can be packed into the compartments in  $P$ . If  $\text{FP}(\tilde{I}, P)$  is feasible, then small items can also be fractionally packed since  $P$  contains at least  $a(\tilde{I})$  bins.

Let  $w'_1, w'_2, \dots, w'_p$  be the distinct widths of wide compartments in  $P$ . Let  $U_j$  be the set of wide compartments in  $P$  having width  $w'_j$ . Let  $h(U_j)$  be the sum of heights of the compartments in  $U_j$ . By Definition 23, we know that  $p \leq |\mathcal{T}|^2/2$ . Let  $w_1, w_2, \dots, w_r$  be the distinct widths of items in  $\tilde{W}$  (recall that  $\tilde{W}$  is the set of wide items in  $\tilde{I}$ ). Let  $\tilde{W}_j$  be the items in  $\tilde{W}$  having width  $w_j$ . Let  $h(\tilde{W}_j)$  be the sum of heights of all items in  $\tilde{W}_j$ . By Claim 27, we get  $r \leq 1/\varepsilon\varepsilon_1$ .

Let  $C := [C_0, C_1, \dots, C_r]$  be a vector, where  $C_0 \in [p]$  and  $C_j \in \mathbb{Z}_{\geq 0}$  for  $j \in [r]$ .  $C$  is called a *wide configuration* iff  $w(C) := \sum_{j=1}^r C_j w_j \leq w'_{C_0}$ . Intuitively, a wide configuration  $C$  represents a set of wide items that can be placed side-by-side into a compartment of width  $w'_{C_0}$ . Let  $\mathcal{C}$  be the set of all wide configurations. Then  $|\mathcal{C}| \leq p/\varepsilon_1^r$ , which is a constant. Let  $\mathcal{C}_j := \{C \in \mathcal{C} : C_0 = j\}$ .

To pack  $\tilde{W}$  into wide compartments, we must determine the height of each configuration. Let  $x \in \mathbb{R}_{\geq 0}^{|\mathcal{C}|}$  be a vector where  $x_C$  denotes the height of configuration  $C$ . Then  $\tilde{W}$  can be packed into wide compartments according to  $x$  iff  $x$  is a feasible solution to the following

feasibility linear program, named  $\text{FP}_W(\tilde{I}, P)$ :

$$\begin{aligned} \sum_{C \in \mathcal{C}} C_j x_C &\geq h(\tilde{W}_j) \quad \forall j \in [r] && (\tilde{W}_j \text{ should be covered}) \\ \sum_{C \in \mathcal{C} \text{ and } C_0=j} x_C &\leq h(U_j) \quad \forall j \in [p] && (C_j \text{ should fit in } U_j) \\ x_C &\geq 0 \quad \forall C \in \mathcal{C} \end{aligned}$$

Let  $x^*$  be an extreme point solution to  $\text{FP}_W(\tilde{I}, P)$  (if  $\text{FP}_W(\tilde{I}, P)$  is feasible). By Rank Lemma<sup>4</sup>, at most  $p + r$  entries of  $x^*$  are non-zero. Since the number of variables and constraints is constant,  $x^*$  can be computed in constant time.

Let  $\tilde{H}$  be the set of tall items in  $\tilde{I}$ . Items in  $\tilde{H}$  have at most  $1/\varepsilon\varepsilon_1$  distinct heights. Let there be  $q$  distinct heights of tall compartments in  $P$ . By Definition 23, we get  $q \leq 1/\varepsilon_{\text{cont}} = 6|\mathcal{T}|/\varepsilon\varepsilon_1$ . We can similarly define *tall configurations* and define a feasibility linear program for tall items, named  $\text{FP}_H(\tilde{I}, P)$ .  $\tilde{H}$  can be packed into tall compartments in  $P$  iff  $\text{FP}_H(\tilde{I}, P)$  is feasible. Let  $y^*$  be an extreme point solution to  $\text{FP}_H(\tilde{I}, P)$ . Then  $y^*$  can be computed in constant time and  $y^*$  has at most  $q + 1/\varepsilon\varepsilon_1$  positive entries.

Hence,  $\tilde{I}$  can be packed into  $P$  iff  $\text{FP}(\tilde{I}, P) := \text{FP}_W(\tilde{I}, P) \wedge \text{FP}_H(\tilde{I}, P)$  is feasible. The solution  $(x^*, y^*)$  shows us how to split each compartment into *shelves*, where each shelf corresponds to a configuration  $C$  and the shelf can be split into  $C_j$  *containers* of width  $w_j$  and one container of width  $w'_{C_0} - w(C)$ . Let there be  $m$  bins in  $P$ . After splitting the configurations across compartments, we get at most  $p + q + 2/\varepsilon\varepsilon_1 + m(n_W + n_H)$  shelves.

### 4.3.3 Converting a Fractional Packing to a Non-Fractional Packing

Consider a packing  $P$  of empty compartments into  $m$  bins. Let  $x^*$  and  $y^*$  be extreme-point solutions to  $\text{FP}_W(\tilde{I}, P)$  and  $\text{FP}_H(\tilde{I}, P)$ , respectively (assuming  $\tilde{I}$  can fit into  $P$ ). Then  $(x^*, y^*)$  gives us a fractional compartmental packing of  $\tilde{I}$  into  $m$  bins. We now show how to convert this to a non-fractional compartmental packing by removing some items from  $\tilde{I}$ .

Formally, we give an algorithm called  $\text{greedyCPack}(\tilde{I}, P, x^*, y^*)$ . It returns a pair  $(Q, D)$ , where  $Q$  is a (non-fractional) compartmental bin packing of items  $\tilde{I} - D$ , where the compartments in the bins are as per  $P$ .  $D$  is called the set of discarded items.

$\text{greedyCPack}$  is based on standard techniques. We prove in Appendix C.2 that

$$a(D) < \frac{52|\mathcal{T}|\varepsilon_2}{\varepsilon_1} m + 4\varepsilon_2 \left( \frac{|\mathcal{T}|^2}{2} + \frac{6|\mathcal{T}|}{\varepsilon\varepsilon_1} + \frac{2}{\varepsilon\varepsilon_1} \right). \quad (1)$$

We give an outline of  $\text{greedyCPack}$  here and defer the details to Appendix C.2.

1. For each  $j$ , iteratively assign wide items from  $\tilde{W}_j$  to a container of width  $w_j$ . When the total height of assigned items exceeds the height of the container, discard the last-assigned item and switch to a new container and repeat.
2. Similarly assign tall items to tall containers.
3. Identify rectangular regions where we can pack small items:
  - For each configuration  $C$ , there is a free region of width  $w'_{C_0} - w(C)$  and height  $x^*_C$  in a wide compartment. Similarly, we get free regions in tall compartments.

<sup>4</sup> Rank Lemma: the number of non-zero variables in an extreme-point solution to a linear program is at most the number of non-trivial constraints [34, Lemma 2.1.4].

- In each bin, the number of compartments is constant, so the space outside compartments be divided into a constant number of rectangular regions (see Lemma 29).  
Pack most of the small items into these free regions using NFDH. Discard the rest.

#### 4.3.4 Summary

See Appendix C.3 for a precise description of `skewedCPack`.

Recall the function  $f$  from Section 4.1. Since  $\varepsilon_2 := f(\varepsilon_1)$ , we get

$$\varepsilon_2 = f(\varepsilon_1) = \frac{\varepsilon\varepsilon_1}{104(1 + 1/\varepsilon\varepsilon_1)^{2/\varepsilon_1-2}} \leq \frac{\varepsilon\varepsilon_1}{104|\mathcal{T}|}. \quad (2)$$

The last inequality follows from the fact that  $|\mathcal{T}| \leq (1 + 1/\varepsilon\varepsilon_1)^{2/\varepsilon_1-2}$ .

► **Theorem 26.** *The number of bins used by `skewedCPackε( $\tilde{I}$ )` is less than*

$$(1 + 20\varepsilon) \text{opt}(I) + \frac{1}{13} \left(1 + \frac{1}{\varepsilon\varepsilon_1}\right)^{2/\varepsilon_1-2} + 23.$$

**Proof.** In an optimal fractional compartmental bin packing of  $\tilde{I}$ , let  $P^*$  be the corresponding packing of empty compartments into bins. Hence,  $P^*$  contains  $m := \text{fcopt}(\tilde{I})$  bins. Since `iterPackings( $\tilde{I}$ )` iterates over all bin packings of compartments,  $P^* \in \text{iterPackings}(\tilde{I})$ . Since wide and tall items in  $\tilde{I}$  can be packed into the compartments of  $P^*$ , we get that  $x^*$  and  $y^*$  are not null. By Lemma 3, the number of bins used by NFDH to pack  $I_{\text{med}} \cup D$  is less than  $2a(I_{\text{med}} \cup D)/(1 - \delta) + 3 + 1/(1 - \delta)$ . Therefore, the number of bins used by `skewedCPack( $I$ )` is less than

$$\begin{aligned} & m + \frac{2a(I_{\text{med}} \cup D)}{1 - \delta} + 3 + \frac{1}{1 - \delta} \\ & < m + \frac{2\varepsilon}{1 - \delta}a(I) + \frac{2\varepsilon_2}{1 - \delta} \left( \frac{52|\mathcal{T}|}{\varepsilon_1}m + 4 \left( \frac{|\mathcal{T}|^2}{2} + \frac{6|\mathcal{T}| + 2}{\varepsilon\varepsilon_1} \right) \right) + 3 + \frac{1}{1 - \delta} \\ & \hspace{15em} \text{(by Equation (1) and } a(I_{\text{med}}) \leq \varepsilon a(I)) \\ & = \left(1 + \frac{104\varepsilon_2|\mathcal{T}|}{\varepsilon_1(1 - \delta)}\right) m + \frac{2\varepsilon}{1 - \delta}a(I) + 3 + \frac{1}{1 - \delta} + \frac{8\varepsilon_2}{1 - \delta} \left( \frac{|\mathcal{T}|^2}{2} + \frac{6|\mathcal{T}| + 2}{\varepsilon\varepsilon_1} \right) \\ & = \left(1 + \frac{\varepsilon}{1 - \delta}\right) m + \frac{2\varepsilon}{1 - \delta}a(I) + 3 + \frac{1}{13(1 - \delta)} \left( \frac{\varepsilon\varepsilon_1|\mathcal{T}|}{2} + 19 + \frac{2}{|\mathcal{T}|} \right). \end{aligned}$$

(by Equation (2))

By Theorem 25 and Lemma 18, we get

$$m = \text{fcopt}(\tilde{I}) < (1 + 4\varepsilon) \text{fopt}(\tilde{I}) + 2 < (1 + 4\varepsilon)(1 + \varepsilon) \text{opt}(I) + 4 + 8\varepsilon.$$

Therefore, the number of bins used by `skewedCPack( $I$ )` is less than

$$\begin{aligned} & \left( (1 + 4\varepsilon)(1 + \varepsilon) \left(1 + \frac{\varepsilon}{1 - \delta}\right) + \frac{2\varepsilon}{1 - \delta} \right) \text{opt}(I) \\ & \quad + (4 + 8\varepsilon) \left(1 + \frac{\varepsilon}{1 - \delta}\right) + 3 + \frac{1}{13(1 - \delta)} \left( \frac{\varepsilon\varepsilon_1|\mathcal{T}|}{2} + 19 + \frac{2}{|\mathcal{T}|} \right) \\ & \leq (1 + 20\varepsilon) \text{opt}(I) + \frac{1}{13} \left(1 + \frac{1}{\varepsilon\varepsilon_1}\right)^{2/\varepsilon_1-2} + 23. \end{aligned} \quad \text{(since } \delta \leq \varepsilon_1 \leq \varepsilon \leq 1/2)$$

◀

## References

- 1 Fidaa Abed, Parinya Chalermsook, José Correa, Andreas Karrenbauer, Pablo Pérez-Lantero, José A Soto, and Andreas Wiese. On guillotine cutting sequences. In *International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 1–19, 2015. doi:10.4230/LIPIcs.APPROX-RANDOM.2015.1.
- 2 Anna Adamaszek, Sarel Har-Peled, and Andreas Wiese. Approximation schemes for independent set and sparse subsets of polygons. *Journal of the ACM*, 66(4):29:1–29:40, 2019. doi:10.1145/3326122.
- 3 Soroush Alamdari, Therese Biedl, Timothy M Chan, Elyot Grant, Krishnam Raju Jampani, Srinivasan Keshav, Anna Lubiw, and Vinayak Pathak. Smart-grid electricity allocation via strip packing with slicing. In *Workshop on Algorithms and Data Structures (WADS)*, pages 25–36. Springer, 2013. doi:10.1007/978-3-642-40104-6\_3.
- 4 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. doi:10.1137/080736831.
- 5 Nikhil Bansal, José R Correa, Claire Kenyon, and Maxim Sviridenko. Bin packing in multiple dimensions: Inapproximability results and approximation schemes. *Mathematics of Operations Research*, 31(1):31–49, 2006. doi:10.1287/moor.1050.0168.
- 6 Nikhil Bansal, Marek Eliáš, and Arindam Khan. Improved approximation for vector bin packing. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1561–1579. SIAM, 2016. doi:10.1137/1.9781611974331.ch106.
- 7 Nikhil Bansal and Arindam Khan. Improved approximation algorithm for two-dimensional bin packing. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 13–25, 2014. doi:10.1137/1.9781611973402.2.
- 8 Nikhil Bansal, Andrea Lodi, and Maxim Sviridenko. A tale of two dimensional bin packing. In *Symposium on Foundations of Computer Science (FOCS)*, pages 657–666. IEEE, 2005. doi:10.1109/SFCS.2005.10.
- 9 Alberto Caprara. Packing  $d$ -dimensional bins in  $d$  stages. *Mathematics of Operations Research*, 33:203–215, February 2008. doi:10.1287/moor.1070.0289.
- 10 Alberto Caprara, Andrea Lodi, and Michele Monaci. Fast approximation schemes for two-stage, two-dimensional bin packing. *Mathematics of Operations Research*, 30(1):150–172, 2005. doi:10.1287/moor.1040.0112.
- 11 Miroslav Chlebík and Janka Chlebíková. Hardness of approximation for orthogonal rectangle packing and covering problems. *Journal of Discrete Algorithms*, 7(3):291–305, 2009. doi:10.1016/j.jda.2009.02.002.
- 12 Henrik I. Christensen, Arindam Khan, Sebastian Pokutta, and Prasad Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79, 2017. doi:10.1016/j.cosrev.2016.12.001.
- 13 Fan RK Chung, Michael R Garey, and David S Johnson. On packing two-dimensional bins. *SIAM Journal on Algebraic Discrete Methods*, 3(1):66–76, 1982. doi:10.1137/0603007.
- 14 Edward G. Coffman, Michael R. Garey, David S. Johnson, and Robert E. Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9:808–826, 1980. doi:10.1137/0209062.
- 15 W Fernandez De La Vega and George S. Lueker. Bin packing can be solved within  $1 + \varepsilon$  in linear time. *Combinatorica*, 1(4):349–355, 1981. doi:10.1007/BF02579456.
- 16 Max A. Deppert, Klaus Jansen, Arindam Khan, Malin Rau, and Malte Tutas. Peak demand minimization via sliced strip packing, 2021. arXiv:2105.07219.
- 17 Aleksei V Fishkin, Olga Gerber, and Klaus Jansen. On efficient weighted rectangle packing with large resources. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 1039–1050. Springer, 2005. doi:10.1007/11602613\_103.
- 18 Waldo Gálvez, Fabrizio Grandoni, Afrouz Jabal Ameli, Klaus Jansen, Arindam Khan, and Malin Rau. A tight  $(3/2 + \varepsilon)$  approximation for skewed strip packing. In *International*

- Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.44.
- 19 Waldo Gálvez, Fabrizio Grandoni, Afrouz Jabal Ameli, and Kamyar Khodamoradi. Approximation algorithms for demand strip packing, 2021. arXiv:2105.08577.
  - 20 Waldo Gálvez, Fabrizio Grandoni, Sandy Heydrich, Salvatore Ingala, Arindam Khan, and Andreas Wiese. Approximating geometric knapsack via L-packings. In *Symposium on Foundations of Computer Science (FOCS)*, pages 260–271. IEEE, 2017. doi:10.1109/FOCS.2017.32.
  - 21 Waldo Gálvez, Fabrizio Grandoni, Arindam Khan, Diego Ramirez-Romero, and Andreas Wiese. Improved approximation algorithms for 2-dimensional knapsack: Packing into multiple L-shapes, spirals and more. In *International Symposium on Computational Geometry (SoCG)*, volume 189, pages 39:1–39:17, 2021. doi:10.4230/LIPIcs.SoCG.2021.39.
  - 22 Paul C Gilmore and Ralph E Gomory. Multistage cutting stock problems of two and more dimensions. *Operations Research*, 13(1):94–120, 1965. doi:10.1287/opre.13.1.94.
  - 23 Rolf Harren, Klaus Jansen, Lars Prädél, and Rob Van Stee. A  $(5/3 + \varepsilon)$ -approximation for strip packing. In *Workshop on Algorithms and Data Structures (WADS)*, pages 475–487. Springer, 2011. doi:10.1007/978-3-642-22300-6\_40.
  - 24 Rebecca Hoberg and Thomas Rothvoss. A logarithmic additive integrality gap for bin packing. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2616–2625, 2017. doi:10.1137/1.9781611974782.172.
  - 25 Klaus Jansen and Lars Prädél. New approximability results for two-dimensional bin packing. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 919–936, 2013. doi:10.1007/s00453-014-9943-z.
  - 26 Klaus Jansen and Rob van Stee. On strip packing with rotations. In *ACM Symposium on Theory of Computing (STOC)*, pages 755–761, 2005. doi:10.1145/1060590.1060702.
  - 27 Klaus Jansen and Guochuan Zhang. On rectangle packing: maximizing benefits. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, volume 4, pages 204–213, 2004.
  - 28 Claire Kenyon and Eric Rémila. Approximate strip packing. In *Symposium on Foundations of Computer Science (FOCS)*, pages 31–36, 1996. doi:10.1109/SFCS.1996.548461.
  - 29 Arindam Khan, Arnab Maiti, Amatya Sharma, and Andreas Wiese. On guillotine separable packings for the two-dimensional geometric knapsack problem. In *International Symposium on Computational Geometry (SoCG)*, volume 189, pages 48:1–48:17, 2021. doi:10.4230/LIPIcs.SoCG.2021.48.
  - 30 Arindam Khan and Madhusudhan Reddy Pittu. On guillotine separability of squares and rectangles. In *International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.47.
  - 31 Arindam Khan and Eklavya Sharma. Tight approximation algorithms for geometric bin packing with skewed items. *ArXiv*, 2105.02827, 2021. arXiv:2105.02827.
  - 32 Arindam Khan, Eklavya Sharma, and K. V. N. Sreenivas. Approximation algorithms for generalized multidimensional knapsack. *ArXiv*, 2102.05854, 2021. arXiv:2102.05854.
  - 33 Arindam Khan, Eklavya Sharma, and K. V. N. Sreenivas. Geometry meets vectors: Approximation algorithms for multidimensional packing. *ArXiv*, 2106.13951, 2021. arXiv:2106.13951.
  - 34 Lap Chi Lau, Ramamoorthi Ravi, and Mohit Singh. *Iterative Methods in Combinatorial Optimization*, volume 46. Cambridge University Press, 2011.
  - 35 János Pach and Gábor Tardos. Cutting glass. In *International Symposium on Computational Geometry (SoCG)*, pages 360–369, 2000. doi:10.1145/336154.336223.
  - 36 Lars Dennis Prädél. *Approximation Algorithms for Geometric Packing Problems*. PhD thesis, Kiel University, 2012. URL: [https://macau.uni-kiel.de/servlets/MCRFileNodeServlet/dissertation\\_derivate\\_00004634/dissertation-praedel.pdf?AC=N](https://macau.uni-kiel.de/servlets/MCRFileNodeServlet/dissertation_derivate_00004634/dissertation-praedel.pdf?AC=N).
  - 37 Sai Sandeep. Almost optimal inapproximability of multidimensional packing problems. *ArXiv*, 2101.02854, 2021. arXiv:2101.02854.

- 38 Steven S. Seiden and Gerhard J. Woeginger. The two-dimensional cutting stock problem revisited. *Mathematical Programming*, 102(3):519–530, 2005. doi:10.1007/s10107-004-0548-1.
- 39 Eklavya Sharma. Harmonic algorithms for packing  $d$ -dimensional cuboids into bins. *ArXiv*, 2011.10963, 2020. arXiv:2011.10963.
- 40 A. Steinberg. A strip-packing algorithm with absolute performance bound 2. *SIAM Journal on Computing*, 26(2):401–409, 1997. doi:10.1137/S0097539793255801.
- 41 Paul E. Sweeney and Elizabeth Ridenour Paternoster. Cutting and packing problems: A categorized, application-orientated research bibliography. *Journal of the Operational Research Society*, 43(7):691–706, 1992. doi:10.1057/jors.1992.101.

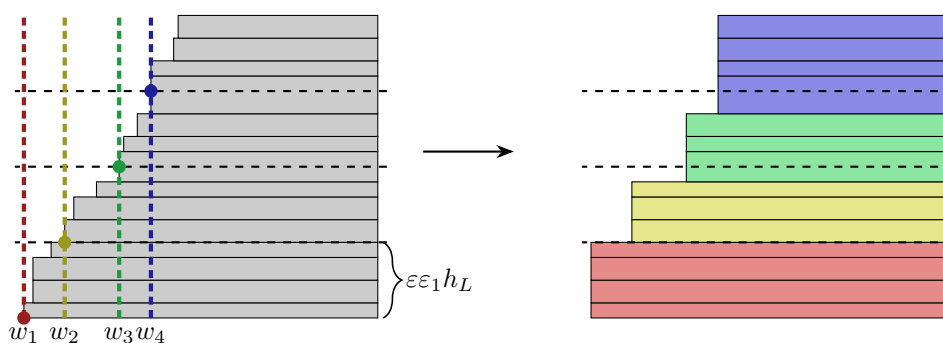
## A Linear Grouping

In this section, we describe the *linear grouping* technique [15, 28] for wide and tall items.

Let  $\varepsilon$  and  $\varepsilon_1$  be constants in  $(0, 1)$ . Let  $W$  be a set of items where each item has width more than  $\varepsilon_1$ . We will describe an algorithm, called `lingroupWide` that takes  $W$ ,  $\varepsilon$  and  $\varepsilon_1$  as input and returns the set  $\widehat{W}$  as output, where  $\widehat{W}$  is obtained by increasing the width of each item in  $W$ . `lingroupWide`( $W, \varepsilon, \varepsilon_1$ ) first arranges the items  $W$  in decreasing order of width and stacks them one-over-the-other (i.e., the widest item in  $W$  is at the bottom). Let  $h_L$  be the height of the stack. Let  $y(i)$  be the  $y$ -coordinate of the bottom edge of item  $i$ . Split the stack into sections of height  $\varepsilon\varepsilon_1 h_L$  each. For  $j \in [1/\varepsilon\varepsilon_1]$ , let  $w_j$  be the width of the widest item intersecting the  $j^{\text{th}}$  section from the bottom, i.e.,

$$w_j := \max(\{w(i) : i \in W \text{ and } (y(i), y(i) + h(i)) \cap ((j-1)\varepsilon\varepsilon_1 h_L, j\varepsilon\varepsilon_1 h_L) \neq \emptyset\}).$$

Round up the width of each item  $i$  to the smallest  $w_j$  that is at least  $w(i)$  (see Figure 4). Let  $W_j$  be the items whose width got rounded to  $w_j$  and let  $\widehat{W}_j$  be the resulting rounded items. (There may be ties, i.e., there may exist  $j_1 < j_2$  such that  $w_{j_1} = w_{j_2}$ . In that case, define  $W_{j_2} := \widehat{W}_{j_2} = \emptyset$ . This ensures that all  $W_j$  are disjoint.) Finally, define  $\widehat{W} := \bigcup_j \widehat{W}_j$ .



■ **Figure 4** Example invocation of `lingroupWide` for  $\varepsilon = \varepsilon_1 = 1/2$ .

We can similarly define the algorithm `lingroupTall`. Let  $H$  be a set of items where each item has height more than  $\varepsilon_1$ . `lingroupTall` that takes  $H$ ,  $\varepsilon$  and  $\varepsilon_1$  as input and returns  $\widehat{H}$ , where  $\widehat{H}$  is obtained by increasing the height of each item in  $H$ .

▷ **Claim 27.** Items in `lingroupWide`( $W, \varepsilon, \varepsilon_1$ ) have at most  $1/(\varepsilon\varepsilon_1)$  distinct widths. Items in `lingroupTall`( $H, \varepsilon, \varepsilon_1$ ) have at most  $1/(\varepsilon\varepsilon_1)$  distinct heights.

► **Lemma 28.** *Let  $W$ ,  $H$  and  $S$  be sets of items, where items in  $W$  have width more than  $\varepsilon_1$  and items in  $H$  have height more than  $\varepsilon_1$ . Let  $\widehat{W} := \text{lingroupWide}(W, \varepsilon, \varepsilon_1)$  and  $\widehat{H} := \text{lingroupTall}(H, \varepsilon, \varepsilon_1)$ . If we allow slicing items in  $\widehat{W}$  and  $\widehat{H}$  using horizontal and vertical cuts, respectively, then we can pack  $\widehat{W} \cup \widehat{H} \cup S$  into less than  $(1 + \varepsilon) \text{opt}(W \cup H \cup S) + 2$  bins.*

**Proof.** (cf. Appendix C in [31]) ◀

## B skewed4Pack: Packing Items into Containers

► **Lemma 13.** *Let  $P$  be a packing of  $\widetilde{I}$  into  $m$  bins, where we made at most  $m - 1$  horizontal cuts in wide shelves and at most  $m - 1$  vertical cuts in tall shelves. Then we can (without slicing) pack a large subset of  $\widehat{I}$  into the shelves in  $P$  such that the unpacked items (also called discarded items) from  $\widehat{W}$  have total area less than  $\varepsilon \text{hsum}(\widehat{W}) + \delta_H(1 + \varepsilon)(m + 1/\varepsilon^2)$ , and the unpacked items from  $\widehat{H}$  have area less than  $\varepsilon \text{wsum}(\widehat{H}) + \delta_W(1 + \varepsilon)(m + 1/\varepsilon^2)$ .*

**Proof.** For each  $j \in [1/\varepsilon^2]$ , number the type- $j$  wide containers arbitrarily, and number the items in  $\widehat{W}_j^{(L)}$  arbitrarily. Now greedily assign items from  $\widehat{W}_j^{(L)}$  to the first container  $C$  until the total height of the items exceeds  $h(C)$ . Then move to the next container and repeat. As per the constraints of the linear program, all items in  $\widehat{W}_j^{(L)}$  will get assigned to some type- $j$  wide container. Similarly, number the type-0 wide containers arbitrarily and number the items in  $W^{(S)}$  arbitrarily. Greedily assign items from  $W^{(S)}$  to the first container  $C$  until the total area of the items exceeds  $a(C)$ . Then move to the next container and repeat. As per the constraints of the linear program, all items in  $W^{(S)}$  will get assigned to some type-0 wide container. Similarly, assign all items from  $\widehat{H}$  to tall containers.

Let  $C$  be a type- $j$  wide container and  $\widehat{J}$  be the items assigned to it. If we discard the last item from  $\widehat{J}$ , then the items can be packed into  $C$ . The area of the discarded item is at most  $w(C)\delta_H$ . Let  $C$  be a type-0 wide container and  $\widehat{J}$  be the items assigned to it. Arrange the items in  $\widehat{J}$  in decreasing order of height and pack the largest prefix  $\widehat{J}' \subseteq \widehat{J}$  into  $C$  using NFDW (Next-Fit Decreasing Width).

Discard the items  $\widehat{J} - \widehat{J}'$ . By Lemma 2,  $a(\widehat{J} - \widehat{J}') < \varepsilon h(C) + \delta_H w(C) + \varepsilon \delta_H$ . Therefore, for a wide shelf  $S$ , the total area of discarded items is less than  $\varepsilon h(S) + \delta_H(1 + \varepsilon)$ .

After slicing the shelves in  $\widetilde{I}$  to get  $P$ , we get at most  $m + 1/\varepsilon^2$  wide shelves and at most  $m + 1/\varepsilon^2$  tall shelves. Therefore, the total area of discarded items from  $W$  is less than  $\varepsilon \text{hsum}(\widehat{W}) + \delta_H(1 + \varepsilon)(m + 1/\varepsilon^2)$ , and the total area of discarded items from  $H$  is less than  $\varepsilon \text{wsum}(\widehat{H}) + \delta_W(1 + \varepsilon)(m + 1/\varepsilon^2)$ . ◀

## C Details of skewedCPack

### C.1 Removing Medium Items

Let  $T := \lceil 2/\varepsilon \rceil$ . Let  $\mu_0 = \varepsilon$ . For  $t \in [T]$ , define  $\mu_t := f(\mu_{t-1})$  and define

$$J_t := \{i \in I : w(i) \in (\mu_t, \mu_{t-1}] \text{ or } h(i) \in (\mu_t, \mu_{t-1}]\}.$$

Let  $r := \text{argmin}_{t=1}^T a(J_t)$ ,  $I_{\text{med}} := J_r$ ,  $\varepsilon_1 := \mu_{r-1}$ . Each item belongs to at most 2 sets  $J_t$ , so

$$a(I_{\text{med}}) = \min_{t=1}^T a(J_t) \leq \frac{1}{T} \sum_{t=1}^T a(J_t) \leq \frac{2}{\lceil 2/\varepsilon \rceil} a(I) \leq \varepsilon a(I).$$

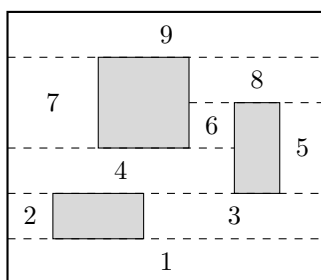


### C.2 Converting a Fractional Packing to a Non-Fractional Packing

► **Lemma 29.** *Let there be a set  $I$  of rectangles packed inside a bin. Then there is a polynomial-time algorithm that can decompose the empty space in the bin into at most  $3|I| + 1$  rectangles by making horizontal cuts only.*

**Proof.** Extend the top and bottom edge of each rectangle leftwards and rightwards till they hit another rectangle or an edge of the bin. This decomposes the empty region into rectangles  $R$ . See Figure 5.

For each rectangle  $i \in I$ , the top edge of  $i$  is the bottom edge of a rectangle in  $R$ , the bottom edge of  $i$  is the bottom edge of two rectangles in  $R$ . Apart from possibly the rectangle in  $R$  whose bottom edge is at the bottom of the bin, the bottom edge of every rectangle in  $R$  is either the bottom or top edge of a rectangle in  $I$ . Therefore,  $|R| \leq 3|I| + 1$ . ◀



■ **Figure 5** Horizontal cuts partition empty space around the 3 items into 9 rectangular regions.

Let  $(Q, D) := \text{greedyCPack}(\tilde{I}, P, x^*, y^*)$ , where  $P$  is a packing of empty compartments into  $m$  bins. We will describe  $\text{greedyCPack}$  and show that

$$a(D) < \frac{52|\mathcal{T}|\varepsilon_2}{\varepsilon_1}m + 4\varepsilon_2 \left( \frac{|\mathcal{T}|^2}{2} + \frac{6|\mathcal{T}|}{\varepsilon\varepsilon_1} + \frac{2}{\varepsilon\varepsilon_1} \right).$$

For a configuration  $C$  in a wide compartment, there is a container of width  $w'_{C_0} - w(C)$  available for packing small items. Hence, there are  $p + q + 2/\varepsilon\varepsilon_1 + m(n_W + n_H)$  containers available inside compartments for packing small items. By Lemma 29, we can partition the space outside compartments into at most  $m(3(n_W + n_H) + 1)$  containers. Therefore, the total number of containers available for packing small items is at most

$$m_S := (p + q + 2/\varepsilon\varepsilon_1) + m(4(n_W + n_H) + 1) \leq \left( \frac{|\mathcal{T}|^2}{2} + \frac{6|\mathcal{T}|}{\varepsilon\varepsilon_1} + \frac{2}{\varepsilon\varepsilon_1} \right) + \frac{16|\mathcal{T}|}{\varepsilon_1}m.$$

Greedy assign small items to small containers, i.e., keep assigning small items to a container till the area of items assigned to it is at least the area of the container, and then resume from the next container. Each small item will get assigned to some container. For each container  $C$ , pack the largest possible prefix of the assigned items using the Next-Fit Decreasing Height (NFDH) algorithm. By Lemma 2, the area of unpacked items would be less than  $\varepsilon_2 + \delta + \varepsilon_2\delta$ . Summing over all containers, we get that the unpacked area is less than  $(\varepsilon_2 + \delta + \varepsilon_2\delta)m_S \leq 3\varepsilon_2m_S$ .

For each  $j$ , greedily assign wide items from  $\tilde{W}_j$  to containers of width  $w_j$ , i.e., keep assigning items till the height of items exceeds the height of the container. Each wide item will get assigned to some container. Then discard the last item from each container. For each shelf in a wide compartment having configuration  $C$ , the total area of items we discard is at most  $\delta w(C)$ . Similarly, we can discard tall items of area at most  $\delta h(C)$  from each shelf in a tall compartment having configuration  $C$ .

## 22:22 Geometric Bin Packing with Skewed Items

Hence, across all configurations, we discard wide and tall items of area at most

$$\delta((p + q + 2/\varepsilon\varepsilon_1) + m(n_W + n_H)) \leq \delta \left( \frac{|\mathcal{T}|^2}{2} + \frac{6|\mathcal{T}|}{\varepsilon\varepsilon_1} + \frac{2}{\varepsilon\varepsilon_1} \right) + \frac{4\delta|\mathcal{T}|}{\varepsilon_1} m.$$

Therefore, for  $(Q, D) := \text{greedyCPack}(\tilde{I}, P, x^*, y^*)$ , we get

$$a(D) < \frac{52|\mathcal{T}|\varepsilon_2}{\varepsilon_1} m + 4\varepsilon_2 \left( \frac{|\mathcal{T}|^2}{2} + \frac{6|\mathcal{T}|}{\varepsilon\varepsilon_1} + \frac{2}{\varepsilon\varepsilon_1} \right).$$

### C.3 Pseudocode for skewedCPack

■ **Algorithm 1**  $\text{skewedCPack}_\varepsilon(I)$ : Packs a set  $I$  of  $\delta$ -skewed rectangular items into bins without rotating the items.

---

```

1:  $(\tilde{I}, I_{\text{med}}) = \text{round}_\varepsilon(I)$ .
2: Initialize  $Q_{\text{best}}$  to null.
3: for  $P \in \text{iterPackings}(\tilde{I})$  do                                // iterPackings is defined in Section 4.3.1.
4:    $x^* = \text{opt}(\text{FP}_W(\tilde{I}, P))$ .                                // FPW and FPH are defined in Section 4.3.2.
5:   // If  $\text{FP}_W(\tilde{I}, P)$  is feasible,  $x^*$  is an extreme-point solution to  $\text{FP}_W(\tilde{I}, P)$ .
6:   // If  $\text{FP}_W(\tilde{I}, P)$  is infeasible,  $x^*$  is null.
7:    $y^* = \text{opt}(\text{FP}_H(\tilde{I}, P))$ .
8:   if  $x^* \neq \text{null}$  and  $y^* \neq \text{null}$  then                    // if  $\tilde{I}$  can be packed into  $P$ 
9:      $(Q, D) = \text{greedyCPack}(\tilde{I}, P, x^*, y^*)$ . // greedyCPack is defined in Section 4.3.3.
10:     $Q_D = \text{NFDH}(D \cup I_{\text{med}})$ .
11:    if  $Q \cup Q_D$  uses less bins than  $Q_{\text{best}}$  then
12:       $Q_{\text{best}} = Q \cup Q_D$ .
13:    end if
14:  end if
15: end for
16: return  $Q_{\text{best}}$ 

```

---

## D Lower Bound on APoG

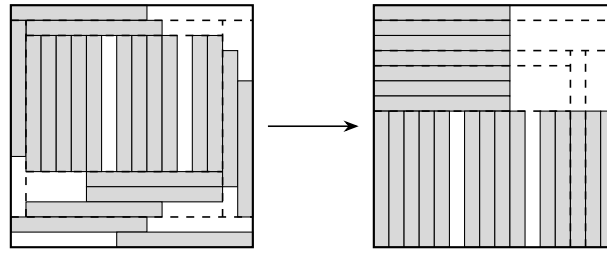
In this section, we prove a lower bound of roughly  $4/3$  on the APoG for skewed rectangles.

► **Lemma 30.** *Let  $m$  and  $k$  be positive integers and  $\varepsilon \in (0, 1)$ . Let  $J$  be a set of items packed into a bin, where each item has the longer dimension equal to  $(1 + \varepsilon)/2$  and the shorter dimension equal to  $(1 - \varepsilon)/2k$ . If the bin is guillotine-separable, then  $a(J) \leq 3/4 + \varepsilon/2 - \varepsilon^2/4$ .*

**Proof sketch.** For an item packed in the bin, if the height is  $(1 - \varepsilon)/2k$ , call it a wide item, and if the width is  $(1 - \varepsilon)/2k$ , call it a tall item. Let  $W$  be the set of wide items in  $J$ .

We can rearrange the items in the bin so that all wide items touch the left edge of the bin and all tall items touch the bottom edge of the bin. See Appendix E in [31] for a formal proof and Figure 6 for an example.

Therefore, the square region of side length  $(1 - \varepsilon)/2$  at the top-right corner of the bin is empty. Hence, the area occupied in each bin is at most  $3/4 + \varepsilon/2 - \varepsilon^2/4$ . ◀



■ **Figure 6** Structuring a guillotine-separable packing.

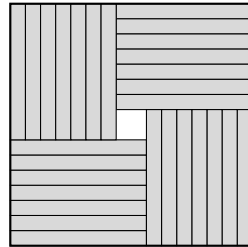
► **Theorem 31.** Let  $m$  and  $k$  be positive integers and  $\varepsilon \in (0, 1)$ . Let  $I$  be a set of  $4mk$  items, where  $2mk$  items have width  $(1 + \varepsilon)/2$  and height  $(1 - \varepsilon)/2k$ , and  $2mk$  items have height  $(1 + \varepsilon)/2$  and width  $(1 - \varepsilon)/2k$ . Let  $\text{opt}(I)$  be the number of bins in the optimal packing of  $I$  and  $\text{opt}_g(I)$  be the number of bins in the optimal guillotinable packing of  $I$ . Then

$$\frac{\text{opt}_g(I)}{\text{opt}(I)} \geq \frac{4}{3}(1 - \varepsilon).$$

This holds true even if items in  $I$  are allowed to be rotated.

**Proof.** For an item  $i \in I$ , if  $h(i) = (1 - \varepsilon)/2k$ , call it a wide item, and if  $w(i) = (1 - \varepsilon)/2k$ , call it a tall item. Let  $W$  be the set of wide items and  $H$  be the set of tall items.

Partition  $W$  into groups of  $k$  elements. In each group, stack items one-over-the-other. This gives us  $2m$  containers of width  $(1 + \varepsilon)/2$  and height  $(1 - \varepsilon)/2$ . Similarly, get  $2m$  containers of height  $(1 + \varepsilon)/2$  and height  $(1 - \varepsilon)/2$  by stacking items from  $H$  side-by-side. We can pack 4 containers in one bin, so  $I$  can be packed into  $m$  bins. See Figure 7 for an example. Therefore,  $\text{opt}(I) \leq m$ .



■ **Figure 7** Packing  $4k$  items in one bin. Here  $k = 7$ .

We will now show a lower-bound on  $\text{opt}_g(I)$ . In any guillotine-separable packing of  $I$ , the area occupied by each bin is at most  $3/4 + \varepsilon/2 - \varepsilon^2/4$  (by Lemma 30). Note that  $a(I) = m(1 - \varepsilon^2)$ . Therefore,

$$\begin{aligned} \text{opt}_g(I) &\geq \frac{m(1 - \varepsilon^2)}{3/4 + \varepsilon/2 - \varepsilon^2/4} \\ \implies \frac{\text{opt}_g(I)}{\text{opt}(I)} &\geq \frac{4}{3} \times \frac{1 - \varepsilon^2}{1 + 2\varepsilon/3 - \varepsilon^2/3} = \frac{4}{3} \times \frac{1 - \varepsilon}{1 - \varepsilon/3} \geq \frac{4}{3}(1 - \varepsilon). \end{aligned}$$



# Approximating Two-Stage Stochastic Supplier Problems

**Brian Brubach** ✉

Wellesley College, MA, USA

**Nathaniel Grammel** ✉

University of Maryland at College Park, MD, USA

**David G. Harris** ✉

University of Maryland at College Park, MD, USA

**Aravind Srinivasan** ✉

University of Maryland at College Park, MD, USA

**Leonidas Tsepenekas** ✉

University of Maryland at College Park, MD, USA

**Anil Vullikanti** ✉

University of Virginia, Charlottesville, VA, USA

---

## Abstract

The main focus of this paper is radius-based (supplier) clustering in the two-stage stochastic setting with recourse, where the inherent stochasticity of the model comes in the form of a budget constraint. We also explore a number of variants where additional constraints are imposed on the first-stage decisions, specifically matroid and multi-knapsack constraints.

Our eventual goal is to provide results for supplier problems in the most general distributional setting, where there is only black-box access to the underlying distribution. To that end, we follow a two-step approach. First, we develop algorithms for a restricted version of each problem, in which all possible scenarios are explicitly provided; second, we employ a novel *scenario-discarding* variant of the standard *Sample Average Approximation (SAA)* method, in which we crucially exploit properties of the restricted-case algorithms. We finally note that the scenario-discarding modification to the SAA method is necessary in order to optimize over the radius.

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

**Keywords and phrases** Approximation Algorithms, Stochastic Optimization, Two-Stage Recourse Model, Clustering Problems, Knapsack Supplier

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.23

**Category** APPROX

**Related Version** *Full Version:* <https://arxiv.org/abs/2008.03325>

**Funding** *Brian Brubach:* Brian Brubach was supported in part by NSF awards CCF-1422569 and CCF-1749864, and by research awards from Adobe.

*Nathaniel Grammel:* Nathaniel Grammel was supported in part by NSF awards CCF-1749864 and CCF-1918749, and by research awards from Amazon and Google.

*Aravind Srinivasan:* Aravind Srinivasan was supported in part by NSF awards CCF-1422569, CCF-1749864 and CCF-1918749, and by research awards from Adobe, Amazon, and Google.

*Leonidas Tsepenekas:* Leonidas Tsepenekas was supported in part by NSF awards CCF-1749864 and CCF-1918749, and by research awards from Amazon and Google.

**Acknowledgements** The authors want to sincerely thank Chaitanya Swamy as well as referees of earlier versions of the paper, for their precious feedback and helpful suggestions.



© Brian Brubach, Nathaniel Grammel, David G. Harris, Aravind Srinivasan, Leonidas Tsepenekas, and Anil Vullikanti;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 23; pp. 23:1–23:22



Leibniz International Proceedings in Informatics

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

## 1 Introduction

Stochastic optimization, first introduced in the work of Beale [2] and Dantzig [5], provides a way for modeling uncertainty in the realization of the input data. In this paper, we give approximation algorithms for a family of problems in stochastic optimization, and more precisely in the *2-stage recourse model* [22]. Our formal problem definitions follow.

We are given a set of clients  $\mathcal{C}$  and a set of facilities  $\mathcal{F}$ , in a metric space characterized by a distance function  $d$ . We let  $n = |\mathcal{C}|$  and  $m = |\mathcal{F}|$ . Our paradigm unfolds in two stages. In the first, each  $i \in \mathcal{F}$  has a cost  $c_i^I$ , but at that time we do not know which clients from  $\mathcal{C}$  will need service, and we only have a description of the distribution  $\mathcal{D}$  that governs the arrivals of clients later on. In the second stage, a *scenario*  $A \subseteq \mathcal{C}$  is realized with probability  $p_A$  according to  $\mathcal{D}$ , and now each  $i \in \mathcal{F}$  has a cost  $c_i^A$ . The clients of the realized scenario are precisely those that will require service from the facilities of  $\mathcal{F}$ . Using only the description of the distribution  $\mathcal{D}$ , we can proactively open a set of facilities  $F_I$  in *stage-I*. Subsequently, when a scenario  $A$  arrives in *stage-II*, we can augment the already constructed solution by opening some additional facilities  $F_A$ .

Throughout the paper, the objective function we minimize is the *maximum covering distance or radius*. Let  $d(j, S) = \min_{i \in S} d(i, j)$  for any  $j \in \mathcal{C}$  and for any  $S \subseteq \mathcal{F}$ . We then ask for  $F_I$  and  $F_A$ , such that  $d(j, F_I \cup F_A) \leq R$  for every  $A$  that materializes and all  $j \in A$ , **for the minimum  $R$  possible**. Furthermore, the expected opening cost of the returned solution is required to be at most some given budget  $B$ , i.e.,  $\sum_{i \in F_I} c_i^I + \mathbb{E}_{A \sim \mathcal{D}} \left[ \sum_{i \in F_A} c_i^A \right] \leq B$ . We call this problem *Two-Stage Stochastic Supplier* or **2S-Sup** for short.

Finally, we assume that for every  $j \in \mathcal{C}$  we have  $\Pr_{A \sim \mathcal{D}}[j \in A] > 0$ ; note that if this is not the case, then the presence of  $j$  in the input is completely redundant.

**Additional Stage-I Constraints.** Beyond the basic version of the problem, we also consider variants where there are additional hard constraints on the set of chosen stage-I facilities.

In *Two-Stage Stochastic Matroid Supplier* or **2S-MatSup** for short, the input also includes a matroid  $\mathcal{M} = (\mathcal{F}, \mathcal{I})$ , where  $\mathcal{I} \subseteq 2^{\mathcal{F}}$  is the family of independent sets of  $\mathcal{M}$ . In this case, we additionally require  $F_I \in \mathcal{I}$ .

In *Two-Stage Stochastic Multi-knapsack Supplier* or **2S-MuSup** for short,  $L$  additional knapsack constraints are imposed on  $F_I$ . Specifically, we are given budgets  $W_\ell \geq 0$  and weights  $f_i^\ell \geq 0$  for every  $i \in \mathcal{F}$  and every integer  $\ell \in [L]$ , such that the stage-I facilities should satisfy  $\sum_{i \in F_I} f_i^\ell \leq W_\ell$  for every  $\ell \in [L]$ . We also call a **2S-MuSup** instance *discrete*, if all weights  $f_i^\ell$  are integers, and for such an instance we further define a parameter  $\Lambda = \prod_{\ell=1}^L W_\ell$ .

**Modeling the Stage-I Distributional Knowledge.** To complete the description of a two-stage problem, one needs to define how knowledge of the distribution  $\mathcal{D}$  is represented in stage-I.

The most general representation is the *black-box* model [20, 8, 17, 14, 19], where we only have access to an oracle that can sample scenarios  $A$  according to  $\mathcal{D}$ . In this model, every time a scenario  $A$  is revealed, either through the oracle or through an actual data realization, we also learn the facility-cost vector  $c^A$  associated with it. We also consider the more restricted *polynomial-scenarios* model [18, 11, 16, 7], where all scenarios  $A$ , together with their occurrence probabilities  $p_A$  and their corresponding facility-cost vectors  $c^A$ , are explicitly provided.

We use the suffixes **BB** and **Poly** to distinguish these settings. For example, **2S-Sup-BB** is the previously defined **2S-Sup** in the black-box model.

In both distributional settings, our algorithms must have runtime polynomial in  $n, m$ . For the polynomial-scenarios case, the runtime should also be polynomial in the number of explicitly provided scenarios.

## 1.1 Motivation

To our knowledge, we are the first to consider this type of radius minimization problems in the two-stage stochastic paradigm. Regarding clustering problems in this regime, most prior work has focused on *Facility Location* [18, 20, 16, 17, 9, 14, 19]. On similar lines, [1] studies a stochastic  $k$ -center variant, where points arrive independently, *but each point only needs to get covered with some given probability*. Moreover, **2S-Sup** is the natural two-stage counterpart of the well-known **Knapsack-Supplier** problem [10]. **Knapsack-Supplier** has a 3-approximation, which is also the best ratio possible unless  $P=NP$  [10].

To see a practical application for our problems, consider healthcare resource allocation, when trying to mitigate a disease outbreak through the preventive placement of testing sites. Suppose that  $\mathcal{F}$  corresponds to potential locations that can host a testing center (e.g., hospitals, private clinics, university labs),  $\mathcal{C}$  to populations that can be affected by a possible disease outbreak, and each scenario  $A \in \mathcal{D}$  to which populations suffer the outbreak. Since immediate testing is of utmost importance, a central decision maker may prepare testing sites, such that under every scenario, each infected population has the closest possible access to a testing center. Assembling these sites in advance, i.e., in stage-I, has multiple benefits; for example, the necessary equipment and materials might be much cheaper and easier to obtain before the onset of the disease. Furthermore, the choice to minimize the maximum covering distance, as opposed to the opening cost, would reflect a policy valuing societal welfare more than economic performance.

In addition, there may be further constraints on  $F_I$ , irrespective of the stage-II decisions, which cannot be directly reduced to the budget  $B$ . For instance, we might have a constraint on the total number of personnel we want to occupy prior to the outbreak of the disease, assuming that facility  $i$  requires  $f_i$  people to keep it operational during the waiting period. To our knowledge, this is the first time additional stage-I constraints are studied in the two-stage stochastic regime.

## 1.2 Our Generalization Scheme and Comparison with Previous Results

Our ultimate goal is to devise algorithms for the black-box setting. As is usual in two-stage stochastic problems, we do this in three steps. First, we develop algorithms for the less complicated polynomial-scenarios model. Second, we sample a small number of scenarios from the black-box oracle and use our polynomial-scenarios algorithms to (approximately) solve the problems on them. Finally, we extrapolate this solution to the original black-box problem. This overall methodology is called *Sample Average Approximation (SAA)*.

Unfortunately, standard SAA approaches [21, 4] cannot be directly applied in radius minimization problems. On a high level, the obstacle here is that we need to compute the true cost of the approximate solution, something that is impossible using already existing results. Because this is a delicate technical issue, we refer the reader to Appendix A for an in-depth discussion.

**Our Sampling Framework.** Since the optimal black-box radius  $R^*$  is always the distance between a client and a facility, there are at most  $nm$  different options for it. Thus, we consider each separately, and assume for now that we work with a specific guess  $R$ . Given

this, we sample some  $N$  scenarios from the oracle, and let  $Q = \{S_1, S_2, \dots, S_N\}$  be that sampled set. We then run our polynomial-scenarios  $\eta$ -approximation algorithms on  $Q$ , which are guaranteed to provide solutions that cover each client within distance  $\eta R$ . Crucially, we show that if  $R \geq R^*$  and  $N$  is chosen appropriately, these solutions have cost at most  $(1 + \epsilon)B$  on  $Q$ , for any  $\epsilon > 0$ . Hence, in the end we keep the minimum guess for  $R$  whose cost over the samples is at most  $(1 + \epsilon)B$ .

For this minimum guess  $R$  (which obviously satisfies  $R \leq R^*$ ), the polynomial-scenarios algorithm returned a stage-I set  $F_I$ , and a stage-II set  $F_{S_v}$  for each  $S_v \in Q$ . **Our polynomial-scenarios algorithms are also designed to satisfy two additional key properties.** First, given  $F_I$  and any  $A \notin Q$ , there is an *efficient* process to *extend* the algorithm's output to a stage-II solution  $F_A$  with  $d(j, F_I \cup F_A) \leq \eta R$  for all  $j \in A$ . Second, irrespective of  $Q$ , the set  $\mathcal{S}$  of possible black-box solutions the extension process might produce, has only exponential size as a function of  $n$  and  $m$  (by default, it could have size  $2^{m|\mathcal{D}|}$ , and note that  $\mathcal{D}$  may be exponentially large or even uncountably infinite). **We call algorithms satisfying these properties *efficiently generalizable*.**

After using the extension process to construct a solution for every  $A$  that materializes, there is a final *scenario-discarding* step to our framework. Specifically, for some given  $\alpha \in (0, 1)$ , we first determine a threshold value  $T$  corresponding to the  $\lceil \alpha |Q| \rceil^{\text{th}}$  costliest scenario of  $Q$ . Then, if for an arriving  $A$  the computed set  $F_A$  has stage-II cost more than  $T$ , we perform no stage-II openings by setting  $F_A = \emptyset$  (i.e., we “give up” on  $A$ ). This step coupled with the bounds on  $|\mathcal{S}|$  ensure that the overall opening cost of our solution is at most  $(1 + \epsilon)B$ . At this point, note that discarding implies that there may exist scenarios  $A$  with  $d(j, F_I \cup F_A) > \eta R$  for some  $j \in A$ . However, we show such scenarios occur with probability at most  $\alpha$ , and the latter can be made inverse polynomially small.

### 1.3 Outline and Contributions

In Section 2, we present our generalization scheme. We summarize it as follows:

► **Theorem 1.** *Suppose we have an efficiently generalizable,  $\eta$ -approximation algorithm for the polynomial-scenarios variant of any of the problems we study. Let  $\mathcal{S}$  be the set of all potential black-box solutions its extension process may produce. Then, for any  $\gamma, \epsilon, \alpha \in (0, 1)$  and with  $\mathcal{O}\left(\frac{1}{\epsilon\alpha} \log\left(\frac{nm|\mathcal{S}|}{\gamma}\right) \log\left(\frac{nm}{\gamma}\right)\right)$  samples, we compute a radius  $R$  and a black-box solution  $F_I, F_A$  for all  $A \in \mathcal{D}$ :*

1.  $F_I$  satisfies the stage-I specific constraints of the problem (matroid or multiknapsack).
2. With probability at least  $1 - \gamma$ , we have  $R \leq R^*$  and  $\sum_{i \in F_I} c_i^I + \mathbb{E}_{A \sim \mathcal{D}}[\sum_{i \in F_A} c_i^A] \leq (1 + \epsilon)B$ , where  $R^*$  the optimal radius of the black-box variant.
3. With probability at least  $1 - \gamma$ , there holds  $\Pr_{A \sim \mathcal{D}}[d(j, F_I \cup F_A) \leq \eta R, \forall j \in A] \geq 1 - \alpha$ .

► **Theorem 2.** *We provide the following efficiently generalizable algorithms:*

- A 3-approximation for **2S-Sup-Poly** with  $|\mathcal{S}| \leq (n + 1)!$ .  
For the black-box case, the sample complexity of Theorem 1 is  $\tilde{O}\left(\frac{n}{\epsilon\alpha}\right)$ .
- A 5-approximation for **2S-MatSup-Poly** with  $|\mathcal{S}| \leq 2^m n!$ .  
For the black-box case, the sample complexity of Theorem 1 is  $\tilde{O}\left(\frac{m+n}{\epsilon\alpha}\right)$ .
- A 5-approximation for discrete instances of **2S-MuSup-Poly**, with  $|\mathcal{S}| \leq 2^m$  and runtime  $\text{poly}(n, m, \Lambda)$ . In the black-box case, the sample complexity of Theorem 1 is  $\tilde{O}\left(\frac{m}{\epsilon\alpha}\right)$ .

Here,  $\tilde{O}()$  hides  $\text{polylog}(n, m, 1/\gamma)$  terms. The 3-approximation for **2S-Sup-Poly** is presented in Section 3. It relies on a novel LP rounding technique, not used in clustering problems before. Notably, its approximation ratio matches the lower bound of the non-stochastic counterpart [10] (**Knapsack Supplier**), something very rare in the two-stage



paradigm. The 5-approximation for **2S-MatSup-Poly** is presented in Section 4. It relies on solving an auxiliary LP, whose optimal solution is guaranteed to be integral. The 5-approximation for **2S-MuSup-Poly** is presented in Appendix C, and is based on a reduction to a deterministic supplier problem with outliers. Specifically, if we view stage-I as consisting of a deterministic robust problem, stage-II is interpreted as trying to cover all outliers left over by stage-I.

**The main advantages of our generalization scheme are.**

1. Unlike standard SAA approaches [4, 21], it can handle problems based on the maximum-radius objective function.
2. The approximation ratio  $\eta$  is preserved with high probability during the generalization. By contrast, in typical two-stage problems, the approximation ratio usually gets inflated when generalizing the polynomial-scenarios setting to the black-box one.
3. The adaptive selection of  $T$  yields crisp sample bounds in terms of  $\alpha$  and  $\epsilon$ . By contrast, simpler non-adaptive approaches (e.g.,  $T = \frac{B}{\alpha}$ ) would still give the same guarantees, but the dependence of the sample bounds on  $\alpha$ ,  $\epsilon$  would be worse ( $\frac{1}{\epsilon^2\alpha^2}$  compared to  $\frac{1}{\epsilon\alpha}$  as we achieve). This adaptive thresholding may also be of independent interest; for instance, we conjecture that it might be able to improve the sample complexity in the SAA analysis of [4].

**Remark 1.** There is an important connection between the design of our generalization scheme and the design of our polynomial-scenarios approximation algorithms.

In any SAA approach, the sample complexity necessarily depends on the set of possible actions over which the generalization is performed. In Theorem 1, the sample bounds are given in terms of the *cardinality* of  $\mathcal{S}$ . Following the lines of [21], it may be possible to replace this dependence with a notion of dimension of the underlying convex program. However, such general bounds would lead to *significantly* larger complexities, consisting of very high order polynomials of  $n$ ,  $m$ .

On the other hand, all of our polynomial-scenarios algorithms are carefully designed, so that the *cardinality* of  $\mathcal{S}$  itself is small. *Indeed, one of the major contributions of this work is to show that this property can still be satisfied for sophisticated approximation algorithms using complex LP rounding.* Consequently, we can use simple generalization bounds. Besides being clear and intuitive, these lead to a much lower dependence on  $n, m$  for the sample complexity (see Theorem 2). To our knowledge, these are the first examples of non-trivial approximation algorithms for two-stage stochastic problems via directly bounding the size of the solution set  $\mathcal{S}$ .

**Remark 2.** If we assume that the maximum stage-II cost of any facility is bounded by some polynomial value  $\Delta$ , then we could use standard SAA results directly for our problems. Alternatively, we can use a variant of our generalization scheme (without scenario-discarding) getting refined sample bounds. A simple modification of our Section 2 analysis yields Theorem 3. However, this additional assumption on the cost function is much stronger than what is typically used in the two-stage stochastic literature, and so our scheme aims at tackling the most general case.

► **Theorem 3.** *Suppose we have an efficiently generalizable,  $\eta$ -approximation algorithm for the polynomial-scenarios variant of any of the problems we study. Let  $\mathcal{S}$  be the set of all possible black-box solutions its extension process can produce. Then, for any  $\gamma, \epsilon \in (0, 1)$  and*

■ **Algorithm 1** GreedyCluster( $\mathcal{Q}, R, g$ ).

---

```

 $H \leftarrow \emptyset;$ 
for each  $j \in \mathcal{Q}$  in non-increasing order of  $g(j)$  do
     $H \leftarrow H \cup \{j\};$ 
    for each  $j' \in \mathcal{Q}$  with  $G_{j,R} \cap G_{j',R} \neq \emptyset$  do
         $\pi(j') \leftarrow j, \mathcal{Q} \leftarrow \mathcal{Q} \setminus \{j'\};$ 
    end
end
Return  $(H, \pi)$ ;
    
```

---

with  $\mathcal{O}\left(\frac{m\Delta}{\epsilon} \log\left(\frac{nm|S|}{\gamma}\right) \log\left(\frac{nm}{\gamma}\right)\right)$  samples, we get a radius  $R$  and a black-box solution  $F_I, F_A$  for all  $A \in \mathcal{D}$ :

1.  $F_I$  satisfies the stage-I specific constraints of the problem (matroid or multiknapsack).
2. With probability at least  $1-\gamma$ , we have  $R \leq R^*$  and  $\sum_{i \in F_I} c_i^I + \mathbb{E}_{A \sim \mathcal{D}}[\sum_{i \in F_A} c_i^A] \leq (1+\epsilon)B$ , where  $R^*$  the optimal radius of the black-box variant.
3. With probability one, we have  $d(j, F_I \cup F_A) \leq \eta R$  for all  $j \in A \in \mathcal{D}$ .

In particular, with our polynomial-scenarios approximation algorithms, the sample bounds of **2S-Sup**, **2S-MatSup** and **2S-MuSup** are  $\tilde{O}\left(\frac{nm\Delta}{\epsilon}\right)$ ,  $\tilde{O}\left(\frac{(n+m)m\Delta}{\epsilon}\right)$  and  $\tilde{O}\left(\frac{m^2\Delta}{\epsilon}\right)$  respectively.

## 1.4 Notation and Important Subroutines

For  $k \in \mathbb{N}$ , we use  $[k]$  to denote  $\{1, 2, \dots, k\}$ . Also, for a vector  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_k)$  and a subset  $X \subseteq [k]$ , we use  $\alpha(X)$  to denote  $\sum_{i \in X} \alpha_i$ . For a client  $j$  and  $R \geq 0$ , we define  $G_{j,R} = \{i \in \mathcal{F} : d(i, j) \leq R\}$ ,  $i_{j,R}^I = \arg \min_{i \in G_{j,R}} c_i^I$  and  $i_{j,R}^A = \arg \min_{i \in G_{j,R}} c_i^A$  for any  $A$ .

We repeatedly use a key subroutine named GreedyCluster(), shown in Algorithm 1. Its input is a set of clients  $\mathcal{Q}$ , a target radius  $R$ , and an ordering function  $g : \mathcal{Q} \mapsto \mathbb{R}$ . Its output is a set  $H \subseteq \mathcal{Q}$  along with a mapping  $\pi : \mathcal{Q} \mapsto H$ . The goal of this subroutine is to sparsify the given input  $\mathcal{Q}$ , by greedily choosing a set of representative clients  $H$ .

► **Observation 4.** For  $(H, \pi) = \text{GreedyCluster}(\mathcal{Q}, R, g)$ , the following two properties hold: (i) for all  $j, j' \in H$  with  $j \neq j'$ , we have  $G_{j,R} \cap G_{j',R} = \emptyset$ ; and (ii) for all  $j \in \mathcal{Q}$  with  $j' = \pi(j)$ , we have  $G_{j,R} \cap G_{j',R} \neq \emptyset$ ,  $d(j, j') \leq 2R$ , and  $g(j') \geq g(j)$ .

## 2 Generalizing to the Black-Box Setting

Let  $\mathcal{P}$  be any of the two-stage problems we consider, with polynomial-scenarios variant **P-Poly** and black-box variant **P-BB**. Moreover, suppose that we have an  $\eta$ -approximation algorithm  $\text{AlgP}$  for **P-Poly**, which we intend to use to solve **P-BB**. Before we proceed to our generalization scheme, we present some important definitions and assumptions.

As a starting point, assume we are given a radius demand  $R$ ; we later discuss how to optimize over this. Hence, we denote a **P-BB** problem instance by the tuple  $\mathcal{J} = (\mathcal{C}, \mathcal{F}, \mathcal{M}_I, c^I, B, R)$ , where  $\mathcal{C}$  is the set of clients,  $\mathcal{F}$  the set of facilities  $i$ , each with stage-I cost  $c_i^I$ ,  $\mathcal{M}_I \subseteq 2^{\mathcal{F}}$  the set of legal stage-I openings (representing the stage-I specific constraints of  $\mathcal{P}$ ),  $B$  the budget, and  $R$  the given covering demand. In addition, there is an underlying distribution  $\mathcal{D}$ , where each scenario  $A \in \mathcal{D}$  appears with some unknown probability  $p_A$ . Our only means of access to  $\mathcal{D}$  is via a sampling oracle. Finally, when a scenario  $A \in \mathcal{D}$  is revealed, we also learn the corresponding facility costs  $c_i^A$ .

► **Definition 5.** We define a strategy  $s$  to be a tuple  $(F_I^s, F_A^s \mid A \in \mathcal{D})$  of facility sets, where  $A$  ranges over  $\mathcal{D}$ . The set  $F_I^s$  represents the facilities  $s$  opens in stage-I, and  $F_A^s$  denotes the facilities  $s$  opens in stage-II, when the arriving scenario is  $A$ . In other words, a strategy is a just potential solution to  $\mathcal{P}$ -**BB**.

► **Assumption 6.** For any strategy  $s$  and  $A \in \mathcal{D}$ , the value  $c^A(F_A^s)$  has a continuous CDF. We can assume this w.l.o.g.; we simply add a dummy facility  $i_d$  in the input, and for all  $s$  and  $A \in \mathcal{D}$ , we include  $i_d$  in the original  $F_A^s$ . Then,  $c_{i_d}^A$  is set to be some infinitesimal smooth noise. Also,  $B$  and  $\mathcal{M}_I$  can trivially be extended to account for  $i_d$ . Finally, the assumption implies that for a finite set of scenarios  $Q$ , the values  $c^A(F_A^s)$  for all  $A \in Q$  are distinct with probability 1.

We say that a given instance  $\mathfrak{J}$  is *feasible* for  $\mathcal{P}$ -**BB**, if there exists a strategy  $s^*$  satisfying:

$$F_I^{s^*} \in \mathcal{M}_I, \quad c^I(F_I^{s^*}) + \sum_{A \in \mathcal{D}} p_A c^A(F_A^{s^*}) \leq B, \quad \forall j \in A \in \mathcal{D} \quad d(j, F_I^{s^*} \cup F_A^{s^*}) \leq R$$

For  $\mathcal{P}$ -**Poly**, consider an instance  $\mathfrak{J} = (\mathcal{C}, \mathcal{F}, \mathcal{M}_I, Q, \vec{q}, \vec{c}, B, R)$ , where  $\mathcal{C}, \mathcal{F}, \mathcal{M}_I, B, R$  are as in the  $\mathcal{P}$ -**BB** setting,  $Q$  is the set of provided scenarios,  $\vec{c}$  the vector of stage-I and stage-II explicitly given costs, and  $\vec{q}$  the vector of occurrence probabilities  $q_A$  of each  $A \in Q$ . We say that the instance  $\mathfrak{J}$  is *feasible* for  $\mathcal{P}$ -**Poly**, if there exist sets  $F_I \subseteq \mathcal{F}$  and  $F_A \subseteq \mathcal{F}$  for every  $A \in Q$ , such that:

$$F_I \in \mathcal{M}_I, \quad c^I(F_I) + \sum_{A \in Q} q_A c^A(F_A) \leq B, \quad \forall j \in A \in Q \quad d(j, F_I \cup F_A) \leq R$$

We also write  $F$  for the overall collection of sets  $F_I$  and  $F_A : A \in Q$ .

► **Definition 7.** An algorithm  $\text{Alg}\mathcal{P}$  is a valid  $\eta$ -approximation algorithm for  $\mathcal{P}$ -**Poly**, if given any problem instance  $\mathfrak{J} = (\mathcal{C}, \mathcal{F}, \mathcal{M}_I, Q, \vec{q}, \vec{c}, B, R)$ , one of the following two cases holds:

- (A) If  $\mathfrak{J}$  is feasible for  $\mathcal{P}$ -**Poly**, then  $\text{Alg}\mathcal{P}$  returns a collection of sets  $F$  with  $F_I \in \mathcal{M}_I$ ,  $c^I(F_I) + \sum_{A \in Q} q_A c^A(F_A) \leq B$  and  $\forall j \in A \in Q \quad d(j, F_I \cup F_A) \leq \eta R$ .
- (B) If  $\mathfrak{J}$  is not feasible for  $\mathcal{P}$ -**Poly**, then the algorithm either returns “INFEASIBLE”, or returns a collection of sets  $F$  satisfying the properties presented in A.

► **Definition 8.** A valid  $\eta$ -approximation algorithm  $\text{Alg}\mathcal{P}$  for  $\mathcal{P}$ -**Poly** is efficiently generalizable, if for every instance  $\mathfrak{J} = (\mathcal{C}, \mathcal{F}, \mathcal{M}_I, Q, \vec{q}, \vec{c}, B, R)$  for which it returns a solution  $F$ , there is an efficient procedure that implicitly extends this to a strategy  $\bar{s}$ , and satisfies:

- (I) Given any  $A \in \mathcal{D}$ , it returns a set  $F_A^{\bar{s}} \subseteq \mathcal{F}$ , with  $d(j, F_I^{\bar{s}} \cup F_A^{\bar{s}}) \leq \eta R$  for all  $j \in A$ .
- (II)  $F_I^{\bar{s}} = F_I$  and  $F_A^{\bar{s}} = F_A$  for every  $A \in Q$ .
- (III) Given  $\mathfrak{J}$ , let  $\mathcal{S}$  be the set of all possible strategies that are potentially achievable using the extension procedure for any set  $Q$ . Then  $|\mathcal{S}| \leq t_{\mathcal{P}}(n, m)$  for some function  $t_{\mathcal{P}}(n, m)$ , with  $\log(t_{\mathcal{P}}(n, m)) = \text{poly}(n, m)$ .

Note that property III is not trivial, since by default  $|\mathcal{S}| \leq 2^{m|\mathcal{D}|}$ , and  $|\mathcal{D}|$  can be exponentially large or even uncountably infinite.

The first step of our generalization is based on sampling a set  $Q$  of scenarios from  $\mathcal{D}$ , and then applying the efficiently-generalizable  $\text{Alg}\mathcal{P}$  on  $Q$ . When running the latter, we also increase the available budget to  $(1 + \epsilon)B$ , for some  $\epsilon > 0$ . The purpose of this step is to verify whether or not the given instance of  $\mathcal{P}$ -**BB** is feasible, and to achieve this we may have to repeat it a polynomial number of times. See Algorithm 2 for the full details.

**Algorithm 2** Determining Feasibility for  $\mathcal{P}$ -BB.

---

**Input :** Parameters  $\epsilon, \gamma, \alpha \in (0, 1)$ ,  $N \geq 1$  and a  $\mathcal{P}$ -BB instance  
 $\mathfrak{J} = (\mathcal{C}, \mathcal{F}, \mathcal{M}_I, c^I, B, R)$ .

**If**  $\exists j \in \mathcal{C} : d(j, \mathcal{F}) > R$  **then** return “INFEASIBLE”; // For points not sampled

**for**  $h = 1, \dots, \left\lceil \log_{\frac{13}{12}}(1/\gamma) \right\rceil$  **do**

Draw  $N$  independent samples from the oracle, obtaining set  $Q = \{S_1, \dots, S_N\}$ ;  
 Let  $\vec{c}$  the vector containing  $c^I$  and the stage-II facility-cost vectors of all  $S_v \in Q$ ;  
 For every  $S_v \in Q$  set  $q_{S_v} \leftarrow 1/N$ ;  
**if**  $\text{AlgP}(\mathcal{C}, \mathcal{F}, \mathcal{M}_I, Q, \vec{q}, \vec{c}, (1 + \epsilon)B, R)$  returns  $F$  **then**

Let  $T$  be the  $\lceil \alpha N \rceil^{\text{th}}$  largest value of  $c^{S_v}(F_{S_v})$  among all scenarios in  $Q$ ;  
 Return  $(F, T)$ ;

**end**

**end**

Return “INFEASIBLE”;

---

If Algorithm 2 returns “INFEASIBLE”, then our approach would deem that  $\mathfrak{J}$  is not feasible for  $\mathcal{P}$ -BB. Otherwise, let  $F$  be the solution returned by  $\text{AlgP}$  at the last “successful” iteration of the while loop. Because  $\text{AlgP}$  is efficiently-generalizable, we can apply its extension procedure to any arriving scenario, and therefore implicitly construct a strategy  $\bar{s}$ . By the properties of  $\text{AlgP}$  and II, I, we have  $F_I^{\bar{s}} \in \mathcal{M}_I$  and  $d(j, F_I^{\bar{s}} \cup F_A^{\bar{s}}) \leq \eta R$  for every  $A \in \mathcal{D}$  and  $j \in A$ .

However, we are not yet done. The second step of our generalization framework consists of slightly modifying the strategy  $\bar{s}$ . For that reason, we use the value  $T$  returned by Algorithm 2, which corresponds to the  $\lceil \alpha N \rceil^{\text{th}}$  largest value  $c^{S_v}(F_{S_v}^{\bar{s}})$  among all  $S_v \in Q$ , with  $Q$  the sampled set in the last iteration of the while loop ( $F_{S_v}^{\bar{s}} = F_{S_v}$  by II). Note here that Assumption 6 ensures that the choice of  $T$  is well-defined.

If now an arriving scenario  $A$  has  $c^A(F_A^{\bar{s}}) > T$ , we will perform no stage-II opening. This modification eventually constructs a new strategy  $\hat{s}$ , with  $F_I^{\hat{s}} = F_I^{\bar{s}}$ ,  $F_A^{\hat{s}} = \emptyset$  when  $c^A(F_A^{\bar{s}}) > T$ , and  $F_A^{\hat{s}} = F_A^{\bar{s}}$  if  $c^A(F_A^{\bar{s}}) \leq T$ . The latter strategy will determine our final opening actions, and hence we need to analyze its *opening cost*  $C(\hat{s})$  over  $\mathcal{D}$ , and the probability with which it does not return an  $\eta$ -approximate solution. Regarding the latter, note that when  $F_A^{\hat{s}} \neq F_A^{\bar{s}}$ , we can no longer guarantee an approximation ratio of  $\eta$  as implied by property I for  $\bar{s}$ .

► **Lemma 9.** *If instance  $\mathfrak{J}$  is feasible for  $\mathcal{P}$ -BB and  $N \geq 1/\epsilon$ , then with probability at least  $1 - \gamma$  Algorithm 2 does not terminate with “INFEASIBLE”.*

**Proof.** By rescaling, we assume w.l.o.g. that  $B = 1$ . Also, the cost of any strategy  $s$  over  $\mathcal{D}$  is given by  $C(s) = c^I(F_I^s) + \sum_{A \in \mathcal{D}} p_A c^A(F_A^s)$ . For any specific execution of the while loop in Algorithm 2, let  $Y_v^s$  be the second-stage cost of  $s$  on sample  $S_v$ . Finally, for a fixed  $s$  the random variables  $Y_v^s$  are independent, and the empirical cost of  $s$  on  $Q$  is  $\hat{C}(s) = c^I(F_I^s) + \frac{1}{N} \sum_{v=1}^N Y_v^s$ .

If  $\mathfrak{J}$  is feasible, then there exists some strategy  $s^*$  satisfying  $F_I^{s^*} \in \mathcal{M}_I$  and  $d(j, F_I^{s^*} \cup F_A^{s^*}) \leq R$  for every  $A \in \mathcal{Q}$  and  $j \in A$ . We will also show that  $\hat{C}(s^*) \leq (1 + \epsilon)B$  with probability at least  $1/13$ . In this case, the restriction of  $s^*$  to  $Q$  verifies that  $(\mathcal{C}, \mathcal{F}, \mathcal{M}_I, Q, \vec{q}, \vec{c}, (1 + \epsilon)B, R)$  is feasible for  $\mathcal{P}$ -Poly. Thus, since  $\text{AlgP}$  is a valid  $\eta$ -approximation for  $\mathcal{P}$ -Poly, it will not return “INFEASIBLE”.

As  $s^*$  is feasible for  $\mathfrak{J}$  we have  $C(s^*) \leq B$ , implying  $\mathbb{E}[Y_v^{s^*}] = \sum_{A \in \mathcal{D}} p_A \cdot c^A(F_A^{s^*}) \leq B = 1$  for all samples  $v$ . By Lemma 20 with  $\delta = \epsilon BN$ , this yields

$$\Pr \left[ \sum_{v=1}^N Y_v^{s^*} < \mathbb{E} \left[ \sum_{v=1}^N Y_v^{s^*} \right] + \epsilon BN \right] \geq \min \left\{ \frac{\epsilon BN}{1 + \epsilon BN}, \frac{1}{13} \right\}$$

When  $N \geq \frac{B}{\epsilon} = \frac{1}{\epsilon}$ , we see that  $\epsilon BN / (1 + \epsilon BN) \geq 1/13$ . Hence, with probability at least  $1/13$  we have  $\sum_{v=1}^N Y_v^{s^*} < \mathbb{E}[\sum_{v=1}^N Y_v^{s^*}] + \epsilon BN$ , in which case we get  $\hat{C}(s^*) \leq (1 + \epsilon)B$  as shown below:

$$\begin{aligned} \hat{C}(s^*) &= c^I(F_I^{s^*}) + \frac{1}{N} \sum_{v=1}^N Y_v^{s^*} \leq c^I(F_I^{s^*}) + \frac{1}{N} \sum_{v=1}^N \mathbb{E}[Y_v^{s^*}] + \epsilon B \\ &\leq c^I(F_I^{s^*}) + \sum_{A \in \mathcal{D}} p_A \cdot c^A(F_A^{s^*}) + \epsilon B \leq (1 + \epsilon)B \end{aligned}$$

So each iteration terminates successfully with probability at least  $1/13$ . To bring the error probability down to at most  $\gamma$ , we repeat the process for  $\left\lceil \log_{\frac{13}{12}}(1/\gamma) \right\rceil$  iterations.  $\blacktriangleleft$

Let  $\mathcal{T}$  be the event that Algorithm 2 terminates **without returning** “INFEASIBLE”, and  $\mathcal{T}_h$  the event that Alg $\mathcal{P}$  found a solution  $F$  at the  $h^{\text{th}}$  iteration of the while loop. We denote by *Invalid* the event that Algorithm 2 returns an invalid output; specifically, if  $\mathcal{T}$  occurs, *Invalid* is the event of having  $C(\hat{s}) > (1 + 2\epsilon)B$ , otherwise it is the event of mistakenly deciding that  $\mathfrak{J}$  is not feasible. Let now  $Q_h$  be the set of scenarios sampled at the  $h^{\text{th}}$  iteration of Algorithm 2, and for any strategy  $s$  let  $T_s^h$  be the  $\lceil \alpha N \rceil^{\text{th}}$  largest value  $c^{S_v}(F_{S_v}^s)$  among all  $S_v \in Q_h$ . We then denote by  $\mathcal{E}_h$  the event that for all  $s \in \mathcal{S}$ , we have  $\Pr_{A \sim \mathcal{D}}[c^A(F_A^s) > T_s^h] \geq \frac{\alpha}{4}$ . Finally, note that due to III the set  $\mathcal{S}$  is deterministically given in the event  $\mathcal{E}_h$ .

► **Lemma 10.** For any  $\gamma, \alpha \in (0, 1)$  and  $N = \mathcal{O}\left(\frac{1}{\alpha} \log\left(\frac{t_{\mathcal{P}}(n, m)}{\gamma}\right)\right)$ , we have  $\Pr[\bar{\mathcal{E}}_h] \leq \gamma / (\log_{\frac{13}{12}}(\frac{1}{\gamma}) + 1)$ .

**Proof.** Focus on a specific iteration  $h$ . Consider a strategy  $s \in \mathcal{S}$ , and for each  $S_v \in Q_h$  let  $X_v$  be an indicator random variable that is 1 iff  $c^{S_v}(F_{S_v}^s) > T_s^h$ . Also let  $X = \sum_{v=1}^N X_v$ , and note that by Assumption 6 we have  $X = \lceil \alpha N \rceil - 1$ . This implies that the empirical probability of scenarios with stage-II cost more than  $T_s^h$  is  $q_s^h = (\lceil \alpha N \rceil - 1)/N$ . Finally, let  $p_s^h = \Pr_{A \sim \mathcal{D}}[c^A(F_A^s) > T_s^h]$ .

If  $p_s^h \geq \alpha$  then we immediately get  $\Pr[p_s^h < \alpha/4] = 0$ . Therefore, assume that  $p_s^h < \alpha$ . If  $N \geq 4/\alpha$  then we have:

$$q_s^h - \frac{\alpha}{2} = \frac{\lceil \alpha N \rceil - 1}{N} - \frac{\alpha}{2} \geq \frac{\alpha N - 1}{N} - \frac{\alpha}{2} = \frac{\alpha}{2} - \frac{1}{N} \geq \frac{\alpha}{2} - \frac{\alpha}{4} = \frac{\alpha}{4}$$

Hence, if  $p_s^h \geq q_s^h - \frac{\alpha}{2}$  and  $N \geq 4/\alpha$ , we get  $p_s^h \geq \frac{\alpha}{4}$ . Using Lemma 22 with  $p_s^h < \alpha$ ,  $\delta = \alpha/2$  and  $N = \frac{8}{\alpha} \log\left(\frac{t_{\mathcal{P}}(n, m)}{\gamma} (\log_{\frac{13}{12}}(\frac{1}{\gamma}) + 1)\right) \geq 4/\alpha$  yields the following:

$$\Pr[p_s^h < \frac{\alpha}{4}] \leq \Pr[p_s^h < q_s^h - \alpha/2] \leq e^{-\left(\frac{t_{\mathcal{P}}(n, m)}{\gamma} (\log_{\frac{13}{12}}(\frac{1}{\gamma}) + 1)\right)} = \frac{\gamma}{t_{\mathcal{P}}(n, m) (\log_{\frac{13}{12}}(1/\gamma) + 1)}$$

A union bound over all  $s \in \mathcal{S}$  and property III will finally give  $\Pr[\bar{\mathcal{E}}_h] \leq \gamma / (\log_{\frac{13}{12}}(\frac{1}{\gamma}) + 1)$ .  $\blacktriangleleft$

► **Theorem 11.** For any  $\epsilon, \gamma, \alpha \in (0, 1)$  and  $N = \mathcal{O}\left(\frac{1}{\epsilon \alpha} \log\left(\frac{t_{\mathcal{P}}(n, m)}{\gamma}\right)\right)$ ,  $\Pr[\text{Invalid}] \leq 3\gamma$ .

## 23:10 Approximating Two-Stage Stochastic Supplier Problems

**Proof.** Using the definition of the Invalid event and Lemmas 9, 10 we get the following.

$$\begin{aligned}
\Pr[\text{Invalid}] &= \Pr[\text{Invalid} \mid \bar{\mathcal{T}}] \Pr[\bar{\mathcal{T}}] + \Pr[\text{Invalid} \mid \mathcal{T}] \Pr[\mathcal{T}] \leq \gamma + \sum_h \Pr[\text{Invalid} \wedge \mathcal{T}_h] \\
&= \gamma + \sum_h \left( \Pr[\text{Invalid} \wedge \mathcal{T}_h \mid \mathcal{E}_h] \Pr[\mathcal{E}_h] + \Pr[\text{Invalid} \wedge \mathcal{T}_h \mid \bar{\mathcal{E}}_h] \Pr[\bar{\mathcal{E}}_h] \right) \\
&\leq 2\gamma + \sum_h \Pr[\text{Invalid} \wedge \mathcal{T}_h \wedge \mathcal{E}_h] \tag{1}
\end{aligned}$$

For each  $s \in \mathcal{S}$ , let  $t_s$  be value such that  $\Pr_{A \sim \mathcal{D}}[c^A(F_A^s) > t_s] = \frac{\alpha}{4}$ . Note that the existence of  $t_s$  is guaranteed by Assumption 6. Further, for each  $s \in \mathcal{S}$ ,  $A \in \mathcal{D}$ , define  $\tilde{c}^A(F_A^s)$  to be  $c^A(F_A^s)$  if  $c^A(F_A^s) \leq t_s$ , and 0 otherwise. In addition, for an iteration  $h$  let  $Y_{v,h}^s$  be a random variable denoting the second-stage  $\tilde{c}$  cost of  $s$  for the  $v$ -th sample of  $h$ , and  $Z_{v,h}^s$  be an indicator random variable that is 1 iff the original second-stage cost of  $s$  on the  $v$ -th sample of  $h$  is greater than  $t_s$ . We use the following cost functions:

$$\hat{C}_h(s) = c^I(F_I^s) + \frac{1}{N} \sum_{v=1}^N Y_{v,h}^s + \frac{t_s}{N} \sum_{v=1}^N Z_{v,h}^s \quad \text{and} \quad \tilde{C}(s) = c^I(F_I^s) + \sum_{A \in \mathcal{D}} p_A \cdot \tilde{c}^A(F_A^s)$$

Also, if  $p_s = \Pr_{A \sim \mathcal{D}}[c^A(F_A^s) > t_s]$ , then  $\mathbb{E}[\hat{C}_h(s)] = \tilde{C}(s) + p_s t_s$ . Finally, let  $\hat{C}_h^{II}(s) = \hat{C}_h(s) - c^I(F_I^s)$  and  $\tilde{C}^{II}(s) = \tilde{C}(s) - c^I(F_I^s)$ .

Now observe that if  $\text{Invalid} \wedge \mathcal{T}_h \wedge \mathcal{E}_h$  occurs, then there must exist some  $s \in \mathcal{S}$  with  $\hat{C}_h(s) \leq (1 + \epsilon)B$  and  $\tilde{C}(s) > (1 + 2\epsilon)B$ . Specifically we have  $\hat{C}_h(\bar{s}) \leq (1 + \epsilon)B$  and  $\tilde{C}(\bar{s}) > (1 + 2\epsilon)B$ . To see why  $\hat{C}_h(\bar{s}) \leq (1 + \epsilon)B$  is true, note that under this event AlgP finds a solution in iteration  $h$ . The empirical cost of this solution (which corresponds to a restriction of  $\bar{s}$ ) is at most  $(1 + \epsilon)B$ , and the pruning based on the value  $t_s$  can only decrease this cost. Regarding  $\tilde{C}(\bar{s}) > (1 + 2\epsilon)B$ , under  $\text{Invalid} \wedge \mathcal{T}_h \wedge \mathcal{E}_h$  we at first have  $C(\hat{s}) > (1 + 2\epsilon)B$ . In addition,  $\tilde{C}(\bar{s}) \leq C(\hat{s})$ , because by the definitions of  $t_s$  and  $\mathcal{E}_h$  we have  $t_s \geq T_s^h$ . Hence, we upper bound the probability of  $\text{Invalid} \wedge \mathcal{T}_h \wedge \mathcal{E}_h$  as follows:

$$\begin{aligned}
\Pr[\text{Invalid} \wedge \mathcal{T}_h \wedge \mathcal{E}_h] &\leq \Pr[\exists s \in \mathcal{S} : \hat{C}_h(s) \leq (1 + \epsilon)B \wedge \tilde{C}(s) > (1 + 2\epsilon)B] \\
&\leq \Pr[\exists s \in \mathcal{S} : \hat{C}_h(s) \leq (1 + \epsilon)B \wedge \tilde{C}(s) + p_s t_s > (1 + 2\epsilon)B + p_s t_s] \\
&\leq \Pr[\exists s \in \mathcal{S} : \hat{C}_h(s) \leq (1 + \epsilon)B \wedge \mathbb{E}[\hat{C}_h(s)] > (1 + 2\epsilon)B + p_s t_s] \\
&\leq \Pr[\exists s \in \mathcal{S} : \hat{C}_h^{II}(s) \leq (1 - \delta_s) \mathbb{E}[\hat{C}_h^{II}(s)]] \\
&\leq \sum_{s \in \mathcal{S}} \Pr \left[ \hat{C}_h^{II}(s) \leq (1 - \delta_s) \mathbb{E}[\hat{C}_h^{II}(s)] \right] \\
&= \sum_{s \in \mathcal{S}} \Pr \left[ N \cdot \hat{C}_h^{II}(s) / t_s \leq (1 - \delta_s) N \cdot \mathbb{E}[\hat{C}_h^{II}(s)] / t_s \right] \tag{2}
\end{aligned}$$

In the above we defined  $\delta_s$  such that  $\delta_s \geq \frac{\epsilon + p_s t_s}{1 + 2\epsilon + p_s t_s}$ , and also we made use of  $B = 1$  and  $\mathbb{E}[\hat{C}_h(s)] = \tilde{C}(s) + p_s t_s > 1 + 2\epsilon + p_s t_s$ . Applying Lemma 21 gives

$$\Pr \left[ N \cdot \hat{C}_h^{II}(s) / t_s \leq (1 - \delta_s) N \cdot \mathbb{E}[\hat{C}_h^{II}(s)] / t_s \right] \leq e^{-\frac{N(\epsilon + p_s t_s)^2}{2t_s(1 + 2\epsilon + p_s t_s)}} \tag{3}$$

We now focus on the quantity  $\frac{(\epsilon + p_s t_s)^2}{2t_s(1 + 2\epsilon + p_s t_s)}$ , and consider two distinct cases for  $p_s t_s$ .

- Suppose  $p_s t_s \geq \epsilon$ . Then  $\frac{(\epsilon + p_s t_s)^2}{2t_s(1 + 2\epsilon + p_s t_s)} \geq \frac{p_s^2 t_s^2}{2t_s(1 + 3p_s t_s)} \geq \frac{p_s}{2} \frac{p_s t_s}{1 + 3p_s t_s} \geq \frac{\epsilon \cdot p_s}{2(1 + 3\epsilon)}$ , where the last inequality follows because  $x/(1 + 3x)$  is increasing and in our case  $x \geq \epsilon$ .

- Suppose  $p_s t_s < \epsilon$ . Then  $\frac{(\epsilon + p_s t_s)^2}{2t_s(1+2\epsilon+p_s t_s)} \geq \frac{\epsilon^2}{2t_s(1+3\epsilon)} \geq \frac{\epsilon \cdot p_s}{2(1+3\epsilon)}$ , where in the last inequality we used the fact that in this case  $t_s < \epsilon/p_s$ .

Therefore, by definition of  $p_s$ , we have  $\frac{(\epsilon + p_s t_s)^2}{2t_s(1+2\epsilon+p_s t_s)} \geq \frac{\epsilon \cdot \alpha}{8(1+3\epsilon)}$  in every case. Plugging that in (3), (2), and setting  $N = \frac{8(1+\epsilon)}{\epsilon \alpha} \log\left(\frac{t_{\mathcal{P}}(n,m)}{\gamma} (\log_{\frac{13}{12}}(\frac{1}{\gamma}) + 1)\right)$  gives  $\Pr[\text{Invalid} \wedge \mathcal{T}_h \wedge \mathcal{E}_h] \leq \gamma / (\log_{\frac{13}{12}}(\frac{1}{\gamma}) + 1)$ . Finally, using this in (1) gives the desired error probability of at most  $3\gamma$ . ◀

► **Theorem 12.** For any  $\gamma, \alpha \in (0, 1)$  and  $N = \mathcal{O}\left(\frac{1}{\alpha} \log\left(\frac{t_{\mathcal{P}}(n,m)}{\gamma}\right)\right)$ , the solution strategy  $\hat{s}$  satisfies  $\Pr_{A \sim \mathcal{D}}[d(j, F_I^{\hat{s}} \cup F_A^{\hat{s}}) \leq \eta R, \forall j \in A] \geq 1 - 2\alpha$  with probability at least  $1 - \gamma$ .

**Proof.** Consider some iteration  $h$  and strategy  $s \in \mathcal{S}$ . Let  $p_s^h = \Pr_{A \sim \mathcal{D}}[c^A(F_A^s) > T_s^h]$ , and  $\mathcal{B}_s^h$  the event of having  $p_{T_s^h} > 2\alpha$ . Suppose that  $p_s^h > \alpha$ , otherwise  $\mathcal{B}_s^h$  cannot occur. Let  $X_v$  an indicator random variable that is 1 iff  $s$  has stage-II cost larger than  $T_s^h$  in the  $v$ -th sample. Also, let  $X = \sum_{v=1}^N X_v$ , and recall that  $X = \lceil \alpha N \rceil - 1 \leq \alpha N$ . Moreover, we have  $\mathbb{E}[X] = p_s^h N$  and notice that  $2X > \mathbb{E}[X]$  implies  $p_s^h < 2\alpha$ . Using Lemma 21 with  $\delta = 1/2$  we get  $\Pr[X \leq \mathbb{E}[X]/2] \leq e^{-p_s^h N/8}$ . Because  $p_s^h > \alpha$ , setting  $N = \frac{8}{\alpha} \log\left(\frac{t_{\mathcal{P}}(n,m)}{\gamma} (\log_{\frac{13}{12}}(\frac{1}{\gamma}) + 1)\right)$  gives  $\sum_h \sum_{s \in \mathcal{S}} \Pr[\mathcal{B}_s^h] \leq \gamma$ . ◀

Finally, by optimizing over the radius, we get our main generalization result:

► **Theorem 13.** Assume we have an efficiently generalizable  $\eta$ -approximation for  $\mathcal{P}$ -Poly. Then, using  $\mathcal{O}\left(\frac{1}{\epsilon \alpha} \log\left(\frac{nm \cdot t_{\mathcal{P}}(n,m)}{\gamma}\right) \log \frac{nm}{\gamma}\right)$  samples, we obtain a strategy  $\hat{s}$  and a radius  $R$ , such that with probability at least  $1 - \mathcal{O}(\gamma)$  the following hold: (i)  $C(\hat{s}) \leq (1 + 2\epsilon)B$ , (ii)  $F_I^{\hat{s}} \in \mathcal{M}_I$ ; (iii)  $R \leq R^*$ , where  $R^*$  is the optimal radius for  $\mathcal{P}$ -BB; (iv)  $\Pr_{A \sim \mathcal{D}}[d(j, F_I^{\hat{s}} \cup F_A^{\hat{s}}) \leq \eta R, \forall j \in A] \geq 1 - 2\alpha$ .

**Proof.** Because  $R^*$  is the distance between some facility and some client, there are at most  $nm$  alternatives for it. Thus, we can run Algorithm 2 for all possible  $nm$  target radius values, using error parameter  $\gamma' = \frac{\gamma}{nm}$ . We then return the smallest radius that did not yield “INFEASIBLE”. By a union bound over all radius choices, the probability of the *Invalid* event in any of them is at most  $3\gamma$ . Thus, with probability at least  $1 - 3\gamma$ , the chosen radius  $R$  satisfies  $R \leq R^*$ , and the opening cost of the corresponding strategy is at most  $(1 + 2\epsilon)B$ . Finally, for the returned strategy Theorem 12 holds as well, and the sample bound accounts for all iteration of Algorithm 2.

Additionally, note that we do not need fresh samples for each radius guess  $R$ ; we can draw an appropriate number of samples  $N$  upfront, and test all guesses in “parallel” with the same data. ◀

In light of Theorem 13 and the generic search step for the radius  $R$ , we assume for all our  $\mathcal{P}$ -poly problems that a target radius  $R$  is given explicitly.

We conclude with some final remarks. At first, III guarantees  $N = \text{poly}(n, m, \frac{1}{\epsilon}, \frac{1}{\alpha}, \log \frac{1}{\gamma})$ . Also, the probability  $2\alpha$  of not returning an  $\eta$ -approximate solution can be made inverse polynomially small, without affecting the polynomial nature of the sample complexity.

### 3 Approximation Algorithm for 2S-Sup-BB

In this section we tackle **2S-Sup-BB**, by first designing a 3-approximation algorithm for **2S-Sup-Poly**, and then proving that the latter is efficiently generalizable.

**Algorithm 3** Correlated LP-Rounding Algorithm for **2S-Sup-Poly**.

---

Solve LP (4)-(6) to get a feasible solution  $y^I, y^A : A \in Q$ ;  
**if** no feasible LP solution exists **then**  
     | Return “INFEASIBLE”;  
**end**  
 $(H_I, \pi^I) \leftarrow \text{GreedyCluster}(\mathcal{C}, R, g^I)$ , where  $g^I(j) = y^I(G_j)$  ;  
**for** each scenario  $A \in Q$  **do**  
     |  $(H_A, \pi^A) \leftarrow \text{GreedyCluster}(A, R, g^A)$ , where  $g^A(j) = -y^I(G_{\pi^I(j)})$  ;  
**end**  
 Order the clients of  $H_I$  as  $j_1, j_2, \dots, j_h$  such that  $y^I(G_{j_1}) \leq y^I(G_{j_2}) \leq \dots \leq y^I(G_{j_h})$ ;  
 Consider an additional “dummy” client  $j_{h+1}$  with  $y^I(G_{j_{h+1}}) > y^I(G_{j_\ell})$  for all  $\ell \in [h]$ ;  
**for** all integers  $\ell = 1, 2, \dots, h+1$  **do**  
     |  $F_I^\ell \leftarrow \{i_{j_k}^I \mid j_k \in H_I \text{ and } y^I(G_{j_k}) \geq y^I(G_{j_\ell})\}$ ;  
     **for** each  $A \in Q$  **do**  
         |  $F_A^\ell \leftarrow \{i_j^A \mid j \in H_A \text{ and } F_I^\ell \cap G_{\pi^I(j)} = \emptyset\}$ ;  
     **end**  
     |  $S_\ell \leftarrow c^I(F_I^\ell) + \sum_{A \in Q} p_A \cdot c^A(F_A^\ell)$ ;  
**end**  
 Return  $F_I^{\ell^*}, F_A^{\ell^*} : A \in Q$  such that  $\ell^* = \arg \min_\ell S_\ell$ ;  


---

**3.1 A 3-Approximation Algorithm for 2S-Sup-Poly**

We are given a list of scenarios  $Q$  together with their probabilities  $p_A$  and cost vectors  $c^A$ , a target radius  $R$ , and let  $G_j = G_{j,R}$ ,  $i_j^I = i_{j,R}^I$ ,  $i_j^A = i_{j,R}^A$  for every  $j \in \mathcal{C}$  and  $A \in Q$ . Consider LP (4)-(6).

$$\sum_{i \in \mathcal{F}} y_i^I \cdot c_i^I + \sum_{A \in Q} p_A \sum_{i \in \mathcal{F}} y_i^A \cdot c_i^A \leq B \quad (4)$$

$$\sum_{i \in G_j} (y_i^I + y_i^A) \geq 1, \quad \forall j \in A \in Q \quad (5)$$

$$0 \leq y_i^I, y_i^A \leq 1 \quad (6)$$

Constraint (4) captures the total expected cost, and constraint (5) the fact that for all  $A \in Q$ , every  $j \in A$  must have an open facility within distance  $R$  from it. In addition, note that if the LP is infeasible, then there cannot be a solution of radius at most  $R$  for the given **2S-Sup-Poly** instance. The rounding algorithm appears in Algorithm 3.

**► Theorem 14.** For any scenario  $A \in Q$  and every  $j \in A$ , we have  $d(j, F_I^{\ell^*} \cup F_A^{\ell^*}) \leq 3R$ .

**Proof.** Focus on some  $A \in Q$ . Recall that  $d(j, \pi^I(j)) \leq 2R$  and  $d(j, \pi^A(j)) \leq 2R$  for any  $j \in A$ . For  $j \in H_A$  the statement is clearly true, because either  $G_{\pi^I(j)} \cap F_I^{\ell^*} \neq \emptyset$  or  $G_j \cap F_A^{\ell^*} \neq \emptyset$ . So consider some  $j \in A \setminus H_A$ . If  $G_{\pi^A(j)} \cap F_A^{\ell^*} \neq \emptyset$ , then any facility  $i \in G_{\pi^A(j)} \cap F_A^{\ell^*}$  will be within distance  $3R$  from  $j$ . If on the other hand  $G_{\pi^A(j)} \cap F_A^{\ell^*} = \emptyset$ , then our algorithm guarantees  $G_{\pi^I(\pi^A(j))} \cap F_I^{\ell^*} \neq \emptyset$ . Further, the stage-II greedy clustering yields  $g^A(\pi^A(j)) \geq g^A(j) \implies y^I(G_{\pi^I(j)}) \geq y^I(G_{\pi^I(\pi^A(j))})$ . Therefore, from the way we formed  $F_I^{\ell^*}$  and the fact that  $G_{\pi^I(\pi^A(j))} \cap F_I^{\ell^*} \neq \emptyset$ , we infer that  $G_{\pi^I(j)} \cap F_I^{\ell^*} \neq \emptyset$ . The latter ensures that  $d(j, G_{\pi^I(j)} \cap F_I^{\ell^*}) \leq 3R$ . ◀

**► Theorem 15.** The opening cost  $S_{\ell^*}$  of Algorithm 3 is at most  $B$ .



■ **Algorithm 4** Generalization Procedure for **2S-Sup-Poly**.

---

**Input :** Returned sets  $F_I, F_A : A \in Q$  and inner execution details of Algorithm 3  
 Let  $\bar{s}$  the strategy we will define, and for the stage-I actions set  $F_I^{\bar{s}} \leftarrow F_I$ ;  
 Suppose scenario  $A \in \mathcal{D}$  arrived in the second stage;  
 For every  $j \in A$  set  $g(j) \leftarrow -y^I(G_{\pi^I(j)})$ , where  $y^I, \pi^I$  are the LP solution vector and stage-I mapping computed in Algorithm 3;  
 $(H_A, \pi^A) \leftarrow \text{GreedyCluster}(A, R, g)$ ;  
 $F_A^{\bar{s}} \leftarrow \{i_j^A \mid j \in H_A \text{ and } F_I \cap G_{\pi^I(j)} = \emptyset\}$ ;

---

**Proof.** Consider the following process to generate a random solution: we draw a random variable  $\beta$  uniformly from  $[0, 1]$ , and then set  $F_I^\beta = \{i_j^I \mid j \in H_I \text{ and } y^I(G_j) \geq \beta\}$ ,  $F_A^\beta = \{i_j^A \mid j \in H_A \text{ and } F_I \cap G_{\pi^I(j)} = \emptyset\}$  for all  $A \in Q$ . For each possible draw for  $\beta$ , the resulting sets  $F_I^\beta, F_A^\beta$  correspond to sets  $F_I^\ell, F_A^\ell$  for some integer  $\ell \in [h+1]$ . Hence, in order to show the existence of an  $\ell$  with  $S_\ell \leq B$ , it suffices to show  $\mathbb{E}_{\beta \sim [0,1]}[c^I(F_I^\beta) + \sum_{A \in Q} p_A \cdot c^A(F_A^\beta)] \leq B$ .

We start by calculating the probability of opening a given facility  $i_j^I$  with  $j \in H_I$  in stage-I. This will occur only if  $\beta \leq y^I(G_j)$ , and so  $\Pr[i_j^I \text{ is opened at stage-I}] \leq \min(y^I(G_j), 1)$ . Therefore, due to  $G_j \cap G_{j'} = \emptyset$  for all distinct  $j, j' \in H_I$ , we get:

$$\mathbb{E}_{\beta \sim [0,1]}[c^I(F_I^\beta)] \leq \sum_{j \in H_I} c_{i_j^I}^I \cdot y^I(G_j) \leq \sum_{i \in \mathcal{F}} y_i^I \cdot c_i^I \quad (7)$$

Moreover, for any  $j \in H_A$  and any  $A \in Q$  we have  $\Pr[i_j^A \text{ is opened at stage-II} \mid A] = 1 - \min(y^I(G_{\pi^I(j)}), 1) \leq 1 - \min(y^I(G_j), 1) \leq y^A(G_j)$ . The first inequality results from the greedy clustering of stage-I that gives  $y^I(G_{\pi^I(j)}) \geq y^I(G_j)$ , and the second follows from (5). Thus, due to  $G_j \cap G_{j'} = \emptyset$  for all distinct  $j, j' \in H_A$ , we get:

$$\mathbb{E}_{\beta \sim [0,1]}[c^A(F_A^\beta)] \leq \sum_{j \in H_A} c_{i_j^A}^A \cdot y^A(G_j) \leq \sum_{i \in \mathcal{F}} y_i^A \cdot c_i^A \quad (8)$$

Combining (7), (8) and (4) gives  $\mathbb{E}_{\beta \sim [0,1]}[c^I(F_I^\beta)] + \sum_{A \in Q} p_A \cdot \mathbb{E}_{\beta \sim [0,1]}[c^A(F_A^\beta)] \leq B$ . ◀

### 3.2 Generalizing to the Black-Box Setting

To show that Algorithm 3 fits the framework of Section 2, we must show that it is efficiently generalizable as in Definition 8. For one thing, it is obvious that Algorithm 3 satisfies the properties of Definition 7, and therefore is a valid 3-approximation. Hence, we only need a process to efficiently extend its output to any arriving scenario  $A \in \mathcal{D}$ , where  $\mathcal{D}$  the black-box distribution. This is demonstrated in Algorithm 4, which mimics the stage-II actions of Algorithm 3. Here we crucially exploit the fact that the stage-II decisions of Algorithm 3 only depend on information from the LP about stage-I variables.

Since Algorithm 4 exactly imitates the stage-II actions of Algorithm 3, it is easy to see that property II is satisfied. Further, the arguments in Theorem 14 would still apply, and eventually guarantee  $d(j, F_I^{\bar{s}} \cup F_A^{\bar{s}}) \leq 3R$  for all  $j \in A$  and any  $A \in \mathcal{D}$ , thus verifying property I. To conclude, we only need to prove III. Let  $\mathcal{S}_K$  the set of strategies achievable via Algorithm 4.

► **Lemma 16.** *Algorithm 3 satisfies property III with  $|\mathcal{S}_K| \leq (n+1)!$ .*

■ **Algorithm 5** Rounding Algorithm for **2S-MatSup-Poly**.

---

Solve LP (9)-(12) to get a feasible solution  $y^I, y^A$  for all  $A \in Q$ ;  
**if** no feasible LP solution exists **then**  
 | Return “INFEASIBLE”;  
**end**  
 $(H_I, \pi^I) \leftarrow \text{GreedyCluster}(\mathcal{C}, R, g^I)$  where  $g^I(j) = y^I(G_j)$  ;  
 Let  $g^{II} : \mathcal{C} \mapsto [n]$  be some fixed and given bijective mapping;  
**for** each scenario  $A \in Q$  **do**  
 |  $(H_A, \pi^A) \leftarrow \text{GreedyCluster}(A, R, g^{II})$  ;  
**end**  
 Solve LP (13)-(16) and get an optimal integral solution  $z^*$ , such that  
 $z_i^* \in \{0, 1\} \forall i \in \mathcal{F}$ ;  
 $F_I \leftarrow \{i \in \mathcal{F} \mid z_i^* = 1\}$ ;  
 $F_A \leftarrow \{i_j^A \in \mathcal{F} \mid j \in H_A \text{ and } G_{\pi^I(j)} \cap F_I = \emptyset\}$  for every  $A \in Q$ .

---

**Proof.** The constructed final strategy is determined by 1) the sorted order of  $y^I(G_j)$  for all  $j \in \mathcal{C}$ , and 2) a minimum threshold  $\ell'$  such that  $G_{j_{\ell'}} \cap F_I \neq \emptyset$  with  $j_{\ell'} \in H_I$ . Given those, we know exactly what  $H_I$  and  $H_A$  for every  $A \in \mathcal{D}$  will be, as well as  $F_I$  and  $F_A$  for every  $A \in \mathcal{D}$ . The set of all possible such options is also independent  $Q$ . Since there are  $n!$  total possible orderings for the  $y^I(G_j)$  values, and the threshold parameter  $\ell'$  can take at most  $n + 1$  values, we get  $|\mathcal{S}_K| \leq (n + 1)!$ . ◀

■ **4 Approximation Algorithm for 2S-MatSup-BB**

The outline of this section is similar to that of Section 3. We begin with a 5-approximation algorithm for **2S-MatSup-Poly**, and then show that it is also efficiently generalizable.

**4.1 A 5-Approximation Algorithm for 2S-MatSup-Poly**

We are given a radius  $R$ , and a list of scenarios  $Q$  together with their probabilities  $p_A$  and cost vectors  $c^A$ . Moreover, assume that  $r_{\mathcal{M}}$  is the rank function of the input matroid  $\mathcal{M} = (\mathcal{F}, \mathcal{I})$ . We also use the notation  $G_j = G_{j,R}$ , and  $i_j^A = i_{j,R}^A$  for every  $j \in \mathcal{C}$  and  $A \in Q$ . Consider LP (9)-(12).

$$\sum_{i \in \mathcal{F}} y_i^I \cdot c_i^I + \sum_{A \in Q} p_A \sum_{i \in \mathcal{F}} y_i^A \cdot c_i^A \leq B \quad (9)$$

$$\sum_{i \in G_j} (y_i^I + y_i^A) \geq 1, \quad \forall j \in A \in Q \quad (10)$$

$$\sum_{i \in U} y_i^I \leq r_{\mathcal{M}}(U), \quad \forall U \subseteq \mathcal{F} \quad (11)$$

$$0 \leq y_i^I, y_i^A \leq 1 \quad (12)$$

Compared to LP (4)-(6), the only difference lies in constraint (11), which exactly represents the stage-I matroid requirement. Hence, it is a valid relaxation for the problem. Although the LP has an exponential number of constraints, it can be solved in polynomial time via the Ellipsoid algorithm, with a separation oracle based on minimizing a submodular function [13].

Assuming LP feasibility, our algorithm (presented in full detail in Algorithm 5), begins with two greedy clustering steps, one for each stage, that produce sets  $H_I, H_A : A \in Q$  with

corresponding mappings  $\pi^I$  and  $\pi^A$ . We then set up and solve the auxiliary LP shown in (13)-(16), and use this solution to determine sets  $F_I$  and  $F_A$ .

$$\text{minimize } \sum_{i \in \mathcal{F}} z_i \cdot c_i^I + \sum_{A \in Q} p_A \sum_{j \in H_A} c_{i_j^A}^A (1 - z(G_{\pi^I(j)})) \quad (13)$$

$$\text{subject to } z(G_j) \leq 1, \forall j \in H_I \quad (14)$$

$$z(U) \leq r_{\mathcal{M}}(U), \forall U \subseteq \mathcal{F} \quad (15)$$

$$0 \leq z_i \leq 1 \quad (16)$$

► **Lemma 17.** *If LP (9)-(12) is feasible, then the optimal solution  $z^*$  of the auxiliary LP (13)-(16) has objective function value at most  $B$ , and is integral (i.e. for all  $i \in \mathcal{F}$  we have  $z_i^* \in \{0, 1\}$ ).*

**Proof.** Solution  $z^*$  is integral since the LP (13)-(16) is the intersection of two matroid polytopes, namely, the polytope corresponding to  $\mathcal{M}$ , and a partition matroid polytope over all  $G_j$  with  $j \in H_I$ . (Recall that sets  $G_j$  for  $j \in H_I$  are pairwise disjoint.)

Now let  $y^I, y^A$  be a feasible solution of (9)-(12). For all  $j \in H_I$  with  $y^I(G_j) \leq 1$ , set  $z_i = y_i^I$  for all  $i \in G_j$ . For all  $j \in H_I$  with  $y^I(G_j) > 1$ , set  $z_i = y_i^I / y^I(G_j)$  for all  $i \in G_j$ . For the rest of the facilities set  $z_i = 0$ . This solution obviously satisfies (14). Also, because  $y^I$  satisfies (11) and  $z_i \leq y_i^I$  for all  $i$ , we know that  $z$  satisfies (15) too. Finally, regarding the objective function:

$$\sum_{i \in \mathcal{F}} z_i \cdot c_i^I \leq \sum_{i \in \mathcal{F}} y_i \cdot c_i^I \quad (17)$$

For the second-stage cost we then get:

$$\begin{aligned} \sum_{A \in Q} p_A \sum_{j \in H_A} c_{i_j^A}^A (1 - z(G_{\pi^I(j)})) &\leq \sum_{A \in Q} p_A \sum_{\substack{j \in H_A: \\ y^I(G_{\pi^I(j)}) \leq 1}} c_{i_j^A}^A (1 - y^I(G_{\pi^I(j)})) \\ &\leq \sum_{A \in Q} p_A \sum_{\substack{j \in H_A: \\ y^I(G_{\pi^I(j)}) \leq 1}} c_{i_j^A}^A (1 - y^I(G_j)) \\ &\leq \sum_{A \in Q} p_A \sum_{\substack{j \in H_A: \\ y^I(G_{\pi^I(j)}) \leq 1}} c_{i_j^A}^A y^A(G_j) \leq \sum_{A \in Q} p_A \sum_{i \in \mathcal{F}} y_i^A c_i^A \quad (18) \end{aligned}$$

The second line follows from the stage-I greedy clustering, which ensures  $y^I(G_{\pi^I(j)}) \geq y^I(G_j)$  for all  $j \in \mathcal{C}$ . The last line is due to (10), and the fact that for all  $A \in Q$  and all distinct  $j, j' \in H_A$  we have  $G_j \cap G_{j'} = \emptyset$ . Finally, combining (9), (17) and (18) we get the desired bound on the cost. ◀

► **Theorem 18.** *For the sets  $F_I, F_A : A \in Q$  returned by Algorithm 5 the following three properties hold: (i)  $F_I \in \mathcal{I}$ , (ii)  $c^I(F_I) + \sum_{A \in Q} p_A c^A(F_A) \leq B$ , and (iii)  $d(j, F_I \cup F_A) \leq 5R$  for all  $j \in A \in Q$ .*

**Proof.** (i) is obvious, since  $z^*$  satisfies constraint (15). For (ii), the opening cost of the solution coincides with the value of the objective (13) for  $z^*$ , and hence by Lemma 17 it is at most  $B$ .

For (iii), consider  $A \in Q$ , and recall that  $d(j, \pi^I(j)) \leq 2R$  and  $d(j, \pi^A(j)) \leq 2R$  for any  $j \in A$ . For  $j \in H_A$  the bound (iii) holds, because either  $G_{\pi^I(j)} \cap F_I \neq \emptyset$  or  $G_j \cap F_A \neq \emptyset$ . So suppose that  $j \in A \setminus H_A$ . If  $G_{\pi^A(j)} \cap F_A \neq \emptyset$ , then any facility  $i \in G_{\pi^A(j)} \cap F_A$  will be within distance  $3R$  from  $j$ . If on the other hand  $G_{\pi^A(j)} \cap F_A = \emptyset$ , then there exists  $i \in G_{\pi^I(\pi^A(j))} \cap F_I$ . Therefore,  $d(i, j) \leq d(i, \pi^I(\pi^A(j))) + d(\pi^I(\pi^A(j)), \pi^A(j)) + d(\pi^A(j), j) \leq 5R$ . ◀

---

**Algorithm 6** Generalization Procedure for 2S-MatSup-Poly.
 

---

**Input:** Returned sets  $F_I, F_A : A \in Q$  and inner execution details of Algorithm 5  
 Let  $\bar{s}$  the strategy we will define, and for the stage-I actions set  $F_I^{\bar{s}} \leftarrow F_I$ ;  
 Suppose scenario  $A \in \mathcal{D}$  arrived in the second stage;  
 Let  $\pi^I$  the stage-I mapping and  $g^{II}$  the bijective function, both used in Algorithm 5;  
 Set  $(H_A, \pi^A) \leftarrow \text{GreedyCluster}(A, R, g^{II})$ ;  
 Open the set  $F_A^{\bar{s}} = \{i_j^A \mid j \in H_A \text{ and } F_I \cap G_{\pi^I(j)} = \emptyset\}$ ;

---

## 4.2 Generalizing to the Black-Box Setting

It is clear that Algorithm 5 satisfies Definition 7, and therefore is a valid 5-approximation. Consider now Algorithm 6 to efficiently extend its output to any arriving scenario  $A \in \mathcal{D}$ . Since Algorithm 6 exactly imitates the stage-II actions of Algorithm 5, it is easy to see that property II is satisfied. Furthermore, the arguments in Theorem 18 would still go through, and eventually guarantee  $d(j, F_I^{\bar{s}} \cup F_A^{\bar{s}}) \leq 5R$  for all  $j \in A$  and any  $A \in \mathcal{D}$ , thus verifying property I. To conclude, we only need to prove III. Let  $\mathcal{S}_M$  the set of strategies achievable via Algorithm 6.

► **Lemma 19.** *Algorithm 5 satisfies property III with  $|\mathcal{S}_M| = 2^m \cdot n!$ .*

**Proof.** Since  $g^{II}$  can be thought of as part of the input,  $\bar{s}$  depends only on 1) the set  $F_I$  returned by Algorithm 5, and 2) the sorted order of  $y^I(G_j)$  for all  $j \in \mathcal{C}$ , which ultimately dictates the mapping  $\pi^I$ . Given those, we can determine the stage-II openings for every possible scenario  $A \in \mathcal{D}$ . These options do not depend on scenarios  $Q$ . The total number of possible outcomes for  $F_I$  is  $2^m$ , and the total number of orderings for the clients of  $\mathcal{C}$  is  $n!$ . Hence,  $|\mathcal{S}_M| = 2^m \cdot n!$ . ◀

---

## References

- 1 Shipra Agrawal, Amin Saberi, and Yinyu Ye. Stochastic combinatorial optimization under probabilistic constraints, 2008. [arXiv:0809.0460](https://arxiv.org/abs/0809.0460).
- 2 E. M. L. Beale. On minimizing a convex function subject to linear inequalities. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 173–184, 1955.
- 3 Deeparnab Chakrabarty and Maryam Negahbani. Generalized center problems with outliers. *ACM Trans. Algorithms*, 2019.
- 4 Moses Charikar, Chandra Chekuri, and Martin Pal. Sampling bounds for stochastic optimization. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 257–269, 2005.
- 5 George B. Dantzig. Linear programming under uncertainty. *Management Science*, pages 197–206, 1955.
- 6 Uriel Feige. On sums of independent random variables with unbounded variance and estimating the average degree in a graph. *SIAM Journal on Computing*, 35(4):964–984, 2006.
- 7 A. Gupta, R. Ravi, and A. Sinha. An edge in time saves nine: Lp rounding approximation algorithms for stochastic network design. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 218–227, 2004.
- 8 Anupam Gupta, Martin Pál, R. Ravi, and Amitabh Sinha. Boosted sampling: Approximation algorithms for stochastic optimization. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '04, page 417–426, 2004.
- 9 Anupam Gupta, Martin Pal, R. Ravi, and Amitabh Sinha. Sampling and cost-sharing: Approximation algorithms for stochastic optimization problems. *SIAM J. Comput.*, pages 1361–1401, 2011.

- 10 Dorit S. Hochbaum and David B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *J. ACM*, 1986.
- 11 Nicole Immorlica, David Karger, Maria Minkoff, and Vahab S. Mirrokni. On the costs and benefits of procrastination: Approximation algorithms for stochastic combinatorial optimization problems. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 691–700, 2004.
- 12 Anton J. Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.
- 13 Lap-Chi Lau, R. Ravi, and Mohit Singh. *Iterative Methods in Combinatorial Optimization*. Cambridge University Press, USA, 1st edition, 2011.
- 14 Andre Linhares and Chaitanya Swamy. Approximation algorithms for distributionally-robust stochastic optimization with black-box distributions. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 768–779, 2019.
- 15 Andrea Pietracaprina, Geppino Pucci, and Federico Solda. Coreset-based strategies for robust center-type problems, 2020. [arXiv:2002.07463](https://arxiv.org/abs/2002.07463).
- 16 R. Ravi and Amitabh Sinha. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. In *Integer Programming and Combinatorial Optimization*, 2004.
- 17 David Shmoys and Chaitanya Swamy. An approximation scheme for stochastic linear programming and its application to stochastic integer programs. *J. ACM*, 53:978–1012, November 2006.
- 18 Aravind Srinivasan. Approximation algorithms for stochastic and risk-averse optimization. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1305–1313, 2007.
- 19 C. Swamy and D. B. Shmoys. Sampling-based approximation algorithms for multi-stage stochastic optimization. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 357–366, 2005.
- 20 Chaitanya Swamy. Risk-averse stochastic optimization: Probabilistically-constrained models and algorithms for black-box distributions: (extended abstract). *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1627–1646, 2011.
- 21 Chaitanya Swamy and David Shmoys. The sample average approximation method for 2-stage stochastic optimization. *Survey Paper*, April 2008.
- 22 Chaitanya Swamy and David B. Shmoys. Approximation algorithms for 2-stage stochastic optimization problems. *SIGACT News*, pages 33–46, 2006.

## **A** Applying the Standard SAA Method in Supplier Problems

Consider the standard two-stage stochastic setting. In the first stage, we are allowed to take some proactive actions and commit to an anticipatory part of the solution  $x$ , which will incur some cost  $c(x)$ . In the second stage, a scenario  $A$  is sampled from the distribution  $\mathcal{D}$ , and we can take some *stage-II* recourse actions  $y_A$  with cost  $f_A(x, y_A)$ . If  $X$  is the set of stage-I actions and  $Y$  the set of recourse actions, the goal is to find a solution  $x^* \in X$  to minimize  $f(x) = c(x) + \mathbb{E}_{A \sim \mathcal{D}}[q_A(x)]$ , where  $q_A(x) = \min_{y \in Y} \{f_A(x, y) \mid (x, y) \text{ is a valid solution for } A\}$ .

**The Standard SAA Method.** Consider minimizing  $f(x)$  in the black-box model. If  $S$  is a set of scenarios sampled from the black-box oracle, let  $\hat{f}(x) = c(x) + (\sum_{A \in S} q_A(x))/|S|$  be the empirical estimate of  $f(x)$ . Also, let  $x^*$  and  $\bar{x}$  be the minimizers of  $f(x)$  and  $\hat{f}(x)$  respectively.

The work [21] shows that if  $f(x)$  is modeled as a convex program, then for any  $\epsilon, \gamma \in (0, 1)$  and with  $|S| = \text{poly}(n, m, \lambda, \epsilon, 1/\gamma)$ , we have  $f(\bar{x}) \leq (1 + \epsilon)f(x^*)$  with probability at least  $1 - \gamma$  ( $\lambda$  is the maximum multiplicative factor by which an element's cost is increased in

stage-II). An alternate proof of this appeared in [4], which also covered the case of  $f(x)$  being an integer program. Moreover, [4] proves that if  $\bar{x}$  is an  $\alpha$ -approximate minimizer of  $\hat{f}(x)$ , then a slight modification to the sampling still gives  $f(\bar{x}) \leq (\alpha + \epsilon)f(x^*)$  with probability at least  $1 - \gamma$ .

The result of [4] further implies that the black-box model can be effectively reduced to the polynomial-scenarios one, via the following process. Assuming that  $f(x)$  corresponds to the integer program modeling our problem, first find an  $\alpha$ -approximate minimizer  $\bar{x}$  of  $\hat{f}(x)$ , and treat  $\bar{x}$  as the stage-I actions. Then, given any arriving  $A$ , re-solve the problem using any known  $\rho$ -approximation algorithm for the non-stochastic counterpart, with  $\bar{x}$  as a fixed part of the solution. This process eventually leads to an overall approximation ratio of  $\alpha\rho + \epsilon$ .

**Roadblocks for the Standard SAA Analysis in Supplier Problems.** A natural way to fit our models within the existing framework, is to first assume knowledge of the optimal radius  $R^*$  and then use the opening cost as the objective function  $f_{R^*}(x)$ , by turning the radius requirement into a simple covering constraint. In other words, we set  $f_{R^*}(x) = c^I(x) + \mathbb{E}_{A \sim \mathcal{D}}[q_{A,R^*}(x)]$  with  $q_{A,R^*}(x) = \min_y \{c^A(y) \mid (x, y) \text{ covers all } j \in A \text{ within distance } R^*\}$ . Note that  $f_{R^*}(x)$  may represent both the convex and the integer program corresponding to the underlying problem.

To avoid any overhead in the approximation ratio (from re-solving the problem in stage-II), one should apply SAA to the function  $f_{R^*}(x)$  corresponding to the convex program describing the problem (the roadblock described here trivially extends to the case of  $f_{R^*}(x)$  being an integer function as well). If there exists a rounding that turns the empirical minimizer  $\bar{x}_{R^*}$  into a solution that covers each client within distance  $\alpha R^*$ , while also having an opening cost of at most  $f_{R^*}(\bar{x}_{R^*})$ , we get the desired result because  $f_{R^*}(\bar{x}_{R^*}) \leq (1 + \epsilon)f_{R^*}(x_{R^*}^*)$  and  $f_{R^*}(x_{R^*}^*) \leq B$ . With slight modifications, all our polynomial-scenarios algorithms can be interpreted as such rounding procedures.

Nonetheless, we still have to identify a good guess for  $R^*$ , **and this constitutes an unavoidable roadblock in applying standard SAA** in supplier problems. Since  $R^*$  is one of  $nm$  alternative options, one can test each of those individually. Hence, assume we work with some guess  $R$ , and define the corresponding cost functions  $f_R, \hat{f}_R$  with minimizers  $x_R^*, \bar{x}_R$  respectively. Observe that  $R$  is a good guess iff  $f_R(x_R^*) \leq (1 + \mathcal{O}(\epsilon))B$ , since in this way vanilla SAA combined with our rounding procedures yields an opening cost of  $f_R(\bar{x}_R) \leq (1 + \epsilon)f_R(x_R^*)$ , and minimizing over the radius is just a matter of finding the minimum good guess. However, because  $f_R(x)$  is not efficiently computable, the only way to test if  $R$  is a good guess, is through  $\hat{f}_R(x)$ . Unfortunately, empirically estimating  $f_R(x)$  within an  $(1 + \epsilon)$  factor may require a super-polynomial number of samples [12]. The reason for this is the existence of scenarios with high stage-II cost appearing with small probability, which drastically increase the variance of  $\hat{f}_R(x)$ . **On a high level, the obstacle in supplier problems stems from the need to not only find a minimizer  $\bar{x}_R$ , but also compute its corresponding value  $f_R(\bar{x}_R)$ .** This makes it impossible to know which guesses  $R$  are good, and consequently there is no way to optimize over the radius.

Finally, note that if the stage-II cost of every scenario is polynomially bounded, the variance of  $\hat{f}_R(x)$  is also polynomial, and standard SAA arguments go through without difficulties. However, this assumption is much stronger than is typically used for the two-stage stochastic model.

## B Auxiliary Lemmas

► **Lemma 20** ([6]). *Let  $X_1, \dots, X_K$  be non-negative independent random variables, with expectations  $\mu_1, \dots, \mu_K$  respectively, where  $\mu_k \leq 1$  for every  $k$ . Let  $X = \sum_{k=1}^K X_k$ , and let  $\mu = \sum_{k=1}^K \mu_k = \mathbb{E}[X]$ . Then for every  $\delta > 0$  we have  $\Pr[X < \mu + \delta] \geq \min\{\frac{\delta}{1+\delta}, \frac{1}{13}\}$ .*

The two following lemmas are standard Chernoff bounds.

► **Lemma 21.** *Let  $X_1, X_2, \dots, X_K$  be independent random variables with  $X_k \in [0, 1]$  for every  $k$ . For  $X = \sum_{k=1}^K X_k$  with  $\mu = \mathbb{E}[X]$  and any  $\delta > 0$ , we have  $\Pr[X \leq (1 - \delta)\mu] \leq e^{-\frac{\mu\delta^2}{2}}$ .*

► **Lemma 22.** *Let  $X_1, X_2, \dots, X_K$  be independent Bernoulli random variables with parameter  $p$ . Let  $X = \sum_{k=1}^K X_k$  the corresponding binomial random variable. If for the realization of  $X$  we have  $X = qK$ , then for any  $\delta > 0$  we have  $\Pr[p < q - \delta] \leq e^{-K\delta^2/2p}$ .*

## C Approximation Algorithm for 2S-MuSup-BB

To tackle this, we construct an efficiently generalizable algorithm for **2S-MuSup-Poly**, via an intriguing reduction to a non-stochastic clustering problem with outliers. Specifically, if we view stage-I as consisting of a deterministic robust problem, stage-II is interpreted as covering all outliers left over by stage-I. Formally, we use the following robust problem:

**Robust Weighted Multi-Knapsack-Supplier.** We are given a set of clients  $\mathcal{C}$  and a set of facilities  $\mathcal{F}$ , in a metric space with distance function  $d$ . The input also includes parameters  $V, R \in \mathbb{R}_{\geq 0}$ , and for every client  $j \in \mathcal{C}$  an associated weight  $v_j \in \mathbb{R}_{\geq 0}$ . In addition, we have the same types of multi-knapsack constraints as in **2S-MuSup**: there are  $L$  in total budgets  $W_\ell$ , and every facility  $i \in \mathcal{F}$  has costs  $f_i^\ell$  for  $\ell \in [L]$ . The goal is to choose a set of facilities  $S \subseteq \mathcal{F}$ , such that  $\sum_{j \in \mathcal{C}: d(j,S) > R} v_j \leq V$  and  $f^\ell(S) \leq W_\ell$  for every  $\ell \in [L]$ . Clients  $j$  with  $d(j, S) > R$  are called outliers. Finally, an instance of this problem is called *discrete*, if the values  $f_i^\ell$  are all integers.

We first show that any  $\rho$ -approximation for **Robust Weighted Multi-Knapsack-Supplier** can be used in order to get an efficiently generalizable  $(\rho + 2)$ -approximation algorithm for **2S-MuSup-Poly**. In addition, we argue that already existing work [3, 15] gives a 3-approximation for discrete instances of **Robust Weighted Multi-Knapsack-Supplier**, thus leading to an efficiently generalizable 5-approximation for discrete instances of **2S-MuSup-Poly**.

### C.1 Reducing 2S-MuSup-Poly to Robust Weighted Multi-Knapsack-Supplier

We first suppose that the costs  $c_i^j$  are polynomially bounded integers, and claim that this restriction will be removed when we generalize to the black-box setting. Once more, let  $Q$  be a set of provided scenarios,  $R$  a target radius, and  $G_j = G_{j,R}$ ,  $i_j^A = i_{j,R}^A$  for all  $j \in \mathcal{C}$  and  $A \in Q$ . Furthermore, suppose that we have a  $\rho$ -approximation algorithm  $RW$  for **Robust Weighted Multi-Knapsack-Supplier**. For a feasible instance  $\mathcal{J}'$  of the latter problem,  $RW$  returns a solution  $S$  satisfying all knapsack constraints and also  $\sum_{j \in \mathcal{C}: d(j,S) > \rho R} v_j \leq V$ . Otherwise, it either returns “INFEASIBLE”, or again a solution with the previous properties.

If the provided instance  $\mathcal{J}$  of **2S-MuSup-Poly** is feasible, the first step in tackling the problem is figuring out the portion of the budget, say  $B_I$ , that is used in the first stage of a

**Algorithm 7** Approximation Algorithm for **2S-MuSup-Poly**.

---

```

Let  $g^{II} : \mathcal{C} \mapsto [n]$  be some fixed and given bijective mapping;
for each scenario  $A \in Q$  do
    |  $(H_A, \pi^A) \leftarrow \text{GreedyCluster}(A, R, g^{II});$ 
end
Construct instance  $\mathcal{J}'$  of Robust Weighted Multi-Knapsack-Supplier as
discussed;
if  $RW(\mathcal{J}') = \text{"INFEASIBLE"}$  then
    | Return "INFEASIBLE";
end
 $F_I \leftarrow RW(\mathcal{J}')$ ; // Stage-I facilities
for each scenario  $A \in Q$  do
    |  $F_A \leftarrow \{i_j^A \mid j \in H_A \text{ with } d(j, F_I) > \rho R\}$ ; // Stage-II facilities
end
    
```

---

feasible solution. Since the costs  $c_i^I$  are polynomially bounded integers, we can guess  $B_I$  in polynomial time through solving the problem for all different alternatives for it. So from this point on, assume w.l.o.g. that we have the correct  $B_I$ , and also let  $B_{II} = B - B_I$ .

Algorithm 7 shows how to use  $RW$  to approximate **2S-MuSup-Poly**. It begins with greedy clustering steps for each  $A$ , and given  $H_A, \pi^A$  it constructs an instance  $\mathcal{J}'$  of **Robust Weighted Multi-Knapsack-Supplier** as follows.  $\mathcal{C}, \mathcal{F}, d$ , and  $R$  are the same for both problems. For all  $j \in \mathcal{C}$  we set  $v_j = \sum_{A \in Q: j \in H_A} p_A \cdot c_{i_j^A}^A$  and also  $V = B_{II}$ . Finally, the instance  $\mathcal{J}'$  has  $L' = L + 1$  knapsack constraints, where the first  $L$  are the stage-I constraints of **2S-MuSup-Poly** ( $f^\ell(S) \leq W_\ell$ ), and the last is  $c^I(S) \leq B_I$ .

► **Lemma 23.** *If the original **2S-MuSup-Poly** instance  $\mathcal{J}$  is feasible, then the **Robust Weighted Multi-Knapsack-Supplier** instance  $\mathcal{J}'$  is also feasible.*

**Proof.** Consider some feasible solution  $F_I^*, F_A^*$  for **2S-MuSup-Poly**. We claim that  $F_I^*$  is a valid solution for  $\mathcal{J}'$ . It clearly satisfies the  $L$  knapsack constraints of the form  $f^\ell(F_I^*) \leq W_\ell$ , and if our guess  $B_I$  is the right one, it also satisfies  $c^I(F_I^*) \leq B_I$ . Now, for any  $A \in Q$ , any client  $j \in H_A$  with  $d(j, F_I^*) > R$  must be covered by some facility  $x_j^A \in G_j \cap F_A^*$ . Since  $B_{II}$  is the second-stage portion of the budget used by  $F_I^*, F_A^*$ , and  $G_{j'} \cap G_{j''} = \emptyset$  for all distinct  $j', j'' \in H_A$  we have:

$$B_{II} \geq \sum_A p_A \sum_{i \in F_A^*} c_i^A \geq \sum_A p_A \sum_{\substack{j \in H_A: \\ d(j, F_I^*) > R}} c_{x_j^A}^A \geq \sum_A p_A \sum_{\substack{j \in H_A: \\ d(j, F_I^*) > R}} c_{i_j^A}^A = \sum_{\substack{j \in \mathcal{C}: \\ d(j, F_I^*) > R}} v_j$$

This implies that  $S = F_I^*$  satisfies the constraint  $\sum_{j: d(j, S) > R} v_j \leq B_{II}$  of instance  $\mathcal{J}'$ . ◀

► **Theorem 24.** *Algorithm 7 is a valid  $(\rho + 2)$ -approximation for **2S-MuSup-Poly**.*

**Proof.** First of all, Lemma 23 guarantees that if the given instance of **2S-MuSup-Poly** is feasible, we will get a solution  $F_I, F_A$ . By specification of  $RW$ ,  $c^I(F_I) \leq B_I$  and  $f^\ell(F_I) \leq W_\ell$  for every  $\ell$ . The stage-II cost  $C_{II}$  of this solution is given by:

$$C_{II} = \sum_A p_A \sum_{\substack{j \in H_A: \\ d(j, F_I) > \rho R}} c_{i_j^A}^A = \sum_{\substack{j \in \mathcal{C}: \\ d(j, F_I) > \rho R}} v_j \leq B_{II},$$



■ **Algorithm 8** Generalization Procedure for **2S-MuSup-Poly**.

---

**Input :** Returned sets  $F_I, F_A : A \in Q$  and inner execution details of Algorithm 7  
 Let  $\bar{s}$  the strategy we will define, and for the stage-I actions set  $F_I^{\bar{s}} \leftarrow F_I$ ;  
 Suppose scenario  $A$  arrived in the second stage;  
 $(H_A, \pi^A) \leftarrow \text{GreedyCluster}(A, R, g^{II})$ , where  $g^{II}$  the bijective function used in  
 Algorithm 7;  
 Open the set  $F_A^{\bar{s}} \leftarrow \{i_j^A \mid j \in H_A \text{ and } d(j, F_I) > \rho R\}$ ;

---

where the last inequality follows because  $F_I$  is the output of  $RW(\mathcal{J}')$ .

Consider now a  $j \in A$  for some  $A \in Q$ . The distance of  $j$  to its closest facility will be at most  $d(\pi^A(j), F_I \cup F_A) + d(j, \pi^A(j))$ . Since  $\pi^A(j) \in H_A$ , there will either be a stage-I open facility within distance  $\rho R$  from it, or we perform a stage-II opening in  $G_{\pi(j)}$ , which results in a covering distance of at most  $R$ . Also, by the greedy clustering step, we have  $d(j, \pi^A(j)) \leq 2R$ . So in the end we get  $d(j, F_I \cup F_A) \leq (\rho + 2)R$ . ◀

By combining Algorithm 7 with existing 3-approximation algorithms for **Robust Weighted Multi-Knapsack-Supplier**, we get the following result:

► **Theorem 25.** *There is a 5-approximation algorithm for discrete instances of **2S-MuSup-Poly**, where additionally all  $c_i^I$  are polynomially bounded integers. The runtime of it is  $\text{poly}(n, m, \Lambda)$ .*

**Proof.** The results of [3] give a 3-approximation for discrete instances of **Robust Weighted Multi-Knapsack-Supplier**, when  $v_j = 1$  for all  $j$ . The work of [15] extends this to allow arbitrary  $v_j$  values. Note that by our assumption that the values  $c^I$  are polynomially bounded integers, the instance  $\mathcal{J}'$  is discrete, and hence the algorithm of [15] can be utilized in Algorithm 7 and give a 5-approximation for **2S-Sup-Poly**. Finally, given the results in [3, 15], the runtime of the whole process will be  $\text{poly}(n, m, \Lambda)$ . ◀

## C.2 Generalizing to the Black-Box Setting

Since the algorithm of Section C.1 is a valid  $(\rho + 2)$ -approximation, consider the process in Algorithm 8, which efficiently extends its output to any arriving scenario  $A \in \mathcal{D}$ .

Because Algorithm 8 exactly mimics the stage-II actions of Algorithm 7, it is easy to see that property II is satisfied. Further, the arguments of Theorem 24 would still ensure  $d(j, F_I \cup F_A) \leq (\rho + 2)R$  for every  $j \in A$  and  $A \in \mathcal{D}$ , thus guaranteeing property I. To conclude, we again only need to prove property III. Let  $\mathcal{S}_{MK}$  the set of strategies achievable via Algorithm 8.

► **Lemma 26.** *Algorithm 7 satisfies property III with  $|\mathcal{S}_{MK}| = 2^m$ .*

**Proof.** The returned final strategy depends solely on the set  $F_I$ . Given that, we can exactly determine all possible stage-II openings, since every  $H_A$  for  $A \in \mathcal{D}$  can be computed using the fixed function  $g^{II}$ . There are  $2^m$  choices for  $F_I$ , and therefore  $|\mathcal{S}_{MK}| = 2^m$ . Finally, it is easy to see that the set  $\mathcal{S}_{MK}$  is independent of  $Q$ . ◀

Our algorithm for **2S-MuSup-Poly** requires the values  $c_i^I$  to be polynomially bounded integers. As we show next, this assumption can be removed by a standard rescaling trick:

## 23:22 Approximating Two-Stage Stochastic Supplier Problems

► **Theorem 27.** *Suppose that the  $c_i^I$  are arbitrary numbers. By appropriate cost-quantization for any  $\epsilon \in (0, 1)$ , Algorithm 7 can be modified to give a solution  $F_I, F_A : A \in Q$  for **2S-MuSup-Poly**, where  $d(j, F_I \cup F_A) \leq (\rho + 2)R$  for all  $A \in Q, j \in A$ , and also  $c^I(F_I) + \sum_{A \in Q} p_A c^A(F_A) \leq (1 + \epsilon)B$ .*

**Proof.** For convenience, let us assume that  $B = 1$ , and suppose that all facilities have  $c_i^I \leq B = 1$  (as otherwise they can never be opened). Given some  $\epsilon > 0$ , let us define  $q = \epsilon/m$ , and form new costs by  $\tilde{c}_i^I = \lceil c_i^I/q \rceil, \tilde{c}_i^A = c_i^A/q, B' = B(1 + \epsilon)/q$ . The costs  $\tilde{c}_i^I$  are at most  $\lceil 1/q \rceil$ , and hence are polynomially-bounded integers. Therefore, the reduction of Section C.1 can be applied.

Suppose now that  $F_I, F_A$  is a solution to the original instance of **2S-MuSup-Poly**, with opening cost at most  $B$ . For the modified cost of this solution we then have:

$$\tilde{c}^I(F_I) + \sum_A p_A \tilde{c}^A(F_A) \leq (c^I(F_I) + \sum_A p_A c^A(F_A))/q + \sum_{i \in \mathcal{F}} 1 \leq B/q + m \leq B'$$

Thus,  $F_I, F_A$  is also a solution to the modified instance, implying that the latter is feasible. Hence, consider any solution  $\tilde{F}_I, \tilde{F}_A$  to the modified instance, that we would get after running Algorithm 7 with the new costs; its opening cost in the original instance is  $c^I(\tilde{F}_I) + \sum_A p_A c^A(\tilde{F}_A) \leq q\tilde{c}^I(\tilde{F}_I) + q\sum_A p_A \tilde{c}^A(\tilde{F}_A) \leq qB' = B(1 + \epsilon)$ . Therefore, since  $\tilde{F}_I, \tilde{F}_A$  is a  $(\rho + 2)$ -approximate solution, we get the desired result. ◀

Note that applying our generalization framework on this solution would make the overall cost over  $\mathcal{D}$  be at most  $(1 + \mathcal{O}(\epsilon))(1 + \epsilon)B = (1 + \mathcal{O}(\epsilon))B$ , which implies that in the black-box setting we do not need the initial assumption for the costs  $c_i^I$ .

### C.3 Connections to 2S-MatSup

Suppose we define our non-stochastic robust problem as having one knapsack and one matroid constraint, instead of  $L$  knapsack constraints. Then the reduction of Section C.1 would yield a  $(\rho + 2)$ -approximation for **2S-MatSup-Poly** in the exact same manner, where  $\rho$  the ratio of the algorithm used to solve the corresponding deterministic outliers problem.

A result of [3, Theorem 16] gives a 3-approximation for this outliers problem, which in turn would give a 5-approximation for **2S-MatSup-Poly**. However, the algorithm obtained in this way would be randomized (its solution may not be a valid one), would only work for polynomially bounded values  $v_j$ , and would also be significantly more complex than the algorithm of Section 4.

# Fast Approximation Algorithms for Bounded Degree and Crossing Spanning Tree Problems

Chandra Chekuri ✉

University of Illinois at Urbana-Champaign, IL, USA

Kent Quanrud ✉

Purdue University, West Lafayette, IN, USA

Manuel R. Torres ✉

University of Illinois at Urbana-Champaign, IL, USA

---

## Abstract

We develop fast approximation algorithms for the minimum-cost version of the Bounded-Degree MST problem (BD-MST) and its generalization the Crossing Spanning Tree problem (CROSSING-ST). We solve the underlying LP to within a  $(1 + \epsilon)$  approximation factor in near-linear time via the multiplicative weight update (MWU) technique. This yields, in particular, a near-linear time algorithm that outputs an estimate  $B$  such that  $B \leq B^* \leq \lceil (1 + \epsilon)B \rceil + 1$  where  $B^*$  is the minimum-degree of a spanning tree of a given graph. To round the fractional solution, in our main technical contribution, we describe a fast near-linear time implementation of swap-rounding in the spanning tree polytope of a graph. The fractional solution can also be used to sparsify the input graph that can in turn be used to speed up existing combinatorial algorithms. Together, these ideas lead to significantly faster approximation algorithms than known before for the two problems of interest. In addition, a fast algorithm for swap rounding in the graphic matroid is a generic tool that has other applications, including to TSP and submodular function maximization.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Approximation algorithms analysis; Theory of computation  $\rightarrow$  Graph algorithms analysis

**Keywords and phrases** bounded degree spanning tree, crossing spanning tree, swap rounding, fast approximation algorithms

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.24

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/abs/2011.03194v2> [19]

**Funding** *Chandra Chekuri*: Supported in part by NSF grant CCF-1910149.

*Manuel R. Torres*: Supported in part by fellowships from NSF and the Sloan Foundation, and NSF grant CCF-1910149.

## 1 Introduction

Spanning trees in graphs are a fundamental object of study and arise in a number of settings. Efficient algorithms for finding a minimum-cost spanning tree (MST) in a graph are classical. In a variety of applications ranging from network design, TSP, phylogenetics, and others, one often seeks to find a spanning tree with additional constraints. An interesting and well-known problem in this space is the BOUNDED-DEGREE SPANNING TREE (BD-ST) problem in which the goal is to find a spanning tree in a given graph  $G = (V, E)$  that minimizes the maximum degree in the tree. We refer to the minimum-cost version of BD-ST as BD-MST where one seeks a spanning tree of minimum cost subject to a given degree bound  $B$  on the vertices. The decision version of BD-ST (Given  $G, B$  is there a spanning tree with maximum degree  $B$ ?) is already NP-Complete for  $B = 2$  since it captures the Hamilton-Path problem. In an influential paper, Fürer and Raghavachari [24], building on earlier work of Win [49],



© Chandra Chekuri, Kent Quanrud, and Manuel R. Torres;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 24; pp. 24:1–24:21



Leibniz International Proceedings in Informatics

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

described a simple local-search type algorithm that runs in  $\tilde{O}(mn)$  time (here  $m$  is number of edges and  $n$  number of nodes) that outputs a spanning tree with degree at most  $B + 1$ , or certifies that  $G$  does not have a spanning tree with degree at most  $B$  (we use  $\tilde{O}$  notation to suppress poly-logarithmic factors in  $n, m, 1/\epsilon$  for notational simplicity). Their algorithm, in fact, works even in the non-uniform setting where each vertex  $v$  has a specified degree bound  $B_v$ . The Fürer-Raghavachari result spurred a substantial line of work that sought to extend their clean result to the minimum-cost setting. This was finally achieved by Singh and Lau [43] who described a polynomial-time algorithm that outputs a tree  $T$  such that the degree of each  $v$  in  $T$  is at most  $B_v + 1$  and the cost of the tree is at most  $\text{OPT}$ . Their algorithm is based on iterative rounding of a natural LP relaxation. We refer the reader to [35, 13, 28, 43, 22] for several ideas and pointers on BD-ST and BD-MST.

Motivated by several applications, Bilo et al. [9] defined the CROSSING SPANNING TREE problem (CROSSING-ST). In CROSSING-ST the input is a graph  $G = (V, E)$ , a collection of cuts  $C_1, C_2, \dots, C_k$ , and integers  $B_1, B_2, \dots, B_k$ . Each cut  $C_i$  is a subset of the edges though in many applications we view  $C_i$  as  $\delta_G(S_i)$  for some  $S_i \subset V$  (where  $\delta_G(S_i) = \{uv \in E \mid u \in S_i, v \in V \setminus S_i\}$  is the standard definition of a cut set with respect to  $S_i$ ). The goal is to find a spanning tree  $T$  such that  $|E(T) \cap C_i| \leq B_i$ , that is,  $T$  crosses each cut  $C_i$  at most  $B_i$  times. It is easy to see that BD-ST is a special case of CROSSING-ST where the cuts correspond to singletons. We refer to the min-cost version of CROSSING-ST as CROSSING-MST. CROSSING-ST gained substantial prominence in the context of the asymmetric traveling salesman problem (ATSP) – Asadpour et al. [5] showed the importance of thin spanning trees for approximating ATSP and obtained an  $O(\log n / \log \log n)$ -approximation (now we have constant factor approximations for ATSP via other methods [46, 47]). Motivated by the thin tree conjecture and its applications to ATSP (see [5, 2]) and other technical considerations, researchers have studied CROSSING-ST, its generalization to the matroid setting, and various special cases [20, 8, 7, 37, 36]. The best known approximation algorithms for CROSSING-ST and its special cases have mainly relied on the natural LP relaxation. For general CROSSING-ST the best known approximation ratio is  $\min\{O(\log k / \log \log k), (1 + \epsilon)B + O(\log k / \epsilon^2)\}$ . A variety of sophisticated and interesting rounding techniques have been designed for CROSSING-ST and its special cases. An outstanding open problem is whether CROSSING-ST admits a constant factor approximation via the natural LP relaxation. This is challenging due its implications for the thin tree conjecture.

Most of the focus on BD-MST and CROSSING-ST has been on the quality of the approximation. The best known approximation bounds rely on LP relaxations and complex rounding procedures. The overall running times are very large polynomials in the input size and are often unspecified. In this paper we are interested in the design of *fast* approximation algorithms for BD-MST, CROSSING-ST and related problems. In recent years there has been significant progress in designing fast, and often near-linear time, approximation algorithms for a number of problems in discrete and combinatorial optimization. This has been led by, and also motivated, synergy between continuous/convex optimization, numerical linear algebra, dynamic data structures, sparsification techniques, and structural results, among several others. For BD-ST with uniform degree, Duan, He and Zhang [22] described a combinatorial algorithm that for any given  $\epsilon > 0$ , runs in  $O(m \log^7 n / \epsilon^7)$  time, and either outputs a spanning tree with degree  $(1 + \epsilon)B + O(\log n / \epsilon^2)$  or reports that there does not exist a tree with maximum degree  $\leq B$ . This paper is partly motivated by the goal of improving their results: dependence on  $\epsilon$ , a better approximation, handling non-uniform bounds, cost, CROSSING-MST, and connection to the LP relaxation.

A second motivation for this paper is to develop a fast algorithm for swap-rounding in the spanning tree polytope. It is a dependent rounding technique that has several applications ranging from TSP to submodular function maximization (see [20, 26, 17, 23]). The question of developing a fast swap-rounding procedure for spanning trees was explicitly raised in [17] in the context of Metric-TSP.

## 1.1 Results

In this paper we develop fast approximation algorithms for BD-MST, CROSSING-MST and related problems in a unified fashion via broadly applicable methodology based on the LP relaxation. We consider the following problem with general packing constraints. The input to this problem is an undirected graph  $G = (V, E)$ , a non-negative edge-cost vector  $c : E \rightarrow \mathbb{R}_+$ , a non-negative matrix  $A \in [0, 1]^{k \times m}$ , and a vector  $b \in [1, \infty)^k$ . The goal is to find a spanning tree  $T$  of minimum cost such that  $A\mathbf{1}_T \leq b$  where  $\mathbf{1}_T \in \{0, 1\}^m$  is the characteristic vector of the edge set of  $T$ . This is a special case of a more general problem considered in [20]: min-cost matroid base with packing constraints. Here we restrict attention to spanning trees (graphic matroid). We refer to this slightly more general problem also as CROSSING-MST.

Our first result is a near-linear time algorithm to approximately solve the underlying LP relaxation for CROSSING-MST. For a multigraph  $G$  we let  $\mathcal{T}(G)$  denote the set of all spanning trees of  $G$  and let  $\text{ST}(G)$  denote the spanning tree polytope of  $G$  (which is the convex hull of the characteristic vectors  $\{\mathbf{1}_T \mid T \in \mathcal{T}(G)\}$ ).

► **Theorem 1.** *Let  $G = (V, E)$  be a multigraph with  $m$  edges and  $n$  nodes and consider the linear program  $\min\{c^T x : Ax \leq b, x \in \text{ST}(G)\}$  where  $A \in [0, 1]^{k \times m}$ ,  $b \in [1, \infty)^k$ ,  $c \in [0, \infty)^m$ . Let  $N$  be the maximum of  $m$  and number of non-zeroes in  $A$ . There is a randomized polynomial time algorithm that for any given  $\epsilon \in (0, 1/2]$  runs in  $\tilde{O}(N/\epsilon^2)$  time and with high probability either correctly certifies that the LP is infeasible or outputs a solution  $y \in \text{ST}(G)$  such that  $c^T y \leq (1 + \epsilon)\text{OPT}$  and  $Ay \leq (1 + \epsilon)b$  where  $\text{OPT}$  is the minimum value of a feasible solution.*

► **Remark 2.** We describe a randomized algorithm for the sake of simplicity, however we believe that a deterministic algorithm with similar guarantees can be obtained via ideas in [16].

Solving the LP relaxation quickly enables to estimate the optimum integer solution *value* via existing rounding results [43, 20, 8, 7, 37, 36]. For instance, when specialized to BD-ST, we obtain a near-linear time algorithm to estimate the optimum value arbitrarily closely (modulo the additive 1).

► **Corollary 3.** *There is a randomized  $\tilde{O}(m/\epsilon^2)$ -time algorithm that outputs a value  $B$  such that  $B \leq B^* \leq \lceil (1 + \epsilon)B \rceil + 1$  where  $B^*$  is the minimum maximum degree over all spanning trees (that is,  $B^* = \min_{T \in \mathcal{T}(G)} \max_{v \in V} \deg_T(v)$  where  $\deg_T(v)$  is the degree of  $v$  in  $T$ ).*

Our second result shows the utility of the LP solution to sparsify the original graph  $G$ .

► **Theorem 4.** *Let  $x \in \text{ST}(G)$  be such that  $Ax \leq b$  for a matrix  $A \in [0, 1]^{k \times m}$  and  $b \in [1, \infty)^k$ . Consider a random subgraph  $G' = (V, E')$  of  $G$  obtained by picking each edge  $e \in G$  with probability  $\alpha_e := \min\{1, \frac{36 \log(k+m)}{\epsilon^2} \cdot x_e\}$ . Then with high probability the following hold: (i)  $|E'| = O(n \ln(k+m)/\epsilon^2)$  (ii) there exists a fractional solution  $z \in \text{ST}(G)$  in the support of  $G'$  such that  $Az \leq (1 + 3\epsilon)b$ .*

One can run a combinatorial algorithm such as the Fürer-Raghavchari algorithm [24] on the sparse graph rather than on the original graph  $G$ . This yields the following corollary which improves the  $\tilde{O}(mn)$  running time substantially when  $G$  is dense.

► **Corollary 5.** *There is a randomized algorithm for BD-ST that given a graph  $G$  on  $n$  nodes runs in  $\tilde{O}(n^2/\epsilon^2)$  time, and with high probability outputs a spanning tree  $T$  with maximum degree  $\lceil(1 + \epsilon)B^*\rceil + 2$  where  $B^*$  is the optimal degree bound.*

► **Remark 6.** Corollaries 3 and 5 can be generalized to the non-uniform degree version of BD-ST. Input is  $G$  and degree bounds  $B_v, v \in V$ , and the algorithm either decides that there is no spanning tree satisfying the degree bounds or outputs a tree that approximately satisfies them.

Our final result is a fast algorithm to round the LP solution. Several different rounding strategies have been developed for BD-MST and CROSSING-MST and they yield different guarantees and take advantage of the special structure of the given instance. Iterated rounding has been one of the primary and powerful techniques, however it requires basic feasible solutions to the LP relaxation; it seems far from obvious how to obtain fast algorithms with comparable guarantees and is a challenging open problem. We are here interested in *oblivious randomized rounding* strategies that take a point  $x \in \text{ST}(G)$  and round it to a random spanning tree  $T \in \mathcal{T}(G)$  such that the coordinates of the resulting random edge vector are *negatively correlated*<sup>1</sup>. Negative correlation implies concentration for linear constraints as shown by Panconesi and Srinivasan [38]. These strategies, when combined with the LP solution, yield bicriteria approximation algorithms for CROSSING-MST of the form  $(1 + \epsilon, \min\{O(\log k / \log \log k)b_i, (1 + \epsilon)b_i + O(\log k)/\epsilon^2\})$  where the first part is the approximation with respect to the cost and the second part with respect to the packing constraints. For CROSSING-ST and CROSSING-MST these are currently the best known approximation ratios (although special cases such as BD-MST admit much better bounds). Several dependent randomized rounding techniques achieving negative correlation in the spanning tree polytope are known: maximum entropy rounding [5], pipage rounding and *swap rounding* [20]. These rounding techniques generally apply to matroids and have several other applications. In this paper we show that given  $x \in \text{ST}(G)$ , one can swap-round  $x$  to a spanning tree in near-linear time provided it is given in an implicit fashion; alternately one can obtain an implicit *approximate* representation  $x'$  of  $x$  and then apply an efficient swap-rounding on  $x'$ . Since swap-rounding is a flexible procedure and does not generate a unique distribution, a precise technical statement requires more formal notation and we refer the reader to Section 3. Here we state a theorem in a general form so that it can be used in other contexts.

► **Theorem 7.** *Let  $G = (V, E)$  be a multigraph with  $m$  edges and let  $x \in [0, 1]^m$ . For any  $\epsilon \in (0, 1/2)$  there is a randomized algorithm that runs in  $\tilde{O}(m/\epsilon^2)$  time and either correctly decides that  $x \notin \text{ST}(G)$  or outputs a random vector  $T = (X_1, X_2, \dots, X_m) \in \{0, 1\}^m$  such that (i)  $T$  is the characteristic vector of a spanning tree of  $G$  (ii)  $\mathbb{E}[X_i] \leq (1 + \epsilon)x_i$  for  $1 \leq i \leq m$  and (iii)  $X_1, X_2, \dots, X_m$  are negatively correlated. In particular  $T$  is obtained as a swap-rounding of a vector  $y$  such that  $y \leq (1 + \epsilon)x$ .*

Combining Theorems 1 and 7 and existing results on swap rounding [20] we obtain the following. The approximation ratio matches the best known for CROSSING-MST and the algorithm runs in near-linear time.

<sup>1</sup> A collection of  $\{0, 1\}$  random variables  $X_1, X_2, \dots, X_r$  are negatively correlated if, for all subsets  $S \subseteq [r]$ ,  $\mathbb{E}[\prod_{i \in S} X_i] \leq \prod_{i \in S} \mathbb{E}[X_i]$  and  $\mathbb{E}[\prod_{i \in S} (1 - X_i)] \leq \prod_{i \in S} (1 - \mathbb{E}[X_i])$ .

► **Corollary 8.** *For the feasibility version of CROSSING-MST, there is a randomized algorithm that runs in near-linear time and outputs a spanning tree  $T$  such that*

$$A\mathbb{1}_T \leq \min\{O(\log k / \log \log k)b_i, (1 + \epsilon)b_i + O(\log k)/\epsilon^2\}$$

*with high probability. For the cost version of CROSSING-MST, there is a randomized algorithm that outputs a*

$$(1 + \epsilon, \min\{O(\log k / \log \log k)b_i, (1 + \epsilon)b_i + O(\log k)/\epsilon^2\})$$

*bicriteria approximation with probability  $\Omega(\epsilon)$ . After  $\tilde{O}(1/\epsilon)$  independent repetitions of this algorithm, we can obtain the same guarantees with high probability.*

Our algorithm, when specialized to BD-ST and BD-MST is more general than the one in [22] in terms of handling cost and non-uniform degrees. In addition we obtain a very close estimate of  $B^*$ , a much better dependence on  $\epsilon$ , and also obtain an approximation of the form  $O(\log n / \log \log n)B^*$  which is better than  $(1 + \epsilon)B^* + O(\log n)/\epsilon^2$  for small  $B^*$ .

We mainly focused on BD-MST and a high-level result for CROSSING-MST. One can obtain results for related problems that involve multiple costs, lower bounds in addition to upper bounds, and other applications of swap-roundings. We discuss these in more detail in Section A.

## 1.2 Overview of main ideas

Faster approximation algorithms for LPs that arise in combinatorial optimization have been developed via several techniques. We follow a recent line of work [16, 18, 39, 14] that utilizes features of the multiplicative weight update (MWU) method and data structures to speed up *implicit* LPs. In particular, the LP for CROSSING-MST that we seek to solve can be addressed by the randomized MWU algorithm from [18] and data structures for dynamic MST [29]. The overall approach follows some ideas from past work [16]. The sparsification result is inspired by recent applications of similar ideas [16, 15, 11] and utilizes Karger’s theorem on random sampling for packing disjoint bases in matroids [30].

Our main novel contribution is Theorem 7 which we believe is of independent interest beyond the applications outlined here. Dependent randomized rounding techniques have had many spectacular applications. In particular maximum entropy rounding in the spanning tree polytope gave a strong impetus to this line of work via its applications to ATSP [5] and metric-TSP [27]. Swap-rounding is a simpler scheme to describe and analyze, and suffices for several applications that only require negative correlation. However, all the known dependent rounding schemes are computationally expensive. Recent work has led to fantastic progress in sampling spanning trees [4], however the bottleneck for maximum entropy rounding is to compute, from a given point  $x \in \text{ST}(G)$ , the maximum entropy distribution with marginals equal to  $x$ ; polynomial time (approximation) algorithms exist for this [5, 44] but they are rather slow. Swap-rounding [20] requires one to decompose  $x \in \text{ST}(G)$  (or more generally a point in the matroid base polytope) into a convex combination of spanning trees; that is we write  $x = \sum_{T \in \mathcal{T}} \lambda_T \mathbb{1}_T$  such that  $\sum_T \lambda_T = 1$  and  $\lambda_T \geq 0, T \in \mathcal{T}$ . This is a non-trivial problem to do exactly. The starting point here is a theorem in [16] that shows that one can solve this decomposition problem *approximately* and deterministically in near-linear time via a reduction to the problem of spanning tree packing; this is done via MWU techniques. The near-linear time algorithm implies that any  $x \in \text{ST}(G)$  can be decomposed efficiently into an *implicit* convex decomposition of total size  $\tilde{O}(m/\epsilon^2)$  where  $\epsilon$  is the approximation parameter in the decomposition. To store the convex combination  $\sum_{i=1}^h \lambda_i \mathbb{1}_{T_i}$  implicitly, we store the

first tree  $T_1$  explicitly and to obtain  $T_{i+1}$  from  $T_i$  for  $i \in [h-1]$ , we store the edges in the symmetric difference of  $T_{i+1}$  and  $T_i$ . The size of the decomposition is then the sum of the sizes of the symmetric differences and the size of  $T_1$ . We give a more formal definition of an implicit decomposition in Section 3.2. We show in this paper that this implicit sparse decomposition is well-suited to the swap-rounding algorithm. We employ a divide-and-conquer strategy with appropriate tree data structures to obtain an implementation that is near-linear in the size of the implicit decomposition. Putting these ingredients together yields our result.<sup>2</sup>

The seemingly fortuitous connection between the MWU based algorithm for packing spanning trees and its implicit representation leading to a fast algorithm for swap-rounding is yet another illustration of the synergy between tools coming together in the design of fast algorithms.

### 1.3 Other related work

We overview some known results on CROSSING-ST and CROSSING-MST and special cases. BD-MST can be viewed as a special case of CROSSING-MST where each edge participates in 2 constraints. Bansal et al. [8] showed that if each edge participates in at most  $\Delta$  constraints of  $A$  (and  $A$  is a binary matrix) then one can obtain a  $(1, b + \Delta - 1)$ -approximation generalizing the BD-MST result; this was further extended to matroids by Lau, Kiraly and Singh [33]. It is shown in [7] that for CROSSING-ST one cannot obtain a purely additive approximation better than  $O(\sqrt{n})$  via the natural LP relaxation. For this they use a reduction from discrepancy minimization; it also implies, via the hardness result in [12] for discrepancy, that it is NP-Hard to obtain a purely additive  $o(\sqrt{n})$  bound. Bansal et al. [7] consider the *laminar* case of CROSSING-MST where the cuts form a laminar family and obtained a  $(1, b + O(\log n))$  approximation via iterative rounding (this problem generalizes BD-MST). Olver and Zenklusen [37] consider chain-constrained CROSSING-ST which is a further specialization when the laminar family is a chain (a nested family of cuts). For this special case they obtained an  $O(1)$ -factor approximation in the unit cost setting; Linhares and Swamy [36] considered the min-cost version and obtained an  $(O(1), O(1))$ -approximation. [37] also showed that even in the setting of chain-constrained CROSSING-ST, it is NP-Hard to obtain a purely additive bound better than  $c \log n / \log \log n$  for some fixed constant  $c$ .

Dependent randomized rounding has been an active area of research with many applications. Pipage rounding, originally developed by Ageev and Sviridenko [1] in a deterministic way, was generalized to the randomized setting by Srinivasan [45] and by Gandhi et al. [25] and [10, 20] and has led to a number of applications. Maximum entropy rounding satisfies additional properties beyond negative correlation and this is important in applications to metric-TSP (see [27] and very recent work [31, 32]). There has been exciting recent progress on sampling spanning trees and bases in matroids and we refer the reader to some recent work [41, 3, 4] for further pointers. Concentration bounds via dependent rounding can also be obtained without negative correlation (see [21] for instance) and recent work of Bansal [6] combines iterative rounding with dependent rounding in a powerful way.

---

<sup>2</sup> In an earlier version of the paper (see [19]) we described our fast swap rounding using two ideas. The first was a fast near-linear time algorithm to merge two spanning trees using the link-cut tree data structure. We were unaware of prior work of Ene and Nguyễn [23] that had already given such an algorithm in the context of fast algorithms for submodular function maximization in graphic matroids. In this version of the paper we use their algorithm as a black box. We focus on our second idea which exploits the implicit representation. We thank Alina Ene and Huy Nguyễn for pointing out to us their fast algorithm for merging two trees.



Fast approximation algorithms for solving positive LPs and SDPs has been an extensive area of research starting from the early 90s. Lagrangean relaxation techniques based on MWU and other methods have been extensively studied in the past, and continue to provide new insights and results for both explicit and implicit problems. Recent work based on a convex optimization perspective has led to a number of new results and improvements. It is infeasible to do justice to this extensive research area and we refer the reader to two recent PhD theses [40, 48]. Spectacular advances in fast algorithms based on the Laplacian paradigm, interior point methods, cutting plane methods, spectral graph theory, and several others have been made in the recent past and is a very active area of research with frequent ongoing developments.

## Organization

Section 2 introduces some relevant notation, technical background and tree data structures that we rely on. Section 3 describes our fast swap-rounding algorithm and proves Theorem 7. Section 4 describes the sparsification process of Theorem 4. Section 5 discusses the LP relaxation for CROSSING-ST and Theorem 1. Section A brings together results from previous sections to prove some of the corollaries stated in the introduction and provides details of some extensions and related problems.

## 2 Preliminaries and notation

For a set  $S$ , we use the convenient notation  $S - i$  to denote  $S \setminus \{i\}$  and  $S + i$  to denote  $S \cup \{i\}$ .

### Matroids

We discuss some basics of matroids to establish some notation as well as present some useful lemmas that will be used later. A *matroid*  $\mathcal{M}$  is a tuple  $(N, \mathcal{I})$  with  $\mathcal{I} \subseteq 2^N$  satisfying the following three properties: (1)  $\emptyset \in \mathcal{I}$ , (2) if  $A \in \mathcal{I}$  and  $B \subseteq A$ , then  $B \in \mathcal{I}$ , and (3) if  $A, B \in \mathcal{I}$  such that  $|A| < |B|$  then there exists  $b \in B \setminus A$  such that  $A + b \in \mathcal{I}$ . We refer to the sets in  $\mathcal{I}$  as *independent sets* and say that maximal independent sets are *bases*. The *rank* of  $\mathcal{M}$  is the size of a base. For a set  $A \subseteq 2^N$ , we refer to  $r_{\mathcal{M}}(A) = \max\{|S| : S \subseteq A, S \in \mathcal{I}\}$  as the *rank* of  $A$ .

A useful notion that we utilize in our fast implementation of swap rounding is that of contraction of a matroid. We say that the *contraction* of  $e$  in  $\mathcal{M}$  results in the matroid  $\mathcal{M}/e = (N - e, \{I \subseteq N - e : I + e \in \mathcal{I}\})$  if  $r_{\mathcal{M}}(\{e\}) = 1$  and  $\mathcal{M}/e = (N - e, \{I \subseteq N - e : I \in \mathcal{I}\})$  if  $r_{\mathcal{M}}(\{e\}) = 0$ . This definition extends naturally to contracting subsets  $A \subseteq N$ . It can be shown that contracting the elements of  $A$  in any order results in the same matroid, which we denote as  $\mathcal{M}/A$ .

The following statements are standard results in the study of matroids (e.g. see [42]). The following theorem is important in the analysis of swap rounding. It is often called the strong base exchange property of matroids.

► **Theorem 9.** *Let  $\mathcal{M} = (N, \mathcal{I})$  be a matroid and let  $B, B'$  be bases. For  $e \in B \setminus B'$ , there exists  $e' \in B' \setminus B$  such that  $B - e + e' \in \mathcal{I}$  and  $B' - e' + e \in \mathcal{I}$ .*

The next lemma shows that if one contracts elements of an independent set in a matroid, bases in the contracted matroid can be used to form bases in the initial matroid.

► **Lemma 10.** *Let  $\mathcal{M} = (N, \mathcal{I})$  be a matroid and let  $A \in \mathcal{I}$ . Let  $B_A$  be a base in  $\mathcal{M}/A$ . Then  $A \cup B_A$  is a base in  $\mathcal{M}$ .*

### A forest data structure

We need a data structure to represent a forest that supports the necessary operations we need to implement randomized swap rounding in Section 3. The data structure mainly needs to facilitate the contraction of edges, including being able to recover the identity of the original edges after any number of contractions. We enable this by enforcing that when the data structure is initialized, every edge  $e$  is accompanied with a unique identifier. This identifier will be associated with the edge regardless of the edge's endpoints changing due to contraction. The implementation of this step is important to guarantee a fast running time.

The data structure is initialized via the function `init`, which takes as input the vertices, edges, and unique edge identifiers of the forest. `init` initializes an adjacency list  $A$ , stores a mapping  $f$  of edges to their unique edge identifiers, and creates a disjoint-set data structure  $R$  where every vertex initially is in its own set. The operation `contract` contracts an edge  $uv$  in the forest by identifying the vertices  $u$  and  $v$  as the same vertex. This requires choosing  $u$  or  $v$  to be the new representative (suppose we choose  $u$  without loss of generality), merging the sets corresponding to  $u$  and  $v$  in  $R$  while making  $u$  the representative of the set in  $R$ , and modifying the adjacency list  $A$  to reflect the changes corresponding to contracting  $uv$  and making  $u$  the representative. After an edge is contracted, the vertex set changes. We need to support the ability to obtain the edge identifier of an edge in the contracted forest. The data structure maintains  $f$  under edge contractions and returns unique identifiers with the operation `orig-edge`. Given an edge  $uv$  that was in the contracted forest at some point in the past, we also need to support the ability to obtain the edge in the vertex set of the current contracted forest. We do this using  $R$ , which stores all of the vertices that have been contracted together in disjoint sets. This operation is supported by the operation `represented-edge`. Finally, we can copy the graph via the operation `copy`, which simply enumerates over the vertices, edges, and stored unique identifiers of the edges to create a new data structure.

The following lemma formalizes the preceding description of the data structure. We leave the formal details of a specific implementation and the proof of the following lemma for the full version.

► **Lemma 11.** *Let  $F = (V, E)$  be a forest and for all  $e \in E$ , let  $id(e)$  be the unique identifier of  $e$ . The data structure can be initialized via a call to `init` in  $\tilde{O}(|V|)$  time. For  $i = 0, 1, \dots, k$ , let  $F_i = (V_i, E_i)$  be the forest after  $i$  calls to `contract`, so  $F_0 = F$  and  $F_k$  is the current state of the forest. The data structure supports the following operations.*

- `orig-edge( $e$ )`: *input is an edge  $e \in E_k$ . Output is the identifier  $id(e)$  that was provided when  $e$  was added to the data structure. Running time is  $\tilde{O}(1)$ .*
- `represented-edge( $e$ )`: *input is two vertices  $u, v \in V_i$  for some  $i = 0, 1, \dots, k$ . Output is the pair  $\{u_r, v_r\}$ , where  $u_r$  and  $v_r$  are the vertices in  $V_k$  that correspond to  $u$  and  $v$  in the contracted forest  $F_k$ . Running time is  $O(1)$ .*
- `contract( $uv, z \in \{u, v\}$ )`: *input is two vertices  $u, v \in V_k$ . The operation contracts  $uv$  in  $E_k$ , setting the new vertex in  $F_{k+1} = (V_{k+1}, E_{k+1})$  to be  $\{u, v\} \setminus \{z\}$ . The amortized running time is  $\tilde{O}(\deg_{F_k}(z))$ .*
- `copy()`: *output is a forest data structure with vertices  $V_k$  and edges  $E_k$  along with the stored edge identifiers. Running time is  $\tilde{O}(|V_k|)$ .*

<pre> merge-bases(<math>\delta, B, \delta', B'</math>)   while <math>B \setminus B' \neq \emptyset</math> do     <math>e \leftarrow</math> arbitrary element of <math>B \setminus B'</math>     <math>e' \leftarrow</math> element of <math>B' \setminus B</math> such that <math>B - e + e' \in \mathcal{I}</math> and <math>B' - e' + e \in \mathcal{I}</math>     <math>b \leftarrow 1</math> with probability <math>\frac{\delta}{\delta + \delta'}</math> and 0 otherwise     if <math>b = 1</math> then       <math>B \leftarrow B - e + e'</math>     else       <math>B' \leftarrow B' - e' + e</math>     end if   end while   return <math>B</math> </pre>
<pre> swap-round(<math>\delta_1, B_1, \dots, \delta_h, B_h</math>)   <math>C_1 \leftarrow B_1</math>   for <math>k</math> from 1 to <math>h - 1</math> do     <math>C_{k+1} \leftarrow</math> merge-bases(<math>\sum_{i=1}^k \delta_i, C_k, \delta_{k+1}, B_{k+1}</math>)   end for   return <math>C_h</math> </pre>

■ **Figure 1** The randomized swap rounding algorithm from [20].

### 3 Fast swap rounding in the spanning tree polytope

Randomized swap rounding, developed in [20], is a dependent rounding scheme for rounding a fractional point  $x$  in the base polytope of a matroid to a random base  $X$ . The rounding preserves expectation in that  $\mathbb{E}[X] = x$ , and more importantly, the coordinates of  $X$  are negatively correlated. In this section we prove Theorem 7 on a fast algorithm for swap-rounding in the spanning tree polytope. We begin by describing swap-rounding.

#### 3.1 Randomized swap rounding

Let  $\mathcal{M} = (N, \mathcal{I})$  be a matroid and let  $\mathcal{P}$  be the base polytope of  $\mathcal{M}$  (convex hull of the characteristic vectors of the bases of  $\mathcal{M}$ ). Any  $x \in \mathcal{P}$  can be written as a finite convex combination of bases:  $x = \sum_{i=1}^h \delta_i \mathbb{1}_{B_i}$ . Note that this combination is not necessarily unique. As in [20], we give the original algorithm for randomized swap rounding via two routines. The first is `merge-bases`, which takes as input two bases  $B, B'$  and two real values  $\delta, \delta' \in (0, 1)$ . If  $B = B'$  the algorithm outputs  $B$ . Otherwise the algorithm finds a pair of elements  $e, e'$  such that  $e \in B \setminus B'$  and  $e' \in B' \setminus B$  where  $B - e + e' \in \mathcal{I}$  and  $B' - e' + e \in \mathcal{I}$ . For such  $e$  and  $e'$ , we say that they are a *valid exchange pair* and that we *swap*  $e$  with  $e'$ . The existence of such elements is guaranteed by the strong base exchange property of matroids in Theorem 9. The algorithm randomly retains  $e$  or  $e'$  in both bases with appropriate probability and this increases the intersection size of  $B$  and  $B'$ . The algorithm repeats this process until  $B = B'$ . The overall algorithm `swap-round` utilizes `merge-bases` as a subroutine and repeatedly merges the bases until only one base is left. A formal description is in Figure 1 along with the pseudocode for `merge-bases`.

It is shown in [20] that swap-rounding generates a random base/extreme point  $X \in \mathcal{P}$  (note that the extreme points of  $\mathcal{P}$  are characteristic vectors of bases) such that  $\mathbb{E}[X] = x$  and the coordinates  $X_1, X_2, \dots, X_n$  (here  $|N| = n$ ) are negatively correlated. We observe

that swap-rounding does not lead to a unique probability distribution on the bases (that depends only  $x$ ). First, as we already noted, the convex decomposition of  $x$  into bases is not unique. Second, both **merge-bases** and **swap-round** are non-deterministic in their choices of which element pairs to swap and in which order to merge bases in the convex decomposition. The key property for negative correlation, as observed in [20], is to view the generation of the final base  $B$  as a vector-valued martingale (which preserves expectation in each step) that changes only two coordinates in each step. Another rounding strategy, namely pipage rounding, also enjoys this property. Nevertheless swap-rounding is a meta algorithm that has certain clearly defined features. The flexibility offered by **merge-bases** and **swap-round** are precisely what allow for faster implementation in specific settings.

We say that  $\bar{B} \stackrel{d}{=} \text{merge-bases}(\delta, B, \delta', B')$  if for some non-deterministic choice of valid exchange pairs in the algorithm,  $\bar{B}$  is the random output of **merge-bases** $(\delta, B, \delta', B')$ . Similarly we say that  $B \stackrel{d}{=} \text{swap-round}(\delta_1, B_1, \dots, \delta_h, B_h)$  if  $B$  is the random output of the **swap-round** process for some non-deterministic choice of the order in which bases are merged and some non-deterministic choices in the merging of bases. It follows from [20] that if  $B \stackrel{d}{=} \text{swap-round}(\delta_1, B_1, \dots, \delta_h, B_h)$  then  $B$  satisfies the property that  $\mathbb{E}[B] = x$  and coordinates of  $B$  are negatively correlated.

### 3.2 Setup for fast implementation in graphs

Let  $G = (V, E)$  be a multigraph with  $|V| = n$  and  $|E| = m$  and let  $x \in \text{ST}(G)$  be a fractional spanning tree. Swap rounding requires decomposing  $x$  into a convex combination of spanning trees. This step is itself non-trivial; existing algorithms have a high polynomial dependence on  $n, m$ . Instead we will settle for an *approximate* decomposition that has some very useful features. We state a theorem (in fact a corollary of a theorem) from [16] in a slightly modified form suitable for us.

► **Theorem 12** (Paraphrase of Corollary 1.2 in [16]). *Given a graph  $G = (V, E)$  with  $n = |V|$  and  $m = |E|$  and a rational vector  $x \in [0, 1]^m$  there is a deterministic polynomial-time algorithm that runs in  $\tilde{O}(m/\epsilon^2)$  time and either correctly reports that  $x \notin \text{ST}(G)$  or outputs an implicit convex decomposition of  $z$  into spanning trees such that  $z \leq (1 + \epsilon)x$ .*

The MWU algorithm behind the preceding theorem outputs a convex decomposition of  $z = \sum_{i=1}^h \delta_i \mathbf{1}_{T_i}$  for  $h = \tilde{O}(m/\epsilon^2)$  but in an implicit fashion. It outputs  $T = T_1$  and a sequence of tuples  $(\delta_i, E_i, E'_i)$  where  $T_{i+1} = T_i - E_i + E'_i$  for  $1 \leq i < h$  and has the property that  $\sum_{i=1}^{h-1} (|E_i| + |E'_i|) = \tilde{O}(m/\epsilon^2)$ . Thus the convex decomposition of  $z$  is rather sparse and near-linear in  $m$  for any fixed  $\epsilon > 0$ . We will take advantage of this and swap-round  $z$  via this implicit convex decomposition. For many applications of interest, including **CROSSING-MST**, the fact that we randomly round  $z$  instead of  $x$  does not make much of a difference in the overall approximation since  $x$  itself in our setting is the output of an approximate LP solver.

► **Remark 13.** The output of the approximate LP solver based on MWU for **CROSSING-MST** has the implicit decomposition as outlined in the preceding paragraph. However, for the sake of a self-contained result as stated in Theorem 7, we use the result from [16] which also has the advantage of being deterministic.

The rest of the section describes a fast implementation for **swap-round**. The algorithm is based on a divide and conquer strategy for implementing **swap-round** when the convex combination is described in an implicit and compact fashion. An important ingredient is a fast black-box implementation of **merge-bases**. For this we use the following result; as we remarked earlier, an earlier version of this paper obtained a similar result.

► **Theorem 14** (Ene and Nguyễn [23]). *Let  $T$  and  $T'$  be spanning trees of a graph  $G = (V, E)$  with  $|V| = n$  and  $E = T \cup T'$  and let  $\delta, \delta' \in (0, 1)$ . There exists an algorithm `fast-merge` such that  $\text{fast-merge}(\delta, T, \delta', T') \stackrel{d}{=} \text{merge-bases}(\delta, T, \delta', T')$  and the call to `fast-merge` runs in  $O(n \log^2 n)$  time.*

### 3.3 Fast implementation of swap-round

In this subsection the goal is to prove the following theorem.

► **Theorem 15.** *Let  $\sum_{i=1}^h \delta_i \mathbb{1}_{T_i}$  be a convex combination of spanning trees of the graph  $G = (V, E)$  where  $n = |V|$ . Let  $T$  be a spanning tree such that  $T = T_1$  and let  $\{(E_i, E'_i)\}_{i=1}^{h-1}$  be a sequence of sets of edges such that  $T_{i+1} = T_i - E_i + E'_i$  for all  $i \in [h-1]$  and  $E_i \cap E'_i = \emptyset$  for all  $i \in [h-1]$ . Then there exists an algorithm that takes as input  $T$ ,  $\{(E_i, E'_i)\}_{i=1}^{h-1}$ , and  $\{\delta_i\}_{i=1}^h$  and outputs a tree  $T_S$  such that  $T_S \stackrel{d}{=} \text{swap-round}(\delta_1, T_1, \dots, \delta_h, T_h)$ . The running time of the algorithm is  $\tilde{O}(n + \gamma)$  time where  $\gamma = \sum_{i=1}^{h-1} (|E_i| + |E'_i|)$ .*

#### A divide and conquer approach

We consider the swap rounding framework in the setting of arbitrary matroids for simplicity. We work with the implicit decomposition of the convex combination of bases  $\sum_{i=1}^h \delta_i \mathbb{1}_{B_i}$  of the matroid  $\mathcal{M} = (N, \mathcal{I})$ , as described in Theorem 15. That is, the input is a base  $B$  such that  $B = B_1$ , a sequence of sets of elements  $\{(E_i, E'_i)\}_{i=1}^{h-1}$  such that  $B_{i+1} = B_i - E_i + E'_i$  and  $E_i \cap E'_i = \emptyset$  for all  $i \in [h-1]$ , and the sequence of coefficients  $\{\delta_i\}_{i=1}^h$ .

The pseudocode for our divide and conquer algorithm `divide-and-conquer-swap` is given in Figure 2. The basic idea is simple. We *imagine* constructing an explicit convex decomposition  $B_1, B_2, \dots, B_h$  from the implicit one. The high-level idea is to recursively apply swap rounding to  $B_1, \dots, B_{h/2}$  to create a base  $B$ , and similarly create a base  $B'$  by recursively applying swap rounding to  $B_{h/2+1}, \dots, B_h$ , and then merging  $B$  and  $B'$ . The advantage of this approach is manifested in the implicit case. To see this, we observe that in  $\text{merge-bases}(\delta, B, \delta', B')$ , the intersection  $B \cap B'$  is always in the output, and this implies that the intersection  $\bigcap_{i=1}^h B_i$  will always be in the output of  $\text{swap-round}(\delta_1, B_1, \dots, \delta_h, B_h)$ . Therefore, at every recursive level, we simply contract the intersection prior to merging any bases. Note that this is slightly complicated by the fact that the input is an implicit representation. However, we note that  $B \cap \bigcup_{i=1}^{h-1} (E_i \cup E'_i) = B \setminus \bigcap_{i=1}^h B_i$  as  $E_i \cup E'_i = B_i \Delta B_{i+1}$  for all  $i \in [h-1]$  where  $S \Delta U$  denotes the symmetric difference of the sets  $S, U$  (see full version for more details). (We note later how the contraction of elements helps in the running time when specializing to the graphic matroid.) After contracting the intersection, the algorithm recursively calls itself on the first  $h/2$  bases and the second  $h/2$  bases, then merges the output of the two recursive calls via `merge-bases`. With the given implicit representation, this means that the input to the first recursive call is  $B_1, \{(E_i, E'_i)\}_{i=1}^{h/2-1}, \{\delta_i\}_{i=1}^{h/2}$  and the input to the second recursive call is  $B_{h/2+1}, \{(E_i, E'_i)\}_{i=h/2+1}^{h-1}, \{\delta_i\}_{i=h/2+1}^h$  (note we can easily construct  $B_{h/2+1}$  via the implicit representation). The underlying matroid in the call to `merge-bases` is the matroid  $\mathcal{M}$  with the intersection  $\bigcap_{i=1}^h B_i$  contracted.

The following lemma shows that `divide-and-conquer-swap` is a proper implementation of `swap-round`. We leave the details of the proof for the full version of the paper.

► **Lemma 16.** *Let  $\sum_{i=1}^h \delta_i \mathbb{1}_{B_i}$  be a convex combination of bases in the matroid  $\mathcal{M}$  and  $\{(E_i, E'_i)\}_{i=1}^{h-1}$  be a sequence of elements such that  $B_{i+1} = B_i - E_i + E'_i$  and  $E_i \cap E'_i = \emptyset$  for all  $i$ . Then  $\text{swap-round}(\delta_1, B_1, \dots, \delta_h, B_h) \stackrel{d}{=} \text{divide-and-conquer-swap}(B_1, \{(E_i, E'_i)\}_{i=1}^{h-1}, \{\delta_i\}_{i=1}^h)$ .*

```

divide-and-conquer-swap( $B, \{(E_i, E'_i)\}_{i=s}^{t-1}, \{\delta_i\}_{i=s}^t$ )
  if  $s = t$  then
    return  $B$ 
  end if
   $\ell \leftarrow \max \left\{ \ell' \in \{s, s+1, \dots, t\} : \sum_{i=s}^{\ell'-1} |E_i| \leq \frac{1}{2} \sum_{i=s}^{t-1} |E_i| \right\}$ 
   $\hat{B} \leftarrow B \cap \bigcup_{i=s}^{\ell-1} (E_i \cup E'_i)$ 
   $\hat{B}_C \leftarrow \hat{B}$ 
  for  $i$  from  $s$  to  $\ell$  do
     $\hat{B}_C \leftarrow \hat{B}_C - E_i + E'_i$ 
  end for
   $\hat{B}_L \leftarrow \text{divide-and-conquer-swap}(\hat{B}, \{(E_i, E'_i)\}_{i=s}^{\ell-1}, \{\delta_i\}_{i=s}^{\ell})$ 
   $\hat{B}_R \leftarrow \text{divide-and-conquer-swap}(\hat{B}_C, \{(E_i, E'_i)\}_{i=\ell+1}^{t-1}, \{\delta_i\}_{i=\ell+1}^t)$ 
   $\hat{B}_M \leftarrow \text{merge-bases}(\sum_{i=s}^{\ell} \delta_i, \hat{B}_L, \sum_{i=\ell+1}^t \delta_i, \hat{B}_R)$ 
  return  $\hat{B}_M \cup (B \setminus \bigcup_{i=s}^{\ell-1} (E_i \cup E'_i))$ 

```

■ **Figure 2** A divide-and-conquer implementation of swap rounding with an implicit representation.

### A fast implementation of divide-and-conquer-swap for spanning trees

The pseudocode for our fast implementation `fast-swap` of `divide-and-conquer-swap` is given in Figure 4.

As in `divide-and-conquer-swap`, the algorithm `fast-swap` contracts the intersection of the input. Suppose we contract the intersection  $\bigcap_{i=1}^h T_i$  in  $T_j$  and call this contracted tree  $\hat{T}_j$ . Then  $|\hat{T}_j| \leq |T_j \setminus \bigcap_{i=1}^h T_i|$ . A simple argument shows that  $T_j \setminus \bigcap_{i=1}^h T_i \subseteq \bigcup_{i=1}^{h-1} (T_i \Delta T_{i+1}) = \bigcup_{i=1}^{h-1} (E_i \cup E'_i)$  (see full version for more details). Thus, the size of the contracted tree is bounded by the size of the implicit representation  $\gamma := \sum_{i=1}^{h-1} |E_i| + |E'_i|$ . One can write a convex combination of bases in any matroid using the implicit representation, and contraction could even be implemented quickly as is the case in the graphic matroid. The main point for improving the running time is having an implementation of `merge-bases` that runs in time proportional to the size of the *contracted* matroid. This is key to the speedup achieved for the graphic matroid. `fast-merge` runs in time proportional to the size of the input trees, which have been contracted to have size  $O(\min\{n, \gamma\})$ , which yields a running time of  $\tilde{O}(\min\{n, \gamma\})$ . This speedup at every recursive level combined with the divide-and-conquer approach of `fast-swap` is sufficient to achieve a near-linear time implementation of `swap-round`.

Recall that as we are working with contracted trees, an edge in the contracted trees might have different endpoints than it did in the initial trees. The identifiers of edges do not change, regardless of whether the endpoints of the edge change due to contraction of edges in a tree. We therefore will refer to id's of edges throughout the algorithm `fast-swap` to work from contracted edges back to edges in the initial trees. This extra bookkeeping will mainly be handled implicitly.

Contraction of the intersection of the input trees in `fast-swap` using only the implicit representation is handled by the algorithm `shrink-intersection` and we give the pseudocode in Figure 3. Consider spanning trees  $T_s, T_{s+1}, \dots, T_t$ . The input to `shrink-intersection` is  $T_s$  and a sequence of sets of edges  $\{(E_i, E'_i)\}_{i=s}^{t-1}$  such that  $T_{i+1} = T_i - E_i + E'_i$  and  $E_i \cap E'_i = \emptyset$  for  $i \in \{s, s+1, \dots, t-1\}$ . Then `shrink-intersection` contracts  $\bigcap_{i=s}^t T_i$  in  $T_s$ . It is then easy to see that one can compute the intersection via the edges  $\{(E_i, E'_i)\}_{i=s}^{t-1}$  as  $\bigcap_{i=s}^t T_i = T_s \setminus \bigcup_{i=s}^{t-1} (E_i \cup E'_i)$  (see full version for more details). Let  $\hat{T}_s$  be  $T_s$  with  $\bigcap_{i=s}^t T_i$

```

shrink-intersection( $T, \{(E_i, E'_i)\}_{i=s}^{t-1}$ )
 $\hat{T} \leftarrow T.\text{copy}()$ 
for  $e \in T \setminus \bigcup_{i=s}^{t-1} (E_i \cup E'_i)$  do
     $uv \leftarrow \hat{T}.\text{represented-edge}(e)$ 
    assume  $\deg_{\hat{T}}(u) \leq \deg_{\hat{T}}(v)$  (otherwise rename)
     $\hat{T}.\text{contract}(uv, u)$ 
end for
let  $id(e)$  denote the unique identifier of an edge  $e$ 
for  $i$  from  $s$  to  $t - 1$  do
     $\hat{E}_i \leftarrow \bigcup_{e \in E_i} (\hat{T}.\text{represented-edge}(e), id(e))$ 
     $\hat{E}'_i \leftarrow \bigcup_{e \in E'_i} (\hat{T}.\text{represented-edge}(e), id(e))$ 
end for
return  $(\hat{T}, \{(\hat{E}_i, \hat{E}'_i)\}_{i=s}^{t-1})$ 

```

■ **Figure 3** A subroutine used in our fast implementation **fast-swap** of randomized swap rounding; used to implicitly contract the trees of the given convex combination.

contracted. The vertex set of  $\hat{T}_s$  is different than the vertex set of  $T_s$ . Then as the sets of edges  $E_i$  and  $E'_i$  for all  $i$  are defined on the vertices in  $T_s$ , we need to map the endpoints of edges in  $E_i$  to the new vertex set of  $\hat{T}_s$ . Using the data structure presented in Lemma 11, this is achieved using the operations **represented-edge** and **orig-edge**, which handle mapping the edge to its new endpoints and maintaining the edge identifier, respectively.

The following lemma shows that **shrink-intersection** indeed contracts the intersection of the trees via the implicit representation. We leave the details of the proof for the full version.

► **Lemma 17.** *Let  $T_1, \dots, T_h$  be spanning trees and let  $\{(E_i, E'_i)\}_{i=1}^{h-1}$  be a sequence of edge sets defined on the same vertex set such that  $T_{i+1} = T_i - E_i + E'_i$  and  $E_i \cap E'_i = \emptyset$  for all  $i \in [h-1]$ . Contract  $\bigcap_{i=1}^h T_i$  in  $T_1, \dots, T_h$  to obtain  $\hat{T}_1, \dots, \hat{T}_h$ , respectively.*

*Let  $n_{T_1} = |T_1|$  and  $\gamma = \sum_{i=1}^{h-1} (|E_i| + |E'_i|)$ . Then **shrink-intersection** $(T_1, \{(E_i, E'_i)\}_{i=1}^{h-1})$  runs in time  $\tilde{O}(n_{T_1} + \gamma)$  and outputs  $(\hat{T}, \{(\hat{E}_i, \hat{E}'_i)\}_{i=1}^{h-1})$  where  $\hat{T} = \hat{T}_1$  and  $\hat{T}_{i+1} = \hat{T}_i - \hat{E}_i + \hat{E}'_i$  for all  $i \in [h-1]$ . Moreover,  $|E_i| = |\hat{E}_i|$  and  $|E'_i| = |\hat{E}'_i|$  for all  $i \in [h-1]$  and  $|\hat{T}| \leq \min\{n_{T_1}, \gamma\}$ .*

We use the algorithm in Theorem 14 for **merge-bases**. In **fast-swap**, the two trees that are merged  $\hat{T}_L$  and  $\hat{T}_R$  are the return values of the two recursive calls to **fast-swap**. The algorithm at this point has explicit access to the adjacency lists of both  $\hat{T}_L$  and  $\hat{T}_R$ , which are used as input to the algorithm **fast-merge**. The output of **fast-merge** will be the outcome of merging the two trees  $\hat{T}_L$  and  $\hat{T}_R$ , which are edges of potentially contracted trees from the original convex combination. We can use the operation **orig-edge** of the forest data structure of Lemma 11 for  $\hat{T}_L$  and  $\hat{T}_R$  to obtain the edges from the trees of the original convex combination. This extra bookkeeping will be handled implicitly.

We next prove that **fast-swap** is implementing **swap-round** and that it runs in near-linear time.

► **Lemma 18.** *Let  $\sum_{i=1}^h \delta_i \mathbb{1}_{T_i}$  be a convex combination of spanning trees of the graph  $G = (V, E)$  where  $n = |V|$ . Let  $T$  be a spanning tree such that  $T = T_1$  and let  $\{(E_i, E'_i)\}_{i=1}^{h-1}$  be a sequence of sets of edges such that  $T_{i+1} = T_i - E_i + E'_i$  and  $E_i \cap E'_i = \emptyset$  for all  $i \in [h-1]$ .*

*Then **fast-swap** $(T, \{(E_i, E'_i)\}_{i=1}^{h-1}, \{\delta_i\}_{i=1}^h) \stackrel{d}{=} \text{swap-round}(\delta_1, T_1, \dots, \delta_h, T_h)$  and the call to **fast-swap** runs in  $\tilde{O}(n_T + \gamma)$  time where  $n_T = |T|$  and  $\gamma = \sum_{i=1}^{h-1} (|E_i| + |E'_i|)$ .*

```

fast-swap( $T, \{(E_i, E'_i)\}_{i=s}^{t-1}, \{\delta_i\}_{i=s}^t$ )
  if  $s = t$  then
    return  $T$ 
  end if
   $\ell \leftarrow \max \left\{ \ell' \in \{s, s+1, \dots, t\} : \sum_{i=s}^{\ell'-1} |E_i| \leq \frac{1}{2} \sum_{i=s}^{t-1} |E_i| \right\}$ 
   $(\hat{T}, \{(\hat{E}_i, \hat{E}'_i)\}_{i=s}^{t-1}) \leftarrow \text{shrink-intersection}(T, \{(E_i, E'_i)\}_{i=s}^{t-1})$ 
   $\hat{E}_C \leftarrow E(\hat{T})$ 
  for  $i$  from  $s$  to  $\ell$  do
     $\hat{E}_C \leftarrow \hat{E}_C - \hat{E}_i + \hat{E}'_i$ 
  end for
   $\hat{T}_C \leftarrow \text{init}(V(\hat{T}), \hat{E}_C)$ 
   $\hat{T}_L \leftarrow \text{fast-swap}(\hat{T}, \{(\hat{E}_i, \hat{E}'_i)\}_{i=s}^{\ell-1}, \{\delta_i\}_{i=s}^{\ell})$ 
   $\hat{T}_R \leftarrow \text{fast-swap}(\hat{T}_C, \{(\hat{E}_i, \hat{E}'_i)\}_{i=\ell+1}^{t-1}, \{\delta_i\}_{i=\ell+1}^t)$ 
   $\hat{T}_M \leftarrow \text{fast-merge}(\sum_{i=s}^{\ell} \delta_i, \hat{T}_L, \sum_{i=\ell+1}^t \delta_i, \hat{T}_R)$ 
  return  $\hat{T}_M \cup (T \setminus \bigcup_{i=s}^{t-1} (E_i \cup E'_i))$ 
    
```

■ **Figure 4** A fast implementation of randomized swap rounding from [20].

**Proof.** One can immediately see `fast-swap` is an implementation of `divide-and-conquer-swap`. There are some bookkeeping details that are left out of the implementation, such as maintaining the set of edges returned by `fast-merge`, but these are easily handled. Lemma 16 shows that  $\text{divide-and-conquer-swap}(\delta_1, T_1, \dots, \delta_h, T_h) \stackrel{d}{=} \text{swap-round}(\delta_1, T_1, \dots, \delta_h, T_h)$ , implying  $\text{fast-swap}(T, \{(E_i, E'_i)\}_{i=1}^{h-1}, \{\delta_i\}_{i=1}^h) \stackrel{d}{=} \text{swap-round}(\delta_1, T_1, \dots, \delta_h, T_h)$ .

Now we prove the running time bound. Let  $R(n_T, \gamma)$  denote the running time of the call to `fast-swap`( $T, \{(E_i, E'_i)\}_{i=1}^{h-1}, \{\delta_i\}_{i=1}^h$ ). By Lemma 17, the running time of the call to `shrink-intersection` is  $\tilde{O}(n_T + \gamma)$ . Let  $(\hat{T}, \{(\hat{E}_i, \hat{E}'_i)\}_{i=1}^{h-1})$  be the output of `shrink-intersection`. Lemma 17 also guarantees that  $\gamma = \sum_{i=1}^{h-1} |\hat{E}_i| + |\hat{E}'_i|$  and  $|\hat{T}| \leq \min\{n_T, \gamma\}$ . Then by Lemma 11, constructing  $\hat{T}_C$  requires  $\tilde{O}(n_{\hat{T}} + \gamma)$  time. As the size of  $\hat{T}_L$  and  $\hat{T}_R$  is the same as  $\hat{T}$  and  $\hat{T}_C$ , the call to `fast-merge` runs in  $\tilde{O}(n_T + \gamma)$  time by Theorem 14. The time it takes to compute the returned tree is  $\tilde{O}(n_T + \gamma)$  as we have enough time to scan all of  $T$  and  $\bigcup_{i=1}^{h-1} (E_i \cup E'_i)$ . So the total time excluding the recursive calls is  $(n_T + \gamma) \cdot \alpha$  for where  $\alpha = O(\log^c(n_T + \gamma))$  for some fixed integer  $c$ .

As for the recursive calls, first define  $\gamma(s, t) := \sum_{i=s}^t (|E_i| + |E'_i|)$ . Then the running time of the first recursive call is  $R(n_{\hat{T}}, \gamma(1, \ell-1))$  and the second recursive call is  $R(n_{\hat{T}_C}, \gamma(\ell+1, h-1))$ . By choice of  $\ell$ , we always have that  $\gamma(1, \ell-1) \leq \frac{\gamma}{2}$ . As  $\ell$  is the largest integer such that  $\gamma(1, \ell-1) \leq \frac{\gamma}{2}$ , then  $\gamma(1, \ell) > \frac{\gamma}{2}$ . Therefore, we have  $\gamma(\ell+1, h-1) = \gamma - \gamma(1, \ell) < \frac{\gamma}{2}$ . Combining this with the fact that `shrink-intersection` guarantees that  $|\hat{T}| \leq \min\{n_T, \gamma\}$  and  $|\hat{T}_C| \leq \min\{n_T, \gamma\}$ , we have

$$R(n_T, \gamma) \leq 2R(\min\{n_T, \gamma\}, \gamma/2) + \alpha \cdot (n_T + \gamma).$$

Note that  $R(n_T, \gamma) = O(1)$  when  $n_T = O(1)$  and  $\gamma = O(1)$ .

We claim that  $R(a, b) \leq \alpha\beta \cdot (a + 8b \log b)$  is a valid solution to this recurrence for some sufficiently large but fixed constant  $\beta \geq 1$ . By choosing  $\beta$  sufficiently large it is clear that it holds for the base case. To prove the inductive step we see the following:

$$R(a, b) \leq 2R(\min\{a, b\}, b/2) + \alpha \cdot (a + b) \leq 2[\alpha\beta \cdot (\min\{a, b\} + 4b \log(b/2))] + \alpha \cdot (a + b).$$



Hence we need to verify that

$$2[\alpha\beta(\min\{a, b\} + 4b \log(b/2))] + \alpha \cdot (a + b) \leq \alpha\beta \cdot (a + 8b \log b). \quad (1)$$

Since  $\beta \geq 1$ , rearranging, it suffices to verify that

$$2 \min\{a, b\} + 8b \log(b/2) + b \leq 8b \log b.$$

As  $8b \log b - 8b \log(b/2) = 8b$  and  $2 \min\{a, b\} + b \leq 3b$ , this proves (1) and therefore  $R(n_T, \gamma) \leq \alpha\beta(n_T + 8\gamma \log \gamma) = \tilde{O}(n_T + \gamma)$ . This concludes the proof.  $\blacktriangleleft$

The proof of Theorem 7 then follows by combining Theorem 12 (and remarks after the theorem statement) and Theorem 15.

## 4 Sparsification via the LP Solution

Let  $G = (V, E)$  be a graph on  $n$  nodes and  $m$  edges and let  $x$  be a point in  $\text{ST}(G)$ . In this section we discuss Theorem 4, which shows that, via random sampling, one can obtain a sparse point  $x' \in \text{ST}(G)$  from  $x$ . The random sampling approximately preserves linear constraints and thus one can use this technique to obtain sparse LP solutions to the packing problems involving spanning tree (and more generally matroid) constraints. The sampling and analysis rely on Karger's well-known work on random sampling for packing disjoint bases in matroids. We paraphrase the relevant theorem.

► **Theorem 19** (Corollary 4.12 from [30]). *Let  $\mathcal{M}$  be a matroid with  $m$  elements and non-negative integer capacities on elements such that  $\mathcal{M}$  has  $k$  disjoint bases. Suppose each copy of a capacitated element  $e$  is sampled independently with probability  $p \geq 18(\ln m)/(k\epsilon^2)$  yielding a matroid  $\mathcal{M}(p)$ . Then with high probability the number of disjoint bases in  $\mathcal{M}(p)$  is in  $[(1 - \epsilon)pk, (1 + \epsilon)pk]$ .*

We restate Theorem 4 for the reader's convenience. We leave the details of the proof to the full version.

► **Theorem 20.** *Let  $x \in \text{ST}(G)$  be a rational vector such that  $Ax \leq b$  for a matrix  $A \in [0, 1]^{r \times m}$  and  $b \in [1, \infty)^r$ . Consider a random subgraph  $G' = (V, E')$  of  $G$  obtained by picking each edge  $e \in G$  with probability  $\alpha_e := \min\{1, \frac{36 \log(r+m)}{\epsilon^2} \cdot x_e\}$ . Then with high probability the following hold: (i)  $|E'| = O(n \ln(r+m)/\epsilon^2)$  (ii) there exists a fractional solution  $z \in \text{ST}(G)$  in the support of  $G'$  such that  $Az \leq (1 + 3\epsilon)b$ .*

► **Remark 21.** For problems that also involve costs, we have a fractional solution  $\mathbf{x}$  and an objective  $\sum_e c_e x_e$ . Without loss of generality we can assume that  $c_e \in [0, 1]$  for all  $e$ . The preceding proof shows that the sparse graph obtained by sampling supports a fractional solution  $\mathbf{z}$  such that  $\mathbb{E}[\sum_e c_e z_e] \leq (1 + \epsilon) \sum_e c_e x_e$ . Further,  $\sum_e c_e z_e \leq (1 + 3\epsilon) \sum_e c_e x_e$  holds with high probability as long as  $\max_e c_e \leq \sum_e c_e x_e$ . This condition may not hold in general but can be typically guaranteed in the overall algorithm by guessing the largest cost edge in an optimum integer solution.

► **Remark 22.** The proof in the preceding theorem also shows that with high probability the graph  $G'$  is connected and satisfies the property that  $A\mathbf{1}_{E'} \leq O(\log n/\epsilon^2)b$ . Thus any spanning tree of  $G'$  satisfies the constraints to a multiplicative  $O(\log n)$ -factor by fixing  $\epsilon$  to a small constant. This is weaker than the guarantee provided by swap-rounding.

## 5 Fast approximation scheme for solving the LP relaxation

In this section, we discuss Theorem 1, which gives a fast approximation scheme to solve the LP relaxation for CROSSING-ST. We recall the LP for CROSSING-ST.

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e y_e \\ \text{subject to} \quad & Ay \leq b \\ & y \in \text{ST}(G) \end{aligned} \tag{P}$$

Note that even the feasibility problem (whether there exists  $y \in \text{ST}(G)$  such that  $Ay \leq b$ ) is interesting and important. We will mainly focus on the feasibility LP and show how we can incorporate costs in the full version of the paper. We recast the feasibility LP as a pure packing LP using an implicit formulation with an exponential number of variables. This is for technical convenience and to more directly apply the framework from [18]. For each tree  $T \in \mathcal{T}(G)$  we have a variable  $x_T$  and we consider the problem of packing spanning trees.

$$\begin{aligned} \text{maximize} \quad & \sum_{T \in \mathcal{T}} x_T \\ \text{subject to} \quad & \sum_{T \in \mathcal{T}} (A\mathbf{1}_T)_i \cdot x_T \leq b_i, \quad \forall i \in [k] \\ & x_T \geq 0, \quad \forall T \in \mathcal{T} \end{aligned} \tag{C}$$

The following is easy to establish from the fact that  $y \in \text{ST}(G)$  iff  $y$  can be written as a convex combination of the characteristic vectors of spanning trees of  $G$ .

► **Observation 23.** *There exists a feasible solution  $y$  to  $\mathcal{P}$  iff there exists a feasible solution  $x$  to  $\mathcal{C}$  with value at least 1. Further, if  $x$  is a feasible solution to  $\mathcal{C}$  with value  $(1 - \epsilon)$  there exists a solution  $y$  such that  $y \in \text{ST}(G)$  and  $Ay \leq \frac{1}{1-\epsilon}b$ .*

$\mathcal{C}$  is a pure packing LP, albeit in *implicit* form. In the full version of the paper, we show how to approximately solve the LP using MWU techniques and in particular we use the randomized MWU framework from [18] for positive LPs. The full version reviews the MWU framework and shows how to apply it along with other implementation details to achieve the concrete run times that we claim in Theorem 1.

---

## References

- 1 Alexander A. Ageev and Maxim Sviridenko. Pipeage rounding: A new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8:307–328, 2004.
- 2 N. Anari and S. O. Gharan. Effective-resistance-reducing flows, spectrally thin trees, and asymmetric TSP. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 20–39, 2015.
- 3 Nima Anari, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials, entropy, and a deterministic approximation algorithm for counting bases of matroids. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 35–46. IEEE, 2018.
- 4 Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials IV: exchange properties, tight mixing times, and faster sampling of spanning trees, 2020. [arXiv:2004.07220](https://arxiv.org/abs/2004.07220).
- 5 Arash Asadpour, Michel X Goemans, Aleksander Mádry, Shayan Oveis Gharan, and Amin Saberi. An  $O(\log n / \log \log n)$ -approximation algorithm for the asymmetric traveling salesman problem. *Operations Research*, 65(4):1043–1061, 2017.

- 6 Nikhil Bansal. On a generalization of iterated and randomized rounding. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1125–1135, 2019.
- 7 Nikhil Bansal, Rohit Khandekar, Jochen Könemann, Viswanath Nagarajan, and Britta Peis. On generalizations of network design problems with degree bounds. *Mathematical Programming*, 141(1-2):479–506, 2013.
- 8 Nikhil Bansal, Rohit Khandekar, and Viswanath Nagarajan. Additive guarantees for degree-bounded directed network design. *SIAM Journal on Computing*, 39(4):1413–1431, January 2010. doi:10.1137/080734340.
- 9 Vittorio Bilo, Vineet Goyal, Ramamoorthi Ravi, and Mohit Singh. On the crossing spanning tree problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 51–60. Springer, 2004.
- 10 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.
- 11 Deeparnab Chakrabarty, Yin Tat Lee, Aaron Sidford, Sahil Singla, and Sam Chiu-wai Wong. Faster matroid intersection. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1146–1168. IEEE, 2019.
- 12 Moses Charikar, Alantha Newman, and Aleksandar Nikolov. Tight hardness results for minimizing discrepancy. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1607–1614. SIAM, 2011.
- 13 Kamalika Chaudhuri, Satish Rao, Samantha Riesenfeld, and Kunal Talwar. What would Edmonds do? augmenting paths and witnesses for degree-bounded MSTs. *Algorithmica*, 55(1):157–189, November 2007. doi:10.1007/s00453-007-9115-5.
- 14 Chandra Chekuri, Sarel Har-Peled, and Kent Quanrud. Fast LP-based approximations for geometric packing and covering problems. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1019–1038. SIAM, 2020.
- 15 Chandra Chekuri and Kent Quanrud. Approximating the Held-Karp bound for metric TSP in nearly-linear time. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 789–800. IEEE, 2017.
- 16 Chandra Chekuri and Kent Quanrud. Near-linear time approximation schemes for some implicit fractional packing problems. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 801–820. SIAM, 2017.
- 17 Chandra Chekuri and Kent Quanrud. Fast approximations for Metric-TSP via linear programming. *arXiv preprint arXiv:1802.01242*, 2018.
- 18 Chandra Chekuri and Kent Quanrud. Randomized MWU for positive LPs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 358–377. SIAM, 2018.
- 19 Chandra Chekuri, Kent Quanrud, and Manuel R. Torres. Fast approximation algorithms for bounded degree and crossing spanning tree problems. *CoRR*, abs/2011.03194, 2020. arXiv:2011.03194.
- 20 Chandra Chekuri, Jan Vondrak, and Rico Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 575–584. IEEE, 2010.
- 21 Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Multi-budgeted matchings and matroid intersection via dependent rounding. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1080–1097. SIAM, 2011.
- 22 Ran Duan, Haoqing He, and Tianyi Zhang. Near-linear time algorithms for approximate minimum degree spanning trees. *ArXiv*, abs/1712.09166, 2017.
- 23 Alina Ene and Huy L Nguyen. Towards nearly-linear time algorithms for submodular maximization with a matroid constraint. In *International Colloquium on Automata, Languages, and Programming*, volume 132, 2019.

- 24 M. Fürer and B. Raghavachari. Approximating the minimum-degree Steiner tree to within one of optimal. *Journal of Algorithms*, 17(3):409–423, November 1994. doi:10.1006/jagm.1994.1042.
- 25 Rajiv Gandhi, Samir Khuller, Srinivasan Parthasarathy, and Aravind Srinivasan. Dependent rounding and its applications to approximation algorithms. *Journal of the ACM (JACM)*, 53(3):324–360, 2006.
- 26 Kyle Genova and David P Williamson. An experimental evaluation of the best-of-many Christofides’ algorithm for the traveling salesman problem. *Algorithmica*, 78(4):1109–1130, 2017.
- 27 Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. A randomized rounding approach to the traveling salesman problem. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 550–559. IEEE, 2011.
- 28 Michel Goemans. Minimum bounded degree spanning trees. *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS ’06)*, 2006. doi:10.1109/focs.2006.48.
- 29 Jacob Holm, Kristian De Lichtenberg, and Mikkel Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *Journal of the ACM (JACM)*, 48(4):723–760, 2001.
- 30 David R Karger. Random sampling and greedy sparsification for matroid optimization problems. *Mathematical Programming*, 82(1-2):41–81, 1998.
- 31 Anna R Karlin, Nathan Klein, and Shayan Oveis Gharan. An improved approximation algorithm for TSP in the half integral case. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 28–39, 2020.
- 32 Anna R Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved approximation algorithm for metric TSP. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 32–45, 2021.
- 33 Tamás Király, Lap Chi Lau, and Mohit Singh. Degree bounded matroids and submodular flows. *Combinatorica*, 32(6):703–720, 2012.
- 34 Jochen Könemann and R Ravi. Primal-dual meets local search: approximating MST’s with nonuniform degree bounds. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 389–395, 2003.
- 35 Jochen Könemann and Ramamoorthi Ravi. Primal-dual meets local search: approximating MSTs with nonuniform degree bounds. *SIAM Journal on Computing*, 34(3):763–773, 2005.
- 36 André Linhares and Chaitanya Swamy. Approximating min-cost chain-constrained spanning trees: a reduction from weighted to unweighted problems. *Mathematical Programming*, 172(1-2):17–34, 2018.
- 37 Neil Olver and Rico Zenklusen. Chain-constrained spanning trees. *Mathematical Programming*, 167(2):293–314, 2018.
- 38 Alessandro Panconesi and Aravind Srinivasan. Randomized distributed edge coloring via an extension of the Chernoff-Hoeffding bounds. *SIAM J. Comput.*, 26:350–368, 1997.
- 39 Kent Quanrud. Fast and deterministic approximations for  $k$ -cut. *arXiv preprint arXiv:1807.07143*, 2018.
- 40 Kent Quanrud. *Fast approximations for combinatorial optimization via multiplicative weight updates*. PhD thesis, University of Illinois, Urbana-Champaign, 2019. URL: <https://www.ideals.illinois.edu/handle/2142/106153>.
- 41 Aaron Schild. An almost-linear time algorithm for uniform random spanning tree generation. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 214–227, 2018.
- 42 Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.
- 43 Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. *Journal of the ACM (JACM)*, 62(1):1–19, 2015.

- 44 Mohit Singh and Nisheeth K Vishnoi. Entropy, optimization and counting. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 50–59, 2014.
- 45 Aravind Srinivasan. Distributions on level-sets with applications to approximation algorithms. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 588–597. IEEE, 2001.
- 46 Ola Svensson, Jakub Tarnawski, and László A Végh. A constant-factor approximation algorithm for the asymmetric traveling salesman problem. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 204–213, 2018.
- 47 Vera Traub and Jens Vygen. An improved approximation algorithm for ATSP. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1–13, 2020.
- 48 Di Wang. *Fast Approximation Algorithms for Positive Linear Programs*. PhD thesis, EECS Department, University of California, Berkeley, July 2017. URL: <http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-126.html>.
- 49 Sein Win. On a connection between the existence of  $k$ -trees and the toughness of a graph. *Graphs and Combinatorics*, 5(1):201–205, 1989.

## A Putting things together and extensions

In the main body of the paper we discussed the main technical results corresponding to Theorems 1, 4, and 7. In this section we give formal proofs of the corollaries stated in Section 1. In addition, we also describe some extensions and other related results that follow from the ideas in the preceding sections.

### A.1 Proofs of corollaries

We start with Corollary 3.

**Proof of Corollary 3.** Let  $G = (V, E)$  be the input graph with  $m$  edges and  $n$  vertices and let  $\epsilon > 0$ . Consider the LP relaxation to test whether  $G$  has a spanning tree with degree at most a given parameter  $B'$ . Theorem 1 implies that there exists a randomized algorithm that, with high probability, either correctly determines that there is no feasible solution to the LP, or outputs a fractional spanning tree  $y \in \text{ST}(G)$  such that  $\sum_{e \in \delta(v)} y_e \leq (1 + \epsilon)B'$  for all  $v$ . Using the algorithm, we can do binary search over the integers from 2 to  $n - 1$  to find the smallest value  $B$  for which the algorithm outputs a solution. We will focus on the scenario where the algorithm from Theorem 1 is correct in each of the  $O(\log n)$  calls in the binary search procedure; this happens with high probability. For the value of  $B$  found in the binary search, let  $y \in \text{ST}(G)$  be the solution that is output; we have  $\sum_{e \in \delta(v)} y_e \leq (1 + \epsilon)B$  for all  $v$ . Since the approximate LP solver correctly reported infeasibility for all  $B' < B$ , we have  $B - 1 < B^*$ , which implies  $B \leq B^*$ . As there is a feasible fractional spanning tree  $y$  such that  $\sum_{e \in \delta(v)} y_e \leq \lceil (1 + \epsilon)B \rceil$  for all  $v$ , the result of [43] implies that  $B^* \leq \lceil (1 + \epsilon)B \rceil + 1$ .

Regarding the run time, each call to the algorithm in Theorem 1 takes  $\tilde{O}(m/\epsilon^2)$  time since  $N = O(m)$  in the setting of BD-ST. Binary search adds only an  $O(\log n)$  multiplicative-factor overhead, leading to the claimed running time. ◀

We next prove Corollary 5. The algorithm described in the corollary takes advantage of Theorem 4 to sparsify the input graph, then runs the Fürer-Raghavachari algorithm [24].

**Proof of Corollary 5.** We will assume that the input is a graph  $G = (V, E)$  with  $m$  edges and  $n$  vertices. Let  $\epsilon > 0$ . As in the proof of Corollary 3, we can use Theorem 1 and binary search to find in  $\tilde{O}(\frac{m}{\epsilon^2})$  time, with high probability, a fractional spanning tree  $x \in \text{ST}(G)$  such that  $\sum_{e \in \delta(v)} x_e \leq (1 + \epsilon)B^*$  for all  $v \in V$ . By Theorem 4, we can use random

sampling based on  $x$  to obtain a subgraph  $G' = (V, E')$  of  $G$  such that with high probability we have  $|E'| = O(\frac{n \log(n+m)}{\epsilon^2})$  and there exists a fractional solution  $z \in \text{ST}(G)$  in the support of  $G'$  such that  $\sum_{e \in \delta(v)} z_e \leq (1 + 3\epsilon)(1 + \epsilon)B^* \leq (1 + 7\epsilon)B^*$  for all  $v \in V$  (for sufficiently small  $\epsilon$ ). The result of [43] implies there exists a spanning tree  $T$  in  $G'$  such that  $\max_{v \in V} \deg_T(v) \leq \lceil (1 + 7\epsilon)B^* \rceil + 1$ . For a graph  $H$  on  $n$  vertices and  $m$  edges, the algorithm of [24] runs in  $\tilde{O}(mn)$  time and outputs a spanning tree  $T$  such that the maximum degree of  $T$  is at most  $\text{OPT}(H) + 1$ , where  $\text{OPT}(H) = \min_{T \in \mathcal{T}(H)} \max_{v \in V} \deg_T(v)$ . Thus, the algorithm of [24] when applied to  $G'$ , outputs a spanning tree with degree at most  $\lceil (1 + 7\epsilon)B^* \rceil + 2$ , and runs in  $\tilde{O}(\frac{n^2}{\epsilon^2})$  time. ◀

We now prove Corollary 8. This corollary combines the LP solver of Theorem 1 and the fast implementation of randomized swap rounding of Theorem 7 to give a fast algorithm for CROSSING-MST.

**Proof of Corollary 8.** Let  $G = (V, E)$  be the input graph and  $\epsilon > 0$ . We want to solve  $\min\{c^T x : Ax \leq b, x \in \text{ST}(G)\}$ . By Theorem 1, there exists a randomized algorithm that runs in  $\tilde{O}(N/\epsilon^2)$  time and with high probability certifies the LP is infeasible or outputs  $y \in \text{ST}(G)$  such that  $c^T y \leq (1 + \epsilon)\text{OPT}$  and  $Ay \leq (1 + \epsilon)b$ . We then apply the fast implementation of randomized swap rounding of Theorem 7 to  $y$ , which runs in  $\tilde{O}(m/\epsilon^2)$  time and outputs a spanning tree  $T$ . If we only considered the feasibility version (i.e. find  $x \in \text{ST}(G)$  such that  $Ax \leq b$ ), then the existing results on swap rounding [20] imply that  $A\mathbf{1}_T \leq \min\{O(\log k / \log \log k)b, (1 + \epsilon)b + O(\log k)/\epsilon^2\}$  with high probability. In the cost version, [20] implies that  $A\mathbf{1}_T \leq \min\{O(\log k / \log \log k)b, (1 + \epsilon)b + O(\log k)/\epsilon^2\}$  with high probability, and  $\mathbb{E}[c^T \mathbf{1}_T] \leq \sum_e c^T x$ . Thus, the cost is preserved only in expectation. We can, however, apply Markov's inequality and conclude that  $\Pr[c^T \mathbf{1}_T \geq (1 + \epsilon)c^T x] \leq \frac{1}{1 + \epsilon} \leq 1 - \frac{\epsilon}{2}$  (for  $\epsilon$  sufficiently small). For a suitable choice of the high probability bound, we have that  $A\mathbf{1}_T \leq \min\{O(\log k / \log \log k)b, (1 + \epsilon)b + O(\log k)/\epsilon^2\}$  and  $c^T \mathbf{1}_T \leq (1 + \epsilon)c^T x$  with probability at least  $\frac{\epsilon}{2} - \frac{1}{2n^2}$ . We can assume that  $\epsilon > \frac{1}{n^2}$ , for otherwise the  $\frac{1}{\epsilon^2}$  dependence in the run time of the approximate LP algorithm is not meaningful; one can use other techniques including an exact LP solver. Thus, with probability at least  $\frac{\epsilon}{4}$ , we have  $c^T \mathbf{1}_T \leq (1 + O(\epsilon))c^T x \leq (1 + O(\epsilon))\text{OPT}$  and  $A\mathbf{1}_T \leq \min\{O(\log k / \log \log k)b, (1 + \epsilon)b + O(\log k)/\epsilon^2\}$ . To boost the  $\frac{\epsilon}{4}$  probability of success, we can repeat the rounding algorithm  $O(\frac{\log n}{\epsilon})$  times independently; with high probability, one of the trees will yield the desired bicriteria approximation. ◀

### Non-uniform degree bounds

We briefly sketch some details regarding Remark 6. First, we note that the algorithm for solving the LP relaxation handles the non-uniform degree bound case in  $\tilde{O}(m/\epsilon^2)$  time. It either certifies that the given bounds are infeasible or outputs a fractional solution with degree at most  $(1 + \epsilon)B_v$  for each  $v$ . We can then apply the result in [43] to know that there exists a spanning tree  $T$  in which the degree of each  $v$  is at most  $\lceil (1 + \epsilon)B_v \rceil + 1$ . We can apply sparsification from Theorem 4 to the fractional solution to obtain a sparse subgraph that contains a near-optimal fractional solution. It remains to observe that the Fürer-Raghavachari algorithm can be used even in the non-uniform setting via a simple reduction to the uniform setting. This was noted in prior work [34, 43] and we provide the details in the full version of the paper. This results in an  $\tilde{O}(n^2/\epsilon^2)$  time algorithm that either decides that the given non-uniform degree bounds are infeasible or outputs a spanning tree in which the degree of each node  $v$  is at most  $\lceil (1 + \epsilon)B_v \rceil + 2$ .

## A.2 Extensions and related problems

We focused mainly on BD-ST, BD-MST and CROSSING-MST. Here we briefly discuss related problems that have also been studied in the literature to which some of the ideas in this paper apply.

### Estimation of value for special cases of CROSSING-MST

As we remarked in Section 1, various special cases of CROSSING-MST have been studied. For some of these special cases one can obtain a constant factor violation in the constraint [37, 36]. We highlight one setting. One can view BD-MST as a special case of CROSSING-MST where the matrix  $A$  is a  $\{0, 1\}$ -matrix with at most 2 non-zeroes per column (since an edge participates in only two degree constraints); the result in [43] has been extended in [33] (see also [8]) to show that if  $A$  is a  $\{0, 1\}$ -matrix with at most  $\Delta$  non-zeroes per column, then the fractional solution can be rounded such that the cost is not violated and each constraint is violated by at most an additive bound of  $(\Delta - 1)$ . Theorem 1 allows us to obtain a near-linear time algorithm to approximate the LP. Combined with the known rounding results, this gives estimates of the integer optimum solution in near-linear time. Thus, the bottleneck in obtaining a solution, in addition to the value, is the rounding step. Finding faster iterated rounding algorithms is an interesting open problem even in restricted settings.

### Multiple cost vectors

In some applications one has multiple different cost vectors on the edges, and it is advantageous to find a spanning tree that simultaneously optimizes these costs. Such multi-criteria problems have been studied in several contexts. Let  $c_1, c_2, \dots, c_r$  be  $r$  different cost vectors on the edges (which we assume are all non-negative). In this setting it is typical to assume that we are given bounds  $B_1, B_2, \dots, B_r$  and the goal is to find a spanning tree  $T \in \mathcal{T}(G)$  satisfying the packing constraints such that  $c_j(T) \leq B_j$  for  $j \in [r]$ . We can easily incorporate these multiple cost bounds as packing constraints and solve the resulting LP relaxation via techniques outlined in Section 5. Once we have the LP solution we note that swap-rounding is oblivious to the objective, and hence preserves each cost in expectation. With additional standard tricks one can guarantee that the costs can be preserved to within an  $O(\log r)$  factor while ensuring that the constraints are satisfied to within the same factor guaranteed in Corollary 8.

### Lower bounds

BD-MST has been generalized to the setting where there can also be lower bounds on the degree constraints of each vertex. [43] and [33] showed that the additive degree bound guarantees for BD-MST can be extended to the setting with lower bounds in addition to upper bounds. One can also consider such a generalization in the context of CROSSING-MST. The natural LP relaxation for such a problem with both lower and upper bounds is of the form  $\min\{c^T x : Ax \leq b, A'x \geq b', x \in \text{ST}(G)\}$  where  $A, A' \in [0, 1]^{k \times m}$ ,  $b, b' \in [1, \infty)^k$ ,  $c \in [0, \infty)^m$ . Here  $A$  corresponds to upper bounds (packing constraints) and  $A'$  corresponds to lower bounds (covering constraints). This mixed packing and covering LP can also be solved approximately in near-linear time by generalizing the ideas in Section 5. Sparsification as well as swap-rounding can also be applied since they are oblivious to the constraints once the LP is solved. The guarantees one obtains via swap rounding are based on negative correlation and concentration bounds. They behave slightly differently for lower bounds. One obtains a tree  $T$  such that  $A\mathbf{1}_T \leq (1 + \epsilon)b + O(\log k)/\epsilon^2$  and  $A'\mathbf{1}_T \geq (1 - \epsilon)b' - O(\log k)/\epsilon^2$  with high probability. As in the other cases, the LP solution proves the existence of good integer solutions based on the known rounding results.





# Hitting Weighted Even Cycles in Planar Graphs

Alexander Göke ✉

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

Jochen Koenemann ✉

University of Waterloo, Canada

Matthias Mnich ✉ 

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

Hao Sun ✉

University of Waterloo, Canada

---

## Abstract

---

A classical branch of graph algorithms is graph transversals, where one seeks a minimum-weight subset of nodes in a node-weighted graph  $G$  which intersects all copies of subgraphs  $F$  from a fixed family  $\mathcal{F}$ . Many such graph transversal problems have been shown to admit polynomial-time approximation schemes (PTAS) for *planar* input graphs  $G$ , using a variety of techniques like the shifting technique (Baker, J. ACM 1994), bidimensionality (Fomin et al., SODA 2011), or connectivity domination (Cohen-Addad et al., STOC 2016). These techniques do not seem to apply to graph transversals with parity constraints, which have recently received significant attention, but for which no PTASs are known.

In the *even-cycle transversal* (ECT) problem, the goal is to find a minimum-weight hitting set for the set of even cycles in an undirected graph. For ECT, Fiorini et al. (IPCO 2010) showed that the integrality gap of the standard covering LP relaxation is  $\Theta(\log n)$ , and that adding sparsity inequalities reduces the integrality gap to 10.

Our main result is a primal-dual algorithm that yields a  $47/7 \approx 6.71$ -approximation for ECT on node-weighted planar graphs, and an integrality gap of the same value for the standard LP relaxation on node-weighted planar graphs.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Packing and covering problems

**Keywords and phrases** Even cycles, planar graphs, integrality gap

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.25

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/abs/2107.04763>

**Funding** *Jochen Koenemann*: Research funded by the NSERC Discovery Grant program.

*Hao Sun*: Research funded by the NSERC Discovery Grant program.

## 1 Introduction

Transversal problems in graphs have received a significant amount of attention from the perspective of algorithm design. Such problems take as input a node-weighted graph  $G$ , and seek a minimum-weight subset  $S$  of nodes which intersect all graphs  $F$  from a fixed graph family  $\mathcal{F}$  that appears as subgraph in  $G$ . A prominent example in this direction is the fundamental FEEDBACK VERTEX SET (FVS) problem, where  $\mathcal{F}$  is the class of all cycles. FVS is one of Karp's 21 NP-complete problems [17]. It admits a 2-approximation in polynomial time [2, 5], which cannot be improved to a  $(2 - \varepsilon)$ -approximation for any  $\varepsilon > 0$  assuming the Unique Games Conjecture [18].

Recently, several graph transversal problems have been revisited in the presence of additional parity constraints [19, 21, 20, 23]. The natural parity variants of FVS are ODD CYCLE TRANSVERSAL (OCT) and EVEN CYCLE TRANSVERSAL (ECT), where one wishes to intersect the odd-length and even-length cycles of the input graph  $G$ , respectively. The



© Alexander Göke, Jochen Koenemann, Matthias Mnich, and Hao Sun;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 25; pp. 25:1–25:23



Leibniz International Proceedings in Informatics

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

approximability of these problems is much less understood than that of FVS: for OCT, only an  $\mathcal{O}(\sqrt{\log n})$ -approximation is known [1], and for ECT, only a 10-approximation is known [21].

Planar graphs are a natural subclass of graphs in which to consider graph transversal problems. The interest goes back to Baker’s shifting technique [3], which yielded a PTAS for VERTEX COVER in planar graphs (where  $\mathcal{F}$  is the single graph consisting of an edge). The technique was generalized by Demaine et al. [8], who gave EPTASs for graph transversal problems satisfying a certain bidimensionality criterion, including FVS in *unweighted* planar graphs. That result was later extended to yield an EPTAS for FVS in unweighted  $H$ -minor free graphs [13], for any fixed graph  $H$ . In *edge-weighted* planar graphs, PTAS are known for edge-weighted STEINER FOREST and OCT [4, 16, 10].

On *node-weighted* planar graphs, the situation appears to be more complex. First, the existence of a PTAS for FVS on node-weighted planar graphs was a long-standing open question which was resolved only recently in a paper of Cohen-Addad et al. [7]. The authors presented a PTAS for FVS in node-weighted planar graphs, crucially exploiting the fact that the treewidth of  $G - S$  is bounded for feasible solutions  $S$ . The existence of an EPTAS for FVS in node-weighted planar graphs is still open.

To deal with cycle transversal problems (in node-weighted planar graphs) which are more complex than FVS, Goemans and Williamson [14] first proposed a primal-dual based framework. Their framework requires the cycle family  $\mathcal{F}$  to satisfy a certain uncrossing property. The latter property can be seen to be satisfied by OCT, DIRECTED FVS in directed planar graphs, and SUBSET FVS, which seeks a minimum-cost node set hitting all cycles containing a node from a given node set  $T$ . For those problems, the authors obtained 3-approximations<sup>1</sup>. The framework by Berman and Yaroslavtsev [14] also yields a 3-approximation for STEINER FOREST in node-weighted planar graphs [9, 22]. Berman and Yaroslavtsev [6] later improved the approximation factor for the same class of uncrossable cycle transversal problems from 3 to 2.4. For none of those problems, though, the existence of a PTAS is known.

The main question driving our work is whether the framework of Goemans and Williamson [14] (and its improvement by Berman and Yaroslavtsev [6]) can be extended to cycle transversal problems that do not satisfy uncrossability. In this paper we focus on ECT in node-weighted planar graphs as a natural such problem: even cycles are not uncrossable, and hence the frameworks by Goemans and Williamson [14] does not apply. Furthermore, the framework of Cohen-Addad et al. [7] requires that contracting edges only reduces the solution value, which is not the case for even cycles either. For *unweighted* planar graphs, it is still possible to obtain an EPTAS for ECT, by building on the work of Fomin et al. [12]. Their main result are EPTASs for bidimensional problems, which ECT is not (as contracting edges can change the parity of cycles). Yet, to obtain their result, they show that any transversal problem that satisfies the “ $\nu$ -transversability” and “reducibility” conditions has an EPTAS on  $H$ -minor free graphs (cf. [12, Theorem 1]). Both conditions are met by unweighted ECT<sup>2</sup>, which thus admits an EPTAS on  $H$ -minor free graphs. For ECT on node-weighted planar graphs, though, reducibility fails, and the existence of a PTAS is unknown. The currently best result for ECT is a 10-approximation, which was given by Fiorini et al. [11] for general graphs. They showed that the integrality gap of the standard covering LP relaxation for ECT is  $\Theta(\log n)$ , but that adding sparsity inequalities reduces the integrality gap to 10. No better than 10-approximation is known for ECT in node-weighted planar graphs.

<sup>1</sup> 18/7-approximations were claimed but later found to be incorrect [6].

<sup>2</sup>  $\nu$ -transversability follows from as graphs without even cycles have treewidth 2, and reducibility from unit weights and connectedness of the to-be-hit subgraphs  $F$ .

### 1.1 Our results

We prove an improved approximation algorithm for ECT in node-weighted planar graphs.

► **Theorem 1.** *ECT has an efficient  $47/7 \approx 6.71$ -approximation on node-weighted planar graphs.*

This improves the previously best 10-approximation by Fiorini et al. [11] for planar graphs.

Our algorithm takes as input a node-weighted planar graph  $G$  with node weights  $c_v \in \mathbb{N}$  for each  $v \in V(G)$ . We then employ a primal-dual algorithm that is based on the following natural covering LP for ECT and its dual, where  $\mathcal{C}$  denotes the set of even cycles in  $G$ :

$$\begin{array}{l|l}
 \min c^T x & \max \mathbb{1}^T y \\
 \text{(P}_{\text{ECT}}) \text{ s.t. } x(C) \geq 1 \quad \forall C \in \mathcal{C} & \text{s.t. } \sum_{C \in \mathcal{C}, v \in C} y_C \leq c_v \quad \forall v \in V(G) \\
 x \geq 0 & \text{(D}_{\text{ECT}}) \\
 & y \geq 0
 \end{array}$$

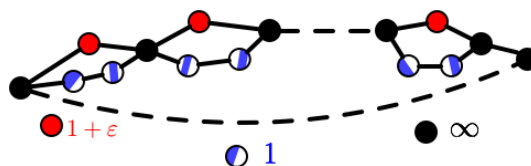
Fiorini et al. [11] proved that the integrality gap of this LP is  $\Theta(\log n)$ . Our main result is an improved integrality gap of this LP for ECT in planar graphs:

► **Theorem 2.** *The integrality gap of the LP (P<sub>ECT</sub>) is at most  $47/7 \approx 6.71$  in planar graphs.*

### 1.2 Our approach

Designing a primal-dual algorithm is far from trivial, as the imposed parity constraints rule out a direct application of the framework proposed by Goemans and Williamson [14]. Unlike in their work, *face-minimal even cycles* (even cycles containing a minimal set of faces in their interior) are not necessarily faces, and may thus overlap. Indeed, increasing the dual variables of face-minimal even cycles does not yield a constant-factor approximation in general.

Consider Figure 1, and let  $F$  be the inner face that is only incident to blue and black nodes. For an even number of 5-cycles surrounding  $F$ ,  $F$  is the only face-minimal even cycle



■ **Figure 1** The bottom path has odd length, and the number of length-5 faces at the top is even.

in the graph. Using only  $F$  for the dual increase, even including a reverse-delete step, leaves one blue node of each 5-cycle. Yet, an optimal solution would take a single red and blue node from one 5-cycle.

To circumvent this impediment, we establish strong structural properties of planar graphs related to ECT. Those properties along with results from matching theory allow us to algorithmically find a large set of pairwise face-disjoint even cycles whose dual variables we can then increment. Even with this set of cycles found, it remains technically challenging to bound the integrality gap. For this purpose, we first use the structure of minimal hitting sets of our graph to associate each such set with a hitting set in a subdivision of the so called 2-compression of our graph; the latter is a certain minor that we define in detail shortly. We then show that faces that are contained in even cycles we increment are incident to few

nodes on average. Crucial in this step is a technical result that is implicit in the work of Berman and Yaroslavtsev [6]. Eventually, this approach leads to an integrality gap of  $47/7$ , and an algorithm with the same approximation guarantee.

Due to space constraints, we defer proofs of statements marked by  $(\star)$  to the full version of the paper [15].

## 2 Primal-dual algorithm for ECT on node-weighted planar graphs

We describe a primal-dual, constant-factor approximation for ECT on node-weighted planar graphs. Our algorithm borrows some ideas from Fiorini et al. [11] for the DIAMOND HITTING SET (DHS) problem, which seeks a minimum-cost set of nodes in a node-weighted graph  $G$  that hits all *diamonds* (sub-divisions of the graph consisting of three parallel edges). For DHS, Fiorini et al. [11] employ a primal-dual algorithm to prove that the natural covering LP ( $P_{\text{ECT}}$ ) (where  $\mathcal{C}$  is replaced by the set of diamonds) has integrality gap  $\Theta(\log n)$ . We develop several new ideas to obtain a constant integrality gap.

We now outline the ideas of our primal-dual approach. Consider a planar input graph  $G$  with node costs  $c_v \in \mathbb{N}$  for each  $v \in V(G)$ . Given feasible dual solution  $y$  to ( $D_{\text{ECT}}$ ), let the *residual cost* of node  $v \in V(G)$  be  $c_v - \sum_{C \in \mathcal{C}, v \in C} y_C$ . Our primal-dual method begins with a trivial feasible dual solution  $y = \mathbb{0}$ , and the empty, infeasible hitting set  $S = \emptyset$ .

Then, in each iteration, we increase  $y_C$  for all  $C$  in some carefully chosen subset  $\mathcal{C}' \subseteq \mathcal{C}$  of even cycles, while maintaining dual feasibility, and until some *primary condition* is achieved. A common such primary condition is that some dual node-constraint becomes *tight* in the increase process, and hence the corresponding node ends up having residual cost 0.

When this happens, we add the node to  $S$ . Once  $S$  is a feasible ECT, our algorithm ends its first phase, and executes a problem-specific *reverse-delete* procedure. Here, we consider all nodes in  $S$  in reverse order of addition to  $S$ , and we delete such a node if the feasibility of  $S$  is maintained. We will later describe a subtle and crucial refinement of this reverse-delete procedure. Call the resulting final output of the algorithm  $S'$ .

During our algorithm, we will use the term *hitting set* to refer to  $S$ , and during the analysis we will use the term *hitting set* to refer to  $S'$ . We will say a hitting set is *feasible* if it is a feasible ECT, and refer to nodes of the hitting set as *hit nodes*.

In the next subsections, we will fill in the details of the algorithm, and analyze the cost of  $S'$  compared to the value of an optimal solution. We begin by defining the concept of “blended inequalities” and necessary graph compression operations. Blended inequalities were used by Fiorini et al. [11], and our definitions follow their’s closely.

### 2.1 Blended inequalities and compression

A *block* of  $G$  is an inclusion-maximal 2-connected subgraph of  $G$ . The *block graph* of  $G$  is the bipartite graph  $B_G$  with bipartition  $V(B_G) = B_1 \cup B_2$ , where  $B_1$  are the blocks of  $G$ ,  $B_2$  are the cut nodes of  $G$ , and  $(b_1, b_2) \in B_1 \times B_2$  is an edge if  $b_2$  is a node of  $b_1$ .

Let  $S$  be a partial solution to the given ECT instance at some point during the execution of our algorithm. Let  $G^S$  be the corresponding residual graph that we obtain from  $G - S$  by deleting all nodes that do not lie on even cycles. Our primal-dual algorithm first looks for an even cycle  $C$  in  $G^S$  such that at most two nodes of  $C$  have neighbours outside  $C$ . If such a cycle  $C$  is found, we increment its dual variable  $y_C$  until a node becomes tight. The reason for doing this is that such  $C$  will pay for at most two hit nodes, which we will show later.

If there is no even cycle  $C$  in  $G^S$  such that at most two nodes of  $C$  have neighbours outside  $C$ , we successively compress the residual graph  $G^S$  using two types of graph compression. To this end, first note that any minimal solution will only contain one node in the

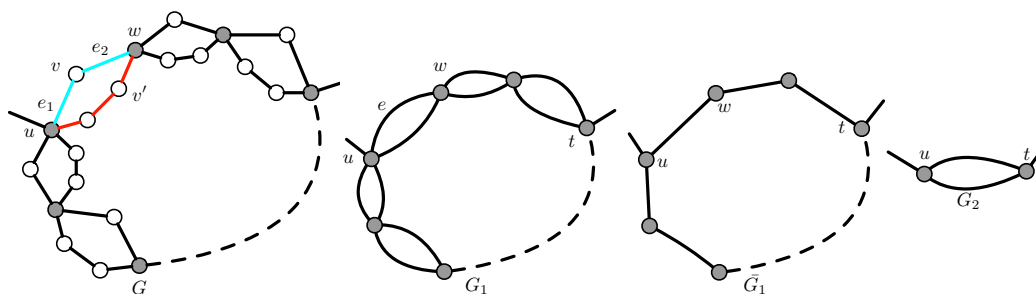
interior of any induced path in  $G^S$ . Suppose we contract some path  $P$  of  $G^S$  of length at least 2 down to an edge  $e$ . Choosing a node in the interior of  $P$  is “equivalent” to choosing the edge  $e$ . This is the motivation for the 1-compression.

Suppose we contract two  $u$ - $v$  paths  $P_1, P_2$  with lengths of different parity down to edges  $e_1, e_2$ , respectively. We will find it helpful to think of these edges as a single *twin* edge between  $u$  and  $v$  whose parity is *flexible*. This is the motivation for the 2-compression.

Formally, we will successively compress  $G^S$  as follows:

- Obtain the 1-compression  $G_1^S$  of  $G^S$  by repeatedly *folding* degree-2 nodes  $v$ , as long as they exist, which means to delete  $v$  and adding the edge  $uw$  between its neighbors  $u, w$ .
- Note that no pair of nodes in  $G_1^S$  is connected by more than two edges. Obtain  $\bar{G}_1^S$  from  $G_1^S$  by replacing each pair of parallel edges by a *twin* edge. In  $\bar{G}_1^S$ , we now once again fold degree-2 nodes as long as those exist. The resulting graph is the 2-compression  $G_2^S$  of  $G^S$ .

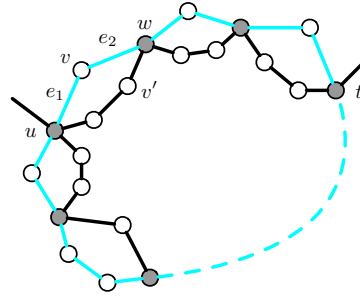
See Figure 2 for examples of 1- and 2-compression of a graph. In the following, we will omit the superscript  $S$  from  $G_1^S, \bar{G}_1^S,$  and  $G_2^S$  if this is clear from the context. Let  $G_3$  be obtained from  $G_2$  by replacing every edge of  $G_2$  with a path of length two. If a twin edge was replaced, call the two edges of the path added *twin edges*. By an abuse of notation, we call a cycle of  $G_1, G_2$  or  $G_3$  even if it contains a twin edge, or if its preimage in  $G$  is even.



■ **Figure 2** The graph  $G$  and its 1- and 2-compression  $G_1$  and  $G_2$ .

In the following, we will sometimes call the subgraph  $Q$  of  $G$  whose contraction yields a subgraph  $R$  of  $G_2$  the *preimage* of  $R$ . If  $R$  is an edge, call  $Q$  a *piece*, and say  $Q$  *corresponds* to  $R$ . Furthermore, call  $u, v$  *ends* of  $Q$  and other nodes of  $Q$  *internal nodes*. If the edge was twin, call the piece *twin*, otherwise, call the piece *single*. The blocks of a piece are cycles and paths, and the block graph of a piece is a path. Each cycle of a piece is called an *elementary cycle*. For an elementary cycle  $C$ , call its two nodes  $u_C$  and  $v_C$  with neighbours outside  $C$  *branch nodes*. Call the two  $u_C - v_C$ -paths  $P_1, P_2$  in  $C$  the *handles* of  $C$ , which form the *handle pair*  $(P_1, P_2)$ . For an illustration, see the red and light blue edges in Figure 2.

The reason for defining  $G_3$  is that intuitively selecting a node inside a piece corresponds to selecting the edge corresponding to the piece in  $G_2$ . It will be simpler for us if our hitting set consists of only nodes, so we subdivide each edge of  $G_2$ . Suppose that  $S$  is the partial (and infeasible) hitting set for the cycles in  $\mathcal{C}$  at some point during the algorithm. Further, assume that  $G^S$  has even cycles, but none with at most two outside neighbours. In this case, if an even cycle  $C'$  in  $G^S$  contains an internal node of some piece  $Q$ , then  $C' \cap Q$  is a path between the ends of  $Q$ ; see Figure 3. It follows that  $C'$  has the form  $v_1 P_1 v_2 P_2 \dots v_k P_k v_1$ , where for  $i = 1, \dots, k$  nodes  $v_i, v_{i+1 \bmod k}$  are ends of some piece  $Q_i$ , and  $P_i$  is a  $v_i - v_{i+1}$  path in  $Q_i$ . For  $i = 1, \dots, k$ , pieces  $Q_i, Q_j$  for  $i \neq j$  are disjoint except for their ends.



■ **Figure 3** The light blue cycle in  $G$  has two  $u$ - $t$  paths lying in different pieces of  $G$ ; the dashed path has odd length.

We say that  $C'$  in  $G^S$  corresponds to cycle  $C = (v_1, \dots, v_k)$  in  $G_2^S$ . For such  $C$ , its *blended inequality* is

$$\sum_v a_v^C x_v \geq 1, \tag{*}$$

where  $a_v^C \in \{0, 1/2, 1\}$  for all nodes  $v$ , and where the support of  $a^C$  is contained in the node set of the preimage of  $C$ . We next provide a precise definition of the coefficients of (\*). With those, one can show that (\*) is dominated by a convex combination of inequalities  $x(C) \geq 1$  in  $(P_{ECT})$ .

Consider an elementary cycle of the preimage of  $C$  and let  $h_1, h_2$  be its two handles. For each of these handles, we define its residual cost as the smallest residual cost of any of its internal nodes. Suppose that the residual cost of  $h_2$  is at most that of  $h_1$ . We will also call  $h_1$  the *dominant*, and  $h_2$  the *non-dominant* handle of this cycle. As an invariant, our algorithm maintains that the designation of dominant and non-dominant handles of an elementary cycle does not change throughout the algorithm's execution.

Suppose first that the residual cost of  $h_1$  is strictly larger than that of  $h_2$ . In this case, let  $a_v^C = 1$  for all internal nodes of handle  $h_1$ , and let  $a_v^C = 0$  of the internal nodes of  $h_2$ . If the residual cost of both handles is the same, we let  $a_v^C = 1/2$  on internal nodes of both handles.

In certain cases, we need to *correct the parity* of the constructed inequality. This is necessary if  $a^C$  as defined above is 0,1 (i.e., if all elementary cycles of  $C$  have a strictly dominant handle), and if the cycle formed by all dominant handles is odd. In this case, we pick an arbitrary elementary cycle on  $C$ , and declare it *special*. For this special cycle, we then set  $a_v^C = 1$  for the internal nodes on *both* handles. Following the same reasoning as Fiorini et al. [11] for DHS, we can show the following for ECT:

► **Lemma 3.** *Each feasible point of our LP  $(P_{ECT})$  satisfies any blended inequality.*

In our algorithm, we assume that inequalities (\*) are part of  $(P_{ECT})$ . Throughout the algorithm, we increase dual variables  $y_{\otimes}$  of such inequalities.

We will sometimes say that variable  $y_{\otimes}$  (or cycle  $C$ ) *pays for*  $\sum_{v \in S'} a_v^C$  hit nodes. It is well-known (see, e.g., Goemans and Williamson [14]) that if during any iteration dual variables for a family of blended inequalities are incremented uniformly, and the dual variables pay for  $\alpha$  hit nodes (of  $S'$ ) on average, then the final solution produced by the algorithm is  $\alpha$ -approximate.

The motivation for blended inequalities is to pay for no more than one node in each piece. Consider the example in Figure 1. Here, the bottom black dashed path is odd, there are an even number of handle pairs in the top part, and  $\varepsilon$  is small. Suppose we set  $a_v^C = 1/2$

on internal nodes of each handle. If we were to increment the inequality  $(\otimes)$ , all the blue nodes of weight 1 would become tight, and after reverse-delete, the algorithm would keep one blue node for each handle pair. However, selecting a red node and a blue node would be a cheaper solution. This could be achieved by setting  $a_v^C = 1$  for red and black nodes, and  $a_v^C = 0$  on blue nodes, until the residual costs of the red nodes become 1, and afterwards setting  $a_v^C = 1/2$  on internal nodes of each handle.

During its execution, the algorithm carefully chooses a family of even cycles  $\mathcal{C}$  in  $G_2^S$  and increments the dual variables of certain blended inequalities for each  $C \in \mathcal{C}$  until a node becomes tight, or the blended inequality changes; i.e. the residual costs of two handles of a handle pair, which were previously not equal, become equal.

In their primal-dual algorithms for cycle transversal problems with uncrossing property, Goemans and Williamson [14] started with the infeasible “hitting set”  $S = \emptyset$ . While  $S$  is infeasible, the dual variables for faces of the residual digraph that are cycles are incremented. A reverse-delete step is applied at the end. The authors show that tight examples for their algorithm feature so called *pocket* subgraphs. Not surprisingly, the improved algorithm of Berman and Yaroslavtsev [6] has to pay special attention to these pockets to obtain the improvement in performance guarantee.

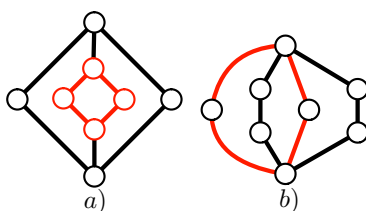
## 2.2 Pockets and their variants

The following definition of crossing cycles was elementary to the approach by Goemans and Williamson [14] for cycle transversal problems in planar graphs.

► **Definition 4.** *In an embedded planar graph, two cycles  $C_1, C_2$  cross if  $C_i$  contains an edge intersecting the interior of the region bounded by  $C_{3-i}$ , for  $i = 1, 2$ . That is, the plane curve corresponding to the embedding of the edge in the plane intersects the interior of the region of the plane bounded by  $C_{3-i}$ . A set of cycles  $\mathcal{C}$  is laminar if no two elements of  $\mathcal{C}$  cross.*

Next, we formally define pockets, and we also introduce the new notion of “pseudo-pockets”, the lack of which will help us “cover” our graph with even cycles.

► **Definition 5.** *Let  $G$  be a graph and let  $\mathcal{C}$  be a collection of cycles in  $G$ . A pseudo-pocket of  $(G, \mathcal{C})$  is a connected subgraph  $G'$  of  $G$  which contains a cycle such that at most two nodes of  $G'$  have neighbours outside  $G'$ . A pocket of  $(G, \mathcal{C})$  is a pseudo-pocket that contains a cycle of  $\mathcal{C}$ . A pocket is minimal if it contains no pocket as a proper induced subgraph.*



■ **Figure 4** (a) Graph formed by red nodes is a pocket. (b) Crossing cycles in red and black.

## 2.3 Identifying families of even cycles via tilings

The  $12/5$ -approximation algorithm of Berman and Yaroslavtsev [6] for DIRECTED FVS in node-weighted planar digraphs  $G$  proceeds roughly as follows.

It starts with the empty hitting set  $S = \emptyset$ . As long as  $S$  is not a hitting set for the directed cycles of  $G$ , it first looks for a pocket  $H$  of the residual digraph  $G^S$ , that is the digraph obtained from  $G - S$  by deleting all nodes not on a directed cycle. It then increments the dual variables for the set of face minimal directed cycles of  $H$ , which happen to be faces. It then adds any nodes that become tight to  $S$ . Once  $S$  is feasible, the algorithm performs a reverse deletion step.

As pointed out, in our setting, face-minimal even cycles may not be faces, and may cross. Following Berman and Yaroslavtsev [6], we wish to “cover” our residual graph with face-minimal even cycles which do not cross, we call this a “tiling”; see Figure 5 iii). As we will see, this tiling allows us to identify the dual variables to increase. Let us formalize the correspondence between edges of the dual between odd faces and even faces.

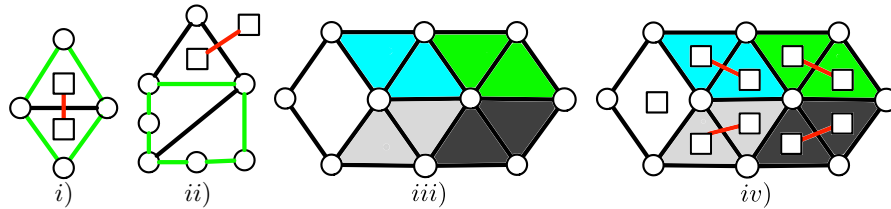
► **Definition 6.** *Let  $H$  be a plane graph without pseudo-pockets. For each face  $f$  of  $H$ , let  $v_f$  be the corresponding node of the planar dual  $H^*$ . A tile of  $H$  is an even cycle  $C$  of  $H$  bounding one or two faces. If  $C$  is a single face  $f$ , we say that  $C$  corresponds to the node  $v_f$ . If  $C$  bounds two faces  $f$  and  $g$ , we say that  $C$  corresponds to the edge  $v_f v_g \in E(H^*)$ . We say that nodes  $v_f, v_g$  and the faces  $f, g$  are covered by the tile.*

For a node  $v$  of  $H^*$ , let  $f_v \subset E(H)$  be the edges on the boundary of the corresponding face of  $H$ . Denote by  $h_\infty$  the node of  $H^*$  corresponding to the infinite face.

Given  $wh_\infty \in E(H^*)$ , a cycle  $C^1 \subset E(H)$  corresponds to  $wh_\infty$  if  $C^1$  is a cycle of  $f_w \Delta f_{h_\infty}$ , or  $C^1 = C' \Delta f_w$  and  $C'$  is a cycle of  $f_w \Delta f_{h_\infty}$ . We also call such a cycle  $C^1$  a tile and say that  $C^1$  covers  $h_\infty, w$ , and the corresponding faces.

Given a matching  $E' \subset E(H^*)$  and  $V' \subset V(H^*)$ , with  $E' = \{e_1, \dots, e_\ell\}$  and  $V' = \{v_1, \dots, v_t\}$ , a set of tiles  $\mathcal{T} = \{C_1, \dots, C_{\ell+t}\}$  corresponds to  $E' \cup V'$  if  $C_i$  corresponds to  $e_i$  for  $i = 1, \dots, \ell$  and  $C_{j+\ell}$  corresponds to  $v_j$  for  $j = 1, \dots, t$ .

In Figure 5 i), cycle  $C$  bounds two faces  $f$  and  $g$ ; see also Figure 5 ii).



■ **Figure 5** Diagrams i) and ii) show cycles in green and corresponding edges of the dual graph in red. (i) The red edge corresponds to the symmetric difference of two finite faces. (ii) The red edge corresponds to the symmetric difference of a finite and infinite face. Diagrams iii) and iv) show a tiling indicated by the boundaries of the various finite regions in white, light grey, etc and the corresponding matching.

► **Definition 7.** *For a plane graph  $H$ , a set  $\mathcal{T}$  of tiles is a pseudo-tiling if no face of  $H$  is covered by more than one tile. If the node  $v_{h_\infty}$  corresponding to the infinite face of  $H$  is not covered by  $\mathcal{T}$ , we call  $\mathcal{T}$  a tiling.*

Certain tilings are particularly desirable; we will define these the next.

► **Definition 8.** *Let  $\alpha \in (0, 1)$ . A tiling is  $\alpha$ -quasi-perfect if it covers all even finite faces, a  $\beta$ -fraction of odd finite faces of  $G^S$ , and a  $\psi$ -fraction of the finite faces of  $G^S$  are even, where  $\beta(1 - \psi) + 2\psi \geq \alpha$ .*



Let  $C$  be an even cycle in  $G_2^S$ , and recall that we say that  $C$  pays for  $\sum_{v \in S} a_v^C$  hit nodes. For an even cycle in a tiling consisting of two faces, we bound the number of hit nodes it pays for by the number of hit nodes each face pays for.

We will show that a finite face of our graph intersects at most  $18/7$  hit nodes on average (over all finite faces). Ideally, we would want to cover all faces by a tiling. Then an even cycle of our tiling is incident to at most  $36/7$  hit nodes on average, twice the amount a face of our graph intersects on average. Alas, tilings covering all faces need not always exist. Thus, we try to find a tiling that covers as many finite faces as possible. Suppose that we find a tiling  $\mathcal{T}$  that covers a set  $\mathcal{T}_{\text{Faces}}$  of finite faces consisting of  $\alpha$ -fraction of the finite faces of our graph. It follows that a face of  $\mathcal{T}_{\text{Faces}}$  will be incident to at most  $18/7\alpha$  hit nodes on average, and so an even cycle of the tiling  $\mathcal{T}$  is incident to at most  $36/7\alpha$  hit nodes on average. Intuitively, even faces pay for fewer hit nodes than even cycles containing two faces, so it is good if a tiling contains many even faces. The motivation for quasi-perfect tilings is that it is good if a large fraction of faces are covered by the tiling and if the tiling contains a lot of even faces. We prove the following key result in Appendix A.

► **Theorem 9.** *Let  $H$  be a 2-compression of some planar graph  $G$ , that has an even cycle and contains no pockets. Then  $H$  has a  $2/3$ -quasi-perfect tiling.*

## 2.4 The algorithm in detail

We formally state our algorithm. It takes as input a planar graph  $G$  with cost function  $c : V(G) \rightarrow \mathbb{N}$ . Let  $\mathcal{C}(G)$  be the set of even cycles of  $G$ , and let  $\text{opt}(G, c)$  be the minimum cost of an *even cycle transversal* of  $G$ , which is a set of nodes intersecting each cycle in  $\mathcal{C}(G)$ .

As we will see, the algorithm returns an even cycle transversal  $S$  of  $G$  whose cost is at most  $(47/7)\text{opt}(G, c)$ . We start with the empty candidate  $S := \emptyset$ . In each iteration, the algorithm looks for an even cycle  $C$  in the residual graph  $G^S$  such that at most two nodes of  $C$  have outside neighbours. If we find such  $C$ , increment the variable  $y_C$  until a node becomes tight. If no such cycle exists, the algorithm computes the 2-compression of  $G^S$ , and in it, we find an inclusion-minimal pocket  $H$  of  $G_2^S$ . Using Theorem 9, we find a  $2/3$ -quasi-perfect tiling  $\mathcal{T}_H$  of  $H$  and increments the dual variables for the blended inequalities for each  $C \in \mathcal{T}_H$ . The algorithm then adds all nodes  $X$  that became tight to our candidate hitting set  $S$ .

During an iteration, for each handle pair  $(Q_1, Q_2)$  for which the set  $X$  of nodes that became tight contains a node in the interior of each handle, our algorithm will choose two nodes  $a, b \in X$  with  $a$  in the interior of  $Q_1$  and  $b$  in the interior of  $Q_2$  and define  $(a, b)$  to be a *node pair*. For instance, in Figure 2 if  $v$  and  $v'$  are the only nodes added during some iteration then the algorithm would define  $(v, v')$  to be a node pair. For a set of nodes  $X$  added during the same iteration, nodes in a pair are considered to be added *before* any node not in a pair.

At the end of the algorithm, we perform a non-trivial reverse-delete procedure. Formally, let  $w_1, \dots, w_\ell$  be the nodes of  $S$  in the order they were added to  $S$  by the algorithm, where for nodes  $w_i, w_j$  that were added during the same iteration if  $w_i$  is in a pair and  $w_j$  is not, then  $i < j$ . That is, for reverse-delete purposes, nodes not in a pair are considered for deletion first. For  $p = \ell, \ell - 1, \dots, 1$ , if  $w_p$  is not in a node pair, then if  $S \setminus \{w_p\}$  is a feasible ECT, the algorithm deletes  $w_p$  from  $S$ ; otherwise, it does not. If  $w_p$  is in a node pair  $(w_p, w')$ , then if  $S \setminus \{w_p, w'\}$  is a feasible hitting set, then delete both  $w_p, w'$  from  $S$ ; else, keep both  $w_p, w'$ .

The intuition behind the caveat in our reverse-delete step is that node pairs are often very useful to keep, because they disconnect a piece. Consider the example in Figure 6. There is a piece with green nodes of cost 2, and an odd number of length-5 faces with red and blue

## 25:10 Hitting Weighted Even Cycles in Planar Graphs

■ **Algorithm 2.1** EvenCycleTransversal( $G, c$ ).

---

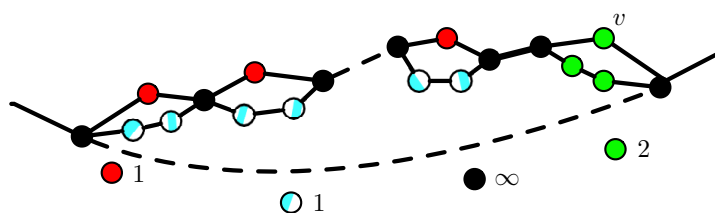
**Input** : A planar graph  $G$  with node costs  $c : V(G) \rightarrow \mathbb{N}$ .  
**Output** : An even cycle transversal  $S$  of  $G$  of cost at most  $\frac{47}{7} \text{opt}(G, c)$ .

- 1  $S \leftarrow \emptyset$
- 2 **while** residual graph  $G^S$  contains an even cycle **do**
- 3     **if**  $G^S$  contains a cycle  $C$  with at most 2 outside neighbours **then**
- 4         | increase the dual variable  $y_C$  for  $C$  until a node  $v$  becomes tight.
- 5     **else**
- 6         | compute the 2-compression  $G_2^S$  of  $G^S$ .
- 7          $H \leftarrow$  minimal pocket of  $G_2^S$ .
- 8          $\mathcal{T}_H \leftarrow$  a 2/3-quasi-perfect tiling of  $H$ .
- 9         Increment dual variables of blended inequalities of all  $C \in \mathcal{T}_H$  until a node  $v$  becomes tight or the blended inequality changes.
- 10         Denote by  $X$  the set of nodes that became tight, and add  $X$  to  $S$ .
- 11         **for** each handle pair  $(Q_1, Q_2)$  **do**
- 12             | **if**  $X$  contains a node in the interior of each handle **then**
- 13                 | choose two nodes  $a, b \in X$  with  $a$  in the interior of  $Q_1$  and  $b$  in the interior of  $Q_2$  and define  $(a, b)$  to be a node pair.
- 14      $w_1, \dots, w_\ell \leftarrow$  nodes of  $S$  in the order they were added, where for nodes  $X$  added during the same iteration, any node of  $X$  in a pair appears before others node of  $X$  not in pairs.
- 15     **for**  $i = \ell$  **downto** 1 **do**
- 16         | **if**  $w_i$  is not part of a pair **then**
- 17             | **if**  $S \setminus \{w_i\}$  is feasible **then**
- 18                 |  $S \leftarrow S \setminus \{w_i\}$ .
- 19         | **else**
- 20             | Let  $(w_i, w_j)$  be the pair containing  $w_i$ . **if**  $S \setminus \{w_i, w_j\}$  is feasible **then**
- 21                 |  $S \leftarrow S \setminus \{w_i, w_j\}$ .
- 22 **return**  $S$

---

striped nodes of cost 1. The black nodes have cost infinity. The bottom dashed path has odd length. In the 2-compression shown on the right, all length-5 faces in the figure belong to one piece. Suppose the blended inequality chooses the length-5 face with the green nodes as the special cycle, and increments the blended inequality for this graph. One sees that the red, blue striped and green nodes become tight simultaneously.

To see that reverse delete orders need to be chosen carefully, consider the following adversarial ordering: in reverse delete, consider the two green nodes other than  $v$  first, then consider the red nodes, and then consider one blue striped node on each handle. Finally, consider the remaining blue striped nodes. One can see that the algorithm would end up with  $v$  and one blue striped node per handle, which is significantly more costly than the optimum which selects the solution consisting of one red and one blue striped node on a handle pair. This completes the description of our approximation algorithm for ECT, whose complete pseudo-code is given as Algorithm 2.1.



■ **Figure 6** The red and blue striped nodes have weight 1, black nodes have infinite weight and green nodes have cost 2. The bottom dashed black path has odd length. The number of length-5 faces at the top is assumed to be even.

## 2.5 Analysis of approximation ratio

We claim the algorithm to be a  $47/7$ -approximation for ECT on node-weighted planar graphs.

Fix an input planar graph  $G$  with node costs  $c_v \in \mathbb{N}$ . Consider a set  $S \subseteq V(G)$  of nodes and a node  $v \in S$ . A cycle  $C$  is a *pseudo-witness cycle* for  $v$  with respect to  $S$  if  $C \cap S = \{v\}$ . If  $C$  is additionally even, then  $C$  is a *witness cycle* for  $v$ . Note that if  $S$  is an inclusion-minimal ECT for  $G$ , then there is a set  $W_v$  of witness cycles for each node in  $v \in S$ . If the reverse-delete procedure does not delete any node of  $S$ , then each node not in a pair has a witness cycle and for each pair, at least one of the nodes in the pair has a witness cycle.

The analyses of the algorithms by Goemans and Williamson [14] and by Berman and Yaroslavtsev [6] for SUBSET FVS on planar graphs rely crucially on the fact that, each node of an inclusion-wise minimal solution has a witness cycle. Goemans and Williamson [14] showed that one can find a laminar collection  $\mathcal{A}$  of witness cycles. Laminar families are well-known to have a natural tree representation. The key argument of both algorithms is that for each *leaf cycle*  $C$  of the laminar family, one can increment the dual variable of at least one face contained in the region defined by  $C$ . Further, this dual variable pays only for the hit node that  $C$  is a witness of. This is used to argue that a large portion of the dual variables they incremented pay for a single hit node. An additional bound on how many nodes the other dual variables pay for is proven exploiting the sparsity of planar graphs.

For ECT, however, we do not have laminar witness cycles. Instead, we must extend the analysis of Berman and Yaroslavtsev [6] to find a set of laminar pseudo-witness cycles.

Consider some time  $\bar{t}$  during the algorithm when applied to  $(G, c)$ . Let  $S_{\bar{t}}$  be the current hitting set and  $G^{S_{\bar{t}}}$  the residual graph. Let  $\{\sum_{v \in V(G)} a_v^C \geq 1\}_{C \in \mathcal{L}}$  be the set of inequalities of the increased dual variables. Here,  $\mathcal{L}$  will be either a single cycle of  $G^{S_{\bar{t}}}$ , or a tiling of  $G_2^{S_{\bar{t}}}$ . We wish to show that the primal increase rate towards the final set  $S'$  at time  $\bar{t}$ ,  $\sum_{C \in \mathcal{L}} \sum_{v \in S'} a_v^C$  is at most  $47/7$  times the dual increase rate  $|\mathcal{L}|$ .

If the algorithm incremented  $y_C$ , where  $C$  was a cycle of  $G$  for which at most two nodes have outside neighbours, then the inequality we increase is  $\sum_{v \in C} x_v \geq 1$ . As  $S'$  is minimal under reverse-delete,  $|C \cap S'| \leq 2$ , and hence the primal increase rate  $\sum_{v \in S'} a_v^C = |C \cap S'|$  is at most twice the dual increase rate 1.

Otherwise, if the algorithm did not increment  $y_C$ , then there is no cycle  $C$  of  $G^{S_{\bar{t}}}$  such that at most two nodes of  $C$  have neighbours outside  $C$ . Hence, the set of increased inequalities are the blended inequalities of a tiling  $\mathcal{T}_H$  of an inclusion-minimal pocket  $H$  of  $G_2^{S_{\bar{t}}}$ . For a cycle  $C$  of  $G_2^{S_{\bar{t}}}$ , let  $\sum_{v \in V(G^{S_{\bar{t}}})} a_v^C \geq 1$  be the blended inequality  $C$  (see Equation  $\circledast$ ).

Recall that informally speaking, we wish to pay for at most one hit node inside a piece. To do this, we need the following theorem which generalizes a result by Fiorini et al. [11, Theorem 5.7] and tells us the structure of a minimal solution within a piece.

► **Theorem 10.** *Let  $S'$  be the output of Algorithm 2.1 on input  $(G, c)$ . Consider an edge  $uw \in E(G_2^{S_i})$  on the even cycle whose dual variable we increase, and let  $Q$  be the piece corresponding to  $uw$  in  $G$ . Then exactly one of the following occurs:*

- (C1)  $S'$  contains no internal node of  $Q$ ,
- (C2)  $S'$  contains exactly one node of  $Q$ , and this node is a cut-node of  $Q$ ,
- (C3)  $S'$  contains exactly two nodes of  $Q$ , and they belong to opposite handles of a cycle of  $Q$ ,
- (C4)  $S'$  contains exactly one node per elementary cycle of  $Q$ , each belonging to the interior of some handle of the corresponding cycle.

**Proof.** If  $S'$  contains two nodes  $a$  and  $b$  in the interiors of different handles of a pair, then since removing both  $a$  and  $b$  disconnects  $u$  from  $w$  in  $Q$ , our algorithm would delete all other nodes of  $V(Q) \setminus \{u, w\}$  from  $S'$ . If  $u$  or  $w$  were in  $S'$ , then our algorithm would delete both  $a$  and  $b$ . Thus,  $u, w \notin S'$ , and case (C3) holds.

Similarly, if  $S'$  contains a cut node  $z$ , then since removing  $z$  disconnects from  $u$  from  $v$  in  $Q$ , our algorithm would delete all other nodes of  $V(Q) \setminus \{u, v\}$  from  $S'$ . If  $u$  or  $w$  were in  $S'$ , then our algorithm would delete  $z$ . Thus,  $u, w \notin S'$ , and case (C2) holds.

If  $u$  or  $w$  is in  $S'$ , then for any  $r \in S' \cap (V(Q) \setminus \{u, w\})$  there cannot be an even cycle of  $G$  which intersects  $S'$  only at  $r$  as such a cycle would have to go through  $u$  or  $w$ , and thus  $S'$  contains no internal node of  $Q$  and case (C1) holds.

Assume that cases (C1), (C2) and (C3) do not hold, so  $u, w \notin S'$ . Let  $(P_1, P_2)$  be a handle pair on  $Q$  such that  $P_1$  contains a hit node  $t$  in its interior and  $P_2$  does not. Suppose that  $Y_1, Y_2$  was another handle pair with no hit node on either of  $Y_1$  or  $Y_2$ . By our deletion procedure, there must be an even cycle  $C$  which intersects  $S'$  at  $t$  only. Such a cycle  $C$  uses the handle  $P_1$  and one handle  $Y_i$  of the pair  $Y_1, Y_2$ . Let  $C'$  be the cycle obtained from  $C$  by replacing the paths  $P_1$  and  $Y_i$  in  $C$  by the paths  $P_2$  and  $Y_{3-i}$ . Since the lengths of different handles of a pair have different parity,  $C'$  is even. Since  $P_2, Y_1$  and  $Y_2$  contain no nodes of  $S'$ ,  $C'$  contains no nodes of  $S'$ , which is a contradiction. Since a handle can only contain one hit node of  $S'$ , this implies that case (C4) holds. ◀

Given a hitting set  $S'$  output by Algorithm 2.1, we wish to construct a corresponding hitting set for  $G_3^{S_i}$  such that the primal increase rate of any particular blended inequality (with respect to  $S'$ ) is equals the number of nodes of  $S'_3$  on the corresponding cycle of  $G_3^{S_i}$ .

► **Definition 11.** *Let  $S'$  be a hitting set output by Algorithm 2.1. The corresponding hitting set for  $G_3^{S_i}$  is the set  $S'_3 \subset V(G_3^{S_i})$  obtained by first taking the nodes of  $S' \cap V(G_3^{S_i})$ . Now, consider an edge  $uv$  of  $G_2^{S_i}$  with corresponding piece  $P$ . Replace  $uv$  by the path  $uw_p v$  in  $G_3^{S_i}$ , and add  $w_p$  to  $S'_3$  if  $P - S'$  has two components.<sup>3</sup>*

▷ **Claim 12.** Let  $C$  be the preimage of an even cycle in  $G_2^{S_i}$ , and  $C_3$  the corresponding cycle in  $G_3^{S_i}$ . We claim  $\sum_{v \in S'} a_v^C \leq |C_3 \cap S'_3| + 1$ . Further, if  $C$  does not contain a twin edge, then  $\sum_{v \in S'} a_v^C \leq |C_3 \cap S'_3|$ .

**Proof.** Define  $b^C$  as follows: For a handle pair, while one handle has greater residual cost than the other set  $b_v^C = 1$  for  $v$  on the handle of greater residual cost  $b_v^C = 0$  on internal nodes of the other handle (change  $b^C$  whenever residual costs become equal). Otherwise,  $b_v^C = 1/2$  on internal nodes of both handles. In short,  $b_v^C$  are the coefficients  $a_v^C$  if we had not redefined  $a_v^C = 1$  for nodes on the special cycle.

<sup>3</sup> Note that the minimality of  $S'$  implies that removing  $S'$  from  $P$  yields at most two connected components.

Let  $uw \in E(G_2^{S_i})$ ,  $Q$  be the preimage of  $uw$  in  $G^{S_i}$  and  $uw_Qw$  be the subdivision of  $uw$  in  $G_3^{S_i}$ . Let  $S'_3$  be the corresponding hitting set of  $S'$  for  $G_3^{S_i}$ . We claim  $\sum_{v \in S' \cap (Q \setminus \{u,w\})} b_v^C = |S'_3 \cap \{w_Q\}|$ . We decide which case of Theorem 10 is satisfied by  $uw$  and  $S'$ .

- If  $uw$  and  $S'$  satisfy (C1), then  $\sum_{v \in S' \cap (Q \setminus \{u,w\})} b_v^C = 0$ . Since  $S'$  contains no internal node of  $Q$ ,  $Q \setminus S$  is connected, and hence  $S'_3$  does not contain  $w_Q$ . Hence  $\sum_{v \in S' \cap (Q \setminus \{u,w\})} b_v^C = |S'_3 \cap \{w_Q\}|$ .
- If  $uw$  and  $S'$  satisfy (C2) or (C3), then  $S'$  does not contain either end node of  $Q$ , and contains either a single cut node of  $Q$ , or exactly two nodes of  $Q$  in the interiors of two handles of a handle pair of  $Q$ . Thus,  $S' \cap Q$  consists either of a single node  $v$  for which  $b_v^C = 1$ , or two nodes  $j, k$  for which  $b_j^C = b_k^C = 1/2$ , and so  $\sum_{v \in S' \cap Q} b_v^C = 1$ .

In case (C2) or (C3),  $Q \setminus S'$  is disconnected, so  $|S'_3 \cap \{w_Q\}| = 1$ . Hence,  $\sum_{v \in S' \cap (Q \setminus \{u,w\})} b_v^C = |S'_3 \cap \{w_Q\}|$ .

- Suppose  $S'$  satisfies (C4). Suppose, for sake of contradiction that, Algorithm 2.1 added a node pair  $(\ell', m)$  on some handle pair  $(P_1, P_2)$  of  $Q$ . It then follows from the reverse-delete step that the final solution  $S'$  contains both  $\ell'$  and  $m$ , or none of them. Since we do not contain a node pair, the deletion procedure of Algorithm 2.1 implies the algorithm did not add a node pair with nodes in  $Q$ . Hence, throughout the algorithm, for each handle pair  $(P_1, P_2)$  of  $Q$ , the handle  $P_i$ , which contains a hit node in its interior must have strictly less residual cost than the other. Hence  $b_v^C = 0$  on handle  $P_i$ . This implies

$$\sum_{v \in (V(Q) \setminus \{u,w\})} b_v^C = 0 . \tag{1}$$

Thus  $\sum_{v \in S' \cap (Q \setminus \{u,w\})} b_v^C = |S'_3 \cap \{w_Q\}|$ .

Let  $C = v_1 v_2 \dots v_\ell v_1$ . Let  $Q_i$  be the piece corresponding to  $v_i v_{i+1 \bmod \ell}$ . Let  $q_i$  be the node resulting from subdividing  $v_i v_{i+1 \bmod \ell}$  in  $G_2^{S_i}$  to obtain  $G_3^{S_i}$ . Let  $C_3 := v_1 q_1 v_2, q_2, \dots, v_\ell q_\ell$  the cycle corresponding to  $C$  in  $G_3^{S_i}$ . We showed

$$\sum_{v \in S' \cap (Q_i \setminus \{u,w\})} b_v^C = |S'_3 \cap \{q_i\}| . \tag{2}$$

Summing (2) for  $i = 1, \dots, \ell$  yields  $\sum_{v \in S' \cap (\cup_{i=1}^{\ell} Q_i \setminus \{v_1, v_2, \dots, v_\ell\})} b_v^C = |\{q_1, q_2, \dots, q_\ell\} \cap C_3|$ .

Noting  $b_{v_i}^C = 1$  for each  $i$  and  $b_v^C = 0$  for  $v \notin \cup_{j=1}^{\ell} Q_j$ , yields

$$\sum_{v \in S'} b_v^C = |C_3 \cap S'_3| . \tag{3}$$

Let us now relate  $a_v^C$  to  $b_v^C$ . If  $C$  has no twin edge, then the blended inequality coefficients  $a_v^C$  are equal to  $b_v^C$ , therefore  $\sum_{v \in S} a_v^C = |C_3 \cap S'_3|$ .

In general,  $C$  may contain a twin edge. In this case,  $a_v^C$  differs from  $b_v^C$  only in the interior of the handles  $H_1, H_2$  of the special cycle: then either  $b_v^C = \frac{1}{2}$  in the interior of  $H_1$  and  $H_2$ , or  $b_v^C = 0$  in the interior of the dominant handle, and  $b_v^C = a_v^C$  everywhere else.

If  $b_v^C = \frac{1}{2}$  in the interior of  $H_1$  and  $H_2$ , then note from Theorem 10 there are at most two nodes of  $S'$  on  $H_1 \cup H_2$ . Thus,  $\sum_{v \in S} a_v^C \leq \sum_{v \in S} b_v^C + 1$ .

Otherwise,  $b_v^C = 0$  in the interior of the dominant handle, and  $b_v^C = a_v^C$  everywhere else. Since  $S$  contains at most one node from the dominant handle  $\sum_{v \in S} a_v^C \leq \sum_{v \in S} b_v^C + 1$ . Thus,  $\sum_{v \in S} a_v^C \leq |C_3 \cap S'_3| + 1$  completing the proof.  $\triangleleft$

To show that  $|C_3 \cap S'_3| + 1$  is small on average we need the fact that  $S'_3$  is a minimal ECT, which is stated in the following remark.

## 25:14 Hitting Weighted Even Cycles in Planar Graphs

► **Remark 13.** Let  $S'$  be the output of Algorithm 2.1 on input  $(G, c)$ . Let  $S'_3$  be the corresponding hitting set for  $G_3^{S'_i}$  in Definition 11. Then each  $v \in S'_3$  has a witness cycle.

For a node  $h$  and cycle  $C$ , denote by  $C \circ h$  that  $h$  lies on  $C$ .

► **Definition 14.** Let  $\mathcal{R}$  be a set of cycles of a graph  $G$ , and let  $S \subset V(G)$ . The debit graph for  $\mathcal{R}$  and  $S$  is the bipartite graph  $\mathcal{D}_G = (\mathcal{R} \cup S, E)$  with edges  $E_{\mathcal{R}} = \{(C, s) \in \mathcal{R} \times S \mid C \circ s\}$ .

Given an embedding of  $G$  and a set  $\mathcal{R}$  of faces of  $G$ , we can obtain an embedding of  $\mathcal{D}_G$  by placing a node  $v_M$  inside the face  $R$  for each  $R \in \mathcal{R}$ . This shows the following observation.

► **Observation 15** ([14, 6]). *If  $\mathcal{R}$  is a set of faces of  $G$ , then the debit graph is planar.*

Note that for  $\mathcal{R}$  a set of cycles, a cycle  $R \in \mathcal{R}$ , the number of nodes  $|R \cap S|$  that  $R$  pays for in the hitting set is the degree of  $R$  in the debit graph.

Recall the definition of the SUBSET FVS problem, which seeks a minimum-weight node set  $X$  which intersects all cycles from  $\mathcal{C}_T$ , the collection of cycles in  $G$  which contain some node from a given set  $T \subseteq V(G)$ . Observe that each node of  $S'_3$  has a witness cycle in  $G_3^{S'_i}$ ; therefore, it is an inclusion-minimal hitting set for the collection  $\mathcal{C}_T$  with  $T = S'_3$ . Goemans and Williamson [14, Lemma 4.2] showed that any inclusion-minimal hitting set for  $\mathcal{C}_T$  has a laminar set of witness cycles, which implies that there is a laminar set of pseudo-witness cycles  $\mathcal{A}$  for hitting set  $S'_3$ .

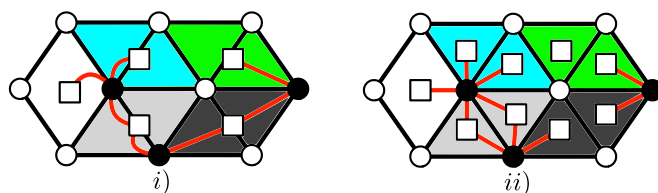
► **Proposition 16** ([14, Lemma 4.2 specialized for SUBSET FVS]). *Let  $G'$  be a planar graph and let  $T \subseteq V(G')$ . Let  $\mathcal{C}_T$  be the set of cycles of  $G'$  containing at least one node of  $T$ , and let  $X$  be an inclusion-minimal hitting set for  $\mathcal{C}_T$ . Then there is a laminar set of cycles  $\mathcal{A} = \{A_x \mid x \in X\}$ , satisfying  $A_x \in \mathcal{C}_T$  and  $A_x \cap X = \{x\}$ .*

Applying Proposition 16 to  $G' = G_3$  and  $X = T = S'_3$  implies there is a laminar set  $\mathcal{A} = \{A_x \mid x \in S'_3\}$  of cycles satisfying  $A_x \cap S'_3 = \{x\}$ . In other words,  $\mathcal{A}$  is a laminar set of pseudo-witness cycles for  $S'_3$ . Note that cycles of  $\mathcal{A}$  may not be even, hence they may be pseudo-witness cycles for  $S'_3$ , but not necessarily witness cycles for nodes of  $S'_3$ .

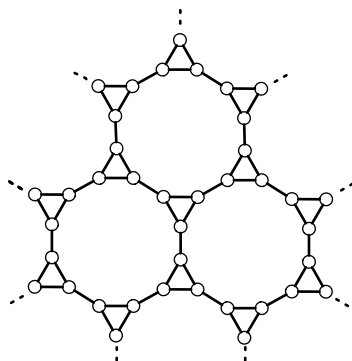
Recall that, during the current iteration, our algorithm incremented the blended inequalities of the cycles in a  $2/3$ -quasi-perfect tiling  $\mathcal{T}_H$  of  $H$ . Recall  $H$  is an inclusion-minimal pocket of  $G_2^{S'_i}$ . By abuse of notation, let  $\mathcal{T}_H$  be the corresponding cycles of  $G_3^{S'_i}$ . Let  $\mathcal{D}$  be the debit graph formed using  $G_3^{S'_i}$ , the cycle set  $\mathcal{T}_H$  and hitting set  $S'_3$ .

Obtain graph  $\mathcal{D}'$  from  $\mathcal{D}$  by replacing each even cycle  $C$  containing two faces with the two faces that compose it. To be precise, construct  $\mathcal{D}'$  by first taking all nodes of  $S'_3$  and all faces of  $H$  that lie inside some even cycle of  $\mathcal{T}_H$  as the vertex set. For each edge  $(C, v) \in E(\mathcal{D})$ , if the cycle  $C$  consist of two faces  $f_1, f_2$  add the edges  $(f_1, v)$  and  $(f_2, v)$  to  $\mathcal{D}'$ , otherwise add the edge  $(C, v)$  to  $\mathcal{D}'$  (see Figure 7). Delete isolated vertices from  $\mathcal{D}'$ . If  $f_i$  is not incident to any hit nodes  $v$ , we remove  $f_i$  from  $\mathcal{D}'$ . Let  $\mathcal{T}_{\text{Faces}(H)}$  be the “face nodes” of  $\mathcal{D}'$ . Let  $\mathcal{F}_{\text{all}(H)}$  denote the finite faces of  $H$ . Let  $\mathcal{F}_H$  denote the set of finite faces of  $H$  that contain a hit node. Observe that  $M \cap S'_3 = \emptyset$  for each  $M \in \mathcal{F}_{\text{all}(H)} \setminus \mathcal{F}_H$ . Now

$$\begin{aligned} \sum_{M \in \mathcal{T}_H} |M \cap S'_3| &\leq \sum_{M \in \mathcal{T}_{\text{Faces}(H)}} |M \cap S'_3| \\ &\leq \sum_{M \in \mathcal{F}_{\text{all}(H)}} |M \cap S'_3| - |\mathcal{F}_H \setminus \mathcal{T}_{\text{Faces}(H)}| = \sum_{M \in \mathcal{F}} |M \cap S'_3| - |\mathcal{F}_H \setminus \mathcal{T}_{\text{Faces}(H)}|. \end{aligned} \quad (4)$$



■ **Figure 7** Left: A possible debit graph  $\mathcal{D}$  with the cycles of the tiling in Figure 5. Right: the graph  $\mathcal{D}'$  obtained by replacing each cycle with the faces that compose it.



■ **Figure 8** A graph consisting of a tessellation of the plane with twice as many triangles as dodecagons. None of the triangles are adjacent, so a maximum tiling covers only the even dodecagons.

The first inequality holds, because for each cycle  $C$  consisting of two faces  $f_1$  and  $f_2$  we have  $|C \cap S'_3| \leq |f_1 \cap S'_3| + |f_2 \cap S'_3|$ . The second inequality holds, because each face of  $\mathcal{F}_H$  contains a hit node, and so  $|C \cap S'_3| \geq 1$  for each  $C \in \mathcal{F}_H$ . The last inequality holds, because by definition  $|M \cap S'_3| = 0$  for all  $M \in \mathcal{F}_{\text{all}(H)} \setminus \mathcal{F}_H$ .

If our tiling covers  $2/3$  of all finite faces, then  $|\mathcal{T}_{\text{Faces}(H)}| \leq 2|\mathcal{T}_H|$  and  $(2/3)|\mathcal{F}_H| \leq |\mathcal{T}_{\text{Faces}(H)}|$ , so  $|\mathcal{F}_H| \leq 3|\mathcal{T}_H|$ . Alas, one can show that a tiling that covers  $2/3$  of all finite faces does not always exist; see Figure 8. To overcome this impediment, we will show that  $|\mathcal{F}_H| \leq 3|\mathcal{T}_H|$  holds for a  $2/3$ -quasi-perfect tiling. Suppose that our  $2/3$ -quasi-perfect tiling covers a  $b$ -fraction of the odd faces in  $\mathcal{F}_H$ , and a  $c$ -fraction of the faces in  $\mathcal{F}_H$  which are even. Let  $\mathcal{F}_{\text{even}(H)}$  be the even finite faces of  $\mathcal{F}_H$ . Then, as  $\mathcal{F}_H \setminus \mathcal{F}_{\text{even}(H)}$  are the odd faces of  $\mathcal{F}_H$ , and  $\mathcal{T}_{\text{Faces}(H)} \setminus \mathcal{F}_{\text{even}(H)}$  are the odd faces covered by our tiling, it holds that  $b|\mathcal{F}_H \setminus \mathcal{F}_{\text{even}(H)}| = |\mathcal{T}_{\text{Faces}(H)} \setminus \mathcal{F}_{\text{even}(H)}|$ . Simplifying, we get

$$b|\mathcal{F}_H| + (1 - b)|\mathcal{F}_{\text{even}(H)}| \leq |\mathcal{T}_{\text{Faces}(H)}| \leq 2|\mathcal{T}_H| - |\mathcal{F}_{\text{even}(H)}| .$$

By rearranging, we get  $b|\mathcal{F}_H \setminus \mathcal{F}_{\text{even}(H)}| + 2|\mathcal{F}_{\text{even}(H)}| \leq 2|\mathcal{T}_H|$ . Noting that  $b(1 - c) + 2c \geq 2/3$ , and rearranging once more, yields

$$\frac{2}{3}|\mathcal{F}_H| \leq b|\mathcal{F}_H \setminus \mathcal{F}_{\text{even}(H)}| + 2|\mathcal{F}_{\text{even}(H)}| \leq |\mathcal{T}_{\text{Faces}(H)}| \leq 2|\mathcal{T}_H| .$$

Noting that  $|\mathcal{F}_{\text{even}(H)}|/|\mathcal{F}_H| = c$  and  $b(1 - c) + 2c \geq 2/3$ , we get

$$3|\mathcal{T}_H| \geq \frac{3}{2}(b(1 - c) + 2c)|\mathcal{F}_H| \geq |\mathcal{F}_H| . \tag{5}$$

By (4), in order to bound  $\sum_{M \in \mathcal{T}_H} |M \cap S'_3|$ , it suffices to bound  $\sum_{M \in \mathcal{F}} |M \cap S'_3|$ . To do this, we prove the following extension of the work by Berman and Yaroslavtsev [6, Theorem 4.1].

## 25:16 Hitting Weighted Even Cycles in Planar Graphs

► **Theorem 17.** *Let  $H$  be an inclusion-wise minimal pocket of  $G$ . Let  $S \subset V(G)$  be a set of nodes with some set  $\mathcal{A}$  of laminar pseudo-witness cycles. Let  $\mathcal{R}$  be a set of finite faces of  $H$  such that each cycle of  $\mathcal{A}$  contains a face of  $\mathcal{R}$  in its interior. Then  $\sum_{M \in \mathcal{R}} |M \cap S| \leq \frac{18}{7} |\mathcal{R}|$ .*

We defer the proof of Theorem 17 to Subsection A.1.

Let  $\mathcal{A}$  be a set of laminar witness cycles for  $S'_3$ . If we were to set  $\mathcal{R} = \mathcal{F}_H$  (the set of finite faces of  $H$  incident to a hit node), then each cycle  $A \in \mathcal{A}$  contains a face of  $\mathcal{R}$  in its interior, namely any face inside  $A$  that is incident to the hit node of  $S'_3$  on  $A$ . Thus,  $S'_3, \mathcal{A}$  and  $\mathcal{R}$  meet the conditions of Theorem 17.

To recap, we wish to bound the primal increase rate  $\sum_{M \in \mathcal{T}_H} \sum_{v \in S} a_v^M$ , so we analyze the expression  $\sum_{M \in \mathcal{T}_H} |M \cap S'_3|$ . Recall from Claim 12 that  $\sum_{v \in S} a_v^M$  is at most one more than  $|M \cap S'_3|$  and  $\sum_{v \in S} a_v^M = |M \cap S'_3|$  if  $M$  contains no twin edge. We bound  $\sum_{M \in \mathcal{T}_H} |M \cap S'_3|$  by looking at the quantity  $\sum_{M \in \mathcal{F}_H} |M \cap S'_3|$ , because  $\mathcal{F}_H$  fits the conditions of Theorem 17. One could then use  $|\mathcal{F}_H| \leq 3|\mathcal{T}_H|$  (by (5)), to bound  $\sum_{M \in \mathcal{T}_H} \sum_{v \in S} a_v^M$  in terms of the dual increase rate  $|\mathcal{T}_H|$ . We will use  $3|\mathcal{T}_H| \geq \frac{3}{2}(b(1-c) + 2c)|\mathcal{F}_H|$  to obtain a stronger bound.

Let  $\mathcal{T}$  be our 2/3-quasi-perfect tiling from Theorem 9. Recall from Definition 8 that the fraction  $\beta$  of odd finite faces that are covered by the tiling, and the fraction  $\psi$  of finite faces of  $H$ , that are even satisfy  $\beta(1-\psi) + 2\psi \geq \alpha$ . Let  $\mathcal{A}$  be a set of pseudo-witness cycles in  $H$  for  $S'_3$ , the corresponding set for the hitting set  $S'$  returned by our algorithm. Define  $\mathcal{R} = \mathcal{F}_H$ . We have that every cycle of  $\mathcal{A}$  contains a face of  $\mathcal{R}$  in its interior. Thus,  $\mathcal{R}, \mathcal{A}$  and  $S'_3$  satisfy the conditions of Theorem 17. Therefore,

$$\sum_{M \in \mathcal{T}_H} |M \cap S'_3| \leq \left( \sum_{M \in \mathcal{F}_H} |M \cap S'_3| \right) - |\mathcal{F}_H \setminus \mathcal{T}_{\text{Faces}(H)}| \leq \frac{18}{7} |\mathcal{F}_H| - |\mathcal{F}_H \setminus \mathcal{T}_{\text{Faces}(H)}|. \quad (6)$$

Note that  $\sum_{v \in S} a_v^M \leq |M \cap S|$ , unless  $M$  contains a twin edge. If  $M \in \mathcal{T}$  is the disjoint union of two odd faces which share an edge, then  $M$  will not contain a twin edge. That is,  $M$  can only contain a twin edge if  $M \in \mathcal{F}_{\text{even}(H)}$ , so  $M$  is an even face then. So

$$\sum_{M \in \mathcal{T}_H} \sum_{v \in S} a_v^M \leq \sum_{M \in \mathcal{T}_H} |M \cap S| + |\mathcal{F}_{\text{even}(H)}| \leq \frac{18}{7} |\mathcal{F}_H| - |\mathcal{F}_H \setminus \mathcal{T}_{\text{Faces}(H)}| + |\mathcal{F}_{\text{even}(H)}|. \quad (7)$$

Recall that  $c = |\mathcal{F}_{\text{even}(H)}|/|\mathcal{F}_H|$  is the fraction of finite faces of  $\mathcal{F}_H$  which are even, and that  $b = |\mathcal{T}_{\text{Faces}(H)} \setminus \mathcal{F}_{\text{even}(H)}|/|\mathcal{F}_H \setminus \mathcal{F}_{\text{even}(H)}|$  is the fraction of odd finite faces of  $\mathcal{F}_H$  covered by our tiling. Note that

$$\begin{aligned} |\mathcal{F} \setminus \mathcal{T}_{\text{Faces}(H)}| &= |\mathcal{F}_H \setminus \mathcal{F}_{\text{even}(H)}| - |\mathcal{T}_{\text{Faces}(H)} \setminus \mathcal{F}_{\text{even}(H)}| \\ &= |\mathcal{F} \setminus \mathcal{F}_{\text{even}(H)}| - b|\mathcal{F}_H \setminus \mathcal{F}_{\text{even}(H)}| = (1-b)(1-c)|\mathcal{F}_H|. \end{aligned}$$

We now recall (5), by which  $3|\mathcal{T}_H| \geq \frac{3}{2}(b(1-c) + 2c)|\mathcal{F}_H|$ .

Substituting these bounds for  $|\mathcal{F}_H|$  and  $|\mathcal{F}_H \setminus \mathcal{T}_{\text{Faces}(H)}|$  into (7), we obtain

$$\begin{aligned} \sum_{M \in \mathcal{T}_H} \sum_{v \in S} a_v^M &\leq c|\mathcal{F}_H| + \frac{18}{7} \left( \frac{2}{b(1-c) + 2c} |\mathcal{T}_H| \right) - (1-b)(1-c)|\mathcal{F}_H| \\ &= \frac{2c}{b(1-c) + 2c} |\mathcal{T}_H| + \frac{18}{7} \left( \frac{2}{b(1-c) + 2c} |\mathcal{T}_H| \right) - \frac{2(1-b)(1-c)}{b(1-c) + 2c} |\mathcal{T}_H|. \end{aligned}$$

If we maximize the right-hand side factor  $\frac{2c}{(b(1-c)+2c)} + \frac{36}{7(b(1-c)+2c)} - \frac{2(1-b)(1-c)}{(b(1-c)+2c)}$  subject to  $b(1-c) + 2c \geq 2/3$ , we obtain that the right-hand side is bounded by  $\frac{47}{7} |\mathcal{T}_H|$ .

This completes the proof of Theorem 1 modulo the proof of Theorem 9; i.e., the fact that large quasi-perfect tilings can be computed efficiently. We deal with this in the appendix.



## 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 *Proc. STOC 2005*, pages 573–581, 2005.
- 2 Vineet Bafna, Piotr Berman, and Toshihiro Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM J. Discrete Math.*, 12:289–297, 1999.
- 3 Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994.
- 4 Mohammadhossein Bateni, Mohammadtaghi Hajiaghayi, and Dániel Marx. Approximation schemes for Steiner forest on planar graphs and graphs of bounded treewidth. *J. ACM*, 58(5), 2011.
- 5 Ann Becker and Dan Geiger. Optimization of Pearl’s method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artif. Intelligence*, 83(1):167–188, 1996.
- 6 Piotr Berman and Grigory Yaroslavtsev. Primal-dual approximation algorithms for node-weighted network design in planar graphs. In *Proc. APPROX 2012*, volume 7408 of *Lecture Notes Comput. Sci.*, pages 50–60, 2012.
- 7 Vincent Cohen-Addad, Éric Colin de Verdière, Philip N. Klein, Claire Mathieu, and David Meierfrankenfeld. Approximating connectivity domination in weighted bounded-genus graphs. In *Proc. STOC 2016*, pages 584–597, 2016.
- 8 Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Bidimensional parameters and local treewidth. *SIAM J. Discrete Math.*, 18(3):501–511, 2004.
- 9 Erik D. Demaine, Mohammadtaghi Hajiaghayi, and Philip N. Klein. Node-weighted Steiner tree and group Steiner tree in planar graphs. *ACM Trans. Algorithms*, 10(3), 2014.
- 10 Yevgeniy Y. Dorfman and Galina Orlova. Finding the maximal cut in a graph. *Engineering Cybernetics*, 10(3), 1972.
- 11 Samuel Fiorini, Gwenaél Joret, and Ugo Pietropaoli. Hitting diamonds and growing cacti. In *Proc. IPCO 2010*, volume 6080 of *Lecture Notes Comput. Sci.*, pages 191–204, 2010.
- 12 Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Bidimensionality and EPTAS. In *Proc. SODA 2011*, pages 748–759, 2011.
- 13 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Bidimensionality and kernels. In *Proc. SODA 2010*, pages 503–510, 2010.
- 14 Michel X. Goemans and David P. Williamson. Primal-dual approximation algorithms for feedback problems in planar graphs. *Combinatorica*, 18(1):37–59, 1998.
- 15 Alexander Göke, Jochen Koenemann, Matthias Mnich, and Hao Sun. Hitting weighted even cycles in planar graphs, 2021. [arXiv:2107.04763](https://arxiv.org/abs/2107.04763).
- 16 Frank Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM J. Comput.*, 4(3), 1975.
- 17 Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.
- 18 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within  $2 - \epsilon$ . *J. Comput. Syst. Sci.*, 74(3):335–349, 2008.
- 19 Daniel Lokshtanov and MS Ramanujan. Parameterized tractability of multiway cut with parity constraints. In *Proc. ICALP 2012*, volume 7391 of *Lecture Notes Comput. Sci.*, pages 750–761, 2012.
- 20 Daniel Lokshtanov, MS Ramanujan, Saket Saurabh, and Meirav Zehavi. Parameterized complexity and approximability of directed odd cycle transversal. In *Proc. SODA 2020*, pages 2181–2200, 2020.
- 21 Pranabendu Misra, Venkatesh Raman, MS Ramanujan, and Saket Saurabh. Parameterized algorithms for even cycle transversal. In *Proc. WG 2012*, volume 7551 of *Lecture Notes Comput. Sci.*, pages 172–183, 2012.

- 22 Carsten Moldenhauer. Primal-dual approximation algorithms for node-weighted Steiner forest on planar graphs. *Inf. Comput.*, 222:748–759, July 2011.
- 23 Martin Nägele and Rico Zenklusen. A new contraction technique with applications to congruency-constrained cuts. *Math. Prog.*, 183:455–481, 2020.

**A** Obtaining a 2/3-quasi-perfect tiling

We now show how to find the 2/3-quasi perfect tiling in line 8 of Algorithm 2.1. The following result states that the minimal pockets picked by the algorithm have such tilings.

► **Theorem 9.** *Let  $H$  be a 2-compression of some planar graph  $G$ , that has an even cycle and contains no pockets. Then  $H$  has a 2/3-quasi-perfect tiling.*

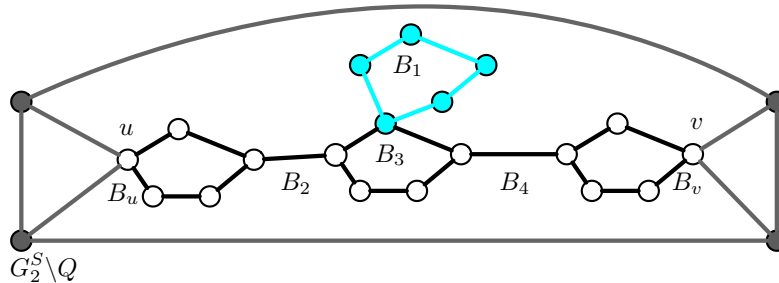
To prove this theorem we will use the following lemma.

► **Lemma 18.** *For any set  $S$ , any pseudo-pocket contained in  $G_2^S$  contains an even cycle.*

**Proof.** Informally speaking, the proof will show that any pseudo-pocket without even cycles contains an odd cycle for which only two nodes have outside neighbours; this, however, cannot appear in the 2-compression, as we would have replaced this cycle by an edge in  $G_2^S$ .

Suppose, for sake of contradiction, that  $G_2^S$  contained a pseudo-pocket  $Q$  without even cycles. Since each node of  $Q$  is in an even cycle of  $G_2$  and  $Q$  contains no even cycle,  $Q$  contains exactly two nodes  $u$  and  $v$  with neighbours outside  $Q$ , and each node of  $Q$  lies on a  $u$ - $v$  path of  $Q$ . Let  $B_u$  and  $B_v$  be the blocks of  $Q$  containing  $u$  and  $v$  in the block graph  $\mathcal{B}$  of  $Q$ , respectively (see Figure 9).

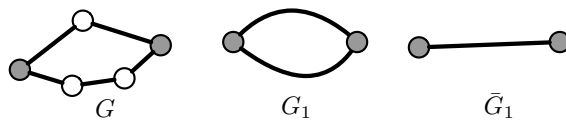
If  $\mathcal{B}$  is not a path, then there would be some block  $B_1$  that does not lie on a  $B_u$ - $B_v$  path in  $\mathcal{B}$ , and thus there would be a node of  $B_1$  that would not lie on a  $u$ - $v$  path in  $Q$  – a contradiction. Hence,  $\mathcal{B}$  is a path. Let  $B$  be a block of  $Q$ . Suppose for a contradiction



■ **Figure 9** Graph  $Q$  consisting of blocks labelled  $B_1, B_2, B_3, B_4, B_u, B_v$ . Block  $B_1$  depicted in blue contains nodes not on any  $u$ - $v$  path, which is a contradiction.

that  $B$  contains a cycle  $C$  and a node  $v'$  of  $C$  with a neighbour  $u' \in V(B)$  outside  $C$ . Since  $v'$  is not a cut node, there is a path  $P$  from  $u'$  to  $C \setminus v'$ . Construct the  $u'$ - $v'$  path  $P'$  from  $P$  by traversing  $P$  from  $u'$  to the first node  $w'$  of  $C \setminus v'$  and appending to that a  $w'$ - $v'$  path in  $C$ . Since  $Q$  contains no even cycles, the cycles  $P' \cup v'u'$  and  $C$  are odd. Then the cycle formed by the edges  $E(C) \Delta E(P' \cup v'u')$ , that is edges of  $C$  or  $P' \cup v'u'$ , but not both, has length  $|E(C)| + |E(P' \cup v'u')| - 2|E(C) \cap E(P' \cup v'u')|$  which is even, and hence a contradiction. Thus if  $B$  contains a cycle then it does not contain nodes outside the cycle, or put simply  $B$  is a cycle. Since we assume  $B$  contains no even cycles,  $B$  is an odd cycle. Thus, the blocks of  $Q$  are odd cycles or edges. Since  $Q$  contains at least one cycle, there is an odd cycle  $C'$ . Since  $\mathcal{B}$  is a path,  $C'$  contains 2 nodes  $a$  and  $b$  with neighbours outside  $C'$ . However,  $G_2^S$

cannot contain such an odd cycle, as that we would have contracted the two  $a$ - $b$  paths of  $C'$  to parallel edges and then replaced them by a twin edge; see Figure 10. This completes the proof. ◀



■ **Figure 10** Cycle is replaced by an edge in 2-compression.

For any set  $S$ , if  $G_3^S$  contained a pseudo-pocket  $Q$  without even cycles, then  $Q$  was obtained from a subgraph  $Q'$  of  $G_2^S$  by subdividing edges. Then  $Q'$  would be a pseudo-pocket of  $G_2^S$  without even cycles. This contradicts Lemma 18. This shows the following corollary.

► **Corollary 19.** *For any set  $S$ , any pseudo-pocket of  $G_3^S$  contains an even cycle.*

Recall from Definition 6 and the paragraph afterwards, that a pseudo-tiling of our graph corresponds to the union of a matching of the dual graph and a set of even faces. A tiling corresponds to the union of a matching of the dual graph not containing any edge incident to the infinite face and a set of even finite faces. Under this correspondence, the existence of large pseudo-tilings is a much more natural thing to prove. Let us first formally define a large pseudo-tiling.

► **Definition 20.** *Let  $\alpha \in (0, 1)$ . A pseudo-tiling  $\mathcal{T}$  is  $\alpha$ -pseudo-perfect if it covers all even faces (including the infinite face if it is even) and a  $\beta$ -fraction of the odd faces, and a  $\psi$ -fraction of the faces of  $H$  are even, where  $\beta(1 - \psi) + 2\psi \geq \alpha$ .*

We will first prove the existence of large pseudo-perfect pseudo-tilings. We fix an embedding of  $H$ . For any multigraph  $W$ , let  $\text{oc}(W)$  be the number of odd components of  $W$ . Recall pseudo-tilings correspond to matchings. Our proof will use Tutte’s Theorem stated below, which informally speaking, says that the absence of a large matching implies the existence of a small set of vertices whose removal results in a graph with a large number of connected components of odd size.

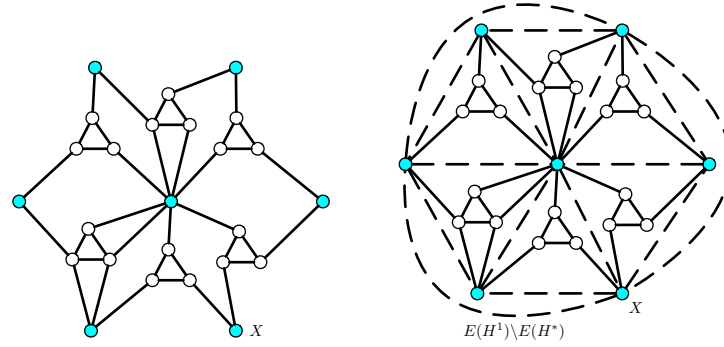
► **Theorem 21 (Tutte’s Theorem).** *For any graph  $G$ , the number of nodes of  $G$  which are not covered by a maximum size matching of  $G$  is at most*

$$\text{oc}(G \setminus X) - |X| . \tag{8}$$

*for some  $X \subset V(G)$ . Further, if some node  $v \in V(G)$  is covered by every maximum matching of  $G$ , then (8) holds for some  $X \subset V(G)$  containing  $v$ .*

The main idea of why such large pseudo-perfect pseudo-tilings should exist is that by Tutte’s Theorem, the absence of a large pseudo-tiling implies that for some set  $X$  of nodes of the dual graph  $H^*$ , the set of odd components of  $H^* \setminus X$  is large relative to  $|X|$ .

Construct a new graph  $H^1$  as follows. Start with the graph  $H^*$  and add as many edges as possible between nodes of  $X$  while preserving planarity and not creating any faces of length two (see Figure 11). We will show that each odd component of  $H^1 \setminus X$  lies in a different face of  $H^1[X]$  and that  $H^1$  contains at most two faces of length two. Thus using Euler’s formula,  $|E(H^1[X])| \leq 3|V(H^1[X])| - 4$ ,  $H^1[X]$  does not have too many edges. The crucial observation is that since each odd component of  $H^1 \setminus X$  lies in a different face of  $H^1[X]$ , each



■ **Figure 11** The graph  $H^*$  with set  $X \subset V(H^*)$  (depicted in blue) on the left. On the right, the graph  $H^1$  obtained from  $H^*$  by adding edges (dashed) between  $X$ .

node  $x \in X$  is adjacent to more other nodes of  $X$  in  $H^1$  than there are odd components of  $H^1 \setminus X$  which contain a neighbour of  $x$ . By facial region, we mean the region of the plane bounded by a face. We will also show there are at most two odd components  $J_1, J_2$  for which at most two nodes of  $X$  have neighbours in  $J_i$ . We can then show that the number of odd components is at most  $2/3$  the number of edges of  $H^1[X]$  plus  $\frac{2}{3}$ , which will contradict that the set of odd components is large.

► **Lemma 22** ( $\star$ ). *Let  $H$  be as in Algorithm 2.1, that is,  $H$  is a minimal pocket of  $G_2^S$ . Then  $H$  has a  $2/3$ -pseudo-perfect pseudo-tiling.*

So let  $\mathcal{T}$  be a  $2/3$ -pseudo-perfect pseudo-tiling of  $H$ . Let  $\beta'$  be the fraction of odd faces of  $H$  which are covered by  $\mathcal{T}$ , and let  $\psi'$  be the fraction of even faces of  $H$ . Next, we will show that if  $\mathcal{T}$  covers more faces than a maximum tiling of  $H$ , then  $\mathcal{T}$  satisfies a slightly stronger condition than  $2/3$ -pseudo-perfect, namely,  $\beta'(1 - \psi')|V(H^*)| + 2\psi'|V(H^*)| \geq \frac{2}{3}|V(H^*)| + \frac{4}{3}$ . Formally, this means:

► **Lemma 23** ( $\star$ ). *Let  $H$  be as in Algorithm 2.1, that is,  $H$  is a minimal pocket of  $G_2^S$ . Suppose that any maximum size pseudo-tiling of  $H$  covers the infinite face. Then  $H$  has a pseudo-tiling covering a  $\beta'$ -fraction of all odd faces such that*

$$\beta'(1 - \psi')|V(H^*)| + 2\psi'|V(H^*)| \geq \frac{2}{3}|V(H^*)| + \frac{4}{3}. \quad (9)$$

► **Theorem 24**. *Let  $H$  be an inclusion-minimal pocket of  $G_2^S$ . Then we can obtain  $2/3$ -quasi-perfect tiling of  $H$  in polynomial time.*

**Proof.** We first show that  $H$  admits a  $2/3$ -quasi-perfect tiling. Let us show that if some tiling  $\mathcal{T}$  is  $2/3$ -pseudo-perfect, then it is  $2/3$ -quasi-perfect. Let  $\beta'$  be the fraction of odd faces of  $H$  that are covered by  $\mathcal{T}$  and  $\psi'$  the fraction of faces of  $H$ , that are even. As  $\mathcal{T}$  is  $2/3$ -pseudo-perfect, it covers all even faces. Since  $\mathcal{T}$  is a tiling, the infinite face is odd. As the number of even finite faces is  $\psi'|V(H^*)|$ , so  $\frac{\psi'|V(H^*)|}{|V(H^*)|-1}$  is the fraction of finite faces of  $H$  that are even.  $(1 - \psi')|V(H^*)|$  is the number of odd faces of  $H$ , so  $\beta'(1 - \psi')|V(H^*)|$  is the number of odd faces of  $H$  covered by  $\mathcal{T}$ . Since the infinite face is odd,  $(1 - \psi')|V(H^*)| - 1$  is the number of odd finite faces. Thus  $\frac{\beta'(1 - \psi')|V(H^*)|}{(1 - \psi')|V(H^*)| - 1}$  is the fraction of odd finite faces of  $H$  covered by  $\mathcal{T}$ . Since

$$\begin{aligned}
 & \frac{\beta'(1-\psi')|V(H^*)|}{(1-\psi')|V(H^*)|-1} \left(1 - \frac{\psi'|V(H^*)|}{|V(H^*)|-1}\right) + \frac{2\psi'|V(H^*)|}{|V(H^*)|-1} \\
 = & \frac{\beta'(1-\psi')|V(H^*)|}{(1-\psi')|V(H^*)|-1} (1-\psi') + 2\psi' + \left(2 - \frac{\beta'(1-\psi')|V(H^*)|}{(1-\psi')|V(H^*)|-1}\right) \left(\psi' - \frac{\psi'|V(H^*)|}{|V(H^*)|-1}\right) \\
 \leq & \frac{\beta'(1-\psi')|V(H^*)|}{(1-\psi')|V(H^*)|-1} (1-\psi') + 2\psi' \leq \frac{2}{3},
 \end{aligned}$$

it holds that  $\mathcal{T}$  is  $2/3$ -quasi-perfect.

If there is a maximum size pseudo-tiling that is also a tiling, then it follows from Lemma 22 that such a tiling is  $2/3$ -quasi-perfect.

Otherwise, if no pseudo-tiling exists, the largest pseudo-tiling is larger than the largest tiling. Let  $\mathcal{T}$  be a maximum size pseudo-tiling.

If the infinite face of  $\mathcal{T}$  is even, consider the tiling  $\mathcal{T}'$  obtained by removing the infinite face from  $\mathcal{T}$ . Let  $\psi^{(1)} := (\psi'|V(H^*)|-1)/(|V(H^*)|-1)$  be the fraction of finite faces of  $H$  which are even. As the infinite face is even,  $\beta'$  is the fraction of odd finite faces of  $H$  which are covered by  $\mathcal{T}'$ . It holds that

$$\begin{aligned}
 \beta'(1-\psi^{(1)})(|V(H^*)|-1) + 2\psi^{(1)}(|V(H^*)|-1) &= \beta'|V(H^*)|(1-\psi') + \psi'|V(H^*)|-1 \\
 &\geq \frac{2}{3}|V(H^*)| + \frac{4}{3} - 1 = \frac{2}{3}(|V(H^*)|-1).
 \end{aligned}$$

So  $\mathcal{T}'$  is  $2/3$ -quasi-perfect.

If the infinite face is odd, consider the tiling  $\mathcal{T}'$  obtained by removing the even cycle covering the infinite face from  $\mathcal{T}$ . Let  $\psi^{(2)} := \psi'|V(H^*)|/(|V(H^*)|-1)$  be the fraction of finite faces of  $H$  that are even. At least  $\beta'|V(H^*)|-2$  of the finite faces of  $H$  are covered by  $\mathcal{T}'$  so the fraction  $\beta''$  of finite odd faces of  $H$  that are covered satisfies  $\beta'' \geq (\beta'|V(H^*)|-1)/(1-\psi^{(2)})(|V(H^*)|-1)$ . Therefore,

$$\begin{aligned}
 \beta''(1-\psi^{(2)})(|V(H^*)|-1) + 2\psi^{(2)}(|V(H^*)|-1) &\geq (\beta'|V(H^*)|-1) + 2c|V(H^*)| \\
 &\geq \frac{2}{3}|V(H^*)| + \frac{4}{3} - 1 = \frac{2}{3}(|V(H^*)|-1).
 \end{aligned}$$

Hence also in this case,  $\mathcal{T}'$  is  $2/3$ -quasi-perfect.

Finally, since a tiling corresponds to the union of a matching and a set of even faces, finding a maximum tiling of  $H$  corresponds to finding a maximum matching of the odd finite faces of  $H$ . Computing such a maximum matching can be done in polynomial time. ◀

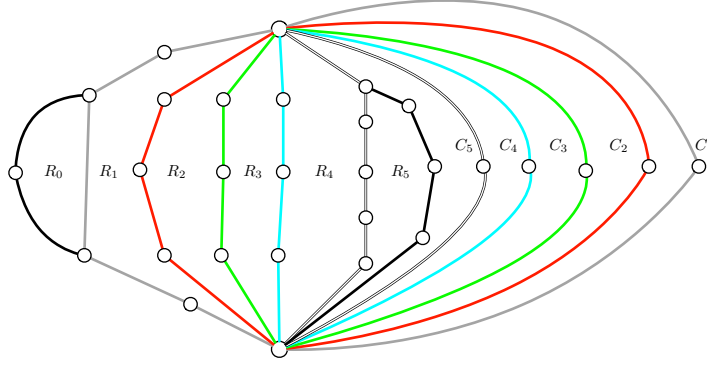
## A.1 Proof of Theorem 17

In this section we will prove Theorem 17. Let  $G, H, \mathcal{R}, S, \mathcal{A}$  be as in the statement of Theorem 17. Let  $\mathcal{D}_G$  be the debit graph of  $G$  with respect to  $S$ .

We introduce the notion of “balance”, which captures for subsets  $\mathcal{R}' \subseteq \mathcal{R}$  of cycles are incident to more or less than  $18/7$  nodes of  $S$  on average.

► **Definition 25.** For each subset  $\mathcal{R}' \subseteq \mathcal{R}$ , its balance  $\text{bal}(\mathcal{R}')$  is the quantity  $|\mathcal{R}'| - \frac{7}{18}|E'_{\mathcal{R}}|$ .

Our proof follows the same methodology as Berman and Yaroslavtsev [6]. First, it shows a pseudo-witness cycle that is not a face and is minimally so, that is any pseudo-witness cycle lying in the finite region bounded by it is a face, has balance at least  $1 - \frac{7}{18}$ . Then it uses this to apply a reduction on  $G$ . We will use the following result of theirs.



■ **Figure 12** Pseudo-witness cycles  $C_1, \dots, C_5$  divide  $H$  into regions  $R_0, R_1, \dots, R_5$ .

► **Proposition 26** ([6, Lemma 4.3]). *Let  $W$  be a planar graph,  $\hat{S}$  be a set of nodes of  $W$  and  $Q \subset \hat{S}$  be a set of nodes of  $W$  that we call outer nodes. Let  $\mathcal{R}_W$  be a set of faces of  $W$  such that each non-outer node of  $\hat{S} \cap W$  has a pseudo-witness cycle in  $\mathcal{R}_W$ . If  $W$  contains a  $\leq 2$  outer nodes, then  $\text{bal}(\mathcal{R}_W) \geq 1 - \frac{7}{18}a$ .*

If all nodes of a pseudo-witness cycle  $A$  are contained in  $H$ , call  $A$  a *hierarchical* pseudo-witness cycle. Otherwise, call  $A$  a *crossing* pseudo-witness cycle. Denote the set of crossing pseudo-witness cycles by  $\hat{\mathcal{A}}$ . We are now ready to complete the proof of Theorem 17. We begin by reductions on our instance  $(G, H, \mathcal{R}, \mathcal{A}, S)$  which simplify our instance and do not increase the balance. If after applying this reduction our instance has positive balance, then our instance had positive balance before the reduction. We define the reduction below.

► **Definition 27.** *We define the following reduction on our instance  $(G, H, \mathcal{R}, \mathcal{A}, S)$ . If  $H$  contains a hierarchical pseudo-witness cycle  $A$  that is not a face of  $\mathcal{R}$ , delete all nodes, edges and faces of  $\mathcal{R}$  inside  $A$  from  $H$  and add  $A$  to  $\mathcal{R}$ . If  $H$  does not contain a hierarchical witness cycle, we call the instance  $(G, H, \mathcal{R}, \mathcal{A}, S)$  reduced.*

Let  $\mathcal{R}_C$  be the faces in  $\mathcal{R}$  contained in the region bounded by  $C$ . Let  $H^1, \mathcal{R}^1$  be the result of applying the reduction in Definition 27 on  $H, \mathcal{R}$ . The balance of  $H^1, \mathcal{R}^1$  is equal to

$$\begin{aligned} & |(\mathcal{R} \setminus \mathcal{R}_C) \cup \{C\}| - \sum_{M \in (\mathcal{R} \setminus \mathcal{R}_C) \cup \{C\}} |M \cap S| \\ &= |\mathcal{R}| - \sum_{M \in \mathcal{R}} |M \cap S| - (|\mathcal{R}_C| + 1 - (\sum_{M \in \mathcal{R}_C} |M \cap S|) + 1) = \text{bal}(H) + 1 - \text{bal}(\mathcal{R}_C) - \frac{7}{18}. \end{aligned}$$

That is to say, the reduction changes the balance by  $1 - \text{bal}(\mathcal{R}_C) - \frac{7}{18}$ , which by Proposition 26 is non-positive. Thus, if after applying the reduction in Definition 27, our instance has positive balance then it initially had positive balance. We know apply the reduction in Definition 27 until our instance is reduced, for simplicity we will continue to call this graph  $H$ .

The crossing pseudo-witness cycles  $\hat{\mathcal{A}}$  partition  $H$  into *regions*, see Figure 12. That is, consider the subgraph  $K \subset H$  consisting of nodes and edges lying on a witness cycle of  $\hat{\mathcal{A}}$  or on the outside face of  $H$ . The regions are defined as the portions of the plane bounded by the finite faces of  $K$ . Define a *subpocket* [6] as the subgraph of  $H$  consisting of the nodes and edges lying in or on the boundary of a region.

► **Proposition 28** ([6]). *The regions that the set of crossing cycles  $\hat{\mathcal{A}}$  partition the plane into satisfy the following. For each region, there is a set  $\tilde{\mathcal{A}}$  of at most two pseudo-witness cycles of  $\hat{\mathcal{A}}$  such that each node bounding the region either does not lie on a pseudo-witness cycle in  $\tilde{\mathcal{A}}$  or lies on a cycle of  $\tilde{\mathcal{A}}$ .*

By the reduction described in Definition 27 each non-crossing cycle of  $\mathcal{A}$  is a face. Since by Proposition 28, the outside face of each subpocket  $W$  contains nodes from at most two crossing pseudo-witness cycles, and contains all nodes that belong to pseudo-witness cycles lie on the outside face, there are at most two hit nodes of  $W$  whose pseudo-witness is not a face and they must lie on the outside face of  $W$ . Hence, each subpocket satisfies the conditions of Proposition 26 and hence has positive balance. Thus,  $H$  has positive balance, that is,  $0 \leq |\mathcal{R}| - \frac{7}{18}|E_{\mathcal{R}}| = |R| - \sum_{M \in R} |M \cap S|$ . Rearranging,  $\sum_{M \in R} |M \cap S| \leq \frac{18}{7}|\mathcal{R}|$ , which completes the proof of Theorem 17.  $\blacktriangleleft$





# Revenue Maximization in Transportation Networks

**Kshipra Bhawalkar** ✉

Google Research, Mountain View, CA, USA

**Kostas Kollias** ✉

Google Research, Mountain View, CA, USA

**Manish Purohit** ✉

Google Research, Mountain View, CA, USA

---

## Abstract

We study the joint optimization problem of pricing trips in a transportation network and serving the induced demands by routing a fleet of available service vehicles to maximize revenue. Our framework encompasses applications that include traditional transportation networks (e.g., airplanes, buses) and their more modern counterparts (e.g., ride-sharing systems). We describe a simple combinatorial model, in which each edge in the network is endowed with a curve that gives the demand for traveling between its endpoints at any given price. We are supplied with a number of vehicles and a time budget to serve the demands induced by the prices that we set, seeking to maximize revenue. We first focus on a (preliminary) special case of our model with unit distances and unit time horizon. We show that this version of the problem can be solved optimally in polynomial time. Switching to the general case of our model, we first present a two-stage approach that separately optimizes for prices and routes, achieving a logarithmic approximation to revenue in the process. Next, using the insights gathered in the first two results, we present a constant factor approximation algorithm that jointly optimizes for prices and routes for the supply vehicles. Finally, we discuss how our algorithms can handle capacitated vehicles, impatient demands, and selfish (wage-maximizing) drivers.

**2012 ACM Subject Classification** Theory of computation → Graph algorithms analysis

**Keywords and phrases** Pricing, networks, approximation algorithms

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.26

**Category** APPROX

## 1 Introduction

The increasing popularity of ride-sharing systems has inspired renewed interest on questions of pricing and routing transportation requests in networks [2, 4, 6, 12, 16]. Typically, such ride sharing platforms have an abundance of data at their disposal, which offers them a good understanding of the market. These data offer insights which can lead to reasonable estimates of the supply of drivers expected at a given time, as well as the number of customers who would be interested in taking a given trip at a given time and price. Similar data is available for more traditional transportation companies, such as airlines and bus agencies. In all these settings it is natural to ask the question:

*How do we maximize revenue in a transportation network, given a) a supply of vehicles and b) demand curves for the possible trips?*

This question appears at face value to be (primarily) a pricing problem. While this is true to a large extent, there is a latent scheduling/routing aspect of how one can serve these demands with an available supply of vehicles. This connection implies that any approach in this setting has to address difficulties encountered both in pricing and in routing problems.

Various efforts have been made at tackling aspects of pricing and routing in ride-sharing platforms. These include queueing approaches [4], mechanism design [6, 12, 16], and Markov chain models [2]. In this work we formulate and study a simple combinatorial model of the



© Kshipra Bhawalkar, Kostas Kollias, and Manish Purohit;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 26; pp. 26:1–26:16



Leibniz International Proceedings in Informatics

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

TRANSPORTATION NETWORK PRICING problem. In our model, the problem is studied on a graph where distances are symmetric (i.e., the distance from node  $u$  to node  $v$  is the same as the distance from node  $v$  to node  $u$ ) but demands are asymmetric (i.e., the demand from node  $u$  to node  $v$  at a price  $p$  is not necessarily the same as the demand from node  $v$  to node  $u$  at the same price  $p$ ). We consider this assumption to be reasonable in real world scenarios but also note that our results hold within a constant approximation when the distance between any  $u$  and  $v$  is within constant bounds of the distance between  $v$  and  $u$ . An available supply of  $k$  vehicles can move from node to node in the graph serving demands in the process.

It is not hard to observe that pricing decisions are interconnected with routing decisions. Knowing how many times vehicles will travel from  $u$  to  $v$  gives insights on how to price the trip from  $u$  to  $v$ . In such a case, we would want to charge as much as possible while still maintaining a demand high enough to utilize the vehicles that make the trip. Similarly, knowing that a specific trip has a large number of customers who are willing to travel at a high price hints that we should send a large number of vehicles their way. This interconnection makes the problem challenging and interesting.

## 1.1 Our Contributions

In Section 3 we attempt to disentangle the pricing component from the routing aspect and understand their difficulties separately. We explore the pricing and supply assignment aspect by abstracting away the routing component in a special case of the model. We show that this pricing and assignment version of the problem can be solved in polynomial time. In Section 4, we transition to the general graph model and present an approach that handles pricing and routing as separate stages, achieving a logarithmic approximation to the revenue in the process. In Section 5, applying the insights gathered in the first two, we bring the pricing and routing components back together in a joint optimization stage and provide a constant factor approximation algorithm for general graphs. In Section 6 of the paper we explain how our solution can handle selfish drivers with a small loss in the approximation factor. In Section 7, we show that our techniques generalize to the setting where edges have different demands depending on the time of the day, a setting that can also handle impatient demands that disappear after a certain period. Finally, in Section 8, we discuss how the capacitated version of the problem reduces to the unit capacity case.

## 1.2 Related Work

Various previous works study pricing in ride-sharing systems. The papers most closely related to ours are [6, 16] who also study a network with price dependent demand curves and seek to maximize revenue. A significant difference in our model is that we consider a general network with arbitrary distances as opposed to the unit distances studied in these two models. The work in [6] considers an infinite supply setting and proves that price discrimination can significantly improve revenue over uniform pricing. The work in [16] considers drivers with preferences for one location over the other and takes a mechanism design approach to achieve incentive compatibility. We note that in our final section we also consider a special, well-motivated, form of driver preferences: wage maximization.

Other papers study more dynamic aspects of ride-sharing platforms such as spatial imbalance and temporal variation [12], dynamic pricing [8], Markov models [2], and queueing models [4]. Other studies focus on market segmentation [1, 3] and car pooling aspects [14].

On the routing side, our work is related to the vehicle routing problem [10, 11] and, more closely, to prize collecting traveling salesperson problems in graphs. Most relevant is work on the *orienteering* problem, the best known algorithms for variants of which are given in [15] and [9]. The work in [15] achieves a 2 approximation for single path orienteering on undirected graphs via a primal-dual algorithm. The work in [9] presents dynamic programming based algorithms, following up on work in [5, 7]. A particular result from [7] that is relevant in our proofs is that undirected orienteering with  $k$  paths can be approximated within a factor 3.

## 2 Model and Preliminaries

In this section we define the specifics of TRANSPORTATION NETWORK PRICING.

Consider a set of locations  $V$  and the possible trips between them  $E = V \times V$ . Let  $l_e$  be the length (in time) of trip  $e \in E$ . We assume trip times are symmetric and  $l_e = l_{e'}$  for  $e = (u, v)$  and  $e' = (v, u)$ . For each trip  $e$ , we are also given a demand curve  $d_e(p)$  that gives the number of agents who are willing to pay a price  $p$  for trip  $e$ . Naturally, we assume that  $d_e(p)$  is a non-increasing function of  $p$ . For convenience, for each trip  $e$ , we also define the price curve

$$p_e(d) = \max\{p \mid d_e(p) \geq d\}$$

as the maximum price  $p$  such that at least  $d$  agents are willing to pay  $p$  for the trip  $e$ . To serve these demands, we have a supply of  $k$  service vehicles who can move from location to location and transport the demands. The total trips a service vehicle can make are limited by a time horizon  $T$  which is an upper bound on the total length of trips a service vehicle can do. For simplicity and without loss of generality we assume that all edge lengths, demand values, and possible prices are integers.

A solution consists of: (a) a price  $q_e$  for each trip  $e$  and (b) a path  $P_i$  of length at most  $T$  for each service vehicle  $i$ . A path is a sequence of trips  $P = \{e_1, e_2, \dots, e_m\}$  such that the destination of trip  $e_j$  is the source of trip  $e_{j+1}$ . The set of service vehicle paths induces a supply  $s_e$  for trip  $e$ , defined as the number of times  $e$  appears in all paths (note that a path might repeat  $e$  multiple times). The revenue for trip  $e$  is then equal to:

$$r_e = q_e \min\{s_e, d_e(q_e)\}.$$

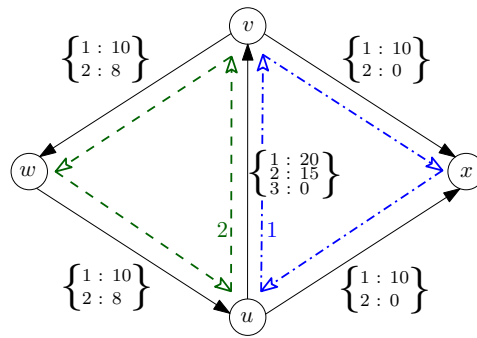
Our objective is to provide prices and paths that maximize the total revenue:

$$R = \sum_{e \in E} r_e.$$

We assume throughout the paper that the number of agents  $k$  and the time horizon  $T$  are both polynomial in the size of the graph  $G$ . We will design (approximation) algorithms that are polynomial in  $k$ ,  $T$ , and the size of the graph. Figure 1 illustrates a simple instance of the TRANSPORTATION NETWORK PRICING problem and its optimal solution.

A key component of our algorithms is the *revenue function* of an edge  $e \in V \times V$  that expresses the maximum amount of revenue that can be obtained from the edge  $e$  for a given supply. Mathematically, we define

$$r_e(\ell) = \max_{0 \leq j \leq \ell} \{j \cdot p_e(j)\}$$



■ **Figure 1** An instance of the TRANSPORTATION NETWORK PRICING problem. Each edge in the digraph represents a trip  $e$  of unit length. The maps labeling each edge represent the corresponding price curve, for instance,  $p_{(u,v)}(1) = 20$  and  $p_{(u,v)}(2) = 15$ . For a time horizon of  $T = 3$  and  $k = 3$  service vehicles, the figure illustrates an optimal solution that assigns two vehicles to the path  $\langle (u, v), (v, w), (w, u) \rangle$  and one vehicle to the path  $\langle (u, v), (v, x), (x, u) \rangle$  for a total revenue of 82.

### 3 Node Model: One Trip Per Vehicle

We begin with a warm-up setting in which each vehicle only makes one trip and our decisions amount to pricing edges and assigning vehicles to them. To fit this framework in our model, we can think of the special case with unit edge lengths and a unit time horizon. Since edges have no interaction with each other in this setting, we may equivalently think of them as simply unconnected nodes. For ease of notation, for this special case, we define a demand curve  $d_i(\cdot)$  for each node  $i$ . The price curve  $p_i(\cdot)$  and revenue curve  $r_i(\cdot)$  are defined similarly. We term this special case as the TRANSPORTATION NODE PRICING PROBLEM. We show that the problem is poly-time solvable.

► **Theorem 1.** *TRANSPORTATION NODE PRICING can be solved in polynomial time when the number  $k$  of service vehicles is polynomial in the size of the graph.*

**Proof.** Consider an arbitrary ordering of the nodes. Let  $\text{Opt}(i, j)$  denote the revenue extracted by the optimal solution for the first  $i$  nodes with  $j$  vehicles. Thus  $\text{Opt}(n, k)$  denotes the revenue extracted by the optimal solution for an instance. The following recurrence shows how one can compute this optimal solution via dynamic programming.

$$\text{Opt}(i, j) = \max_{\ell \in \{0, \dots, j\}} \{ \text{Opt}(i - 1, j - \ell) + r_i(\ell) \} \tag{1}$$

Intuitively, the recurrence searches over all possible number of vehicles to assign to the  $i^{\text{th}}$  node and solves the residual problem optimally. While  $\text{Opt}(n, k)$  only yields the optimal revenue, it is also easy to obtain the actual optimal solution by tracing the path taken by the dynamic program. ◀

As a side-note, we prove that the problem is NP-Hard when  $k$  is super-polynomial. We note though that a FPTAS is possible with an approach similar to the one for KNAPSACK.

► **Theorem 2.** *TRANSPORTATION NODE PRICING is NP-Hard.*

**Proof.** We will prove this by reducing KNAPSACK to our problem. The KNAPSACK problem has a collection of  $n$  items with sizes  $s_i$  and values  $v_i$  for  $i = 1, 2, \dots, n$  and a knapsack of size  $B$ . The goal is to pack items of total size at most  $B$  and maximize the total value picked.

Our reduction is as follows. For each item  $i$ , we construct a node  $i$  with the following demands: for a given large number  $L$ , the demand for the trip to  $i$  is one when the price is  $LS_i v_i$  and  $LS_i$  when the price is  $v_i$ . There are no others interested in the trip to  $i$ . In other words, we set  $r_i(1) = LS_i v_i$  and  $r(LS_i + 1) = LS_i v_i + v_i$ . The total supply of vehicles is  $k = n + LB$ .

Observe that in the induced TRANSPORTATION NODE PRICING instance there are, in effect, two possible prices for each node: either set price  $p_i = LS_i v_i$  and serve the unique customer at that price, or set  $p_i = v_i$  and serve all  $LS_i + 1$  customers. This means we have the option to either extract total revenue  $LS_i v_i$  spending supply 1 or spend an additional supply of  $LS_i$  to extract an extra  $v_i$ . When  $L$  is high enough, it is clear that any optimal solution spends the first  $n$  of the  $n + LB$  supply units to secure the  $LS_i v_i$  from every node, before considering any of the additional  $v_i$ 's. Then the solution will have to allocate the remaining  $LB$  supply units to get additional revenue  $v_i$  from any node  $i$  to which it allocates  $LS_i$ . This is precisely the original KNAPSACK problem where all the sizes are scaled by  $L$ , which implies that any optimal solution to this TRANSPORTATION NODE PRICING instance recovers an optimal solution to the corresponding KNAPSACK instance. ◀

#### 4 Separate Price & Route Optimization

In this section we consider the general TRANSPORTATION NETWORK PRICING model and present an approach that first attempts to determine prices and then to compute routes for the supply vehicles. We show that this algorithm achieves a logarithmic approximation. This section is of interest in itself, but also a warm-up for various aspects we will encounter in our main technical result in the next section (a constant approximation for the same problem that jointly optimizes for prices and routes) such as a reduction to the UNDIRECTED ORIENTEERING PROBLEM:

► **Definition 3.** *In the UNDIRECTED ORIENTEERING problem we are given an undirected graph  $G = (V, E)$  with costs on the edges and values on the nodes, and a cost budget  $T$ . We seek to find  $k$  paths each of cost at most  $T$  that maximize the total value of nodes visited.*

Our algorithm proceeds in two steps as follows:

*Guess a revenue target and set prices accordingly.* We guess a target revenue  $\tilde{r} \in \mathbb{R}$  and attempt to extract a revenue of  $\tilde{r}$  from each edge  $e$  in the network. For every edge  $e$ , set the price that would achieve the revenue target  $\tilde{r}$  with the smallest supply possible. If  $\tilde{r}$  is not achievable on some edge  $e$ , give up on  $e$  and set an infinite price.

*Construct and solve an undirected orienteering instance.* Since prices have been determined, each crossing of an edge by a supply vehicle extracts a known revenue. We formulate and solve an UNDIRECTED ORIENTEERING instance based on this information. Good constant factor approximation algorithms are known for UNDIRECTED ORIENTEERING, something that is the raison d'être of the graph transformation we perform in this stage. In more detail, we construct an auxiliary undirected graph in which we move the value from edges (i.e., the trips in the transportation graph) to new nodes that we introduce between the trip's endpoints. Every time such a node is visited will represent the corresponding trip being performed once. Hence, the value of such a node is equal to the price set for the corresponding trip. The edge lengths in the auxiliary graph are scaled so that the paths returned by the orienteering algorithm can be converted into sequences of trips in the original network.

## 4.1 Price Setting

Our algorithm begins with a guess  $\tilde{r}$  that is the revenue we will try to extract from every edge (i.e., every trip) in the network. Let  $\mathcal{R} = \{r_e(\ell)\}_{e \in E, 0 \leq \ell \leq kT}$  denote the set of all possible revenue values that can be extracted from any edge (note that no trip can be made more than  $kT$  times even if all service vehicles perform that one trip). The algorithm will ultimately be run for all possible guesses  $\tilde{r} \in \mathcal{R}$ . Since  $|\mathcal{R}| \leq n^2 kT$ , trying all possible revenues in  $\mathcal{R}$  can be done in polynomial time.

Once  $\tilde{r}$  is fixed, the price that we should set at any edge  $e$  can be computed as follows. Let  $s_e = \min\{\ell : r_e(\ell) \geq \tilde{r}\}$  be the minimum supply we need to extract value  $\tilde{r}$  at  $e$ . Then  $q_e = p_e(s_e)$  is the price we set for edge  $e$ . We now make the following claim.

► **Lemma 4.** *Let  $(q^*, P^*)$  be an optimal solution with  $q^*$  the vector of prices and  $P^*$  the paths of the service vehicles. Also, let  $\tilde{r}$  be the guess that, with induced prices  $\tilde{q}$  and the same paths  $P^*$ , maximizes the revenue among all guesses. The revenue extracted by solution  $(\tilde{q}, P^*)$  is an  $H_m$ -approximation to the revenue extracted by the solution  $(q^*, P^*)$ , where  $m$  is the number of edges in the graph and  $H_m$  the  $m$ -th harmonic number.*

**Proof.** Order the edges as  $1, 2, \dots, m$ , in order of non-increasing revenue extracted in  $(q^*, P^*)$ . Call these revenues  $r_1^*, r_2^*, \dots, r_m^*$ . Consider the guess  $\tilde{r} = r_j^*$  for our algorithm. Fix the paths  $P^*$  and set the price that achieves  $\tilde{r}$  in each edge  $e$  (or infinite price if not possible) as per our algorithm. Consider any edge  $e \leq j$ . We know that  $\tilde{r} = r_j^*$  is achievable on these edges, since the optimal solution extracts at least that on each one. Moreover, we have enough supply to achieve  $r_j^*$  on these edges, since  $P^*$  allocates enough supply for at least that much. Hence, when the guess is  $\tilde{r} = r_j^*$ , the solution  $(\tilde{q}, P^*)$  extracts value at least  $j r_j^*$ .

Let  $j^* = \arg \max_j \{j r_j^*\}$  be the guess that yields the maximum value. Thus, we have  $j^* r_{j^*}^* \geq j' r_{j'}^*$ , for all  $j' = 1, 2, \dots, m$ . Then:

$$R^* = \sum_{j'=1}^m r_{j'}^* \leq \sum_{j'=1}^m \frac{j^* r_{j^*}^*}{j'} = H_m j^* r_{j^*}^* \leq H_m R,$$

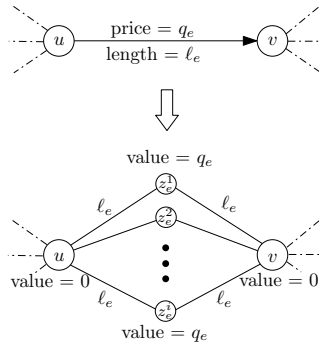
with  $R^*$  the optimal revenue and  $R$  the revenue of  $(\tilde{q}, P^*)$ . This proves the lemma.

We also proceed to prove that this factor is tight. Consider the case when  $r_j^* = 1/j$ . Every  $j r_j^*$  is unit, whereas their sum is  $H_m$ . ◀

## 4.2 Construction of the Orienteering Instance

Given the prices fixed in the previous stage of the algorithm, our goal is to route the  $k$  supply vehicles so that they can extract as much value as possible. This task is similar to the UNDIRECTED ORIENTEERING problem, for which a 2-approximation algorithm exists [15]. The main difference in our setting is that the values are on the trips between nodes and not on the nodes. Moreover, these trips are directed. We now describe a transformation to the graph that handles these issues with a small loss in approximation.

We construct an undirected graph  $H = (N, A)$  with values on the nodes and costs on the edges as follows. We begin with the nodes of the input transportation network  $V$ . All these nodes have value 0. For every ordered pair of nodes  $e = (u, v) \in E$ , we construct  $\min\{kT, d_e(q_e)\}$  nodes  $z_e^i, i = 1, 2, \dots, \min\{kT, d_e(q_e)\}$ , with value  $q_e$ , i.e., the price of the trip from  $u$  to  $v$ . For every such node  $z_e^i$ , we add an (undirected) edge between it and  $u$  and an (undirected) edge between it and  $v$ . Both these edges have length equal to  $l_e$ , the length of the trip from  $u$  to  $v$ . Figure 2 shows an example of the construction of the orienteering instance.



■ **Figure 2** Construction of the orienteering instance with fixed prices.

► **Lemma 5.** *Every path of length at most  $T^*$  in the input graph  $G$  that extracts revenue  $R$  can be expressed as a path of length at most  $2T^*$  in  $H$  that picks value  $R$ .*

**Proof.** Let  $P$  be a path of length at most  $T^*$  in  $G$ . Consider the order in which nodes are visited by the path  $P$  in  $G$ . We visit the same nodes in the same order in the graph  $H$  to obtain a path  $P'$ . The  $i^{\text{th}}$  time we cross an edge from  $u$  to  $v$  (in  $G$ ), we go via the intermediate node  $z_e^i$  in  $H$ . For  $i$  larger than  $d_e(q_e)$ , we go via any of the intermediate nodes (since they all have already been visited). Since the original path  $P$  in  $G$  has length at most  $T^*$ , and every edge  $e = (u, v)$  of length  $l_e$  in  $G$  corresponds to a walk  $(u \rightarrow z_e^i \rightarrow v)$  of length  $2l_e$  in  $H$ , the new path  $P'$  in  $H$  has a total length of at most  $2T^*$ . Let us now compute the value picked up by the path  $P'$  in  $H$ . Let  $s_e$  be the number of times that path  $P$  passes through edge  $e$ . Then, by definition, the total revenue extracted by  $P$  is given by:

$$R = \sum_e q_e \min\{s_e, d_e(q_e)\}.$$

On the other hand, for every edge  $e = (u, v)$  in  $G$ , by construction the path  $P'$  passes through  $\min\{s_e, d_e(q_e)\}$  distinct intermediate vertices ( $z_e^i$ ) each having value  $q_e$ . Thus path  $P'$  picks up value at least  $R$  in  $H$ . ◀

► **Lemma 6.** *Every path of length at most  $\tilde{T}$  in graph  $H$  that picks value  $R$  can be expressed as a path of length at most  $\tilde{T}$  in  $G$  that extracts at least revenue  $R$ .*

**Proof.** Let  $P'$  be a path in  $H$  of length at most  $\tilde{T}$  that picks value  $R$ . Let  $v$  be the first vertex on path  $P'$ . Then the path  $P'$  departs from node  $v$ , visits an intermediate node  $z_e^i$  (where  $e = (u, v)$  or  $e = (v, u)$ ) and either returns back to  $v$  or moves to the opposite node  $u$ . In the former case, it pays a cost of  $2l_e$  and extracts value  $q_e$ . We can construct a path  $P$  in  $G$  in exactly the same way as follows - starting from node  $v$ , visit node  $u$  and come back to  $v$  paying a total cost of  $2l_e$  (since lengths are symmetric) and extracting at least  $q_e$  revenue. In the latter case  $P'$  visits  $v \rightarrow z_e^i \rightarrow u$  and again pays a cost of  $2l_e$  and extracts a revenue of  $q_e$  (unless of course all intermediate nodes  $z_e^i$  have already been visited earlier). In this case, if  $e = (v, u)$ , then we simply cross from node  $v$  to node  $u$  in the path  $P$  to earn revenue  $q_e$  and a cost of only  $l_e$ . On the other hand, if  $e = (u, v)$ , then in path  $P$ , we first take edge  $(v, u)$ , then take  $(u, v)$ , and then again take  $(v, u)$  so that we end up on the same node on both  $P$  and  $P'$ . In this step, path  $P$  extracts a revenue of at least  $q_e$  but pays a cost of  $3l_e$ .

Let  $P'_{\text{rev}}$  denote a path in  $H$  that is the reverse of  $P'$ , i.e., it visits the same set of nodes but in the reverse order. Let  $P_{\text{rev}}$  be the path in  $G$  constructed as above starting from  $P'_{\text{rev}}$ . By construction, both  $P$  and  $P_{\text{rev}}$  extract a revenue of at least  $R$ . However, since for any

step  $v \rightarrow z_e^i \rightarrow u$  in  $P'$ , exactly one of  $P$  and  $P_{\text{rev}}$  pay a cost of  $l_e$  while the other pays  $3l_e$ . Thus, we have:

$$\text{length}(P) + \text{length}(P_{\text{rev}}) = \sum_e 4l_e = 2\text{length}(P')$$

and hence at least one of  $P$  and  $P_{\text{rev}}$  have length of at most  $\tilde{T}$ , proving the lemma. ◀

► **Lemma 7.** *For a set of fixed prices, solving the UNDIRECTED ORIENTEERING problem on graph  $H$  with budget  $T$  and translating the paths of graph  $H$  to paths of graph  $G$  as in Lemma 6, gives a 6-approximation to revenue.*

**Proof.** By Lemma 5 we get that each one of the optimal paths in  $G$  can be expressed as a path of length at most  $2T$  in  $H$ . We solve UNDIRECTED ORIENTEERING with a budget of  $T$ . We note that the optimal solution with budget  $T$  will have at least half the value of the optimal solution with budget  $2T$  since we can simply take the better half. This implies the optimal solution for the instance we solve will have value at least half the optimal revenue. By the fact that UNDIRECTED ORIENTEERING with  $k$  paths can be solved within a 3-approximation, our paths in  $H$  will be within 6 of the optimal revenue. Applying Lemma 6 completes the proof. ◀

Putting Lemma 4 with Lemma 7 together, we get the main theorem of the section. More precisely, Lemma 4 suggests that some prices given by our first stage are such that the optimal paths for them will give an  $H_m$ -approximation to revenue. Lemma 7 proves that, when we try these prices, we will find paths that approximate the optimal paths within a factor 6. We then get the following theorem.

► **Theorem 8.** *Our separate pricing & routing optimization algorithm gives a  $6H_m$ -approximation to revenue, where  $m$  is the number of edges.*

## 5 Joint Price and Route Optimization

In this section we use the insights obtained in the previous two sections to come up with a joint pricing and routing optimization algorithm. The algorithm in effect combines the main ideas of the previous two approaches to price and route at the same time. The algorithm proceeds in the following stages.

*Concave approximate revenue curve construction.* First, we process all demand curves to obtain the corresponding revenue functions  $r_e(\ell)$  (recall that these give the maximum possible revenue that can be achieved at edge  $e$  with supply  $\ell$ ), which in turn we process to obtain *approximate* revenue functions  $\hat{r}_e(\ell)$  that are concave. We prove that we can always find a concave function that satisfies  $r_e(\ell) \leq \hat{r}_e(\ell) \leq 2r_e(\ell)$  for every  $\ell$ . The main reason for performing this step is that the concave approximate revenue functions  $\hat{r}_e(\ell)$  satisfy the nice property that the marginal increase:

$$\Delta \hat{r}_e(\ell) = \hat{r}_e(\ell) - \hat{r}_e(\ell - 1)$$

that is caused by the  $\ell$ -th supply on edge  $e$  is decreasing. This proves useful when we place these marginal contributions as values to be collected from a graph in the second stage. We will also refer to  $\hat{r}_e(\cdot)$  as the *perceived* revenue.

*Auxiliary graph construction.* The main idea of the second stage of our algorithm is to construct an auxiliary graph that, similarly to our approach in the previous section, a) is undirected, b) has values only on nodes, and c) there is an equivalence between paths in the



auxiliary graph and sequences of trips in the input transportation network. Again, the value is moved from edge  $e$ , to a collection of nodes  $z_e^i, i = 1, 2, \dots, kT$ , that are introduced between its endpoints. This time however, the values of these nodes are not the same. Instead, the value of  $z_e^\ell$  is precisely the marginal perceived revenue  $\Delta \hat{r}_e(\ell)$ . The transformation of edge lengths is exactly as in the previous section. We then proceed as in the previous section, to solve the induced UNDIRECTED ORIENTEERING instance and translate the paths of the auxiliary graph  $H$  to paths of the input graph  $G$ . Once this is done, the paths induce supplies on the edges which we can use to infer the prices.

### 5.1 Concave Approximate Revenue Functions

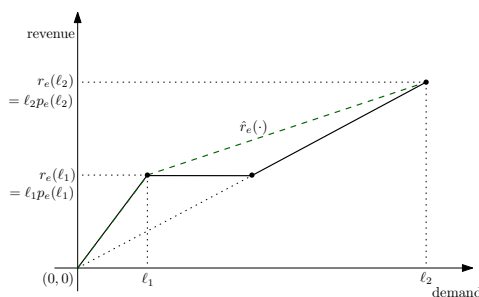
As we also discussed in the preliminary node model, we can express the maximum revenue we can extract from an edge, given supply  $\ell$ , as:

$$r_e(\ell) = \max_{0 \leq j \leq \ell} \{jp_e(j)\},$$

with  $p_e(j)$  the maximum price that induces demand at least  $j$ . As can be seen in Figure 3, we note that  $r_e(\cdot)$  need not be a concave function.

However, we can define a concave function  $\hat{r}_e(\cdot)$  as the concave envelope of  $r_e(\cdot)$ . In other words,  $\hat{r}_e(\cdot)$  is the lowest-valued concave function such that  $\hat{r}_e(\ell) \geq r_e(\ell)$ . Concretely, let  $[\ell_1, \ell_2]$  be a maximal interval such that the function  $r_e(\cdot)$  in this interval is bounded above by the linear interpolation of  $r_e(\ell_1)$  and  $r_e(\ell_2)$ . Then  $\forall \ell \in [\ell_1, \ell_2]$ ,  $\hat{r}_e(\ell)$  is obtained by linearly interpolating between  $(\ell_1, r_e(\ell_1))$  and  $(\ell_2, r_e(\ell_2))$ , i.e.,

$$\hat{r}_e(\ell) = \left( \frac{r_e(\ell_2) - r_e(\ell_1)}{\ell_2 - \ell_1} \right) (\ell - \ell_1) + r_e(\ell_1)$$



■ **Figure 3** Example revenue function and its concave approximation. The bold line shows the original revenue function  $r_e(\cdot)$  for some edge  $e$ , and the green dashed line shows its concave approximation  $\hat{r}_e(\cdot)$ .

We now show that  $\hat{r}_e(\cdot)$  point-wise approximates  $r_e(\cdot)$  within a factor of 2.

▷ **Claim 9.** For all  $0 \leq \ell \leq k$ ,  $\hat{r}_e(\ell) \leq 2r_e(\ell)$

Proof. Let  $[\ell_1, \ell_2]$  be a maximal interval such that  $\hat{r}_e(\ell) > r_e(\ell), \forall \ell \in (\ell_1, \ell_2)$ . Note that by definition of  $\ell_2$ , we have  $r_e(\ell_2) = \ell_2 p_e(\ell_2)$ . Otherwise, if  $r_e(\ell_2) = jp_e(j)$  for some  $j < \ell_2$ , then we have  $r_e(\ell_2) = r_e(j)$  and we cannot have  $\hat{r}_e(j) > r_e(j)$ . Now, since  $p_e(\cdot)$  is a non-increasing function, we have

$$r_e(\ell_1) = \max_{0 \leq j \leq \ell_1} \{jp_e(j)\} \geq \ell_1 p_e(\ell_2) \tag{2}$$

Hence, we have the following,

$$\frac{r_e(\ell_2) - r_e(\ell_1)}{\ell_2 - \ell_1} \leq \frac{r_e(\ell_2) - \ell_1 p_e(\ell_2)}{\ell_2 - \ell_1} \quad (3)$$

$$= \frac{\ell_2 p_e(\ell_2) - \ell_1 p_e(\ell_2)}{\ell_2 - \ell_1} = p_e(\ell_2) \quad (4)$$

Now, for any  $\ell \in (\ell_1, \ell_2)$ , by definition of  $\hat{r}_e(\cdot)$  we have,

$$\hat{r}_e(\ell) = \left( \frac{r_e(\ell_2) - r_e(\ell_1)}{\ell_2 - \ell_1} \right) (\ell - \ell_1) + r_e(\ell_1) \quad (5)$$

$$\leq p_e(\ell_2)(\ell - \ell_1) + r_e(\ell_1) \quad (6)$$

$$\leq 2 \max\{\ell p_e(\ell_2), r_e(\ell_1)\} \quad (7)$$

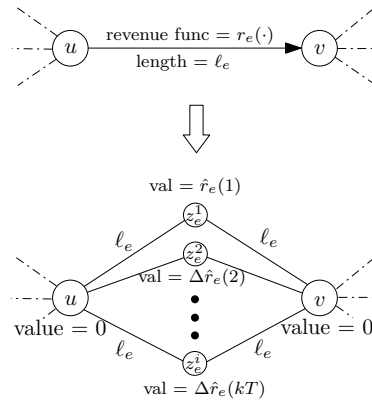
However, since the revenue function  $r_e(\cdot)$  is non-decreasing and the price function  $p_e(\cdot)$  is non-increasing, we have

$$r_e(\ell) \geq \max\{\ell p_e(\ell_2), r_e(\ell_1)\} \quad (8)$$

The claim now follows from equations (7) and (8).  $\triangleleft$

## 5.2 Construction of the Orienteering Instance

As in the previous section, we will construct an undirected graph  $H = (N, A)$  with values on the nodes and costs on the edges. Here also, we begin with the nodes of the input transportation network  $V$  which again have value 0. For every ordered pair of nodes  $e = (u, v) \in E$ , we construct  $kT$  nodes  $z_e^i, i = 1, 2, \dots, kT$ . The value of  $z_e^i$  is  $\Delta \hat{r}_e(i)$ , i.e., the marginal increase in perceived total revenue (as given by the concave approximate revenue functions  $\hat{r}_e(\cdot)$ ) offered by the  $i$ -th trip from  $u$  to  $v$ . For every such node  $z_e^i$ , we add an (undirected) edge between it and  $u$  and an (undirected) edge between it and  $v$ . Both these edges have length equal to  $\ell_e$ , the length of the trip from  $u$  to  $v$ . Figure 4 shows an example of the construction of the orienteering instance.



■ **Figure 4** Construction of the orienteering instance with approximate revenues.

► **Lemma 10.** *Every path of length at most  $T^*$  in the input graph  $G$  that extracts perceived revenue  $R$  can be expressed as a path of length at most  $2T^*$  in  $H$  that picks value  $R$ .*

**Proof.** Let  $P$  be a path in  $G$  of length at most  $T^*$  that extracts a perceived revenue of  $R$ . We construct a path  $P'$  in  $H$  as follows - when  $P$  uses the edge  $(u, v)$  for the  $i^{\text{th}}$  time, our path in  $H$  moves from node  $u$  to node  $v$  via the intermediate node  $z_{(u,v)}^i$ . Thus, if  $P$  passes through an edge  $e$  exactly  $l$  times to extract a perceived revenue of  $\hat{r}_e(l)$ , the path  $P'$  also picks up a value of  $\sum_{i=1}^l \Delta \hat{r}_e(i) = \hat{r}_e(l)$ . Thus  $P'$  also picks up a total value of  $R$ . The length argument follows precisely as in Lemma 5.  $\blacktriangleleft$

► **Lemma 11.** *Every path of length at most  $\tilde{T}$  in graph  $H$  that picks value  $R$  can be expressed as a path of length at most  $\tilde{T}$  in  $G$  that extracts perceived revenue at least  $R$ .*

**Proof.** The path construction and length arguments follow exactly as in Lemma 6. The revenue argument is as follows: Say the path  $P'$  in  $H$  visits  $x$  intermediate nodes corresponding to edge  $e = (u, v)$ . Then our constructed path  $P$  in  $G$  crosses edge  $e$  at least  $x$  times and extracts a perceived revenue of at least  $\hat{r}_e(x)$ . On the other hand, since  $\Delta \hat{r}_e(\cdot)$  is the non-increasing, the value earned by  $P'$  from edge  $e$  is at most  $\sum_{i=1}^x \Delta \hat{r}_e(i) = \hat{r}_e(x)$ .  $\blacktriangleleft$

Lemmas 10 and 11 together imply that the UNDIRECTED ORIENTEERING problem with  $k$  paths is equivalent to the TRANSPORTATION NETWORK PRICING problem with concave revenue functions up to a factor of 2 in the approximation ratio. We can thus directly use a 3-approximation algorithm for UNDIRECTED ORIENTEERING as in Lemma 7 to obtain a 6-approximation to the TRANSPORTATION NETWORK PRICING with concave revenue functions. However, since arbitrary revenue functions can be approximated within a factor of 2 by concave functions, Claim 9 then yields our main result.

► **Theorem 12.** *Our joint pricing & routing optimization algorithm gives a 6-approximation to revenue for concave revenue functions and a 12-approximation to revenue for general revenue functions.*

## 6 Selfish drivers

An additional layer of complexity in the ride-sharing context is added by the fact that drivers are independent and will not follow the paths dictated by our algorithm when this is not the behavior that maximizes their total wages. In this section we discuss the presence of selfish drivers in the TRANSPORTATION NETWORK PRICING setting. We assume that wages are a fixed  $\alpha$  fraction of the revenue (i.e., the drivers and the platform share the earnings with a fixed ratio) and argue that, for any given prices, letting the drivers reach an equilibrium is within a factor 2 of the optimal path selections. In this sense, with a small loss in the approximation factor, we may use our algorithms to compute prices assuming the drivers will comply, advertise them, and let the drivers reach an equilibrium.

For the purposes of our argument, we will need to introduce some additional notation and definitions. First, for simplicity of exposition, we set  $\alpha = 1$ , i.e., assume the drivers receive all revenue. Let  $x_e^i$  be the number of times driver  $i$  crosses edge  $e$  and  $x^i = (x_e^i)_{e \in E}$  the vector for driver  $i$  over all edges which we will refer to as the driver's *strategy*. Let  $x$  be the vector of all driver strategies. Let  $d_e$  be the demand under the current price vector (note that, for simplicity, we have dropped dependence of  $d_e$  on  $q_e$  in the notation, since prices are considered fixed throughout this section) and  $s_e(x)$  the supply under  $x$ . We are now ready to define the (expected) wage of  $i$  on  $e$  as:

$$w_e^i(x) = x_e^i q_e \min \left\{ 1, \frac{d_e}{x_e} \right\}.$$

## 26:12 Revenue Maximization in Transportation Networks

The interpretation of this expression is that  $i$  has probability 1 to get a ride (and hence a payment of  $q_e$ ) every time she crosses the edge when the demand is at least the supply and probability  $d_e/x_e$  when the demand is less than the supply. Another interpretation is that drivers share the total revenue on the edge  $r_e = q_e \min\{s_e(x), d_e\}$  proportionally to the number of times they cross it.

A collection of strategies is a *Nash equilibrium* when for every driver  $i$ , it is the case that a unilateral deviation to some other vector  $y^i$ , induced by a different path selection will not increase her total wages:

$$\sum_{e \in E} w_e^i(x) \geq \sum_{e \in E} w_e^i(x^{-i}, y^i), \quad \forall y^i.$$

The *price of anarchy* is the ratio of the total wages in the optimal solution over the total wages in the worst Nash equilibrium. We get the following observation.

► **Observation 13.** *The price of anarchy in the TRANSPORTATION NETWORK PRICING problem after prices have been fixed is 2.*

**Proof.** The upper bound follows by the fact that the game we have described is a utility game with a submodular utility function (since the drivers cover demands with their path selections). The upper bound then follows from the main result in [18].

The lower bound follows from the following simple instance of the TRANSPORTATION NODE PRICING submodel. Node 1 has a single demand which is priced at  $1 + \epsilon$ . Node 2 has  $1/\epsilon$  demands priced at  $\epsilon$ . There are  $1/\epsilon$  drivers in the game. If all of them head to node 1, their expected wage will be  $\epsilon + \epsilon^2$ , which is larger than the  $\epsilon$  they can get from a ride at node 2. Hence, this is a Nash equilibrium with total wages  $1 + \epsilon$ . The optimal solution assigns 1 driver to node 1 and the rest of them to node 2 for total wages 2. ◀

Hence, we reach the conclusion that, in the presence of selfish drivers, our approximation will be a factor 2 away of the ones achieved by our algorithms, i.e., we achieve a 2-approximation using our joint price and route optimization algorithm.

## 7 Transportation Network Pricing with Dynamic Demands

In this section we consider a natural extension of the TRANSPORTATION NETWORK PRICING problem where the demands on an edge can now vary as a function of time. We let  $d_e(p, t)$  denote the demand on edge  $e$  at time  $t$  when the price is  $p$ . The demand that applies is determined at the moment in time when an agent starts traversing an edge. For ease of notation, we assume that the lengths on edges are specified in the units of time. Since all edge lengths are integral we can assume that the demand changes only at integral time steps. In this section we prove the following theorem.

► **Theorem 14.** *The Transportation Network pricing problem with dynamic demands can be solved in time polynomial in  $n, k$ , and  $T$  to obtain an approximation of  $O(\log n)$ .*

To obtain the best possible result, we proceed in two steps. In step 1, we reduce the TRANSPORTATION NETWORK PRICING WITH DYNAMIC DEMAND problem to single agent TRANSPORTATION NETWORK ROUTING WITH UNIT TIME WINDOWS. In step 2, we reduce the single agent TRANSPORTATION NETWORK ROUTING WITH UNIT TIME WINDOWS problem to DIRECTED ORIENTEERING WITH UNIT TIME WINDOWS problem. In the end, we obtain an approximation factor of  $O(\log n)$ .

## 7.1 Step 1: Transportation Network Pricing to Transportation Network Routing

This reduction is similar to section 5.2. We present an additional step where we reduce the problem from  $k$  agents to a single agent.

First since the demand varies with time, we redevelop some of the notation to depend on time. The price curve  $p_e(d, t) = \max\{p \mid d_e(p, t) \geq d\}$  is the price at which the demand is at least  $d$ . The revenue from assigning  $l$  agents to edge  $e$  at time  $t$  is  $r_e(l, t) = \max_{0 \leq j \leq l} \{j p_e(j, t)\}$ . We approximate the revenue curve using a concave function  $\hat{r}_e(t, i)$  constructed similar to Lemma 9 with the guarantee that  $r_e(l, t) \leq \hat{r}_e(l, t) \leq 2r_e(l, t)$ .

Using the concave revenue functions  $\hat{r}$ , we reduce the problem to one of constructing paths on a graph. We will call this the Transportation Network Routing with Unit Time Windows problem.

► **Definition 15.** *In TRANSPORTATION NETWORK ROUTING WITH UNIT TIME WINDOWS problem, we are given a directed graph  $G(V, E)$  with values  $v_e$  and time window of unit length  $[t_e, t_e + 1]$  associated with each edge. The goal is to find  $k$  paths such that each path has length at most  $T$  and the sum of values  $v_e$  of all edges that appear in at least one path is maximized. The value  $v_e$  on an edge is only collected if the path starts on the edge  $e$  during the time window  $[t_e, t_e + 1]$ . If the same edge  $e$  appears in multiple paths, its value  $v_e$  is collected only once.*

Given our input instance  $G = (V, E)$  of the TRANSPORTATION NETWORK PRICING WITH DYNAMIC DEMANDS problem, we construct an instance  $G' = (V, E')$  of the TRANSPORTATION NETWORK ROUTING WITH UNIT TIME WINDOWS problem as follows. This graph has the same set of vertices  $V$ . For each edge  $(u, v)$  in the original graph  $G$ , we construct  $kT$  parallel edges between  $u$  and  $v$ . The value of the  $(l, t)$ 'th edge for  $l \in \{0, 1, \dots, k\}$  and  $t \in 0, 1, \dots, T - 1$  is  $\Delta \hat{r}(l, t) = \hat{r}(l, t + 1) - \hat{r}(l, t)$ . Then similar arguments as section 5.2 guarantee that an  $\alpha$ -approximation to TRANSPORTATION NETWORK ROUTING WITH UNIT TIME WINDOWS problem yields a  $2\alpha$ -approximation to TRANSPORTATION NETWORK PRICING WITH DYNAMIC DEMANDS. Note that since the paths map one-to-one in time between the two instances the time windows do not create any new challenge.

Next we show that an approximation algorithm for the TRANSPORTATION NETWORK ROUTING WITH TIME WINDOWS problem with one agent can be used to obtain a slightly worse approximation for  $k$  agents. The proof is similar to an analogous result by [7] for orienteering problem.

► **Theorem 16.** *An  $\alpha$ -approximation algorithm for the transportation network problem with time windows for a single agent can be used to obtain an  $(\alpha + 1)$ -approximation for the transportation network problem with time windows for  $k$  agents.*

**Proof.** Given an  $\alpha$ -approximation algorithm for the transportation network problem for a single agent, we use it repeatedly to solve the problem for  $k$  agents. After the algorithm has selected path  $A_i$  for the  $i$ 'th agent. We set the value on all edges used by the path  $A_i$  to zero before calling the algorithm for the next agent. This ensures that all paths constructed by the algorithm are edge disjoint. Let  $O = (O_1, O_2, \dots, O_k)$  denote the optimal solution with  $k$  agents decomposed into the  $k$  agents' paths. Let  $\Delta_i$  denote the edges from path  $O_i$  that have already been used by some path  $A_j$  (where  $j < i$ ) by algorithm before path  $A_i$  is chosen. There is a feasible path using all the edges of  $O_i \setminus \Delta_i$ . Thus we have that  $v(A_i) \geq \frac{1}{\alpha} \{v(O_i) - v(\Delta_i)\}$ . Summing these over all agents,  $\alpha v(A) \geq v(O) - v(\Delta)$ . Moreover  $v(\Delta) \leq v(A)$ . Hence we conclude that  $(\alpha + 1)v(A) \geq v(O)$ . ◀

With this result, it suffices to obtain an approximation for the TRANSPORTATION NETWORK ROUTING WITH TIME WINDOWS problem for a single agent.

## 7.2 Step 2: Transportation Network Routing to Directed Orienteering

We next reduce TRANSPORTATION NETWORK ROUTING WITH TIME WINDOWS to directed orienteering with fixed start locations and unit time windows.

► **Definition 17.** *In DIRECTED ORIENTEERING WITH UNIT TIME WINDOWS AND FIXED START we are given a directed graph  $G = (V, E)$  with costs on the edges and values on the nodes, and a cost budget  $T$ . There is also a time-window of unit length associated with each node. The value from a node is only collected if it is visited within the time window. We seek to find a path of cost at most  $T$  that starts at node  $s \in V$  such that the value collected is maximized.*

This problem can be solved in polynomial time to obtain an approximation of  $O(\log n)$ . This follows from [9] that provide an approximation of  $O(\alpha)$  where  $\alpha$  is approximation for directed orienteering, [13] that provides an  $O(\beta \log n)$  approximation for directed orienteering where  $\beta$  is the integrality gap for asymmetric TSP, and [17] that provides a constant factor integrality gap for asymmetric TSP.

We start with the TRANSPORTATION NETWORK ROUTING instance  $G' = (V, E')$  with length  $l_e$ , value  $v_e$  and unit time window  $[t_e, t_e + 1]$  associated with each edge. We construct a directed graph  $H = (N, A)$  with values on the nodes and costs on the edges. The set of vertices  $N = V \cup I$ . The set  $V$  is the set of original vertices. The set  $I$  is the set of *intermediate* vertices, with one vertex  $z_e$  for each edge  $e$  in  $E'$ . In the graph  $H$ , for each edge  $e = (u, v) \in E'$ , we add an edge  $(u, z_e)$  of length  $l_e$  and an edge  $(z_e, v)$  of length zero. We associate value  $v_e$  and time window  $[t_e + l_e, t_e + l_e + 1]$  with each intermediate node  $z_e$  and value 0 with nodes in  $V$ .

We prove the following lemmas to obtain the final result:

► **Lemma 18.** *Any path of length at most  $T^*$  in graph  $G'$  that picks value  $V$  can be expressed as a path of length  $T^*$  in  $H$  that picks value  $V$*

**Proof.** Consider edge  $e = (u, v)$  in the path. We map it to the edges  $(u, z_e), (z_e, v)$ . The path collects the value if it starts traversing the edge during  $[t_e, t_e + 1]$ . In the orienteering instance the path will get to the intermediate node  $z_e$  in time window  $[t_e + l_e, t_e + l_e + 1]$  so the same value  $v_e$  can be collected. ◀

For mapping a solution in graph  $H$  to a solution in graph  $G'$  the main blocker is that the path may start or end at one of the  $z_e$  nodes. To tackle this we call the orienteering problem with a fixed start node. We prove the following lemma.

► **Lemma 19.** *Given a path of length  $T^*$  that starts at a node a non-intermediate node  $s$  in  $H$  and collects value  $V$ , we can construct a path of length  $T^*$  in the graph  $G'$  with value  $V$  starting at node  $s$  in graph  $G'$*

**Proof.** If the path in  $H$  ends at an intermediate node  $z_e$ , it can be extended to the next non-intermediate node without increasing its length. We can assume that the path begins and ends in non-intermediate nodes. After that there is one-to-one mapping between the portions of the path. An intermediate node  $z_e$  only connects to the end node  $v$  of the edge  $e$ . So we can always find pairs of segments  $(u, z_e), (z_e, v)$  in the path. These can be mapped to  $e = (u, v)$  in graph  $G'$ . The value  $v_e$  is the same, the lengths of the segments are the same and the time window  $[t_e + l_e, t_e + l_e + 1]$  in the graph  $H$  maps to  $[t_e, t_e + 1]$  which is precisely when the constructed path will begin traversing edge  $e$ . ◀

To complete the proof of Theorem 14, we need to call the orienteering subroutine with all possible start nodes in the set  $V$ . We can choose the best solution among those and it will be an  $O(\log n)$ -approximation to the optimal solution to the SINGLE AGENT TRANSPORTATION NETWORK ROUTING problem. Trying different start nodes does not degrade the running time by more than a factor of  $n$ .

## 8 Capacitated Vehicles

For the sake of simplicity, we have studied the problem in terms of unit capacity vehicles that can serve a single demand when crossing an edge. We now explain that a simple transformation can reduce the capacitated version of the problem where each vehicle can serve up to a fixed number  $c$  of demands to the unit capacity case. The main idea is as follows: We will transform any given demand curve into an equivalent one such that a) for any given price, the number of buyers that is willing to buy is a multiple of  $c$  and b) the revenue functions remain intact. Achieving that would then allow us to change the units of measurement by a factor  $c$  and have each unit of demand correspond to a number of buyers equal to the vehicle capacity, in effect recovering the unit capacity model. Note that the revenue functions that give the optimal revenue of an edge as a function of the supply assigned to it are the only input given to our main algorithms. This implies our approximation results are preserved by such a reduction.

Consider a given price curve  $p_e(\cdot)$ . For any given integer  $s$ , let,

$$\rho_s = \max_{(s-1)c < d \leq s \cdot c} d \cdot p_e(d), \quad (9)$$

be the maximum revenue obtained when using exactly  $s$  service vehicles. Our modified curve is such that:

$$\hat{d}_e(p) = s \cdot c, \text{ for all } p \in \left( \frac{\rho_{s+1}}{(s+1)c}, \frac{\rho_s}{s \cdot c} \right]. \quad (10)$$

The following lemma proves that the modified demand curve is well defined.

► **Lemma 20.**

$$\frac{\rho_{s+1}}{(s+1)c} \leq \frac{\rho_s}{s \cdot c}.$$

**Proof.** Note that  $\rho_s \geq p_e(s \cdot c)s \cdot c$ , since using  $d = s \cdot c$  is an option in (9). Also,  $\rho_{s+1} \leq p_e(s \cdot c)(s+1)c$ , since the highest price for which supply  $s+1$  is needed is at most  $p_e(s \cdot c)$  and the highest demand for which supply  $s+1$  is needed is  $(s+1)c$ . The two inequalities can be combined to give:

$$\frac{\rho_{s+1}}{(s+1)c} \leq p_e(s \cdot c) \leq \frac{\rho_s}{s \cdot c},$$

which completes the proof. ◀

The demand curve (10) by definition satisfies the property that only a multiple of  $c$  buyers will show up under any price. Then, the  $j$ -th such group of  $c$  buyers can be replaced with a single buyer with value  $\rho_j/j$ . By Lemma 20 these  $\rho_j/j$  values are nonincreasing, as necessary for demand curves. Moreover, the optimal revenue obtained by any given number of supply vehicles remains the same, which suggests the revenue curves are unchanged and our transformation is completed as desired.

## References

- 1 Reza Alijani, Siddhartha Banerjee, Sreenivas Gollapudi, Kostas Kollias, and Kamesh Munagala. The segmentation-thickness tradeoff in online marketplaces. *POMACS*, 3(1):18:1–18:26, 2019.
- 2 Siddhartha Banerjee, Daniel Freund, and Thodoris Lykouris. Pricing and optimization in shared vehicle systems: An approximation framework. In *Proceedings of the 2017 ACM Conference on Economics and Computation, EC '17, Cambridge, MA, USA, June 26-30, 2017*, page 517, 2017.
- 3 Siddhartha Banerjee, Sreenivas Gollapudi, Kostas Kollias, and Kamesh Munagala. Segmenting two-sided markets. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 63–72, 2017.
- 4 Siddhartha Banerjee, Ramesh Johari, and Carlos Riquelme. Pricing in ride-sharing platforms: A queueing-theoretic approach. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation, EC '15, Portland, OR, USA, June 15-19, 2015*, page 639, 2015.
- 5 Nikhil Bansal, Avrim Blum, Shuchi Chawla, and Adam Meyerson. Approximation algorithms for deadline-tsp and vehicle routing with time-windows. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 166–174, 2004.
- 6 Kostas Bimpikis, Ozan Candogan, and Daniela Sabán. Spatial pricing in ride-sharing networks. *Operations Research*, 67(3):744–769, 2019.
- 7 Avrim Blum, Shuchi Chawla, David R. Karger, Terran Lane, Adam Meyerson, and Maria Minkoff. Approximation algorithms for orienteering and discounted-reward TSP. *SIAM J. Comput.*, 37(2):653–670, 2007.
- 8 Juan-Camilo Castillo, Dan Knoepfle, and Glen Weyl. Surge pricing solves the wild goose chase. In *Proceedings of the 2017 ACM Conference on Economics and Computation, EC '17, Cambridge, MA, USA, June 26-30, 2017*, pages 241–242, 2017.
- 9 Chandra Chekuri, Nitish Korula, and Martin Pál. Improved algorithms for orienteering and related problems. *ACM Trans. Algorithms*, 8(3):23:1–23:27, 2012.
- 10 George B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.
- 11 Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59:345–358, 1992.
- 12 Hongyao Ma, Fei Fang, and David C. Parkes. Spatio-temporal pricing for ridesharing platforms. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019.*, page 583, 2019.
- 13 Viswanath Nagarajan and R. Ravi. The directed orienteering problem. *Algorithmica*, 60(4):1017–1030, 2011.
- 14 Michael Ostrovsky and Michael Schwarz. Carpooling and the economics of self-driving cars. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019.*, pages 581–582, 2019.
- 15 Alice Paul, Daniel Freund, Aaron Ferber, David B. Shmoys, and David P. Williamson. Prize-collecting TSP with a budget constraint. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, pages 62:1–62:14, 2017.
- 16 Duncan Rheingans-Yoo, Scott Duke Kominers, Hongyao Ma, and David C. Parkes. Ridesharing with driver location preferences. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 557–564, 2019.
- 17 Ola Svensson, Jakub Tarnawski, and László A. Végh. A constant-factor approximation algorithm for the asymmetric traveling salesman problem. *J. ACM*, 67(6):37:1–37:53, 2020.
- 18 Adrian Vetta. Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*, page 416, 2002.



# Connected $k$ -Partition of $k$ -Connected Graphs and $c$ -Claw-Free Graphs

Ralf Borndörfer  

Zuse Institute Berlin, Germany

Katrin Casel  

Hasso Plattner Institute, University of Potsdam, Germany

Davis Issac  



Hasso Plattner Institute, University of Potsdam, Germany

Aikaterini Niklanovits  

Hasso Plattner Institute, University of Potsdam, Germany

Stephan Schwartz  

Zuse Institute Berlin, Germany

Ziena Zeif  

Hasso Plattner Institute, University of Potsdam, Germany

---

## Abstract

---

A *connected partition* is a partition of the vertices of a graph into sets that induce connected subgraphs. Such partitions naturally occur in many application areas such as road networks, and image processing. In these settings, it is often desirable to partition into a fixed number of parts of roughly of the same size or weight. The resulting computational problem is called Balanced Connected Partition (BCP). The two classical objectives for BCP are to maximize the weight of the smallest, or minimize the weight of the largest component. We study BCP on  $c$ -claw-free graphs, the class of graphs that do not have  $K_{1,c}$  as an induced subgraph, and present efficient  $(c - 1)$ -approximation algorithms for both objectives. In particular, for 3-claw-free graphs, also simply known as claw-free graphs, we obtain a 2-approximation. Due to the claw-freeness of line graphs, this also implies a 2-approximation for the edge-partition version of BCP in general graphs.

A harder connected partition problem arises from demanding a connected partition into  $k$  parts that have (possibly) heterogeneous target weights  $w_1, \dots, w_k$ . In the 1970s Györi and Lovász showed that if  $G$  is  $k$ -connected and the target weights sum to the total size of  $G$ , such a partition exists. However, to this day no polynomial algorithm to compute such partitions exists for  $k > 4$ . Towards finding such a partition  $T_1, \dots, T_k$  in  $k$ -connected graphs for general  $k$ , we show how to efficiently compute connected partitions that at least approximately meet the target weights, subject to the mild assumption that each  $w_i$  is greater than the weight of the heaviest vertex. In particular, we give a 3-approximation for both the lower and the upper bounded version i.e. we guarantee that each  $T_i$  has weight at least  $\frac{w_i}{3}$  or that each  $T_i$  has weight most  $3w_i$ , respectively. Also, we present a both-side bounded version that produces a connected partition where each  $T_i$  has size at least  $\frac{w_i}{3}$  and at most  $\max(\{r, 3\})w_i$ , where  $r \geq 1$  is the ratio between the largest and smallest value in  $w_1, \dots, w_k$ . In particular for the balanced version, i.e.  $w_1 = w_2 = \dots = w_k$ , this gives a partition with  $\frac{1}{3}w_i \leq w(T_i) \leq 3w_i$ .

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Graph algorithms analysis

**Keywords and phrases** connected partition, Györi-Lovász, balanced partition, approximation algorithms

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.27

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/abs/2107.04837>

**Funding** This research was partially funded by the HPI Research School on Data Science and Engineering.

*Aikaterini Niklanovits*: HPI Research School

*Ziena Zeif*: HPI Research School



© Ralf Borndörfer, Katrin Casel, Davis Issac, Aikaterini Niklanovits, Stephan Schwartz, and Ziena Zeif; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 27; pp. 27:1–27:14



Leibniz International Proceedings in Informatics

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

## 1 Introduction

Partitioning a graph into connected subgraphs is a problem that arises in many application areas such as parallel processing, road network decomposition, image processing, districting problems, and robotics [34, 35, 4, 1, 39]. Often in these applications, it is required to find a partition into a specified number  $k$  of connected subgraphs. For instance, in the parallel processing applications, the number of processors is restricted, and in robotics applications, the number of robots available is restricted. Formally, we call a partition  $T_1, T_2, \dots, T_k$  of the vertex set of graph, a *connected ( $k$ -)partition*, if the subgraph induced by the vertices in  $T_i$  is connected for each  $1 \leq i \leq k$ .

The typical modeling objective in such connected partition problems is to balance sizes among the  $k$  parts. Sometimes one needs to consider a vertex-weighted generalization e.g. weights representing the required amount of work at the entity corresponding to the vertex. The two classical balancing objectives for such  $k$ -partitions are to maximize the total weight of the lightest part, or to minimize the weight of the heaviest part. These objectives yield the following two versions of the *balanced connected partition problem* (BCP).

MAX-MIN BCP (MIN-MAX BCP)

*Input:* A vertex-weighted graph  $G = (V, E, w)$  where  $w : V \rightarrow \mathbb{N}$ , and  $k \in \mathbb{N}$ .

*Task:* Find a connected  $k$ -partition  $T_1, \dots, T_k$  of  $G$  maximizing  $\min_{i \in [k]} w(T_i)$  (minimizing  $\max_{i \in [k]} w(T_i)$  resp.).

On general graphs, both variants of BCP are NP-hard [5], and hence the problems have been mostly studied from the viewpoint of approximation algorithms [6, 8, 9, 11, 12]. Most of the known results are for small values of  $k$ , and there are some results also for special classes like grid graphs or graphs of bounded treewidth (see related work section for further details). The currently best known polynomial-time approximation for general graphs for any  $k$  is a 3-approximation for both MAX-MIN and MIN-MAX BCP by Casel et. al. [6].

Intuitively, an obstacle for getting a balanced connected partition is a large induced star, i.e. a tree with one internal node and  $c$  leaves, denoted by  $K_{1,c}$ . We say a graph is  *$c$ -claw-free* or  *$K_{1,c}$ -free* if it does not contain an induced  $K_{1,c}$  as subgraph. For such graphs, we give a very efficient  $(c - 1)$ -approximation algorithm for both the min-max and max-min objective. In particular by setting  $c = 3$ , we get a 2-approximation on  $K_{1,3}$ -free graphs, better known as *claw-free graphs*.

Claw free graphs have been widely studied by Seymour and Chudnovsky in a series of seven papers under the name *Claw-free graphs I-VII* ([14]-[20]), who also provide a structure theorem for these graphs [21]. Some interesting examples of such graphs are line graphs, proper circular interval graphs and de-Bruijn graphs [22]. Apart from their structural properties, claw-free graphs have been studied in the context of obtaining efficient algorithms for several interesting problems, see e.g. [27, 24, 23].

Although, for  $c > 3$  our algorithm gives a worse guarantee than the algorithm by Casel et. al. [6], we note that their algorithm runs in  $\mathcal{O}(\log(X^*)k^2|V||E|)$  time for MAX-MIN BCP and in  $\mathcal{O}(\log(X^*)|V||E|(\log \log X^* \log(|V|w_{\max}) + k^2))$  time for MIN-MAX BCP, where  $X^*$  denotes the optimum value and  $w_{\max} := \max_{v \in V} w(v)$  the maximum weight of a vertex, whereas our algorithms give an  $\mathcal{O}(\log(X^*)|E|)$  runtime for MAX-MIN BCP and an  $\mathcal{O}(|E|)$  runtime for MIN-MAX BCP. Moreover, our algorithms are less technical and hence much easier to implement. We prove the following statements.

► **Theorem 1.** *Given a vertex-weighted  $K_{1,c}$ -free graph  $G = (V, E, w)$  and  $k \in \mathbb{N}$ , a  $(c - 1)$ -approximation for MIN-MAX BCP can be computed in  $\mathcal{O}(|E|)$  time.*

► **Theorem 2.** *Given a vertex-weighted  $K_{1,c}$ -free graph  $G = (V, E, w)$  and  $k \in \mathbb{N}$ , a  $(c - 1)$ -approximation for MAX-MIN BCP can be computed in time  $\mathcal{O}(\log(X^*)|E|)$ , where  $X^*$  is the optimum value.*

Since line graphs are  $K_{1,3}$ -free, these results directly imply efficient approximations for the following edge-partition versions of BCP. A  $k$ -partition of the edges of a graph, is called a *connected edge  $k$ -partition*, if the subgraph induced by the edges in each part is connected. In the problem Min-Max (Max-Min) balanced connected edge partition (BCEP), one searches for a connected edge  $k$ -partition of an edge-weighted graph minimizing the maximum (resp. maximizing the minimum) weight of the parts. This problem is equivalent to finding a connected  $k$ -partition of the vertices in the line graph of the input graph. The best known approximation for BCEP is for graphs with no edge weight larger than  $w(G)/2k$ . For such graphs, [13] give an algorithm that finds a connected edge  $k$ -partition, such that the weight of the heaviest subgraph is at most twice as large as the weight of the lightest subgraph, implying a 2-approximation for MIN-MAX and MAX-MIN BCEP. In comparison, our algorithms achieve the same approximation guarantee without restrictions on the edge weights.

► **Corollary 3.** *MIN-MAX BCEP and MAX-MIN BCEP have 2-approximations in polynomial time.*

An extension of BCP is demanding for fixed (possibly heterogeneous) size targets for each of the  $k$  parts. More precisely, given a graph  $G$  and  $w_1, \dots, w_k$  with  $\sum_{i=1}^k w_i = n$ , the task is to find a partition  $T_1, \dots, T_k$  where each  $T_i$  has size  $w_i$  and induces a connected subgraph. Such a connected  $k$ -partition with the fixed target weights exists for  $G$  only if  $G$  meets certain structural properties; a  $K_{1,3}$  for example has no connected 2-partition  $T_1, T_2$  with  $|T_1| = |T_2| = 2$ . A characterization of when such a connected partition always exists was independently proved by Györi [26] and Lovász [32]: They showed that in any  $k$ -connected graph a connected  $k$ -partition satisfying the target weights always exists. This result is the famous Györi-Lovász Theorem (GL theorem, for short):

► **Theorem 4** (Györi-Lovász Theorem [26, 32]). *Given a  $k$ -connected graph  $G = (V, E, w)$ ,  $n_1, \dots, n_k \in \mathbb{N}$  such that  $\sum_{i=1}^k n_i = |V|$ , and  $k$  terminal vertices  $t_1, \dots, t_k \in V$ , there exists a connected  $k$ -partition  $T_1, \dots, T_k$  of  $V$  such that for each  $i \in [k]$ ,  $|T_i| = n_i$  and  $t_i \in T_i$ .*

Recently, the theorem was generalized to vertex-weighted graphs as:

► **Theorem 5** (Weighted Györi-Lovász Theorem [7, 10, 28]). *Given a vertex-weighted  $k$ -connected graph  $G = (V, E, w)$ ,  $w_1, \dots, w_k \in \mathbb{N}$  such that  $\sum_{i=1}^k w_i = w(V)$ , and  $k$  terminal vertices  $t_1, \dots, t_k$ , there exists a connected  $k$ -partition  $T_1, \dots, T_k$  of  $V$  such that  $w_i - w_{\max} < w(T_i) < w_i + w_{\max}$ , and  $t_i \in T_i$  for each  $i \in [k]$ , where  $w_{\max}$  is the largest vertex weight.*

We refer to the partition guaranteed by the (weighted) GL theorem as *GL partition*. We will however not consider the terminal vertices in the GL partitions in this work.

The GL theorem has found some applications in the field of algorithms. Chen et. al. [10] use it for proving the existence of low-congestion *confluent flows* in  $k$ -connected graphs. Further, Löwenstein et. al. [33] and Chandran et. al. [7] use it for finding spanning trees with low *spanning tree congestion*. Perhaps, the reason why such a strong combinatorial statement has not found further applications is that we do not know how to efficiently compute GL

partitions. About five decades after the discovery of the GL theorem, polynomial time algorithms for finding a GL partition (even in the unweighted case without terminals) are only known for  $k \leq 4$  [37, 38, 28]. The fastest algorithm for general  $k$  takes  $\Omega(2^n)$  time [7, 29]. Neither are there any impossibility results to exclude efficient computability of such partitions. Even when  $k$  is part of the input, a polynomial time algorithm is not ruled out.

The absence of efficient algorithms for finding exact GL partitions motivates finding GL-style partitions that approximately satisfy the weight targets. In this paper we present polynomial time algorithms for such approximations. First we give an algorithm for a “half-bounded” GL partition, in the sense that we can guarantee an approximate upper or lower bound on the weight of the parts.

► **Theorem 6.** *Let  $G = (V, E, w)$  be a  $k$ -connected vertex-weighted graph and  $w_1, \dots, w_k \in \mathbb{N}$  with  $\sum_{i=1}^k w_i = w(G)$ , and  $\min_{i \in [k]} w_i \geq \max_{v \in V} w(v)$ . A connected  $k$ -partition  $T_1, \dots, T_k$  of  $V$  such that either  $w(T_i) \geq \frac{1}{3}w_i$  for every  $i \in [k]$  (lower-bound version) or  $w(T_i) \leq 3w_i$  for every  $i \in [k]$  (upper-bound version) can be computed in time  $\mathcal{O}(k|V|^2|E|)$ .*

We then extend this result to a lower and upper bounded partition.

► **Theorem 7.** *Let  $G = (V, E, w)$  be a  $k$ -connected vertex-weighted graph and  $w_1, \dots, w_k \in \mathbb{N}$  with  $\sum_{i=1}^k w_i = w(G)$ , and  $\min_{i \in [k]} w_i \geq \max_{v \in V} w(v)$ , and  $r := \frac{\max_{i \in [k]} w_i}{\min_{j \in [k]} w_j}$ . Then, a connected  $k$ -partition  $T_1, \dots, T_k$  of  $V$  such that  $\frac{1}{3}w_i \leq w(T_i) \leq \max\{r, 3\}w_i$  for every  $i \in [k]$  can be found in time  $\mathcal{O}(k|V|^2|E|)$ .*

In particular, Theorem 7 implies the following approximately balanced partition of  $k$ -connected graphs.

► **Corollary 8.** *Let  $G = (V, E, w)$  be a  $k$ -connected vertex-weighted graph such that  $w(G) \geq k \max_{v \in V} w(v)$ . Then, a connected  $k$ -partition  $T_1, \dots, T_k$  of  $V$  such that  $\frac{1}{3} \left\lfloor \frac{w(G)}{k} \right\rfloor \leq w(T_i) \leq 3 \left\lceil \frac{w(G)}{k} \right\rceil$  for every  $i \in [k]$  can be found in time  $\mathcal{O}(k|V|^2|E|)$ .*

To the best of our knowledge, these are the first polynomial time algorithms that *approximate* the GL theorem. We believe that such an efficient approximation will result in the theorem being used for developing algorithms in the future. Especially, we are hopeful that the both-side approximation for balanced connected partition of  $k$ -connected graphs will find applications. We remark, however that for the above mentioned applications of confluent flows and spanning tree congestion, the terminal vertices are essential and hence our algorithms cannot be used. An interesting future direction would be to extend our results to the setting with terminals.

Observe that Corollary 8 in some sense yields a 3-approximation simultaneously for MIN-MAX and MAX-MIN BCP in  $k$ -connected graphs. In this regard, it is interesting to note that the  $+/-w_{\max}$  slack given in the weighted GL theorem is enough to retain hardness in the following sense: even for  $k = 2$ , MIN-MAX BCP and MAX-MIN BCP remain strongly NP-hard when restricted to 2-connected graphs; and the corresponding hardness-proof given in [8] also constructs an instance with  $w(G) \geq k \max_{v \in V} w(v)$ . This hardness can be extended to  $k$ -connected graphs for any fixed  $k \geq 2$  (see [8][Theorem 3] for more details).

Lastly, we point out that this paper is only a short version and refer the reader for more technical details and complete proofs to the full version of it.

## 1.1 Related work

Both variants of BCP were first introduced for trees [36, 31]. Under this restriction, a linear time algorithm was provided for both variants in [25]. This is particularly important since different heuristics transform the original instance to a tree to efficiently solve the problem, see [13, 39]. For both variants of BCP, a 3-approximation is given in [6], which is the best known approximation in polynomial time. With respect to lower bounds, it is known that there exists no approximation for MAX-MIN BCP with a ratio below  $6/5$ , unless  $P = NP$  [8]. For the unweighted case, a  $\frac{k}{2}$ -approximation for MIN-MAX BCP with  $k \geq 3$ , is given in [11].

Balanced connected partitions for fixed small values of  $k$ , denoted  $BCP_k$ , have also been studied extensively. The restriction  $BCP_2$ , i.e. balanced connected bipartition, is already NP-hard [5]. On the positive side, a  $\frac{4}{3}$ -approximation for MAX-MIN  $BCP_2$  is given in [12], and in [11] this result is used to derive a  $\frac{5}{4}$ -approximation for MIN-MAX  $BCP_2$ . Considering tripartitions, MAX-MIN  $BCP_3$  and MIN-MAX  $BCP_3$  can be approximated with ratios  $\frac{5}{3}$  and  $\frac{3}{2}$ , respectively [9].

Regarding special graph classes, BCP has been investigated in grid graphs and series-parallel graphs. While it was shown that BCP is NP-hard for arbitrary grid graphs [2], the MAX-MIN BCP can be solved in polynomial time for ladders, i.e., grid graphs with two rows [3]. For the class of series-parallel graphs, Ito et. al. [30] observed that BCP remains weakly NP-hard (by a simple reduction from the Partition problem) and gave a pseudo-polynomial-time algorithm for both variants of BCP. They also showed that their algorithm can be extended to graphs with bounded tree-width.

The GL Theorem was independently proved by Györi [26] and Lovász [32]. Györi used an elementary graph theoretic approach while Lovász used ideas from topology. Lovász's proof also works for directed graphs. The Györi-Lovász Theorem is extended to weighted directed graphs by Chen et. al. [10] and Györi's original proof was generalized to weighted undirected graphs by Chandran et. al. [7]. Both papers only gave upper bounds of  $w_i + w_{\max}$  on the weight of partition  $T_i$  and did not provide any lower bounds. Later Hoyer [28] showed that the method of Chandran et. al. [7] can be also extended to give the lower bound  $w_i - w_{\max}$ , even for directed graphs. Polynomial algorithms to also compute GL partitions are only known for the particular cases  $k = 2, 3, 4$  [37, 38, 28] and all  $k \geq 5$  are still open.

## 2 Preliminaries

By  $\mathbb{N}$  we denote the natural numbers without zero. We use  $[k]$  to denote the set  $\{1, \dots, k\}$ .

All the graphs that we refer to in this paper are simple, finite and connected. Consider a graph  $G = (V, E)$ . We denote by  $V(G)$  and  $E(G)$  the set of vertices and edges of  $G$  respectively, and if the graph we refer to is clear, we may simply write  $V$  and  $E$ . For a set of vertex sets  $\mathcal{S} \subseteq 2^V$  we use  $V(\mathcal{S})$  to denote  $\bigcup_{S \in \mathcal{S}} S$ . We denote an edge  $e = \{u, v\} \in E(G)$  by  $uv$  and the neighborhood of a vertex  $v \in V$  in  $G$  by  $N_G(v) = \{u \in V \mid uv \in E(G)\}$ . Similarly we denote the neighborhood of a vertex set  $V' \subseteq V$  in  $G$  by  $N_G(V')$ , that is  $\bigcup_{v \in V'} N_G(v) \setminus V'$ . We may omit the subscript  $G$  when the graph is clear from the context. We use  $\Delta(G)$  to denote the maximum degree of  $G$ .

We denote a *vertex-weighted graph* by  $G = (V, E, w)$  where  $w$  is a function assigning integer weights to vertices  $w: V \rightarrow \mathbb{N}$ , and  $V$  and  $E$  are vertex and edge sets. We denote by  $w_{\min}$  and by  $w_{\max}$ ,  $\min_{v \in V} w(v)$  and  $\max_{v \in V} w(v)$ , respectively. For any  $V' \subseteq V$ , we use  $w(V')$  to denote the sum of weights of the vertices in  $V'$ . For a subgraph  $H$  of  $G$  we use  $w(H)$  to denote  $w(V(H))$ , and refer to it as the *weight of the subgraph*  $H$ . For a rooted tree  $T$  and a vertex  $x$  in  $T$ , we use  $T_x$  to denote the rooted subtree of  $T$  rooted at  $x$ .

For  $V' \subseteq V$  we denote by  $G[V']$  the graph induced by  $V'$ , i.e.  $G[V'] = (V', E')$  with  $E' = E \cap (V' \times V')$ . For vertex-weighted graphs, induced subgraphs inherit the vertex-weights given by  $w$ . For  $V' \subseteq V$  we also use  $G - V'$  to denote the subgraph  $G[V \setminus V']$ . Similarly, if  $V'$  is a singleton  $\{v\}$  we also write  $G - v$ . For graphs  $G_1$  and  $G_2$ , we use  $G_1 \cup G_2$  to denote the graph on vertices  $V(G_1) \cup V(G_2)$  with edge set  $E(G_1) \cup E(G_2)$ .

Let  $\mathcal{U} = \{U_1, \dots, U_r\}$  be such that each  $U_i \subseteq V(G)$ . We call  $\mathcal{U}$  a *connected packing* of  $V(G)$  if each  $G[U_i]$  is connected, and the sets in  $\mathcal{U}$  are pairwise disjoint. A connected packing  $\mathcal{U}$  is called *connected vertex partition* (CVP) of  $V$ , if also  $\cup_{i=1}^r U_i = V(G)$ . We denote a CVP that has  $k$  vertex sets as  $\text{CVP}_k$ . For any  $\mathcal{U}' \subseteq \mathcal{U}$ , we define  $V(\mathcal{U}') := \cup_{U' \in \mathcal{U}'} U'$ , and the weight  $w(\mathcal{U}') := w(V(\mathcal{U}'))$ . Let  $\mathcal{I}$  be an interval. If  $\mathcal{U}$  is a CVP and  $w(U_i) \in \mathcal{I}$  for all  $i \in [r]$ , then we say that  $\mathcal{U}$  is an  $\mathcal{I}$ -connected vertex ( $r$ -)partition ( $\mathcal{I}$ -CVP $_k$  or just  $\mathcal{I}$ -CVP) of  $V$ . If  $\mathcal{U}$  is a connected-packing and  $w(U_i) \in \mathcal{I}$  for all  $i \in [r]$ , then we say that  $\mathcal{U}$  is a  $\mathcal{I}$ -connected packing of  $V$ .

### 3 Approximation for BCP on $c$ -claw-free graphs

In this section we give an idea of how to prove Theorems 1 and 2 by giving a  $(c - 1)$ -approximation for MAX-MIN BCP and MIN-MAX BCP on  $K_{1,c}$ -free graphs. We assume  $c \geq 3$  as  $c \leq 2$  gives trivial graph classes. We first show that a connected partition for  $K_{1,c}$ -free graphs with parts of size in  $[\lambda, (c - 1)\lambda]$  for some fixed  $\lambda$  can be found in linear time. For MAX-MIN BCP, this algorithm has to be called many times while doing a binary search for the optimum value. We point out that it is not difficult to adapt these algorithms to unconnected graphs achieving the same approximation results.

Exploiting that each vertex in any DFS-tree of a  $K_{1,c}$ -free graph has at most  $c - 1$  children, we can carefully extract connected components of a fixed size while also maintaining a DFS-tree for the remaining graph. Also, this can be done very efficiently, as stated in the following result.

► **Lemma 9.** *Given a  $K_{1,c}$ -free graph  $G$  and a DFS-tree of  $G$ . For any  $w(G) \geq \lambda \geq w_{\max}$ , there is an algorithm that finds a connected vertex set  $S$  such that  $\lambda \leq w(S) < (c - 1)\lambda$  and  $G - S$  is connected, in  $\mathcal{O}(|V|)$  time. Furthermore, the algorithm finds a DFS-tree of  $G - S$ .*

We use `BalancedPartition` to denote the algorithm that exhaustively applies Lemma 9. Observe that `BalancedPartition` produces a connected partition  $S_1, \dots, S_m$  where  $w(S_i) \in [\lambda, (c - 1)\lambda]$  for every  $i \in [m - 1]$  and  $w(S_m) < (c - 1)\lambda$  in linear time, where the achieved runtime follows by saving already processed subtrees.

Theorem 1 now follows from running `BalancedPartition` with  $\lambda = \max\{w_{\max}, \frac{w(G)}{k}\}$ . Note that this choice of  $\lambda$  is a trivial lower bound for the optimum value.

As already mentioned, to prove Theorem 2, we first need to find an input parameter  $\lambda$  for Algorithm `BalancedPartition` that provides the desired  $(c - 1)$ -approximation.

Let  $(G, k)$  be an instance of MAX-MIN BCP, where  $G$  is a  $K_{1,c}$ -free graph. Let  $X^*$  be the optimal value for the instance  $(G, k)$ . For any given  $X \leq w(G)/k$ , we design an algorithm that either gives a  $[\lfloor X/(c - 1) \rfloor, \infty)$ -CVP $_k$ , or reports that  $X > X^*$ . Note that  $X^* \leq w(G)/k$ . Once we have this procedure in hand, a binary search for the largest  $X$  in the interval  $(0, \lceil w(G)/k \rceil]$  for which we find a  $[\lfloor X/(c - 1) \rfloor, \infty)$ -CVP $_k$  can be used to obtain an approximate solution for MAX-MIN BCP.

**Algorithm MaxMinApx.** First remove all vertices of weight more than  $\lambda = \lfloor X/(c - 1) \rfloor$  and save them in  $H$ . Then save the connected components of weight less than  $\lambda$  in  $\mathcal{Q}$ .

Let  $\mathcal{V} = \{V_1, \dots, V_\ell\}$  be the connected components of  $G - (H \cup V(\mathcal{Q}))$ . Apply algorithm `BalancedPartition` on each  $G[V_i]$  with  $\lambda$  as input parameter to obtain  $\mathcal{S}^i = \{S_1^i, \dots, S_{m_i}^i\}$  for every  $i \in [\ell]$ . If for some  $i \in [\ell]$  the weight  $w(S_{m_i}^i)$  is less than  $\lambda$ , then merge this vertex set with  $S_{m_i-1}^i$  and accordingly update  $\mathcal{S}^i$ . Further, compute a  $(\lambda, \infty)$ -CVP $_{|H|}$   $\mathcal{S}^H$  of  $G[H \cup V(\mathcal{Q})]$  as follows: for each  $h \in H$ , we have a set  $S^h \in \mathcal{S}^H$  with  $h \in S^h$ ; we add each  $Q \in \mathcal{Q}$  to some  $S^h$  such that  $h \in N(Q)$ . Let  $\mathcal{S} = \mathcal{S}^H \cup \bigcup_{i=1}^{\ell} \mathcal{S}^i$ . If  $|\mathcal{S}| \geq k$ , then merge connected sets arbitrarily in  $\mathcal{S}$  until  $|\mathcal{S}| = k$  and return  $\mathcal{S}$ . If  $|\mathcal{S}| < k$ , report that  $X > X^*$ .

We point out that a  $[\lambda, \infty)$ -CVP $_j$  with  $j > k$ , can easily be transformed to a  $[\lambda, \infty)$ -CVP $_k$ , since the input graph is connected. It is not hard to see that if algorithm `MaxMinApx` returns  $\mathcal{S}$  then this is a  $[\lambda, \infty)$ -CVP $_k$  of  $V$ . The most complicated part of proving that `MaxMinApx` works correctly is showing that if it terminates with  $|\mathcal{S}| < k$  and reports  $X > X^*$  that this is indeed true.

► **Lemma 10.** *If Algorithm `MaxMinApx` terminates with  $|\mathcal{S}| < k$ , then  $X > X^*$ .*

**Proof.** Let  $H, \mathcal{Q}, \mathcal{V} = \{V_1, \dots, V_\ell\}$  be the computed vertices and connected vertex sets in the algorithm for  $\lambda = \lfloor X/(c-1) \rfloor$ , respectively. Recall that  $w(V_i) \geq \lambda$  for every  $V_i \in \mathcal{V}$  and  $w(Q) < \lambda$  for every  $Q \in \mathcal{Q}$ . Let  $\mathcal{S}^* = \{S_1^*, \dots, S_k^*\}$  be an optimal solution of  $(G, k)$ , i.e.,  $\mathcal{S}^*$  is an  $[X^*, \infty)$ -CVP $_k$  of  $V$ . Consider the sets  $\mathcal{V}^{H \cup \mathcal{Q}} := \{S_i^* \in \mathcal{S}^* \mid S_i^* \cap (H \cup V(\mathcal{Q})) \neq \emptyset\}$ ,  $\mathcal{V}^1 := \{S_i^* \in \mathcal{S}^* \mid S_i^* \cap V_1 \neq \emptyset\} \setminus \mathcal{V}^{H \cup \mathcal{Q}}$ ,  $\dots$ ,  $\mathcal{V}^\ell := \{S_i^* \in \mathcal{S}^* \mid S_i^* \cap V_\ell \neq \emptyset\} \setminus \mathcal{V}^{H \cup \mathcal{Q}}$ . We claim that these sets are a partition of  $\mathcal{S}^*$ . This follows directly from the fact that  $H$  separates all  $V_i \in \mathcal{V}$  and all  $Q \in \mathcal{Q}$  from each other. That is, for an  $i \in [k]$  and  $j \in [\ell]$  the connected vertex set  $S_i^*$  with  $S_i^* \cap V_j \neq \emptyset$  and  $S_i^* \cap V \setminus V_j \neq \emptyset$  contains at least one  $h \in H$  and hence  $S_i^* \in \mathcal{V}^{H \cup \mathcal{Q}}$ . Otherwise, if  $S_i^* \subseteq V_j$ , then  $S_i^* \in \mathcal{V}^j$ .

Suppose `MaxMinApx` terminates with  $|\mathcal{S}| < k$  although  $X \leq X^*$ . We show that  $|\mathcal{V}^{H \cup \mathcal{Q}}| \leq |H|$  and  $|\mathcal{V}^i| \leq |\mathcal{S}^i|$  for every  $i \in [\ell]$ , implying that  $|\mathcal{S}^*| \leq |H| + \sum_{i=1}^{\ell} |\mathcal{S}^i| = |\mathcal{S}| < k$ , which contradicts  $|\mathcal{S}^*| = k$ .

First, we show  $|\mathcal{V}^{H \cup \mathcal{Q}}| \leq |H|$ . For this, it is sufficient to prove that  $S_i^* \cap H \neq \emptyset$  for each  $S_i^* \in \mathcal{V}^{H \cup \mathcal{Q}}$  as  $\mathcal{S}^*$  is a partition of  $V$ . We prove this by contradiction. Suppose there is an  $S_i^* \in \mathcal{V}^{H \cup \mathcal{Q}}$ , such that  $S_i^* \cap H = \emptyset$ . This implies that  $S_i^* \subseteq Q$  for some  $Q \in \mathcal{Q}$ , since  $H$  separates every  $Q \in \mathcal{Q}$  from every other  $Q' \in \mathcal{Q} \setminus \{Q\}$  and from the vertices  $V \setminus (H \cup V(\mathcal{Q}))$ . Thus,  $w(S_i^*) \leq w(Q) < \lambda$  by the definition of  $\mathcal{Q}$  and therefore  $w(S_i^*) < \lambda = \lfloor X/(c-1) \rfloor \leq \lfloor X^*/(c-1) \rfloor$ , contradicting  $\min_{i \in [k]} w(S_i^*) = X^*$ .

It remains to show that  $|\mathcal{V}^i| \leq |\mathcal{S}^i|$  for every  $i \in [\ell]$ . Fix an  $i \in [\ell]$  and let  $G[V_i]$  with  $\lambda$  be the input when calling algorithm `BalancedPartition`. Observe that the input is valid, since  $G[V_i]$  is connected by definition and  $w(G[V_i]) \geq \lambda \geq \max_{v \in V_i} w(v)$  as  $H$  contains all vertices that have weight more than  $\lambda$ . Algorithm `BalancedPartition` provides a CVP  $\mathcal{S}^i = \{S_1^i, \dots, S_{m_i}^i\}$  of  $V_i$  with  $w(S_j^i) \in [\lambda, (c-1)\lambda)$  for every  $j \in [m_i-1]$  and  $w(S_{m_i}^i) < (c-1)\lambda$ . Consider  $\mathcal{S}^i$  before merging, i.e. we do not merge  $S_{m_i}^i$  to  $S_{m_i-1}^i$  in the algorithm `MaxMinApx` if  $w(S_{m_i}^i) < \lambda$ . That is,  $w(S_{m_i}^i) < \lambda$  is possible, and we need to show  $|\mathcal{V}^i| \leq m_i - 1 = |\mathcal{S}^i| - 1$ . Observe for  $\mathcal{S}^* \in \mathcal{V}^i$  that  $\mathcal{S}^* \subseteq V_i$ , i.e.  $\sum_{j=1}^{m_i} w(S_j^i) \geq \sum_{\mathcal{S}^* \in \mathcal{V}^i} w(\mathcal{S}^*)$ . As a result, we have  $|\mathcal{S}^i|X \geq |\mathcal{S}^i|(c-1)\lambda > \sum_{j=1}^{m_i} w(S_j^i) \geq \sum_{\mathcal{S}^* \in \mathcal{V}^i} w(\mathcal{S}^*) \geq |\mathcal{V}^i|X^*$ . Consequently, by  $X \leq X^*$  we obtain  $|\mathcal{V}^i| < |\mathcal{S}^i|$ , which leads to  $|\mathcal{V}^i| \leq |\mathcal{S}^i| - 1$ . ◀

## 4 Approximation of the Györi-Lovász Theorem for $k$ -connected Graphs

Our algorithms for the approximate GL theorems are based mainly on the following combinatorial lemma concerning certain vertex separators, that leads to useful structures in

$k$ -connected graphs. Let  $G = (V, E, w)$  be a connected vertex-weighted graph and let  $\lambda$  be an integer. We say  $s \in V$  is a  $\lambda$ -separator if all connected components of  $G - \{s\}$  weigh less than  $\lambda$ . We say  $G$  is  $\lambda$ -dividable if there is a  $[\lambda, \infty)$ -CVP<sub>2</sub> of  $V$ .

► **Lemma 11** ([6]). *Let  $G = (V, E, w)$  be a connected vertex-weighted graph and let  $\lambda > w_{\max}$  be an integer. If  $w(G) > 3(\lambda - 1)$ , then either  $G$  is  $\lambda$ -dividable or there is a  $\lambda$ -separator. Furthermore, finding the connected vertex sets in case  $G$  is  $\lambda$ -dividable and finding the  $\lambda$ -separator in the other case can be done in  $\mathcal{O}(|V||E|)$  time.*

#### 4.1 Bounded Partition for $k$ -connected Graphs

In this section, we give an algorithm for computing approximate GL partitions with one-side approximation bound (either lower bound or upper bound), thus proving Theorem 6. For this, we first use the following theorem, from which Theorem 6 follows as below.

► **Theorem 12.** *Let  $G = (V, E, w)$  be a  $k$ -connected vertex-weighted graph and let  $w_1, \dots, w_k \in \mathbb{N}$  with  $\sum_{i=1}^k w_i = w(G)$ , and  $\min_{i \in [k]} w_i \geq \max_{v \in V} w(v)$ . A set of connected vertex sets  $\mathcal{T} = \{T_1, \dots, T_\ell\}$  with  $\ell \leq k$  and  $\alpha w_i \leq w(T_i) \leq 3\alpha w_i$  for every  $i \in [\ell]$  can be computed in time  $\mathcal{O}(k|V|^2|E|)$ . Moreover, if  $\ell < k$ , then  $\mathcal{T}$  is also a CVP of  $V$ .*

By Theorem 12 we can derive Theorem 6 using  $\alpha = 1/3$  and  $\alpha = 1$  for the lower bound and upper bounded version, respectively.

In the following we always assume that  $w_1, \dots, w_k$  is sorted in descending order. To now give the algorithm proving Theorem 12, we make use of Lemma 11. For this, we first need to ensure that  $w_{\max} < \alpha w_k$ . Therefore, we perform a preprocessing step until we reach an instance that satisfies  $w_{\max} < \alpha w_k$ . Suppose  $w_{\max} \geq \alpha w_\ell$ , where  $\ell$  is the smallest index in  $[k]$  that satisfies this inequality. We remove a vertex  $v_{\max}$  with  $w(v_{\max}) = w_{\max}$  from  $G$  and  $w_\ell$  from  $\mathcal{W}$ . Further, we set  $T_\ell = \{v_{\max}\}$  and consider the set for index  $\ell$  to be finished, i.e., we now aim to find a set  $\mathcal{T} = \{T_1, \dots, T_{\ell-1}, T_{\ell+1}, \dots, T_k\}$  according to  $\mathcal{W} = \{w_1, \dots, w_{\ell-1}, w_{\ell+1}, \dots, w_k\}$ . Observe that since we only deleted one vertex,  $G$  now is at least  $(k - 1)$ -connected, and  $|\mathcal{W}| = k - 1$ . Note that since  $w_{\max} \leq w_k \leq w_\ell \leq 3\alpha w_\ell$ , we have  $w(T_\ell) \in [\alpha w_\ell, 3\alpha w_\ell]$  as required. Also, we obtain  $w(G) \geq \sum_{w_j \in \mathcal{W}} w_j$  after removing  $w_\ell$  from  $\mathcal{W}$  and  $v_{\max}$  from  $G$ .

After this preprocessing, we can assume that we have a  $k$ -connected graph  $G = (V, E, w)$  and natural numbers  $w_1, \dots, w_k$  sorted in descending order, where  $\sum_{i=1}^k w_i \leq w(G)$  and  $w_{\max} < \alpha w_k$ .

On such a graph  $G$  we then gradually build a packing  $\mathcal{T}$  with the help of Lemma 11. During our algorithm to build  $\mathcal{T}$  we ensure that at each step  $\mathcal{T} = \{T_1, T_2, \dots, T_{i-1}\}$  where each  $T_j \in \mathcal{T}$  is a connected vertex set with weight in  $[\alpha w_j, 3\alpha w_j]$  for each  $j \in [i - 1]$ . We then search in the remaining graph for the next set  $T_i$  and always use  $\overline{G}$  to denote the graph  $G \setminus V(\mathcal{T})$ . We say a connected subgraph is  $i$ -small if it has weight less than  $\alpha w_i$  and  $i$ -big otherwise. In case we reach a situation, where  $\overline{G}$  has no connected component that is  $i$ -big, we have to alter the already built sets  $T_1, T_2, \dots, T_{i-1}$  to build  $T_i$ . For this, we use  $\mathcal{T}_a$  to denote the set of all  $T_j \in \mathcal{T}$  that have no  $(\alpha w_j)$ -separator, and  $\mathcal{T}_b$  to denote the set of all  $T_j \in \mathcal{T}$  that have an  $(\alpha w_j)$ -separator. For  $T_j \in \mathcal{T}_b$  with an  $(\alpha w_j)$ -separator  $s$  we use  $C(T_j)$  to denote the connected components of  $G[T_j \setminus \{s\}]$  (if there is more than one  $(\alpha w_j)$ -separator, fix one of them arbitrarily). The following Algorithm BoundedGL formally explains our routine to build  $\mathcal{T}$ .



**Algorithm BoundedGL**

1. Initialize  $\mathcal{T} := \emptyset$  as container for the desired connected-vertex-packing  $T_1, \dots, T_k$  of  $G$  and initialize  $i := 1$  as an increment-variable.
2. While  $\overline{G} = G \setminus V(\mathcal{T})$  is not the empty graph: //main loop
  - a. Find a connected vertex set  $T_i$  having weight in  $[\alpha w_i, 3\alpha w_i]$ , add  $T_i$  to  $\mathcal{T}$ , and increment  $i$  by one. If  $i = k + 1$  then terminate the algorithm.  
// See Lemma 14 for correctness of this step
  - b. While  $\overline{G}$  is not empty and has no  $i$ -big connected component: //inner loop  
Pick an  $i$ -small connected component  $Q$  of  $\overline{G}$ . Pick a  $T_j \in \mathcal{T}$  such that either  $T_j \in \mathcal{T}_a$  and  $Q$  has an edge to  $T_j$  (Case 1), or  $T_j \in \mathcal{T}_b$  and  $Q$  has an edge to some component  $Q' \in C(T_j)$  (Case 2). // The occurrence of at least one of these cases is shown in Lemma 15.  
If  $w(T_j \cup Q) \leq 3\alpha w_j$  then update  $T_j$  to  $T_j \cup Q$ . Otherwise:
    - i. Case 1 ( $T_j \in \mathcal{T}_a$ ): Apply the following **Divide-routine** on  $T_j \cup Q$ : Use Lemma 11 to compute a  $[\alpha w_j, \infty)$ -CVP<sub>2</sub>  $V_1, V_2$  of  $T_j \cup Q$ . Set  $T_j = V_2$  (i.e.  $V_1$  goes to  $\overline{G}$ ).
    - ii. Case 2 ( $T_j \in \mathcal{T}_b$ ): remove  $Q'$  from  $T_j$  (i.e.  $Q'$  goes back to  $\overline{G}$ ) if  $T_j \cup Q$  is not  $\alpha w_j$ -dividable. Otherwise, apply divide routine on  $T_j \cup Q$ .

To prove the correctness of the algorithm, we will show that the following invariant is maintained.

► **Lemma 13.** *Algorithm BoundedGL maintains a packing  $\mathcal{T} = \{T_1, T_2, \dots, T_{i-1}\}$  where each  $T_j \in \mathcal{T}$  is a connected vertex set having weight in  $[\alpha w_j, 3\alpha w_j]$ .*

Towards proving this we use the following fact.

► **Lemma 14.** *Whenever the divide routine in algorithm BoundedGL is to be executed, there is at least one  $i$ -big component in  $\overline{G}$ .*

**Proof.** When  $i = 1$  i.e. in the first main loop iteration, this holds because  $\overline{G} = G$  is connected and  $\alpha w_1 \leq w_1 \leq \sum_{i=1}^k w_i \leq w(G)$ . For the subsequent iterations, the divide routine is only applied after the inner loop is terminated which only happens if either  $\overline{G}$  is empty or has an  $i$ -big component. In case  $\overline{G}$  is empty, then the algorithm terminates. So, if the algorithm applies the divide routine in the inner loop, then  $\overline{G}$  has an  $i$ -big component. ◀

**Proof of Lemma 13.** We increment  $i$  only in Step 2a. Before incrementing  $i$ , we add  $T_i$  to  $\mathcal{T}$  while ensuring that  $w(T_i) \in [\alpha w_i, 3\alpha w_i]$  and  $G[T_i]$  is connected. In Lemma 14, we prove that whenever the divide routine is about to be executed, there is an  $i$ -big component in  $\overline{G}$ , ensuring the existence of such a  $T_i$ . Once a  $T_j$  is added to  $\mathcal{T}$ , it is then modified only in Step 2b. So let us look into how it gets modified in Step 2b. If the condition  $w(T_j \cup Q) \leq 3\alpha w_j$  is satisfied then it is clear that the new  $T_j = T_j \cup Q$  also satisfies the weight constraints. Since  $Q$  has an edge to  $T_j$  and  $T_j$  and  $Q$  each were connected, it is also clear that the new  $T_j$  remains connected. So now consider the case when  $w(T_j \cup Q) > 3\alpha w_j$ . In Case 1 ( $T_j \in \mathcal{T}_a$ ), we call the divide routine and the new  $T_j$  is the set  $V_2$  returned by the routine. The set  $V_2$  is connected due to the property of the divide routine. To see that it also satisfies the weight constraints, observe that  $w(V_1 \cup V_2)$  is at most  $4\alpha w_j$  as  $w(T_j)$  was at most  $3\alpha w_j$  and  $w(Q) < \alpha w_j \leq \alpha w_j$ . Since  $w(V_1), w(V_2) \geq \alpha w_j$ , we then have  $w(V_2) \in [\alpha w_j, 3\alpha w_j]$ . So it only remains to consider Case 2 ( $T_j \in \mathcal{T}_b$ ). The case when  $T_j \cup Q$  is  $\alpha w_j$ -dividable is analog to Case 1. We know  $w(Q') < \alpha w_j$  by definition. Also, since  $w(T_j \cup Q)$  was more than  $3\alpha w_j$  and  $w(Q) < \alpha w_j$ , we have that  $w(T_j)$  was at least  $2\alpha w_j$ . Thus the new  $T_j = T_j \setminus Q'$  has weight in  $[\alpha w_j, 3\alpha w_j]$ . Also, the new  $T_j$  is connected by the definition of  $Q'$ . ◀

It is clear from the algorithm that termination occurs only if  $\overline{G}$  is empty or  $i = k + 1$ . Then using Lemma 13, it is clear that  $\mathcal{T}$  contains the required packing as claimed in Theorem 12, provided that Step 2b runs correctly and the inner loop terminates, which we prove in the two lemmas below.

► **Lemma 15.** *In Step 2b at least Case 1 or Case 2 occurs.*

**Proof.** Suppose Case 1 does not occur i.e.,  $Q$  does not have an edge to any  $T_j \in \mathcal{T}_a$ . For  $T_j \in \mathcal{T}_b$ , let  $s_j$  denote the fixed  $(\alpha w_j)$ -separator vertex. Since  $G$  is  $k$ -connected and  $|\mathcal{T}| < k$ , there is an edge from  $Q$  to at least one vertex in  $\bigcup_{T_j \in \mathcal{T}_b} T_j \setminus \{s_j\}$ . Thus, Case 2 occurs. ◀

► **Lemma 16.** *The inner loop runs correctly and terminates after at most  $|V|^2$  iterations.*

**Proof.** The occurrence of one of the two cases in Step 2b is shown in Lemma 15. For the correctness of Case 1, observe that we can use Lemma 11 to divide  $T_j \cup Q$ , as  $T_j \cup Q$  cannot have an  $(\alpha w_j)$ -separator (this would give an  $(\alpha w_j)$ -separator for  $T_j$  implying that  $T_j \in \mathcal{T}_b$ ). It remains to prove that the inner loop terminates as claimed. If Step 2(b)i is executed, then an  $i$ -big component is created in  $\overline{G}$  as  $\overline{G}$  now contains  $V_1$  returned by the divide-routine, and hence the loop is terminated. The same yields if we apply the divide routine in Step 2(b)ii. So, suppose the inner loop never executes the divide routine. In the other cases, either a connected component is deleted from  $\overline{G}$  or new vertices are added to a connected component in  $\overline{G}$ . Also note that new connected components are not introduced to  $\overline{G}$  and vertices are not deleted from existing connected components (except when the whole connected component is removed; also, two or more connected components may merge due to the introduction of new vertices to  $\overline{G}$ ). Thus, after  $|V|^2$  iterations either there is an  $i$ -big component or  $\overline{G}$  is empty. ◀

It is tempting to think that one could use Theorem 12 to derive a CVP  $\mathcal{T} = \{T_1, \dots, T_k\}$  such that  $\alpha w_i \leq w(T_i) \leq 3\alpha w_i$  for each  $i \in [k]$ . If Algorithm **BoundedGL** terminates with  $\ell < k$ , then  $\mathcal{T}$  is a partition of the vertices in  $G$ , and we only have trouble with the lower bound on  $T_j$  for  $\ell < j \leq k$ . Otherwise, if it terminates with  $\ell = k$ , then  $\mathcal{T}$  satisfies all lower bounds, but might not be a partition. Assigning the remaining vertices in  $G$  to turn  $\mathcal{T}$  into a CVP in this case might yield violations of the upper bound. Since  $\alpha = 1$  yields the first, and  $\alpha = \frac{1}{3}$  the second case, one might think that choosing the correct  $\alpha$  in between would result in a CVP with  $\ell = k$ . Unfortunately, Algorithm **BoundedGL** does not have a monotone behaviour w.r.t.  $\alpha \in (\frac{1}{3}, 1)$  in the sense that for two values  $\frac{1}{3} < \alpha_1 < \alpha_2 < 1$ , the case  $\ell = k$  for  $\alpha_1$  does not imply  $\ell = k$  for  $\alpha_2$ . Thus, even if we could prove the existence of an optimal value for  $\alpha$ , we have no way to search for it.

## 4.2 Both-side Bounded Partition for $k$ -connected Graphs

In this section, we give the algorithms to derive Theorem 7 by providing a both-side bounded approximate GL partition. For achieving a simultaneous lower and upper bounded partition, as a starting point, we apply Theorem 12 with  $\alpha = \frac{1}{3}$  obtaining a lower and upper bounded packing  $\mathcal{T} = \{T_1, \dots, T_k\}$  with  $\frac{1}{3}w_i \leq w(T_i) \leq w_i$  for every  $i \in [k]$ . As long as  $\mathcal{T}$  is not a CVP of  $V$  yet, we transfer a subset of the remaining vertices  $V \setminus V(\mathcal{T})$  through a path in an auxiliary graph to elements in  $\mathcal{T}$ , while making sure that for each  $i$ ,  $\frac{1}{3}w_i \leq w(T_i) \leq \max\{r, 3\}w(T_i)$ . We call one such transfer a *transferring-iteration*. We define  $\mathcal{T}^* := \{T_1, T_2, \dots, T_j\}$  where  $j$  is the smallest number such that  $w(T_i) \geq w_i$  for  $i \in [j]$  and  $w(T_{j+1}) < w_{j+1}$ . In case of  $w_a = w_b$  and  $w(T_a) \geq w_a$ , but  $w(T_b) < w_b$  we assume that  $a < b$ . Note that this is easily realizable by a relabeling of indices. Observe that  $\mathcal{T}^* = \emptyset$  if  $w(T_1) < w_1$ . As a measure of progress,

we guarantee in each transferring-iteration that either the cardinality of  $\mathcal{T}^*$  increases, or the number of vertices in  $V(\mathcal{T})$  increases. Also, the cardinality of  $\mathcal{T}^*$  is non-decreasing throughout the algorithm. Note that if  $T_i \in \mathcal{T}^*$  for all  $i$ , then it follows that  $w(T_i) = w_i$  for all  $i$  and moreover,  $\mathcal{T}$  is a CVP.

Let  $\mathcal{T} = \{T_1, \dots, T_k\}$  be a connected packing of  $V$  in  $G$  with  $w(T_i) \leq \max\{r, 3\}w_i$  for every  $i \in [k]$ . We use  $\mathcal{Q}$  to denote the vertex sets forming the connected components of  $G[V \setminus V(\mathcal{T})]$ . We define  $\mathcal{T}^+ := \{T_i \in \mathcal{T} \mid w(T_i) \geq w_i\}$  and  $\mathcal{T}^- := \mathcal{T} \setminus \mathcal{T}^+$ . Note that  $\mathcal{T}^* \subseteq \mathcal{T}^+$ . Analogous to section 4.1, we define  $\mathcal{T}_a^+$  as the set of  $T_i \in \mathcal{T}^+$  that do *not* have a  $w_i$ -separator vertex and  $\mathcal{T}_b^+$  to be the ones in  $\mathcal{T}^+$  having a  $w_i$ -separator. For  $T_i \in \mathcal{T}_b^+$ , we use  $s(T_i)$  to denote its  $w_i$ -separator (if there are multiple we fix one arbitrarily) and  $C(T_i)$  to denote the vertex sets forming the connected components of  $G[T_i \setminus \{s(T_i)\}]$ . We say a vertex  $v \in V$  or a vertex set  $V' \subseteq V$  is  $\mathcal{T}$ -assigned if  $v \in V(\mathcal{T})$  or  $V' \subseteq V(\mathcal{T})$ , respectively. That is, the set of  $\mathcal{T}$ -assigned vertices is  $V(\mathcal{T})$  and  $V(\mathcal{Q})$  is the set of *not*  $\mathcal{T}$ -assigned vertices. We say  $\mathcal{T}$  is *pack-satisfied* if  $|\mathcal{T}| = k$ , each  $T_j \in \mathcal{T}$  is connected,  $w(T_j) \in [\frac{1}{3}w_j, \max\{r, 3\}w_j]$ , and the vertex sets in  $\mathcal{T}$  are pairwise disjoint.

We define the *transfer-graph*  $H = (\mathcal{V}_H, E_H)$  as  $\mathcal{V}_H := (\bigcup_{T \in \mathcal{T}_b^+} C(T)) \cup \mathcal{T}_a^+ \cup \mathcal{T}^- \cup \mathcal{Q}$  and  $E_H := \{(V_1, V_2) \in \binom{\mathcal{V}_H}{2} \mid N_G(V_1) \cap V_2 \neq \emptyset\}$ .

#### Algorithm DoubleBoundedGL

1. Apply Theorem 12 with  $\alpha = \frac{1}{3}$  on  $G$  to obtain a connected packing  $\mathcal{T} = \{T_1, \dots, T_k\}$  with  $w(T_i) \geq \frac{1}{3}w_i$  for every  $i \in [k]$ .
2. While  $\mathcal{Q} \neq \emptyset$ :
  - a. Find a minimal path in  $H$  from  $\mathcal{Q}$  to  $\mathcal{T}^-$ . Let this path be  $P_Q^{T_i}$  where  $Q \in \mathcal{Q}$  and  $T_i \in \mathcal{T}^-$ .  
*// Note that all vertices in  $P_Q^{T_i}$  except the start and end vertex are in  $\mathcal{T}_a^+ \cup \bigcup_{T \in \mathcal{T}_b^+} C(T)$  by minimality of the path.*
  - b. Execute the **TransferVertices** routine given below, which augments vertices through the path  $P_Q^{T_i}$  such that  $\mathcal{T}$  stays pack-satisfied, and either  $|\mathcal{T}^*|$  increases, or  $|\mathcal{T}^*|$  remains the same and the number of  $\mathcal{T}$ -assigned vertices increases.

We need some more notations for describing the **TransferVertices** routine. For  $\mathcal{V}'_H \subseteq \mathcal{V}_H$  and  $\mathcal{T}' \subseteq \mathcal{T}$  we define  $\mathcal{T}'(\mathcal{V}'_H)$  as the set  $\{T_i \in \mathcal{T}' \mid V(T_i) \cap V(\mathcal{V}'_H) \neq \emptyset\}$ . For  $H' \subseteq H$  we define  $\mathcal{T}'(H') := \mathcal{T}'(\mathcal{V}_H(H'))$ , and  $V(H') := V(\mathcal{V}_H(H'))$ . With  $|P_Q^{T_i}|$  we denote the length of the path  $P_Q^{T_i}$ , i.e. the number of edges in  $P_Q^{T_i}$ . We define  $P_Q^\ell$  as the vertex with distance  $\ell$  to  $Q$  in  $P_Q^{T_i}$ , where  $P_Q^0 = Q$ , and define  $T(P_Q^\ell)$  for  $\ell \in [|P_Q^{T_i}|]$  as the function which returns  $T_j$  with  $P_Q^\ell \subseteq T_j$ . For  $\mathcal{T}' \subseteq \mathcal{T}$  we define  $I(\mathcal{T}') := \{i \mid T_i \in \mathcal{T}'\}$ .

The **TransferVertices** routine transfers vertices through the path  $P_Q^{T_i}$ . Our input is a pack-satisfied  $\mathcal{T}$  and a  $P_Q^{T_i}$  path according to Step 2b in algorithm **DoubleBoundedGL**. By the minimality of the path  $P_Q^{T_i}$ , it is clear that  $V_H(P_Q^{T_i}) \setminus \{Q, T_i\} \subseteq \mathcal{T}_a^+ \cup \bigcup_{T \in \mathcal{T}_b^+} C(T)$ . That is, except for the destination  $T_i$  we run only through vertex sets from  $\mathcal{T}^+$  in  $\mathcal{T}(P_Q^{T_i})$ . Roughly, our goal is to transfer vertices of  $V(P_Q^{T_i} - T_i)$  to  $T_i$ , thereby changing the division of the vertex sets  $\mathcal{T}(P_Q^{T_i})$  and preserving the vertex sets in  $\mathcal{T}^*$ .

We often need to do a *truncate* operation on sets  $T_j$  with  $w(T_j) > \max\{r, 3\}w_j$ . We mean by *truncate*  $T_j$  that we remove vertices from  $T_j$  until  $w_j \leq w(T_j) \leq \max\{r, 3\}w_j$  such that  $T_j$  remains connected. This can be done by removing a non-separator vertex from  $T_j$  until the weight drops below  $\max\{r, 3\}w(T_j)$ . Note that any connected graph has at least one non-separator vertex. Since  $w_{\max} \leq w_j$  we know that the weight does not decrease below  $w_j$  during the last deletion.

**Algorithm TransferVertices**

1. Initialize  $X := Q$  and let  $u = \min(I(\mathcal{T}^-))$ .
2. For  $\ell = 1$  to  $|P_Q^{\mathcal{T}^i}|$  do:
  - a. Let  $T_j = T(P_Q^\ell)$ .
  - b. If  $w(X) \geq w_u$ : set  $T_u = X$ . Truncate  $T_u$  if necessary and terminate the algorithm.
  - c. If  $w(X \cup T_j) \leq \max\{r, 3\}w_j$ : update  $T_j$  to  $X \cup T_j$  and terminate the algorithm.
  - d. If  $T_j \notin \mathcal{T}^*$ : Set  $T'_j = T_j \cup X$ ,  $T_j = T_u$  and  $T_u = T'_j$ . Truncate  $T_j$  and  $T_u$  if necessary and terminate the algorithm.
  - e. If  $T_j \in \mathcal{T}_a^+$ : divide  $T_j \cup X$  into connected vertex sets  $V_1, V_2$  with  $w(V_1), w(V_2) \geq w_j$  using the construction given by Lemma 11. Set  $T_j = V_1$  and  $T_u = V_2$ . Truncate  $T_j$  and  $T_u$  if necessary and terminate the algorithm.
  - f. We know  $T_j \in \mathcal{T}_b^+ \cap \mathcal{T}^*$ . Set  $X = X \cup P_Q^\ell$  and remove  $P_Q^\ell$  from  $T_j$ .

---

**References**

---

- 1 Curtis A Barefoot, Roger Entringer, and Henda Swart. Vulnerability in graphs a comparative survey. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 1:13–22, 1998.
- 2 Ronald Becker, Isabella Lari, Mario Lucertini, and Bruno Simeone. Max-min partitioning of grid graphs into connected components. *Networks: An International Journal*, 32(2):115–125, 1998.
- 3 Ronald Becker, Isabella Lari, Mario Lucertini, and Bruno Simeone. A polynomial-time algorithm for max-min partitioning of ladders. *Theory of Computing Systems*, 34(4):353–374, 2001.
- 4 Aydın Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. Recent advances in graph partitioning. In *Algorithm Engineering*, pages 117–158. Springer, 2016.
- 5 Paolo M Camerini, Giulia Galbiati, and Francesco Maffioli. On the complexity of finding multi-constrained spanning trees. *Discrete Applied Mathematics*, 5(1):39–50, 1983.
- 6 Katrin Casel, Tobias Friedrich, Davis Issac, Aikaterini Niklanovits, and Ziena Zeif. Balanced crown decomposition for connectivity constraints. *arXiv preprint arXiv:2011.04528*, 2020.
- 7 L. Sunil Chandran, Yun Kuen Cheung, and Davis Issac. Spanning tree congestion and computation of generalized györi-lovász partition. In *45th International Colloquium on Automata, Languages, and Programming*, volume 107 of *LIPICs*, pages 32:1–32:14, 2018.
- 8 Frédéric Chataigner, Liliane Benning Salgado, and Yoshiko Wakabayashi. Approximation and inapproximability results on balanced connected partitions of graphs. *Discrete Mathematics and Theoretical Computer Science*, 9(1), 2007. URL: <http://dmcs.episciences.org/384>.
- 9 Guangting Chen, Yong Chen, Zhi-Zhong Chen, Guohui Lin, Tian Liu, and An Zhang. Approximation algorithms for the maximally balanced connected graph tripartition problem. *Journal of Combinatorial Optimization*, pages 1–21, 2020.
- 10 Jiangzhuo Chen, Robert D Kleinberg, László Lovász, Rajmohan Rajaraman, Ravi Sundaram, and Adrian Vetta. (almost) tight bounds and existence theorems for single-commodity confluent flows. *Journal of the ACM (JACM)*, 54(4):16, 2007.
- 11 Yong Chen, Zhi-Zhong Chen, Guohui Lin, Yao Xu, and An Zhang. Approximation algorithms for maximally balanced connected graph partition. In *International Conference on Combinatorial Optimization and Applications*, pages 130–141. Springer, 2019.
- 12 Janka Chlebíková. Approximating the maximally balanced connected partition problem in graphs. *Information Processing Letters*, 60(5):225–230, 1996.
- 13 An-Chiang Chu, Bang Ye Wu, and Kun-Mao Chao. A linear-time algorithm for finding an edge-partition with max-min ratio at most two. *Discrete Applied Mathematics*, 161(7-8):932–943, 2013.
- 14 Maria Chudnovsky and Paul Seymour. Claw-free graphs. I. orientable prismatic graphs. *Journal of Combinatorial Theory, Series B*, 97(6):867–903, 2007.

- 15 Maria Chudnovsky and Paul Seymour. Claw-free graphs. II. non-orientable prismatic graphs. *Journal of Combinatorial Theory, Series B*, 98(2):249–290, 2008.
- 16 Maria Chudnovsky and Paul Seymour. Claw-free graphs. III. circular interval graphs. *Journal of Combinatorial Theory, Series B*, 98(4):812–834, 2008.
- 17 Maria Chudnovsky and Paul Seymour. Claw-free graphs. IV. decomposition theorem. *Journal of Combinatorial Theory, Series B*, 98(5):839–938, 2008.
- 18 Maria Chudnovsky and Paul Seymour. Claw-free graphs. V. global structure. *Journal of Combinatorial Theory, Series B*, 98(6):1373–1410, 2008.
- 19 Maria Chudnovsky and Paul Seymour. Claw-free graphs VI. colouring. *Journal of Combinatorial Theory, Series B*, 100(6):560–572, 2010.
- 20 Maria Chudnovsky and Paul Seymour. Claw-free graphs. VII. quasi-line graphs. *Journal of Combinatorial Theory, Series B*, 102(6):1267–1294, 2012.
- 21 Maria Chudnovsky and Paul D Seymour. The structure of claw-free graphs. *Surveys in combinatorics*, 327:153–171, 2005.
- 22 Phillip EC Compeau, Pavel A Pevzner, and Glenn Tesler. How to apply de bruijn graphs to genome assembly. *Nature biotechnology*, 29(11):987–991, 2011.
- 23 Marek Cygan, Geevarghese Philip, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. Dominating set is fixed parameter tractable in claw-free graphs. *Theoretical Computer Science*, 412(50):6982–7000, 2011.
- 24 Yuri Faenza, Gianpaolo Oriolo, and Gautier Stauffer. Solving the weighted stable set problem in claw-free graphs via decomposition. *Journal of the ACM (JACM)*, 61(4):1–41, 2014.
- 25 Greg N. Frederickson. Optimal algorithms for tree partitioning. In *Proceedings of the Second Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms*, pages 168–177. ACM/SIAM, 1991. URL: <http://dl.acm.org/citation.cfm?id=127787.127822>.
- 26 E Gyori. On division of graphs to connected subgraphs, combinatorics. In *Colloquia Mathematica Societatis Janos Bolyai, 1976*, 1976.
- 27 Danny Hermelin, Matthias Mnich, and Erik Jan van Leeuwen. Parameterized complexity of induced graph matching on claw-free graphs. *Algorithmica*, 70(3):513–560, 2014.
- 28 Alexander Hoyer. *On the Independent Spanning Tree Conjectures and Related Problems*. PhD thesis, Georgia Institute of Technology, 2019.
- 29 Davis Issac. *On some covering, partition and connectivity problems in graphs*. PhD thesis, Saarländische Universitäts-und Landesbibliothek, 2019.
- 30 Takehiro Ito, Xiao Zhou, and Takao Nishizeki. Partitioning a graph of bounded tree-width to connected subgraphs of almost uniform size. *Journal of discrete algorithms*, 4(1):142–154, 2006.
- 31 Sukhamay Kundu and Jayadev Misra. A linear tree partitioning algorithm. *SIAM Journal on Computing*, 6(1):151–154, 1977.
- 32 László Lovász. A homology theory for spanning trees of a graph. *Acta Mathematica Academiae Scientiarum Hungarica*, 30(3-4):241–251, 1977.
- 33 Christian Löwenstein, Dieter Rautenbach, and Friedrich Regen. On spanning tree congestion. *Discrete mathematics*, 309(13):4653–4655, 2009.
- 34 Mario Lucertini, Yehoshua Perl, and Bruno Simeone. Most uniform path partitioning and its use in image processing. *Discrete Applied Mathematics*, 42(2-3):227–256, 1993.
- 35 Rolf H Möhring, Heiko Schilling, Birk Schütz, Dorothea Wagner, and Thomas Willhalm. Partitioning graphs to speedup dijkstra’s algorithm. *Journal of Experimental Algorithmics (JEA)*, 11:2–8, 2007.
- 36 Yehoshua Perl and Stephen R Schach. Max-min tree partitioning. *Journal of the ACM (JACM)*, 28(1):5–15, 1981.
- 37 Hitoshi Suzuki, Naomi Takahashi, and Takao Nishizeki. A linear algorithm for bipartition of biconnected graphs. *Information Processing Letters*, 33(5):227–231, 1990.

## 27:14 Connected $k$ -Part. of $k$ -Conn. & $c$ -Claw-Free Graphs

- 38 Koichi Wada and Kimio Kawaguchi. Efficient algorithms for tripartitioning triconnected graphs and 3-edge-connected graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 132–143. Springer, 1993.
- 39 Xing Zhou, Huaimin Wang, Bo Ding, Tianjiang Hu, and Suning Shang. Balanced connected task allocations for multi-robot systems: An exact flow-based integer program and an approximate tree-based genetic algorithm. *Expert Systems with Applications*, 116:10–20, 2019.

# Better Pseudodistributions and Derandomization for Space-Bounded Computation

William M. Hoza   

Simons Institute for the Theory of Computing, Berkeley, CA, USA

---

## Abstract

---

Three decades ago, Nisan constructed an explicit pseudorandom generator (PRG) that fools width- $n$  length- $n$  read-once branching programs (ROBPs) with error  $\varepsilon$  and seed length  $O(\log^2 n + \log n \cdot \log(1/\varepsilon))$  [19]. Nisan’s generator remains the best explicit PRG known for this important model of computation. However, a recent line of work starting with Braverman, Cohen, and Garg [6, 8, 10, 22] has shown how to construct *weighted* pseudorandom generators (WPRGs, aka pseudorandom pseudodistribution generators) with better seed lengths than Nisan’s generator when the error parameter  $\varepsilon$  is small.

In this work, we present an explicit WPRG for width- $n$  length- $n$  ROBPs with seed length  $O(\log^2 n + \log(1/\varepsilon))$ . Our seed length eliminates  $\log \log$  factors from prior constructions, and our generator completes this line of research in the sense that further improvements would require beating Nisan’s generator in the standard constant-error regime. Our technique is a variation of a recently-discovered reduction that converts moderate-error PRGs into low-error WPRGs [10, 22]. Our version of the reduction uses averaging samplers.

We also point out that as a consequence of the recent work on WPRGs, any randomized space- $S$  decision algorithm can be simulated deterministically in space  $O(S^{3/2}/\sqrt{\log S})$ . This is a slight improvement over Saks and Zhou’s celebrated  $O(S^{3/2})$  bound [23]. For this application, our improved WPRG is not necessary.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Pseudorandomness and derandomization; Theory of computation  $\rightarrow$  Complexity classes

**Keywords and phrases** Weighted pseudorandom generator, pseudorandom pseudodistribution, read-once branching program, derandomization, space complexity

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.28

**Category** RANDOM

**Funding** This paper is based on research conducted while the author was a graduate student at the University of Texas at Austin, supported by the NSF GRFP under Grant DGE-1610403 and by a Harrington Fellowship from UT Austin.

**Acknowledgements** I thank David Zuckerman for helpful comments on a draft of this paper. I thank Alicia Torres Hoza for suggesting ways to cut down on footnotes.

## 1 Introduction

### 1.1 Derandomization

Randomization is a versatile technique in algorithm design. However, random bits are not always available. Therefore, we would like to deterministically simulate randomized algorithms as efficiently as possible. In this paper, we focus on space efficiency. After fixing its input, the output of a small-space algorithm as a function of its random bits can be computed by a *read-once branching program* (ROBP).

► **Definition 1.1** (ROBP). *A width- $w$  length- $n$  ROBP is a directed graph consisting of  $n + 1$  layers of vertices  $V_0, \dots, V_n$  with  $w$  vertices in each layer. For each  $i \in [n]$ , each vertex in  $V_{i-1}$  has two outgoing edges labeled 0 and 1 leading to  $V_i$ . On input  $x \in \{0, 1\}^n$ , the program*



© William M. Hoza;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 28; pp. 28:1–28:23



Leibniz International Proceedings in Informatics

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

starts at a designated start vertex  $v_{\text{start}} \in V_0$ , then reads the bits  $x_1, \dots, x_n$  in order and traverses the corresponding edges. The program accepts or rejects depending on whether the final vertex in this path is a designated accept vertex  $v_{\text{acc}} \in V_n$ . In this way, the program computes a function  $f: \{0, 1\}^n \rightarrow \{0, 1\}$ .

Arguably, the most important case is  $w = n$ , which captures  $(\log n)$ -space randomized algorithms that always halt. To derandomize such an algorithm, we would like to estimate the expectation of the corresponding ROBP on a uniform random input.

## 1.2 Pseudorandom Generators

The traditional approach to derandomization is to design a *pseudorandom generator* (PRG).

► **Definition 1.2** (PRG). Let  $\mathcal{F}$  be a class of functions  $f: \{0, 1\}^n \rightarrow \{0, 1\}$ . An  $\varepsilon$ -PRG for  $\mathcal{F}$  is a function  $G: \{0, 1\}^r \rightarrow \{0, 1\}^n$  such that for every  $f \in \mathcal{F}$ ,

$$\left| \mathbb{E}_{x \in \{0, 1\}^r} [f(G(x))] - \mathbb{E}_{x \in \{0, 1\}^n} [f(x)] \right| \leq \varepsilon.$$

Here  $r$  is the seed length of  $G$ .

By the probabilistic method, there exists a (nonexplicit) PRG for width- $n$  length- $n$  ROBPs with seed length  $O(\log(n/\varepsilon))$ . A corresponding explicit<sup>1</sup> construction would imply a complete derandomization of space-bounded computation ( $\mathbf{L} = \mathbf{BPL}$ ), because we could deterministically estimate the expectation of a given ROBP  $f$  by computing  $2^{-r} \cdot \sum_{x \in \{0, 1\}^r} f(G(x))$ . Babai, Nisan, and Szegedy designed the first explicit PRG for width- $n$  length- $n$  ROBPs [4], with seed length

$$2^{O(\sqrt{\log n})} \cdot \log(1/\varepsilon).$$

In a subsequent breakthrough [19], Nisan designed a PRG with a much better seed length of

$$O(\log^2 n + \log n \cdot \log(1/\varepsilon)).$$

## 1.3 Weighted PRGs

In the decades since Nisan's work [19], despite intense effort, the problem of designing PRGs for width- $n$  length- $n$  ROBPs has stubbornly resisted further attacks. Nisan's PRG [19] remains the best explicit PRG known for this model. However, PRGs are not the only possible approach to derandomization. Braverman, Cohen, and Garg recently introduced an intriguing generalization of PRGs called *weighted pseudorandom generators* (WPRGs), aka *pseudorandom pseudodistribution generators* [6].

► **Definition 1.3** (WPRG). Let  $\mathcal{F}$  be a class of functions  $f: \{0, 1\}^n \rightarrow \{0, 1\}$ . An  $\varepsilon$ -WPRG for  $\mathcal{F}$  is a pair of functions  $(G, \rho)$ , where  $G: \{0, 1\}^r \rightarrow \{0, 1\}^n$  and  $\rho: \{0, 1\}^r \rightarrow \mathbb{R}$ , such that for every  $f \in \mathcal{F}$ ,

$$\left| \mathbb{E}_{x \in \{0, 1\}^r} [\rho(x) \cdot f(G(x))] - \mathbb{E}_{x \in \{0, 1\}^n} [f(x)] \right| \leq \varepsilon.$$

Here  $r$  is the seed length of  $(G, \rho)$ . If  $\rho$  maps  $\{0, 1\}^r \rightarrow [-K, K]$ , we say the WPRG is  $K$ -bounded.

<sup>1</sup> We say that a function  $G: \{0, 1\}^r \rightarrow \{0, 1\}^n$  is *explicit* if it can be computed in space  $O(r)$ . More precisely, we are considering a family of functions indexed by one or more parameters (e.g.,  $n$  and  $\varepsilon$ ). The algorithm for computing  $G$  is given both the parameters and the input to  $G$ .



A standard (“unweighted”) PRG is the case  $\rho(x) \equiv 1$ . Just like an unweighted PRG, a WPRG for ROBPs can be used to estimate the expectation of a given ROBP  $f$ , because we can compute  $2^{-r} \cdot \sum_{x \in \{0,1\}^r} \rho(x) \cdot f(G(x))$ . As long as  $r$  is small and  $G$  and  $\rho$  are both efficiently computable, this is still an efficient derandomization. Thus, optimal WPRGs for ROBPs would immediately imply  $\mathbf{L} = \mathbf{BPL}$ . Furthermore, WPRGs imply *hitting set generators* (HSGs).

► **Definition 1.4 (HSG).** *Let  $\mathcal{F}$  be a class of functions  $f: \{0,1\}^n \rightarrow \{0,1\}$ . An  $\varepsilon$ -HSG for  $\mathcal{F}$  is a function  $G: \{0,1\}^r \rightarrow \{0,1\}^n$  such that for every  $f \in \mathcal{F}$ ,*

$$\mathbb{E}_{x \in \{0,1\}^r} [f(G(x))] \geq \varepsilon \implies \exists x \in \{0,1\}^r, f(G(x)) = 1.$$

If  $(G, \rho)$  is an  $\varepsilon$ -WPRG for  $\mathcal{F}$ , then  $G$  is an  $\varepsilon'$ -HSG for  $\mathcal{F}$  for any  $\varepsilon' > \varepsilon$  [6]. HSGs have been studied since the 80s [2], but prior to Braverman, Cohen, and Garg’s work [6], no explicit HSG for width- $n$  length- $n$  ROBPs was known that was any better than Nisan’s PRG (except when  $\varepsilon$  is *extremely* small; see Table 1). For these reasons, it was exciting when Braverman, Cohen, and Garg presented an explicit WPRG that fools width- $n$  length- $n$  ROBPs [6] with seed length

$$\tilde{O}(\log^2 n + \log(1/\varepsilon)),$$

which is better than Nisan’s PRG’s seed length when  $\varepsilon \ll 1/\text{poly}(n)$ .

Admittedly, Braverman, Cohen, and Garg’s result [6] did not yet imply an improved derandomization of space-bounded computation, but still, their innovative and complex work provides valuable insights. The additional flexibility in the definition of a WPRG means that WPRGs can be easier to construct compared to unweighted PRGs. In fact, in one setting (unbounded-width permutation ROBPs with a single accept vertex), Pyne and Vadhan recently showed that there is an explicit WPRG [22] with a seed length that is *provably impossible* to attain by unweighted PRGs [12], a testament to the power of the WPRG approach to derandomization.

Subsequent to Braverman, Cohen, and Garg’s work [6], Chattopadhyay and Liao gave a simpler WPRG construction [8] that fools width- $n$  length- $n$  ROBPs with the improved seed length

$$\tilde{O}(\log^2 n) + O(\log(1/\varepsilon)). \tag{1}$$

Very recently, Cohen, Doron, Renard, Sberlo, and Ta-Shma [10] and Pyne and Vadhan [22] independently obtained an even simpler WPRG that fools width- $n$  length- $n$  ROBPs with seed length

$$O(\log^2 n) + \tilde{O}(\log(1/\varepsilon)). \tag{2}$$

(These last two constructions and analyses are nearly identical [10, 22].)

## 1.4 Main Result: An Improved WPRG

In this work, we present another WPRG for ROBPs with a better seed length.

► **Theorem 1.5.** *For any  $w, n \in \mathbb{N}$  and  $\varepsilon > 0$ , there is an explicit  $\varepsilon$ -WPRG for width- $w$  length- $n$  ROBPs with seed length  $O(\log(wn) \log n + \log(1/\varepsilon))$ . Furthermore, the WPRG is  $\text{poly}(1/\varepsilon)$ -bounded.*

■ **Table 1** Known PRGs, WPRGs, and HSGs for width- $n$  length- $n$  ROBPs. As a reminder, PRG  $\implies$  WPRG  $\implies$  HSG.

Seed length	Type of generator	Reference
$\tilde{O}(\sqrt{n}) + O(\log(1/\varepsilon))$	HSG	[2] <sup>2</sup>
$2^{O(\sqrt{\log n})} \cdot \log(1/\varepsilon)$	PRG	[4]
$O(\log^2 n + \log(1/\varepsilon) \cdot \log n)$	PRG	[19]
$\tilde{O}(\log^2 n + \log(1/\varepsilon))$	WPRG	[6]
$O(\log^2 n + \log(1/\varepsilon))$	HSG	[14]
$\tilde{O}(\log^2 n) + O(\log(1/\varepsilon))$	WPRG	[8]
$O(\log^2 n) + \tilde{O}(\log(1/\varepsilon))$	WPRG	[10, 22]
$O(\log^2 n + \log(1/\varepsilon))$	WPRG	This work
$O(\log n + \log(1/\varepsilon))$	PRG	Optimal (non-explicit)

When  $w = n$ , our WPRG has seed length  $O(\log^2 n + \log(1/\varepsilon))$ , giving the “best of both worlds” compared to Equations (1) and (2). Our WPRG is the first to achieve seed length  $O(\log^2 n)$  with error  $n^{-\log n}$ . Furthermore, our WPRG represents the completion of the research project of designing WPRGs for width- $n$  length- $n$  ROBPs while focusing on the seed length’s dependence on  $\varepsilon$  [6, 8, 10, 22]. After all, even an HSG must have seed length at least  $\Omega(\log(1/\varepsilon))$ , so obtaining a better WPRG for width- $n$  length- $n$  ROBPs requires beating Nisan’s generator in the traditional, challenging constant-error regime. (That being said, see Section 5.)

Our WPRG generalizes some other recent work on the small- $\varepsilon$  regime. Hoza and Zuckerman constructed an explicit  $\varepsilon$ -HSG for width- $n$  length- $n$  ROBPs with seed length  $O(\log^2 n + \log(1/\varepsilon))$  [14], which follows also from our WPRG. Meanwhile, Cheng and Hoza gave a deterministic algorithm for estimating  $\mathbb{E}[f] \pm \varepsilon$  in space  $O(\log^2 n + \log(1/\varepsilon))$  given query access to a constant-width ROBP  $f$  [9]; Theorem 1.5 immediately implies such an algorithm for the more general case of polynomial-width ROBPs.

## 1.5 Derandomization that Beats the Saks-Zhou Bound

Next we turn to the general problem of derandomizing space- $S$  decision algorithms, whether by PRGs, WPRGs, HSGs, or any other method. Early work [24, 16, 5] showed that these algorithms can be simulated deterministically in space  $O(S^2)$  (in fact these early papers show how to simulate more powerful models). Saks and Zhou gave an improved simulation that runs in space  $O(S^{3/2})$  [23], which has remained unbeaten for decades. We point out that as a consequence of the recent progress on WPRGs, it is now possible to slightly improve the bound.

► **Theorem 1.6.** *For any function  $S(N) \geq \log N$ , we have*

$$\mathbf{BSPACE}(S) \subseteq \mathbf{DSPACE}\left(\frac{S^{3/2}}{\sqrt{\log S}}\right).$$

<sup>2</sup> For any  $w \in \mathbb{N}$ , Ajtai, Komlos, and Szemerédi designed an explicit  $(1/w)$ -HSG for width- $w$  length- $n$  ROBPs where  $n = O(\log^2 w / \log \log w)$  with optimal seed length  $O(\log w)$  [2]. Turning things around, for any  $n \in \mathbb{N}$  and  $\varepsilon > 0$ , we can let  $w = 2\sqrt{n \log n} / \varepsilon$  and get an explicit  $\varepsilon$ -HSG for width- $w$  length- $n$  ROBPs (hence also for width- $n$  length- $n$  ROBPs) with seed length  $O(\sqrt{n \log n} + \log(1/\varepsilon))$ .

(We use  $N$  to denote the input length, reserving  $n$  to denote the length of an ROBP. Recall that  $\mathbf{BPSPACE}(S)$  is the class of languages that can be decided by randomized algorithms that run in space  $O(S)$  and always halt.<sup>3</sup>) Admittedly,  $O(S^{3/2}/\sqrt{\log S})$  is barely any better than Saks and Zhou’s  $O(S^{3/2})$  bound [23]. However, we hope that Theorem 1.6 might break a “psychological barrier” by demonstrating that the Saks-Zhou algorithm [23] has room for improvement.

Our improved WPRG is not necessary for proving Theorem 1.6. Instead, Theorem 1.6 follows by combining several prior works [23, 3, 17, 8, 10, 22].

## 1.6 Overview of Proofs

### 1.6.1 Overview of our Improved WPRG

The proof of Theorem 1.5 is similar to the recent WPRG constructions by Cohen et al. and Pyne and Vadhan [10, 22]. Say we would like to fool some width- $n$  length- $n$  ROBP  $f$  with low error  $\varepsilon \ll 1/\text{poly}(n)$ . The starting point is a PRG  $G$  that fools ROBPs with moderate error  $1/\text{poly}(n)$ . Building on work by Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan [1], Cohen et al. and Pyne and Vadhan [10, 22] showed a bound of the form

$$\left| \mathbb{E}[f] - \sum_{i=1}^K \sigma_i \cdot \mathbb{E}[f(A_i)] \right| \leq \varepsilon, \quad (3)$$

where  $K = \text{poly}(1/\varepsilon)$ , each  $\sigma_i = \pm 1$ , and each random variable  $A_i$  is a concatenation of  $O\left(\frac{\log(1/\varepsilon)}{\log n}\right)$  truncations of independent samples from  $G$ . From here, we could immediately obtain an  $\varepsilon$ -WPRG by taking  $G$  to be Nisan’s generator [19], but such a WPRG would have seed length  $\Omega(\log(1/\varepsilon) \cdot \log n)$  due to the cost of sampling  $A_i$  via independent seeds to Nisan’s generator. To get a better seed length, we would like to use *correlated* seeds to Nisan’s generator.

The approach of Cohen et al. and Pyne and Vadhan [10, 22] is to use the Impagliazzo-Nisan-Wigderson (INW) PRG [15] to generate a pseudorandom sequence of seeds to Nisan’s generator. Because the INW generator is non-optimal, this approach leads to the seed length  $O(\log^2 n + \log(1/\varepsilon) \log \log_n(1/\varepsilon))$ .

Our approach is based on a simple observation. The proof of Equation (3) does not actually require that  $G$  fool *all* width- $n$  length- $n$  ROBPs. Indeed, Equation (3) holds under the weaker assumption that  $G$  fools all subprograms of *the specific ROBP*  $f$  that we are analyzing.

To exploit this observation, we apply a trick that uses an “averaging sampler”  $\text{Samp}$ . We start with a PRG  $G_0$  for width- $n$  length- $n$  ROBPs with moderate error  $1/\text{poly}(n)$  and seed length  $O(\log^2 n)$ , such as Nisan’s generator [19]. Our WPRG selects a string  $x$  of length  $O(\log^2 n + \log(1/\varepsilon))$  uniformly at random. The sampler condition implies that for any ROBP  $f$ , with high probability over  $x$ , the PRG  $G(y) \stackrel{\text{def}}{=} G_0(\text{Samp}(x, y))$  fools all subprograms of  $f$  with error  $1/\text{poly}(n)$  and optimal seed length  $O(\log n)$ . Our WPRG now applies Equation (3) to  $G$  rather than  $G_0$ . Because  $G$  has such a short seed length, sampling  $A_i$  only costs us  $O(\log(1/\varepsilon))$  truly random bits now, which we can afford. (Similar tricks have been used previously in space-bounded derandomization [20, 3, 14].)

<sup>3</sup> In the older literature, the notation “ $\mathbf{BPSPACE}(S)$ ” refers to a different model where the algorithm is not required to always halt. The class that we study in this paper is sometimes denoted “ $\mathbf{BP}_{\text{H}}\text{SPACE}(S)$ ” in older papers.

In general, our reduction converts any PRG for width- $w$  length- $n$  ROBPs with error  $1/\text{poly}(wn)$  and seed length  $r$  into a WPRG for width- $w$  length- $n$  ROBPs with any desired error  $\varepsilon$  and seed length  $O(r + \log(wn/\varepsilon))$ . Our reduction is incomparable with the prior reduction by Cohen, Doron, Renard, Sberlo, and Ta-Shma and Pyne and Vadhan [10, 22], because we get a better seed length, but we require the initial PRG to have error  $1/\text{poly}(wn)$ , whereas the prior reduction merely requires the initial PRG to have error  $1/\text{poly}(n)$ . (This is shown by Cohen et al. [10], who give a slightly tighter analysis compared to Pyne and Vadhan [22].)

We also take this opportunity to give a slightly different perspective on the proof of Equation (3), the basis of both our reduction and the earlier reduction [10, 22]. The original proof of Equation (3) is based on “preconditioned Richardson iteration,” a method for improving the accuracy of an approximate matrix inverse [1, 10, 22]. Cohen et al. pointed out that the proof has a resemblance to the notion of *local consistency errors* introduced by Cheng and Hoza [9]. We show how Equation (3) can be understood in terms of local consistency without bringing any matrices into the picture. As we explain in Appendix B, this is not a substantially different proof, but rather a different way of thinking about the same proof. We hope that this alternative perspective might yield new insights in the future.

### 1.6.2 Overview of our Improved Derandomization

The proof of Theorem 1.6 (simulating randomized space  $S$  in deterministic space  $o(S^{3/2})$ ) builds on Saks and Zhou’s algorithm [23]. To derandomize space- $(\log w)$  algorithms, Saks and Zhou rely heavily on Nisan’s PRG for width- $w$  length- $n$  ROBPs. Crucially, Saks and Zhou set  $n$  to be *much smaller* than  $w$ . To fool such programs with error  $\varepsilon$ , Nisan’s PRG has seed length  $O(\log(wn/\varepsilon) \log n)$ , so by choosing  $n = 2^{O(\sqrt{\log w})}$  and  $\varepsilon = 1/\text{poly}(w)$ , the seed length of Nisan’s PRG is only  $O(\log^{3/2} w)$ . The crux of Saks and Zhou’s work [23] is a clever method of reusing a seed of this PRG many times to derandomize a  $(\log w)$ -space algorithm even though it might use up to  $w$  random bits.

Saks and Zhou’s work therefore provides extra motivation for studying width- $w$  length- $n$  ROBPs when  $n \ll w$ . These programs correspond to algorithms that only use a small amount of randomness. In this “low-randomness” regime, PRGs have long been known that are slightly better than Nisan’s PRG. Most famously, Nisan and Zuckerman designed a PRG for the case  $n = \text{polylog } w$  with error  $2^{-\log^{0.99} w}$  and optimal seed length  $O(\log w)$  [21]. Later, Armoni designed a PRG that interpolates between Nisan’s PRG [19] and the Nisan-Zuckerman PRG [21], suitable for the regime  $\text{polylog } w \ll n \ll w$  [3]. Using extractors that were not available to Armoni at the time of his work [3], Armoni’s PRG can be implemented [17] to have seed length

$$O\left(\frac{\log(wn/\varepsilon) \log n}{\max\{1, \log \log w - \log \log(n/\varepsilon)\}}\right).$$

For  $n \ll w$  and  $\varepsilon = 1/\text{poly}(n)$ , this is better than Nisan’s PRG by a factor of  $\Theta(\log \log w)$ .

Furthermore, although Saks and Zhou [23] relied on the specific structure of Nisan’s PRG [19], Armoni showed how to modify the Saks-Zhou algorithm to use any generic PRG for ROBPs [3]. It is therefore natural to try to improve the Saks-Zhou theorem by replacing Nisan’s PRG with Armoni’s, and indeed, it has been suggested that Theorem 1.6 follows already from Armoni’s work.<sup>4</sup>

<sup>4</sup> I have heard a speaker make this claim during an oral presentation, but the speaker clarified that they

However, it seems that Theorem 1.6 does *not* follow directly from Armoni’s work. The trouble is the error parameter. For the Saks-Zhou method to work, it seems to be necessary that the PRG has error  $1/\text{poly}(w)$  rather than  $1/\text{poly}(n)$ . When  $\varepsilon = 1/\text{poly}(w)$ , Armoni’s PRG is no better than Nisan’s PRG, so we get no improvement. Armoni himself understood this issue and did not claim to beat the Saks-Zhou bound in the general case. Instead, he showed how to use his PRG to get an improved derandomization of *low-randomness* algorithms [3].

Today, however, we have new tools for fooling ROBPs with low error. In particular, we can use the recent error reduction procedure due to Cohen et al. and Pyne and Vadhan [10, 22]. Cohen et al. show how to convert a PRG for width- $w$  length- $n$  ROBPs with error  $1/\text{poly}(n)$  and seed length  $r$  into a WPRG for width- $w$  length- $n$  ROBPs with any desired error  $\varepsilon$  and seed length  $r + \tilde{O}(\log(w/\varepsilon))$  [10]. Applying this reduction to Armoni’s PRG with  $n = 2^{\sqrt{\log w \cdot \log \log w}}$  (slightly larger than the choice in Saks and Zhou’s original work [23]), we obtain a WPRG for width- $w$  length- $n$  ROBPs with error  $1/\text{poly}(w)$  and seed length

$$O\left(\frac{\log^{3/2} w}{\sqrt{\log \log w}}\right) + \tilde{O}(\log w) = O\left(\frac{\log^{3/2} w}{\sqrt{\log \log w}}\right).$$

Meanwhile, Chattopadhyay and Liao showed [8] that WPRGs can be used in place of PRGs in Saks and Zhou’s algorithm, provided the WPRG is  $\text{poly}(w)$ -bounded. The WPRG from Cohen et al.’s reduction [10] is indeed  $\text{poly}(1/\varepsilon)$ -bounded, completing the proof of Theorem 1.6. The more detailed proof is in Appendix A.

## 1.7 WPRGs vs. HSGs

We remark that the proof of Theorem 1.6 sheds light on the relative strengths of HSGs and WPRGs. Cheng and Hoza recently showed that optimal HSGs would imply  $\mathbf{L} = \mathbf{BPL}$  [9], which might cause one to question whether WPRGs have value above and beyond the value of HSGs. Chattopadhyay and Liao addressed this concern by showing that WPRGs could hypothetically be used in the Saks-Zhou algorithm to prove a new and improved derandomization of space-bounded computation [8], whereas it is still not known how to use HSGs in the Saks-Zhou algorithm. Theorem 1.6 makes the hypothetical possibility envisioned by Chattopadhyay and Liao a reality<sup>5</sup> and thereby demonstrates the strength of the WPRG approach to derandomization.

## 2 Preliminaries

### 2.1 Pseudodistributions

For most of our analysis, we will work with *pseudodistributions* rather than the WPRG formalism. For our purposes, a pseudodistribution is a generalization of a probability distribution in which probabilities are replaced with “pseudoprobabilities,” which are arbitrary real numbers that do not necessarily sum to one.

---

were not familiar with a careful proof and were merely communicating what someone else had said. I am also aware of an instance in which this claim was made in typeset lecture notes, but the claim was removed after a revision.

<sup>5</sup> To be clear, we only achieve derandomization in space  $O(S^{3/2}/\sqrt{\log S})$ , whereas Chattopadhyay and Liao proposed a route toward the much better bound  $O(S^{4/3})$  [8], developing an earlier proposal by Braverman, Cohen, and Garg [6].

► **Definition 2.1** (Pseudodistribution). A pseudodistribution over  $\{0, 1\}^n$  is a formal real linear combination of  $n$ -bit strings,<sup>6</sup> i.e., a sum of the form

$$A = \sum_{i=1}^R a_i \cdot x^{(i)},$$

where  $a_i \in \mathbb{R}$  and  $x^{(i)} \in \{0, 1\}^n$ . A probability distribution over  $\{0, 1\}^n$  is the special case that  $a_i \geq 0$  and  $\sum_{i=1}^R a_i = 1$ . We define  $U_n$  to be the uniform distribution over  $\{0, 1\}^n$ , i.e.,  $U_n = \sum_{x \in \{0, 1\}^n} 2^{-n} \cdot x$ . We often identify a function  $f$  on  $\{0, 1\}^n$  with the induced probability distribution  $f(U_n)$ . We define the pseudoexpectation of a function  $f: \{0, 1\}^n \rightarrow \mathbb{R}$  under the pseudodistribution  $A$  by

$$\tilde{\mathbb{E}}[f(A)] = \sum_{i=1}^R a_i \cdot f(x^{(i)}).$$

We say that  $A$  fools  $f$  with error  $\varepsilon$  if  $|\mathbb{E}[f] - \tilde{\mathbb{E}}[f(A)]| \leq \varepsilon$ .

► **Definition 2.2** (Operations on Pseudodistributions). Linear combinations of pseudodistributions over  $\{0, 1\}^n$  are defined in the natural way. The tensor product of two pseudodistributions is given by

$$\left( \sum_{i=1}^R a_i \cdot x^{(i)} \right) \otimes \left( \sum_{j=1}^{R'} b_j \cdot y^{(j)} \right) = \sum_{i=1}^R \sum_{j=1}^{R'} a_i b_j \cdot (x^{(i)} \circ y^{(j)}),$$

where  $\circ$  denotes concatenation. Thus if  $A$  is a pseudodistribution over  $\{0, 1\}^n$  and  $B$  is a pseudodistribution over  $\{0, 1\}^{n'}$ , then  $A \otimes B$  is a pseudodistribution over  $\{0, 1\}^{n+n'}$ .

The following facts are easy to verify.

► **Proposition 2.3.** Let  $A$  and  $B$  be pseudodistributions over  $\{0, 1\}^n$ , let  $c \in \mathbb{R}$ , and let  $f: \{0, 1\}^n \rightarrow \mathbb{R}$ . Then  $\tilde{\mathbb{E}}[f(A + cB)] = \tilde{\mathbb{E}}[f(A)] + c \cdot \tilde{\mathbb{E}}[f(B)]$ .

► **Proposition 2.4.** For  $b \in \{0, 1\}$ , let  $n_b \in \mathbb{N}$ , let  $A_b$  be a pseudodistribution over  $\{0, 1\}^{n_b}$ , and let  $f_b: \{0, 1\}^{n_b} \rightarrow \mathbb{R}$ . Let  $f(x, y) = f_0(x) \cdot f_1(y)$ . Then

$$\tilde{\mathbb{E}}[f(A_0 \otimes A_1)] = \tilde{\mathbb{E}}[f_0(A_0)] \cdot \tilde{\mathbb{E}}[f_1(A_1)].$$

## 2.2 Weighted PRGs

As discussed in Section 1, a WPRG is a pair  $(G, \rho)$ , where  $G: \{0, 1\}^r \rightarrow \{0, 1\}^n$  and  $\rho: \{0, 1\}^r \rightarrow \mathbb{R}$ . Each WPRG has a corresponding pseudodistribution, just as a PRG has a corresponding distribution.

► **Definition 2.5** (Pseudodistribution Sampled by a WPRG). If  $(G, \rho)$  is a WPRG with seed length  $r$ , the pseudodistribution sampled by  $(G, \rho)$  is  $A = \sum_{x \in \{0, 1\}^r} 2^{-r} \cdot \rho(x) \cdot G(x)$ . Note that  $(G, \rho)$  is an  $\varepsilon$ -WPRG for  $f$  if and only if  $A$  fools  $f$  with error  $\varepsilon$ .

<sup>6</sup> Equivalently,  $A$  is a vector in the  $n$ -fold tensor product space  $\mathbb{R}^2 \otimes \dots \otimes \mathbb{R}^2 \cong \mathbb{R}^{2^n}$ . The reader might find it helpful to make an analogy with quantum computing; recall that a pure state of an  $n$ -qubit system is a vector in the  $n$ -fold tensor product space  $\mathbb{C}^2 \otimes \dots \otimes \mathbb{C}^2 \cong \mathbb{C}^{2^n}$ . We could even have used ket notation for pseudodistributions:  $A = \sum_{i=1}^R a_i \cdot |x^{(i)}\rangle$ .

WPRGs can be combined in the same ways as pseudodistributions. Consideration of these operations will help us verify the seed length, boundedness, and efficiency of our WPRG.

► **Definition 2.6** (Operations on WPRGs). *Suppose we have two WPRGs  $(G_0, \rho_0)$  and  $(G_1, \rho_1)$ , where  $G_b: \{0, 1\}^{r_b} \rightarrow \{0, 1\}^{n_b}$  and  $\rho_b: \{0, 1\}^{r_b} \rightarrow \mathbb{R}$ . We define the tensor product  $(G_0, \rho_0) \otimes (G_1, \rho_1)$  to be a WPRG  $(G, \rho)$  with seed length  $r_0 + r_1$  given by*

$$G(x, y) = G_0(x) \circ G_1(y) \qquad \rho(x, y) = \rho_0(x) \cdot \rho_1(y).$$

If  $n_0 = n_1$ , we define the sum  $(G_0, \rho_0) + (G_1, \rho_1)$  to be a WPRG  $(G, \rho)$  with seed length  $r + 1$  given by

$$G(x, b) = G_b(x) \qquad \rho(x, b) = 2 \cdot \rho_b(x).$$

(There is a factor of 2 because in the definition of WPRGs, we look at an expectation over seeds rather than a sum.) For a WPRG  $(G, \rho)$  and a real number  $c$ , we define  $c \cdot (G, \rho) = (G, \rho')$ , where  $\rho'(x) = c \cdot \rho(x)$ . Under these definitions, if  $(G_b, \rho_b)$  samples from the pseudodistribution  $A_b$  over  $\{0, 1\}^{n_b}$ , then  $(G_0, \rho_0) \otimes (G_1, \rho_1)$  samples from  $A_0 \otimes A_1$ , and if  $n_0 = n_1$ , then  $(G_0, \rho_0) + c \cdot (G_1, \rho_1)$  samples from  $A_0 + cA_1$ . Furthermore, if  $(G_b, \rho_b)$  is  $K_b$ -bounded, then  $(G_0, \rho_0) \otimes (G_1, \rho_1)$  is  $(K_0K_1)$ -bounded; if  $(G_0, \rho_0)$  and  $(G_1, \rho_1)$  are both  $K$ -bounded, then  $(G_0, \rho_0) + (G_1, \rho_1)$  is  $(2K)$ -bounded; if  $(G, \rho)$  is  $K$ -bounded, then  $c \cdot (G, \rho)$  is  $(cK)$ -bounded.

### 2.3 Applying Pseudodistributions to ROBPs

Let  $f$  be an ROBP with layers  $V_0, \dots, V_n$ . Let  $u \in V_i$  and  $v \in V_j$ . When  $j \geq i$ , we define the *subprogram*  $f_{u \rightarrow v}: \{0, 1\}^{j-i} \rightarrow \{0, 1\}$  to be the length- $(j-i)$  ROBP obtained from  $f$  by setting  $u$  to be the start vertex and  $v$  to be the accept vertex. For convenience, we extend  $f_{u \rightarrow v}$  to a function  $f_{u \rightarrow v}: \{0, 1\}^{\geq j-i} \rightarrow \{0, 1\}$  that ignores all but the first  $j-i$  bits of its input.

If  $A$  is a pseudodistribution over  $\{0, 1\}^d$  with  $i + d \geq j$ , we define  $A[u \rightarrow v]$  to be the pseudoprobability of reaching  $v$  from  $u$  using  $A$ , i.e.,  $A[u \rightarrow v] = \mathbb{E}[f_{u \rightarrow v}(A)]$ . We extend the definition by defining  $A[u \rightarrow v] = 0$  when  $i > j$ .

### 2.4 Local Consistency

As mentioned in Section 1.6.1, we will present a WPRG analysis based on the notion of *local consistency* introduced by Cheng and Hoza [9]. The idea behind local consistency is that we measure the quality of a pseudodistribution by using it to estimate  $\mathbb{E}[f_{u \rightarrow v}]$  in two different ways and comparing the results.

► **Definition 2.7** (Local Consistency Error). *Let  $f$  be an ROBP with layers  $V_0, \dots, V_n$ . Let  $u \in V_i$  and  $v \in V_j$  with  $i < j$ , and let  $A$  be a pseudodistribution over  $\{0, 1\}^d$  with  $i + d \geq j$ . The local consistency error  $\text{LCErr}_{u \rightarrow v}(A)$  is defined by*

$$\text{LCErr}_{u \rightarrow v}(A) = \left( \sum_{t \in V_{j-1}} A[u \rightarrow t] \cdot U_1[t \rightarrow v] \right) - A[u \rightarrow v].$$

We extend the definition by setting  $\text{LCErr}_{u \rightarrow v}(A) = 0$  when  $j \leq i$ . We say that  $A$  is  $\alpha$ -locally consistent with respect to  $f$  if for every  $u, v$  we have  $|\text{LCErr}_{u \rightarrow v}(A)| \leq \alpha$ .

## 28:10 Better Pseudodistributions and Derandomization for Space-Bounded Computation

Note that  $U_n$  is 0-locally consistent. As we explain in Appendix B, local consistency is closely connected to approximating the inverse of the random walk Laplacian matrix of  $f$ . Cheng and Hoza's work [9] shows that local consistency and fooling are equivalent, up to some loss in the error parameter [9]. We repeat the argument here for clarity.

► **Lemma 2.8.** *Let  $A$  be a pseudodistribution over  $\{0, 1\}^n$  and let  $f$  be a width- $w$  length- $n$  ROBPP.*

1. *If  $A$  fools every subprogram  $f_{u \rightarrow v}$  of  $f$  with error  $\alpha$ , then  $A$  is  $(2w\alpha)$ -locally consistent with respect to  $f$ .*
2. *If  $A$  is  $\varepsilon$ -locally consistent with respect to  $f$ , then  $A$  fools every subprogram  $f_{u \rightarrow v}$  of  $f$  with error  $wn\varepsilon$ .*

**Proof.** First, suppose  $A$  fools every subprogram  $f_{u \rightarrow v}$  with error  $\alpha$ . Then if  $u \in V_i$  and  $v \in V_j$  with  $i < j$ , we have

$$\begin{aligned} |\text{LCErr}_{u \rightarrow v}(A)| &= \left| A[u \rightarrow v] - \sum_{t \in V_{j-1}} A[u \rightarrow t] \cdot U_1[t \rightarrow v] \right| \\ &\leq |A[u \rightarrow v] - U_n[u \rightarrow v]| + \left| U_n[u \rightarrow v] - \sum_{t \in V_{j-1}} A[u \rightarrow t] \cdot U_1[t \rightarrow v] \right| \\ &\leq \alpha + \sum_{t \in V_{j-1}} |U_n[u \rightarrow t] - A[u \rightarrow t]| \cdot U_1[t \rightarrow v] \\ &\leq (w+1)\alpha \leq 2w\alpha. \end{aligned}$$

Conversely, suppose  $A$  is  $\varepsilon$ -locally consistent with respect to  $f$ . Then for any  $u \in V_i$  and any  $j > i$ ,

$$\begin{aligned} \sum_{v \in V_j} |A[u \rightarrow v] - U_n[u \rightarrow v]| &\leq \sum_{v \in V_j} \left( \left| \sum_{t \in V_{j-1}} A[u \rightarrow t] U_1[t \rightarrow v] - U_n[u \rightarrow v] \right| + \varepsilon \right) \\ &\leq w\varepsilon + \sum_{v \in V_j} \sum_{t \in V_{j-1}} |A[u \rightarrow t] - U_n[u \rightarrow t]| \cdot U_1[t \rightarrow v] \\ &= w\varepsilon + \sum_{t \in V_{j-1}} |A[u \rightarrow t] - U_n[u \rightarrow t]| \cdot \sum_{v \in V_j} U_1[t \rightarrow v] \\ &= w\varepsilon + \sum_{t \in V_{j-1}} |A[u \rightarrow t] - U_n[u \rightarrow t]|. \end{aligned}$$

By induction on  $j - i$ , it follows that

$$\sum_{v \in V_j} |A[u \rightarrow v] - U_n[u \rightarrow v]| \leq wn\varepsilon,$$

and hence  $A$  fools every subprogram with error  $wn\varepsilon$ . ◀

We remark that there is a version of Lemma 2.8 that eliminates both factors of  $w$ . To obtain such bounds, one can consider the *sum* over all  $v \in V_j$  of each type of error  $u \rightarrow v$ . We have no need of this more refined analysis, so we omit the details.



### 3 Amplifying Local Consistency

Let  $G$  be a given pseudodistribution over  $\{0, 1\}^n$ . (Ultimately we will take  $G$  to be a probability distribution, but this stage of the construction makes sense in the more general setting of pseudodistributions.) We will show how to combine multiple samples from  $G$  to *improve its local consistency*. Throughout this section, fix a length- $n$  ROBP  $f$  with layers  $V = V_0 \cup \dots \cup V_n$ , and for convenience, define  $V_i = \emptyset$  when  $i > n$ .

#### 3.1 Construction

For each  $d \leq n$ , define  $G_d$  to be the pseudodistribution obtained by drawing a sample from  $G$  and truncating to the first  $d$  bits. That is, if  $G = \sum_{i=1}^R a_i \cdot x^{(i)}$ , then

$$G_d = \sum_{i=1}^R a_i \cdot x_{1\dots d}^{(i)}. \quad (4)$$

For  $d \in [n]$ , define a pseudodistribution  $\Delta_d$  over  $\{0, 1\}^d$  by

$$\Delta_d = G_{d-1} \otimes U_1 - G_d.$$

The definition of  $\Delta_d$  should remind the reader of local consistency errors. (See Lemma 3.2.) Now we define a “multi-hop” generalization of  $\Delta_d$ . For  $d \in [n]$  and  $m \in [d]$ , define a pseudodistribution  $\Delta_d^{(m)}$  over  $\{0, 1\}^d$  by

$$\Delta_d^{(m)} = \sum_{d_1 + \dots + d_m = d} \Delta_{d_1} \otimes \dots \otimes \Delta_{d_m},$$

where the sum is over all  $m$ -tuples of positive integers  $(d_1, \dots, d_m)$  that sum to  $d$ . Next, for each  $m \geq 1$ , define a pseudodistribution  $T^{(m)}$  over  $\{0, 1\}^n$  by

$$T^{(m)} = \sum_{d=m}^n \Delta_d^{(m)} \otimes G_{n-d},$$

and finally, for each  $m \geq 0$ , define a pseudodistribution  $G^{(m)}$  over  $\{0, 1\}^n$  by

$$G^{(m)} = G + \sum_{i=1}^m T^{(i)} = G + \sum_{i=1}^m \sum_{d=m}^n \Delta_d^{(i)} \otimes G_{n-d}.$$

We will show that as  $m$  gets large,  $G^{(m)}$  becomes increasingly locally consistent.

#### 3.2 Analysis

For  $m \geq 1$ , define a “multi-hop” generalization of local consistency errors by

$$\text{LCErr}_{u \rightarrow v}^{(m)}(G) = \sum_{u=u_0, u_1, \dots, u_m=v} \prod_{j=1}^m \text{LCErr}_{u_{j-1} \rightarrow u_j}(G),$$

where the sum is over all sequences of  $m+1$  vertices starting with  $u$  and ending with  $v$ . Our goal in this section is to prove the following *exact formula* for the local consistency errors of  $G^{(m)}$  in terms of the local consistency errors of  $G$ .

► **Lemma 3.1.** *For any two vertices  $u, v$  and any  $m \geq 0$ , we have  $\text{LCErr}_{u \rightarrow v}(G^{(m)}) = \text{LCErr}_{u \rightarrow v}^{(m+1)}(G)$ .*

Let us briefly pause to marvel at this phenomenon. In most settings, when several imperfect ingredients are combined, we expect that the errors will compound on each other, so the combination has more error than any individual ingredient. We typically consider ourselves lucky if we can prove that the errors compound mildly. The remarkable feature of Lemma 3.1 is that it involves *products* of errors, i.e., the local consistency errors of  $G$  are actually combining *in our favor!*

Toward proving Lemma 3.1, we begin by analyzing  $\Delta_d$ . It is immediate from the definitions that if  $u \in V_i$  and  $v \in V_{i+d}$ , then  $\Delta_d[u \rightarrow v] = \text{LCErr}_{u \rightarrow v}(G)$ . More generally, we have the following formula.

► **Lemma 3.2.** *Let  $d \in [n]$  and  $i, j \leq n$ . Let  $u \in V_i$  and  $v \in V_j$  and let  $A$  be a pseudodistribution over  $\{0, 1\}^{n-d}$ . Then*

$$(\Delta_d \otimes A)[u \rightarrow v] = \sum_{t \in V_{i+d}} \text{LCErr}_{u \rightarrow t}(G) \cdot A[t \rightarrow v]. \quad (5)$$

**Proof.** For the first case, suppose  $i + d \leq j$ . Then for any  $x \in \{0, 1\}^d$  and any  $y \in \{0, 1\}^{n-d}$ , we have  $f_{u \rightarrow v}(x, y) = \sum_{t \in V_{i+d}} f_{u \rightarrow t}(x) \cdot f_{t \rightarrow v}(y)$ . Therefore, for any pseudodistribution  $B$  over  $\{0, 1\}^d$  whatsoever, we have

$$(B \otimes A)[u \rightarrow v] = \sum_{t \in V_{i+d}} B[u \rightarrow t] \cdot A[t \rightarrow v].$$

Since  $\Delta_d[u \rightarrow t] = \text{LCErr}_{u \rightarrow t}(G)$ , we are done in this case.

For the second case, suppose  $i + d > j$ . Then either  $i > j$ , or else the pseudodistributions  $G_{d-1} \otimes U_1 \otimes A$  and  $G_d \otimes A$  agree on their first  $j - i$  bits.<sup>7</sup> Either way,  $(\Delta_d \otimes A)[u \rightarrow v] = 0$ . Meanwhile, for each  $t \in V_{i+d}$ , trivially  $A[t \rightarrow v] = 0$ , so both sides of Equation (5) are zero in this case. ◀

More generally, we have the following relationship between  $\Delta_d^{(m)}$  and  $\text{LCErr}^{(m)}$ .

► **Lemma 3.3.** *Let  $d \in [n]$ ,  $m \in [d]$ , and  $i, j \leq n$  and  $m \geq 0$ . Let  $u \in V_i$  and  $v \in V_j$  and let  $A$  be a pseudodistribution over  $\{0, 1\}^{n-d}$ . Then*

$$(\Delta_d^{(m)} \otimes A)[u \rightarrow v] = \sum_{t \in V_{i+d}} \text{LCErr}_{u \rightarrow t}^{(m)}(G) \cdot A[t \rightarrow v].$$

**Proof.** The base case  $m = 1$  was proven in Lemma 3.2. For the inductive step, note that

$$\Delta_d^{(m+1)} = \sum_{k=m}^{d-1} \Delta_k^{(m)} \otimes \Delta_{d-k},$$

<sup>7</sup> I.e., the induced pseudodistributions on the first  $j - i$  bits (see Equation (4)) are identical.

so

$$\begin{aligned}
& (\Delta_d^{(m+1)} \otimes A)[u \rightarrow v] \\
&= \sum_{k=m}^{d-1} (\Delta_k^{(m)} \otimes \Delta_{d-k} \otimes A)[u \rightarrow v] && \text{(Linearity)} \\
&= \sum_{k=m}^{d-1} \sum_{s \in V_{i+k}} \text{LCErr}_{u \rightarrow s}^{(m)}(G) \cdot (\Delta_{d-k} \otimes A)[s \rightarrow v] && \text{(Induction)} \\
&= \sum_{k=m}^{d-1} \sum_{s \in V_{i+k}} \sum_{t \in V_{i+d}} \text{LCErr}_{u \rightarrow s}^{(m)}(G) \cdot \text{LCErr}_{s \rightarrow t}(G) \cdot A[t \rightarrow v] && \text{(Lemma 3.2)} \\
&= \sum_{t \in V_{i+d}} \text{LCErr}_{u \rightarrow t}^{(m+1)}(G) \cdot A[t \rightarrow v]. \quad \blacktriangleleft
\end{aligned}$$

**Proof of Lemma 3.1.** For any  $u \in V_j$  and  $v \in V_k$ , by Lemma 3.3,

$$T^{(m)}[u \rightarrow v] = \sum_{d=m}^n \sum_{t \in V_{j+d}} \text{LCErr}_{u \rightarrow t}^{(m)}(G) \cdot G[t \rightarrow v] = \sum_{t \in V} \text{LCErr}_{u \rightarrow t}^{(m)}(G) \cdot G[t \rightarrow v].$$

Therefore, if  $u \in V_j$  and  $v \in V_k$  with  $j < k$ , then

$$\begin{aligned}
& \text{LCErr}_{u \rightarrow v}(T^{(m)}) \\
&= \left( \sum_{s \in V_{k-1}} T^{(m)}[u \rightarrow s] \cdot U_1[s \rightarrow v] \right) - T^{(m)}[u \rightarrow v] \\
&= \left( \sum_{s \in V_{k-1}} \sum_{t \in V} \text{LCErr}_{u \rightarrow t}^{(m)}(G) \cdot G[t \rightarrow s] \cdot U_1[s \rightarrow v] \right) - \sum_{t \in V} \text{LCErr}_{u \rightarrow t}^{(m)}(G) \cdot G[t \rightarrow v] \\
&= \sum_{t \in V} \text{LCErr}_{u \rightarrow t}^{(m)}(G) \cdot \underbrace{\left( \left( \sum_{s \in V_{k-1}} G[t \rightarrow s] \cdot U_1[s \rightarrow v] \right) - G[t \rightarrow v] \right)}_{(*)}.
\end{aligned}$$

Quantity  $(*)$  is exactly the local consistency error  $\text{LCErr}_{t \rightarrow v}(G)$ , except in one edge case: when  $t = v$ ,  $\text{LCErr}_{t \rightarrow t}(G) = 0$ , whereas  $(*) = -1$ . Therefore,

$$\begin{aligned}
\text{LCErr}_{u \rightarrow v}(T^{(m)}) &= \left( \sum_{t \in V} \text{LCErr}_{u \rightarrow t}^{(m)}(G) \cdot \text{LCErr}_{t \rightarrow v}(G) \right) - \text{LCErr}_{u \rightarrow v}^{(m)}(G) \\
&= \text{LCErr}_{u \rightarrow v}^{(m+1)}(G) - \text{LCErr}_{u \rightarrow v}^{(m)}(G).
\end{aligned}$$

Thus, we get a telescoping sum:

$$\begin{aligned}
\text{LCErr}_{u \rightarrow v}(G^{(m)}) &= \text{LCErr}_{u \rightarrow v}(G) + \sum_{i=1}^m \text{LCErr}_{u \rightarrow v}(T^{(i)}) \\
&= \text{LCErr}_{u \rightarrow v}(G) + \sum_{i=1}^m \left( \text{LCErr}_{u \rightarrow v}^{(i+1)}(G) - \text{LCErr}_{u \rightarrow v}^{(i)}(G) \right) \\
&= \text{LCErr}_{u \rightarrow v}^{(m+1)}(G).
\end{aligned}$$

(If  $j \geq k$ , then  $\text{LCErr}_{u \rightarrow v}(G^{(m)}) = \text{LCErr}_{u \rightarrow v}^{(m+1)}(G) = 0$ , so the lemma holds trivially in this case.)  $\blacktriangleleft$

The following corollary corresponds to Equation (3).

► **Corollary 3.4.** *If  $G$  fools every subprogram  $f_{u \rightarrow v}$  with error  $\alpha$ , then for every  $m \geq 0$ ,  $G^{(m)}$  fools  $f$  with error  $wn \cdot (2w^2n\alpha)^{m+1}$ .*

**Proof.** For any  $u$  and  $v$ , by Lemma 3.1,

$$\begin{aligned} |\text{LCErr}_{u \rightarrow v}(G^{(m)})| &= |\text{LCErr}_{u \rightarrow v}^{(m+1)}(G)| \\ &\leq \sum_{u=u_0, u_1, \dots, u_{m+1}=v} \prod_{j=1}^{m+1} |\text{LCErr}_{u_{j-1} \rightarrow u_j}(G)| \\ &\leq (wn)^m \cdot (2w\alpha)^{m+1} && \text{(Lemma 2.8)} \\ &\leq (2w^2n\alpha)^{m+1}. \end{aligned}$$

The corollary follows by Lemma 2.8. ◀

We reiterate that Corollary 3.4 follows already from the work of Cohen et al. and Pyne and Vadhan [10, 22], and indeed the proof we have given is not substantially different (see Appendix B). In keeping with our remark after Lemma 2.8, we also remark that there is a version of Corollary 3.4 that eliminates the factors of  $w$  by assuming that for each layer  $j$ , the *sum* of errors  $|G[u \rightarrow v] - U_n[u \rightarrow v]|$  over all  $v \in V_j$  is at most  $\alpha$ . We omit the details.

## 4 Our Improved WPRG for ROBPs

In this section, we will show how to convert a moderate-error PRG for width- $w$  length- $n$  ROBPs into a low-error WPRG for width- $w$  length- $n$  ROBPs. If the given PRG has error  $1/\text{poly}(wn)$  and seed length  $r$ , then for any  $\varepsilon > 0$ , we will obtain a WPRG with error  $\varepsilon$  and seed length  $O(r + \log(wn/\varepsilon))$ .

### 4.1 Construction

Recall the standard notion of an *averaging sampler*, which is essentially equivalent to a seeded randomness extractor [25].

► **Definition 4.1 (Sampler).** *A function  $\text{Samp}: \{0, 1\}^\ell \times \{0, 1\}^q \rightarrow \{0, 1\}^r$  is an  $(\alpha, \gamma)$ -sampler if for every function  $f: \{0, 1\}^r \rightarrow [0, 1]$ ,*

$$\Pr_{x \in \{0, 1\}^\ell} \left[ \left| \mathbb{E}[f] - 2^{-q} \sum_{y \in \{0, 1\}^q} f(\text{Samp}(x, y)) \right| \leq \alpha \right] \geq 1 - \gamma.$$

Let  $\alpha = \frac{1}{4w^3n^2}$  and let  $G: \{0, 1\}^r \rightarrow \{0, 1\}^n$  be a given  $\alpha$ -PRG for width- $w$  length- $n$  ROBPs. Define

$$m = \left\lceil \frac{\log(wn/\varepsilon)}{\log(wn)} \right\rceil \quad \text{and} \quad \gamma = \frac{\varepsilon}{2w^2n^2 \cdot ((8n)^{m+1} + 1)} = \left( \frac{\varepsilon}{wn} \right)^{O(1)},$$

and let  $\text{Samp}: \{0, 1\}^\ell \times \{0, 1\}^q \rightarrow \{0, 1\}^r$  be an  $(\alpha, \gamma)$ -sampler. For each  $x \in \{0, 1\}^\ell$ , let  $G_x$  be the distribution  $G(\text{Samp}(x, U_q))$ , and let  $G_x^{(m)}$  be the corresponding pseudodistribution with amplified local consistency as defined in Section 3.1. Our final pseudodistribution  $G'$  is  $G_x^{(m)}$  for a uniform random  $x$ , i.e.,

$$G' = \sum_{x \in \{0, 1\}^\ell} 2^{-\ell} \cdot G_x^{(m)}.$$

## 4.2 Correctness

▷ Claim 4.2. If  $f$  is a width- $w$  length- $n$  ROBP, then  $G'$  fools  $f$  with error  $\varepsilon$ .

Proof. For each pair of vertices  $u, v$ , since  $G$  is an  $\alpha$ -PRG for width- $w$  ROBPs,  $G$  fools  $f_{u \rightarrow v}$  with error  $\alpha$ . Therefore, by the sampler condition, with probability  $1 - \gamma$  over a uniform random choice of  $x$ ,  $G_x$  fools  $f_{u \rightarrow v}$  with error  $2\alpha$ . Let BAD be the set of  $x$  such that there exist vertices  $u, v$  such that  $G_x$  does *not*  $(2\alpha)$ -fool  $f_{u \rightarrow v}$ . By the union bound,

$$|\text{BAD}| \leq \gamma \cdot w^2 n^2 \cdot 2^\ell = \frac{\varepsilon}{2 \cdot ((8n)^{m+1} + 1)} \cdot 2^\ell.$$

For any  $x$ , unpacking the definitions, we see that  $G_x^{(m)}$  has the form  $\sum_{i=1}^K \pm A_i$ , where

$$K \leq (m+1) \cdot (n+1) \cdot (n+1)^m \cdot 2^m \leq (8n)^{m+1}$$

and each  $A_i$  is a tensor product of probability distributions. Therefore, for  $x \in \text{BAD}$  (indeed for all  $x$ ), we have  $|\tilde{\mathbb{E}}[f(G_x^{(m)})]| \leq (8n)^{m+1}$ . Meanwhile, for  $x \notin \text{BAD}$ , by Corollary 3.4,

$$\left| \tilde{\mathbb{E}}[f(G_x^{(m)})] - \mathbb{E}[f] \right| \leq wn \cdot (4w^2 n \alpha)^{m+1} = wn \cdot \left( \frac{1}{wn} \right)^{m+1} < \frac{\varepsilon}{2}.$$

Therefore, overall,

$$\begin{aligned} \left| \tilde{\mathbb{E}}[f(G')] - \mathbb{E}[f] \right| &= \left| \sum_{x \in \text{BAD}} 2^{-\ell} \cdot (\tilde{\mathbb{E}}[f(G_x^{(m)})] - \mathbb{E}[f]) + \sum_{x \notin \text{BAD}} 2^{-\ell} \cdot (\tilde{\mathbb{E}}[f(G_x^{(m)})] - \mathbb{E}[f]) \right| \\ &\leq \sum_{x \in \text{BAD}} 2^{-\ell} \left( \left| \tilde{\mathbb{E}}[f(G_x^{(m)})] \right| + 1 \right) + \sum_{x \notin \text{BAD}} 2^{-\ell} \cdot \left| \tilde{\mathbb{E}}[f(G_x^{(m)})] - \mathbb{E}[f] \right| \\ &\leq 2^{-\ell} \cdot |\text{BAD}| \cdot ((8n)^{m+1} + 1) + \frac{\varepsilon}{2} \\ &\leq \varepsilon. \end{aligned} \quad \triangleleft$$

## 4.3 Explicitness and Seed Length

We will instantiate **Samp** using the following explicit sampler.

► **Theorem 4.3** ([8, Appendix B]). *For every  $r \in \mathbb{N}$  and every  $\alpha, \gamma > 0$ , there exists an  $(\alpha, \gamma)$ -sampler **Samp**:  $\{0, 1\}^\ell \times \{0, 1\}^q \rightarrow \{0, 1\}^r$  with  $\ell = r + O(\log(1/\gamma) + \log(1/\alpha))$  and  $q = O(\log(1/\alpha) + \log \log(1/\gamma))$ , such that given  $r, \alpha, \gamma, x$ , and  $y$ , the value **Samp**( $x, y$ ) can be computed in space  $O(r + \log(1/\alpha) + \log(1/\gamma))$ .*

**Proof of Theorem 1.5.** Taking **Samp** to be the sampler of Theorem 4.3, we get  $\ell = O(r + \log(1/\gamma)) = O(r + \log(wn/\varepsilon))$  and  $q = O(\log(wn) + \log \log(1/\varepsilon))$ . For a fixed  $x \in \{0, 1\}^\ell$ , as mentioned in the proof of Claim 4.2,  $G_x^{(m)}$  has the form  $\sum_{i=1}^K \pm A_i$ , where

$$K \leq (8n)^{m+1} \leq \text{poly}(n/\varepsilon),$$

and each  $A_i$  is a tensor product of at most  $2m+1$  terms, each of which is either  $(G_x)_d$  for some value of  $d$  or else  $U_1$ . Using the constructions of Definition 2.6, we can sample  $G_x^{(m)}$  by a WPRG with seed length  $O(\log K + mq)$ , and we can sample  $G'$  by a WPRG with seed length

$$\ell + O(\log K + mq) = O\left(r + \log(wn/\varepsilon) \cdot \left(1 + \frac{\log \log(1/\varepsilon)}{\log(wn)}\right)\right) = O(r + \log(wn/\varepsilon)),$$

where the last equality holds without loss of generality, because either  $\varepsilon > 2^{-n}$ , in which case  $\log \log(1/\varepsilon) < \log(wn)$ , or else  $\varepsilon \leq 2^{-n}$ , in which case we can achieve seed length  $O(r + \log(wn/\varepsilon))$  by simply sampling a truly random  $n$ -bit string. Furthermore, as discussed in Definition 2.6, our WPRG is  $(2K)$ -bounded,<sup>8</sup> and we can assume without loss of generality that  $\varepsilon < 1/n$  (since otherwise  $G$  itself would be a suitable WPRG), so our WPRG is indeed  $\text{poly}(1/\varepsilon)$ -bounded.

Finally, pick  $G$  to be Nisan's generator [19]. Then

$$r = O(\log(wn/\alpha) \log n) = O(\log(wn) \log n),$$

so our WPRG has seed length  $O(\log(wn) \log n + \log(1/\varepsilon))$  as claimed. Explicitness is clear. ◀

We remark that because of the specific structure of Nisan's generator [19], the sampler is actually not necessary. Instead, we can let  $x$  be the description of the hash functions in Nisan's generator and let  $y$  be the input to the hash functions.

## 5 Directions for Further Research

As we mentioned in Section 1.4, getting a better WPRG for width- $n$  length- $n$  ROBPs requires beating Nisan's PRG in the standard constant-error regime. However, there are cases where focusing on error dependence might still be fruitful:

- Recall that Nisan and Zuckerman designed a PRG for width- $w$  length- $n$  ROBPs when  $n = \text{polylog } w$  with optimal seed length  $O(\log w)$  [21] but non-optimal error  $2^{-\log^{0.99} w}$ . There are known  $\varepsilon$ -HSGs for this setting with seed length  $O(\log w)$  even when  $\varepsilon = 1/\text{poly}(w)$  [2, 14]; can we match that seed length by a WPRG? The WPRG construction presented in this paper does not seem to work, because if  $G$  has seed length  $o(\log w)$ , then it seems to have too much error for the local consistency amplification procedure  $G^{(m)}$  to work, whereas if  $G$  has seed length  $\Omega(\log w)$ , then we cannot afford to sample multiple independent seeds.
- Currently, the best explicit PRGs for width-3 ROBPs and constant-width regular ROBPs have seed length  $\tilde{O}(\log n \cdot \log(1/\varepsilon))$  [18, 11, 7]. In a similar spirit as Pyne and Vadhan's recent work on permutation ROBPs [22], can we design WPRGs for these models with error  $1/\text{poly}(n)$  and seed length  $o(\log^2 n)$ ?

We also wonder whether there are other applications of recent WPRG constructions. For example, recall that Nisan showed  $\mathbf{BPL} \subseteq \mathbf{DTISP}(\text{poly}(n), \log^2 n)$  [20]. Can we somehow use WPRGs to simulate  $\mathbf{BPL}$  by a deterministic algorithm that simultaneously uses  $\text{poly}(n)$  time and  $o(\log^2 n)$  space?

---

## References

- 1 AmirMahdi Ahmadinejad, Jonathan Kelner, Jack Murtagh, John Peebles, Aaron Sidford, and Salil Vadhan. High-precision estimation of random walks in small space. In *Proceedings of the 61st Symposium on Foundations of Computer Science (FOCS)*, pages 1295–1306, 2020. doi:10.1109/FOCS46700.2020.00123.
- 2 Miklós Ajtai, János Komlós, and Endre Szemerédi. Deterministic simulation in logspace. In *Proceedings of the 19th Symposium on Theory of Computing (STOC)*, pages 132–140, 1987. doi:10.1145/28395.28410.

---

<sup>8</sup> The factor of 2 is because the number of terms in the sum might not be a power of two, so we might need to pad with dummy terms.

- 3 Roy Armoni. On the derandomization of space-bounded computations. In *Proceedings of the 2nd International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 47–59, 1998. doi:10.1007/3-540-49543-6\_5.
- 4 László Babai, Noam Nisan, and Mária Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *Journal of Computer and System Sciences*, 45(2):204–232, 1992. doi:10.1016/0022-0000(92)90047-M.
- 5 A. Borodin, S. Cook, and N. Pippenger. Parallel computation for well-endowed rings and space-bounded probabilistic machines. *Information and Control*, 58(1-3):113–136, 1983. doi:10.1016/S0019-9958(83)80060-6.
- 6 Mark Braverman, Gil Cohen, and Sumegha Garg. Pseudorandom pseudo-distributions with near-optimal error for read-once branching programs. *SIAM Journal on Computing*, 49(5):STOC18–242–STOC18–299, 2020. doi:10.1137/18M1197734.
- 7 Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom generators for regular branching programs. *SIAM Journal on Computing*, 43(3):973–986, 2014. doi:10.1137/120875673.
- 8 Eshan Chattopadhyay and Jyun-Jie Liao. Optimal Error Pseudodistributions for Read-Once Branching Programs. In *Proceedings of the 35th Computational Complexity Conference (CCC)*, pages 25:1–25:27, 2020. doi:10.4230/LIPIcs.CCC.2020.25.
- 9 Kuan Cheng and William M. Hoza. Hitting Sets Give Two-Sided Derandomization of Small Space. In *Proceedings of the 35th Computational Complexity Conference (CCC)*, pages 10:1–10:25, 2020. doi:10.4230/LIPIcs.CCC.2020.10.
- 10 Gil Cohen, Dean Doron, Oren Renard, Ori Sberlo, and Amnon Ta-Shma. Error Reduction For Weighted PRGs Against Read Once Branching Programs, 2021. URL: <https://eccc.weizmann.ac.il/report/2021/020/>.
- 11 Anindya De. Pseudorandomness for permutation and regular branching programs. In *Proceedings of the 26th Conference on Computational Complexity (CCC)*, pages 221–231, 2011. doi:10.1109/CCC.2011.23.
- 12 William M. Hoza, Edward Pyne, and Salil Vadhan. Pseudorandom Generators for Unbounded-Width Permutation Branching Programs. In *Proceedings of the 12th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 7:1–7:20, 2021. doi:10.4230/LIPIcs.ITCS.2021.7.
- 13 William M. Hoza and Chris Umans. Targeted pseudorandom generators, simulation advice generators, and derandomizing logspace. *SIAM Journal on Computing*, pages STOC17–281–STOC17–304, 2021. doi:10.1137/17M1145707.
- 14 William M. Hoza and David Zuckerman. Simple optimal hitting sets for small-success RL. *SIAM Journal on Computing*, 49(4):811–820, 2020. doi:10.1137/19M1268707.
- 15 Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the 26th Symposium on Theory of Computing (STOC)*, pages 356–364, 1994. doi:10.1145/195058.195190.
- 16 H. Jung. Relationships between probabilistic and deterministic tape complexity. In *Proceedings of the 10th Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 339–346, 1981. doi:10.1007/3-540-10856-4\_101.
- 17 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. Revisiting norm estimation in data streams, 2008. arXiv:0811.3648.
- 18 Raghu Meka, Omer Reingold, and Avishay Tal. Pseudorandom generators for width-3 branching programs. In *Proceedings of the 51st Symposium on Theory of Computing (STOC)*, pages 626–637, 2019. doi:10.1145/3313276.3316319.
- 19 Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992. doi:10.1007/BF01305237.
- 20 Noam Nisan.  $RL \subseteq SC$ . *Computational Complexity*, 4(1):1–11, 1994. doi:10.1007/BF01205052.

- 21 Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996. doi:10.1006/jcss.1996.0004.
- 22 Edward Pyne and Salil Vadhan. Pseudodistributions That Beat All Pseudorandom Generators, 2021. URL: <https://eccc.weizmann.ac.il/report/2021/019/>.
- 23 Michael Saks and Shiyu Zhou.  $\text{BP}_H\text{SPACE}(S) \subseteq \text{DSPACE}(S^{3/2})$ . *Journal of Computer and System Sciences*, 58(2):376–403, 1999. doi:10.1006/jcss.1998.1616.
- 24 Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4:177–192, 1970. doi:10.1016/S0022-0000(70)80006-X.
- 25 David Zuckerman. Randomness-optimal oblivious sampling. *Random Structures & Algorithms*, 11(4):345–367, 1997. doi:10.1002/(SICI)1098-2418(199712)11:4<345::AID-RSA4>3.0.CO;2-Z.

## A Derandomization Beyond Saks-Zhou

In this section, we show that randomized space- $S$  decision algorithms can be simulated deterministically in space  $O(S^{3/2}/\sqrt{\log S})$  (Theorem 1.6). As outlined in Section 1.6.2, the proof does not involve any significant new ideas, but rather amounts to combining several previous works and choosing parameters. For that reason, we will refrain from fully describing the Saks-Zhou method. Instead, we will focus on assisting readers who are already familiar with Saks and Zhou’s work [23] (but who are not necessarily comfortable with WPRGs) in verifying Theorem 1.6. Readers who are not familiar with Saks and Zhou’s work are encouraged to read Chattopadhyay and Liao’s discussion of the Saks-Zhou method in the context of WPRGs [8] or Saks and Zhou’s original paper [23].

Let  $G$  denote Nisan’s PRG [19]. Recall that Saks and Zhou [23] exploited the fact that the seed of Nisan’s PRG can be split into two parts  $(x, y)$ , where  $x = O(\log w \log n)$  and  $y = O(\log w)$ ; for a fixed ROBP  $f$ , if we pick  $x$  at random, then with high probability,  $\mathbb{E}[f] \approx 2^{-r} \cdot \sum_y f(G(x, y))$ . This method of estimating  $\mathbb{E}[f]$  has a key technical feature, which is that each time we read a bit of the input of  $f$ , we only need to be using  $O(\log w)$  bits of work space (not counting the string  $x$ , which we think of as being on a read-only random tape). This feature is beneficial, because in the context of the Saks-Zhou algorithm [23], the program  $f$  is computed recursively, so we would like to use as little space as possible while the recursive computation is taking place. (See the work of Hoza and Umans for further discussion [13].)

Armoni generalized Saks and Zhou’s methods by showing that *any* explicit PRG for ROBPs implies a method of estimating  $\mathbb{E}[f]$  with the same key feature [3]. Later, Chattopadhyay and Liao generalized further by showing that the same holds for any polynomially-bounded explicit WPRG [8]. For clarity, we repeat the argument here (in a slightly different form). It is convenient to generalize the definition of ROBPs to allow a large alphabet.

► **Definition A.1** (ROBP over a large alphabet). *A width- $w$  length- $n$  ROBP over the alphabet  $\Sigma$  is defined as in Definition 1.1, except that each vertex in  $V_{i-1}$  has  $|\Sigma|$  outgoing edges leading to  $V_i$ , labeled with the symbols in  $\Sigma$ . The program computes a function  $f: \Sigma^n \rightarrow \{0, 1\}$  in the natural way.*

► **Lemma A.2** ([8]). *Let  $n = n(w)$ ,  $K = K(w)$ ,  $r = r(w)$ ,  $a = a(w)$ , and  $\varepsilon = \varepsilon(w)$  be functions, each of which can be constructed in space  $O(r)$ . Suppose that for every  $w \in \mathbb{N}$ , there is a  $K$ -bounded  $\varepsilon$ -WPRG  $(G, \rho)$  for width- $w$  length- $n$  ROBPs over the alphabet  $\{0, 1\}^a$  with seed length  $r$  that can be computed in space  $O(r)$ . Then there is an algorithm for estimating the expectation of a given width- $w$  length- $n$  ROBP  $f$  over the alphabet  $\{0, 1\}^a$  with the following properties.*



1. The algorithm uses  $r + O(\log(Kw/\varepsilon))$  random bits from a read-only two-way<sup>9</sup> random tape, and with probability  $1 - \varepsilon/w^2$  it outputs a value that is within  $\pm 2\varepsilon$  of  $\mathbb{E}[f]$ .
2. The algorithm uses  $O(r + a + \log(Kwn/\varepsilon))$  bits of work space. Furthermore, whenever the algorithm reads a bit of the description of  $f$ , it first deletes all but  $O(a + \log(Kwn/\varepsilon))$  bits of its work space.

**Proof.** Let  $\text{Samp}: \{0, 1\}^\ell \times \{0, 1\}^a \rightarrow \{0, 1\}^r$  be the  $(\varepsilon/(2K), \varepsilon/w^2)$ -sampler of Theorem 4.3. To estimate  $\mathbb{E}[f]$ , we pick  $x \in \{0, 1\}^\ell$  uniformly at random, and then we output

$$2^{-q} \cdot \sum_{y \in \{0, 1\}^a} \rho(\text{Samp}(x, y)) \cdot f(G(\text{Samp}(x, y))).$$

To prove that this works, define  $g: \{0, 1\}^r \rightarrow [-K, K]$  by  $g(z) = \rho(z) \cdot f(G(z))$ . The sampler condition implies that with probability  $1 - \varepsilon/w^2$  over the choice of  $x$ , we have

$$\left| \mathbb{E}[g] - 2^{-q} \sum_{y \in \{0, 1\}^a} g(\text{Samp}(x, y)) \right| \leq \varepsilon.$$

Meanwhile, the WPRG condition implies that  $|\mathbb{E}[g] - \mathbb{E}[f]| \leq \varepsilon$ . Thus, with probability  $1 - \varepsilon/w^2$ , our algorithm outputs  $\mathbb{E}[f] \pm 2\varepsilon$ .

Now let us analyze the efficiency of the algorithm. The number of random bits we use is clearly  $\ell = r + O(\log(Kw/\varepsilon))$ . The total space used is  $O(r)$  bits to compute  $G$  and  $\rho$ , plus  $O(r \log(Kw/\varepsilon))$  bits to compute  $\text{Samp}$ , plus  $O(\log(wn))$  bits to keep track of our simulation of  $f$ , plus  $O(q) = O(\log(K/\varepsilon) + \log \log w)$  bits for summing over all  $y$ , which is a total of  $O(r + a + \log(Kwn/\varepsilon))$  bits. Prior to reading a bit of the description of  $f$ , we only need to be storing the  $O(\log(wn))$  bits that keep track of our simulation of  $f$ , plus the  $O(q) = O(\log(K/\varepsilon) + \log \log w)$  bits for summing over all  $y$ , plus a single  $a$ -bit symbol of the string  $G(\text{Samp}(x, y))$  (namely, the single symbol that we are currently feeding into our simulation of  $f$ ), which is a total of  $O(a + \log(wnK/\varepsilon))$  bits. ◀

Having established Lemma A.2, we can now compute the space complexity of the derandomization obtained by plugging any efficient WPRG into the Saks-Zhou framework.

▶ **Theorem A.3** ([23, 8]). *Let  $n = n(w)$ ,  $K = K(w)$ , and  $r = r(w)$  be monotone increasing functions, each of which can be constructed in space  $O(r)$ , with  $n \leq w$ . Define  $\varepsilon = w^{-8}$  and  $a = \lceil 4 \log w \rceil$ . Suppose that for every  $w \in \mathbb{N}$ , there exists a  $K$ -bounded  $\varepsilon$ -WPRG for width- $(w + 1)$  length- $n$  ROBPs over the alphabet  $\{0, 1\}^a$  with seed length  $r$  that can be computed in space  $O(r)$ . Then*

$$\text{BPL} \subseteq \bigcup_{c \in \mathbb{N}} \text{DSPACE} \left( r(N^c) + \frac{\log(N \cdot K(N^c)) \cdot \log N}{\log(n(N^c))} \right),$$

where  $N$  denotes the input length.

**Proof outline.** Suppose we are interested in computing the  $n$ -th power of a given substochastic matrix  $M \in \mathbb{R}^{w \times w}$ , where each entry has  $a$  bits of precision. We can easily construct a width- $(w + 1)$  length- $n$  ROBP  $f$  over the alphabet  $\{0, 1\}^a$  such that for each  $i, j \in [w]$ , if we

<sup>9</sup> I.e., the algorithm is allowed to go back and re-read random bits as many times as it likes, unlike the standard model of randomized space-bounded computation in which the random tape must be read a single time from left to right.

let  $u_i$  be the  $i$ -th vertex in the first layer of  $f$  and we let  $v_j$  be the  $j$ -th vertex in the final layer of  $f$ , then  $\mathbb{E}[f_{u_i \rightarrow v_j}] = (M^n)_{i,j}$ . Using Lemma A.2, we can compute each such value  $\mathbb{E}[f_{u_i \rightarrow v_j}]$  to within  $\pm 2\varepsilon$  with failure probability  $\varepsilon/w^2$ . In this way, we compute a matrix  $P \in \mathbb{R}^{w \times w}$  such that  $\|P - M^n\|_{\max} \leq 2\varepsilon$ . We can reuse the same random bits for each entry of the matrix, so our algorithm uses  $r + O(\log(Kw/\varepsilon))$  random bits from a read-only two-way random tape and succeeds with probability  $1 - \varepsilon$ . Furthermore, this algorithm uses  $O(r + a + \log(Kwn/\varepsilon))$  bits of work space, and whenever it reads a bit of the description of  $M$ , it first deletes all but  $O(a + \log(Kwn/\varepsilon))$  bits of its workspace.

Now, consider some randomized log-space algorithm that we wish to derandomize. There is a constant  $c$  such that the acceptance probability of the **BPL** algorithm on an input of length  $N$  is an entry in  $M^w$ , where  $w = N^c$  and  $M \in \{0, \frac{1}{2}, 1\}^{w \times w}$  is an easily-computable stochastic matrix. We have discussed a randomized algorithm for approximating  $M^n$ . The technique of Saks and Zhou [23] implies [8] an algorithm for computing  $M^w$ . As a reminder, the approach is to repeatedly take approximate  $n$ -th powers, reusing the same random bits each time and randomly rounding each entry of the matrix to  $a$  bits of precision after each iteration to resolve dependency issues. The number of iterations is  $\frac{\log w}{\log n}$ . The algorithm can be implemented to have failure probability  $O(w^3 \cdot (2^a \varepsilon + 2^{-a}))$  and approximation error  $O(w^2 2^{-a})$ , using

$$O\left(r + \log(Kw/\varepsilon) + a \cdot \frac{\log w}{\log n}\right)$$

random bits and

$$O\left(r + (a + \log(Kwn/\varepsilon)) \cdot \frac{\log w}{\log n}\right)$$

bits of space [8, Lemma 43]. By our choices  $\varepsilon = w^{-8}$  and  $a = \lceil 4 \log w \rceil$ , the failure probability is  $O(1/w)$ , the approximation error is  $O(1/w^2)$ , the number of random bits is  $O(r + \log(Kw) + \frac{\log^2 w}{\log n})$ , and the space complexity is  $O(r + \frac{\log(Kw) \log w}{\log n})$ . Trying all possibilities for the random tape completes the proof.  $\blacktriangleleft$

Next, we identify the WPRG family that we will plug into Theorem A.3.

► **Theorem A.4** ([3, 17, 10]). *For every  $w \in \mathbb{N}$ , there exists a  $K$ -bounded  $\varepsilon$ -WPRG for width- $(w + 1)$  length- $n$  ROBPs over the alphabet  $\{0, 1\}^{\lceil 4 \log w \rceil}$  with seed length  $r$  that can be computed in space  $O(r)$ , where*

$$\begin{aligned} n &= \exp\left(\left\lceil \sqrt{\log w \cdot \log \log w} \right\rceil\right), & \varepsilon &= w^{-8}, \\ r &\leq O\left(\frac{\log^{3/2} w}{\sqrt{\log \log w}}\right), & K &\leq \text{poly}(w). \end{aligned}$$

**Proof.** For any  $w, n, a, \alpha$ , Armoni designed an  $\alpha$ -PRG for width- $(w + 1)$  length- $n$  ROBPs over the alphabet  $\{0, 1\}^a$  [3]; with an optimization due to Kane, Nelson, and Woodruff [17], this PRG has seed length

$$r = O\left(a + \frac{\log(wn/\alpha) \log n}{\max\{1, \log \log w - \log \log(n/\alpha)\}}\right)$$

and can be computed in space  $O(r)$ . For  $n = \exp(\lceil \sqrt{\log w \cdot \log \log w} \rceil)$ ,  $\alpha = 1/\text{poly}(n)$ , and  $a = O(\log w)$ , this seed length becomes

$$r = O\left(\frac{\log^{3/2} w}{\sqrt{\log \log w}}\right).$$

Now we apply an error reduction procedure that converts this moderate-error PRG into a low-error WPRG. Specifically, we will use the reduction due to Cohen, Doron, Renard, Sberlo, and Ta-Shma [10]. Given a PRG for width- $w$  length- $n$  ROBPs over the alphabet  $\{0, 1\}^a$  with error  $1/(10n^2)$  and seed length  $r$ , they show [10, Corollary 15] how to construct a WPRG for width- $w$  length- $n$  ROBPs over the alphabet  $\{0, 1\}^a$  with any desired error  $\varepsilon$  and seed length  $r + O(\log(w/\varepsilon) \log \log_n(1/\varepsilon))$ . Furthermore, if the PRG can be computed in space  $O(r)$ , then the WPRG can be computed in space  $O(r + \log \log_n(1/\varepsilon) \cdot (\log \log(w/\varepsilon))^2)$ . Cohen et al. did not explicitly mention it, but by inspection it is easy to see that their WPRG is  $\text{poly}(1/\varepsilon)$ -bounded for the same reason that our main WPRG (Theorem 1.5) is  $\text{poly}(1/\varepsilon)$ -bounded. Since  $\varepsilon = 1/\text{poly}(w)$ , the seed length is  $r + \tilde{O}(\log w) = O(r)$ , the space complexity is  $O(r + \text{poly}(\log \log w)) = O(r)$ , and the WPRG is  $\text{poly}(w)$ -bounded. ◀

► **Corollary A.5.**  $\text{BPL} \subseteq \text{DSPACE}\left(\log^{3/2} N / \sqrt{\log \log N}\right)$ , where  $N$  denotes the input length.

**Proof.** Plugging the WPRG of Theorem A.4 into Theorem A.3, we get a space bound of

$$O\left(\frac{\log^{3/2}(N^c)}{\sqrt{\log \log(N^c)}} + \frac{\log(N \cdot N^{O(c)}) \cdot \log N}{\sqrt{\log(N^c) \cdot \log \log(N^c)}}\right),$$

which simplifies to  $O\left(\log^{3/2} N / \sqrt{\log \log N}\right)$ . ◀

Now we generalize Corollary A.5 to the case of  $\text{BSPACE}(S)$ . When  $S$  is space-constructible, the generalization is a standard padding argument. We now show that  $\text{BSPACE}(S)$  is contained in  $\text{DSPACE}(S^{3/2}/\sqrt{\log S})$  for any  $S(N) \geq \log N$ , whether space-constructible or not.

**Proof of Theorem 1.6.** Observe that the proof of Corollary A.5 extends to promise problems. In particular, for any constants  $0 \leq a < b \leq 1$ , there is a deterministic algorithm  $D_{a,b}$  such that if  $f$  is a width- $w$  length- $w$  ROBP over the binary alphabet, then

$$\mathbb{E}[f] \leq a \implies D_{a,b}(f) = 0,$$

$$\mathbb{E}[f] \geq b \implies D_{a,b}(f) = 1,$$

and  $D_{a,b}(f)$  runs in space  $O\left(\log^{3/2} w / \sqrt{\log \log w}\right)$ .

Let  $A$  be a Turing machine witnessing membership of a language in  $\text{BSPACE}(S)$ . For  $N \in \mathbb{N}$ ,  $y \in \{0, 1\}^N$ , and  $s \in \mathbb{N}$ , there exists a width- $w$  length- $w$  ROBP  $E_{y,s}$ , where  $w = O(N \cdot 2^s)$ , such that  $E_{y,s}(x) = 1$  if and only if the computation  $A(y, x)$  ever touches more than  $s$  cells of the work tape. Furthermore, for the same value of  $w$ , there exists a width- $w$  length- $w$  ROBP  $f_{y,s}$  such that if  $E_{y,s}(x) = 0$ , then  $f_{y,s}(x) = A(y, x) \in \{0, 1\}$ . Given  $y$  and  $s$ , these two ROBPs can be constructed deterministically in space  $O(s + \log N)$ .

On input  $y$ , our deterministic algorithm tries each  $s = 1, 2, 3, \dots$  until it finds the first  $s$  such that  $D_{0,0.01}(E_{y,s}) = 0$ . Then, our deterministic algorithm outputs  $D_{0,4,0.6}(f_{y,s})$ . This works, because if  $D_{0,0.01}(E_{y,s}) = 0$ , then  $\mathbb{E}[E_{y,s}] < 0.01$ , so  $\mathbb{E}[f_{y,s}]$  is within  $\pm 0.01$  of the acceptance probability of  $A(y)$ . Furthermore, our algorithm will find a suitable  $s$  satisfying  $s \leq S(N)$ , because  $\mathbb{E}[E_{y,S(N)}] = 0$ . Therefore, the space complexity of our algorithm is at most  $O(\log^{3/2} w / \sqrt{\log \log w})$ , where  $w = O(N \cdot 2^{S(N)}) = 2^{O(S(N))}$ . This space bound is  $O\left(S(N)^{3/2} / \sqrt{\log S(N)}\right)$  as desired. ◀

## B Local Consistency vs. Approximate Inverse Laplacian

Cohen et al. noted that their WPRG construction is reminiscent of local consistency errors [10]. We now briefly elaborate on the connection, for the sake of readers who are familiar with how prior work used preconditioned Richardson iteration to decrease error in space-bounded derandomization [1, 10, 22].

Prior works [1, 10, 22] looked at the transition probability matrix  $W$  associated with a width- $w$  length- $n$  ROBP  $f$ , considered as a directed graph on  $(n+1) \cdot w$  vertices. This matrix  $W$  is an  $(n+1)w \times (n+1)w$  block matrix of the form

$$W = \begin{bmatrix} 0 & M_1 & 0 & \cdots & 0 \\ 0 & 0 & M_2 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \ddots & M_n \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix},$$

where  $M_i \in \{0, \frac{1}{2}, 1\}^{w \times w}$  is the transition probability matrix for  $V_{i-1} \times V_i$ . Let  $L = I - W$  (the Laplacian matrix). Then  $L$  is invertible with inverse

$$L^{-1} = \begin{bmatrix} M_{0\dots 0} & M_{0\dots 1} & M_{0\dots 2} & \cdots & M_{0\dots n} \\ 0 & M_{1\dots 1} & M_{1\dots 2} & \cdots & M_{1\dots n} \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \ddots & M_{n-1\dots n} \\ 0 & 0 & 0 & \cdots & M_{n\dots n} \end{bmatrix},$$

where  $M_{i\dots j} = M_i \cdot M_{i+1} \cdots M_j$ , i.e.,  $M_{i\dots j}$  is the stochastic matrix containing the probabilities  $U_n[u \rightarrow v]$  for  $u \in V_i$  and  $v \in V_j$ . We are interested in obtaining an approximation  $\widehat{L}^{-1}$  to  $L$ , say

$$\widehat{L}^{-1} = \begin{bmatrix} \widehat{M}_{0\dots 0} & \widehat{M}_{0\dots 1} & \widehat{M}_{0\dots 2} & \cdots & \widehat{M}_{0\dots n} \\ 0 & \widehat{M}_{1\dots 1} & \widehat{M}_{1\dots 2} & \cdots & \widehat{M}_{1\dots n} \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \ddots & \widehat{M}_{n-1\dots n} \\ 0 & 0 & 0 & \cdots & \widehat{M}_{n\dots n} \end{bmatrix},$$

where each  $\widehat{M}_{i\dots j}$  is a matrix of estimates for the probabilities  $U_n[u \rightarrow v]$  with  $u \in V_i$  and  $v \in V_j$ . The approach taken by prior work [1, 10, 22] is to use preconditioned Richardson iteration to convert a moderate-error approximation of  $L^{-1}$  into a low-error approximation of  $L^{-1}$ .

The crucial point is that in this analysis, the approximation quality is measured by comparing  $\widehat{L}^{-1}L$  to  $I$  rather than comparing  $L^{-1}$  and  $\widehat{L}^{-1}$  directly. The error matrix  $E \stackrel{\text{def}}{=} I - \widehat{L}^{-1}L$  is given by

$$E = \begin{bmatrix} 0 & E_{0\dots 1} & E_{0\dots 2} & \cdots & E_{0\dots n} \\ 0 & 0 & E_{1\dots 2} & \cdots & E_{1\dots n} \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \ddots & E_{n-1\dots n} \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix},$$

where

$$E_{i\dots j} = \widehat{M_{i\dots j-1}}M_j - \widehat{M_{i\dots j}}.$$

Thus,  $E$  is precisely the matrix of local consistency errors. (This is also plain from one of Pyne and Vadhan's lemmas [22, Lemma 4.6].)



# On the Hardness of Average-Case $k$ -SUM

Zvika Brakerski ✉

Weizmann Institute of Science, Rehovot, Israel

Noah Stephens-Davidowitz ✉

Cornell University, Ithaca, NY, USA

Vinod Vaikuntanathan ✉

Massachusetts Institute of Technology, Cambridge, MA, USA

---

## Abstract

In this work, we show the first worst-case to average-case reduction for the classical  $k$ -SUM problem. A  $k$ -SUM instance is a collection of  $m$  integers, and the goal of the  $k$ -SUM problem is to find a subset of  $k$  integers that sums to 0. In the average-case version, the  $m$  elements are chosen uniformly at random from some interval  $[-u, u]$ .

We consider the *total* setting where  $m$  is sufficiently large (with respect to  $u$  and  $k$ ), so that we are guaranteed (with high probability) that solutions must exist. In particular,  $m = u^{\Omega(1/k)}$  suffices for totality. Much of the appeal of  $k$ -SUM, in particular connections to problems in computational geometry, extends to the total setting.

The best known algorithm in the average-case total setting is due to Wagner (following the approach of Blum-Kalai-Wasserman), and achieves a running time of  $u^{\Theta(1/\log k)}$  when  $m = u^{\Theta(1/\log k)}$ . This beats the known (conditional) lower bounds for worst-case  $k$ -SUM, raising the natural question of whether it can be improved even further. However, in this work, we show a matching *average-case lower bound*, by showing a reduction from *worst-case lattice problems*, thus introducing a new family of techniques into the field of fine-grained complexity. In particular, we show that any algorithm solving average-case  $k$ -SUM on  $m$  elements in time  $u^{o(1/\log k)}$  will give a super-polynomial improvement in the complexity of algorithms for lattice problems.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational complexity and cryptography

**Keywords and phrases**  $k$ -SUM, fine-grained complexity, average-case hardness

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.29

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2010.08821>

**Funding** *Zvika Brakerski*: Supported by the Binational Science Foundation (Grant No. 2016726), and by the European Union Horizon 2020 Research and Innovation Program via ERC Project REACT (Grant 756482) and via Project PROMETHEUS (Grant 780701).

*Noah Stephens-Davidowitz*: Supported in part by NSF Grants CNS-1350619, CNS-1414119 and CNS-1718161, Microsoft Faculty Fellowship and an MIT/IBM grant.

*Vinod Vaikuntanathan*: Supported in part by NSF Grants CNS-1350619, CNS-1414119 and CNS-1718161, Microsoft Faculty Fellowship and an MIT/IBM grant.

## 1 Introduction

The  $k$ -SUM problem is a parameterized version of the classical subset sum problem. Given a collection of  $m$  integers  $a_1, \dots, a_m$ , the  $k$ -SUM problem asks if there is some subset of cardinality  $k$  that sums to zero.<sup>1</sup> This problem (especially for  $k = 3$ , but more generally for

---

<sup>1</sup> This is the homogeneous version of  $k$ -SUM. One could also define the inhomogeneous version where the instance consists also of a target integer  $t$ , and the goal is to produce a subset of  $k$  elements that sums



arbitrary constant  $k$ ) has been influential in computational geometry, where reductions from  $k$ -SUM have been used to show the conditional hardness of a large class of problems [16, 17]. More generally it has been used in computational complexity, where it has formed the basis for several fine-grained hardness results [33, 2, 38, 28]. We refer the reader to the extensive survey of Vassilevska-Williams [37] for an exposition of this line of work. The  $k$ -SUM problem has also been extensively studied in the cryptanalysis community (see, e.g., [36, 12, 10]).

We know two very different algorithms for  $k$ -SUM: a meet-in-the-middle algorithm that achieves run-time  $O(m^{\lceil k/2 \rceil})$  [23], and dynamic programming or FFT-based algorithms that achieve run-time  $\tilde{O}(um)$  [11] where  $u$  is the largest absolute value of the integers  $a_i$  (A sequence of recent works [27, 14, 8, 25] improve the latter to  $\tilde{O}(u+m)$ ). Note that the latter algorithms outperform the former when  $u \ll m^{\lceil k/2 \rceil}$ , in what is sometimes called the *dense regime* of parameters, a point that we will come back to shortly.

In terms of hardness results for  $k$ -SUM, the work of Pătraşcu and Williams [34] shows that an algorithm that solves the problem in time  $m^{o(k)}$  for all  $m$  will give us better algorithms for SAT, in particular refuting the exponential time hypothesis (ETH). The recent work of Abboud, Bringmann, Hermelin and Shabtay [1] shows that a  $u^{1-\varepsilon}$ -time algorithm (for any constant  $\varepsilon > 0$ ) would refute the strong exponential-time hypothesis (SETH). So, we know that the two algorithms described above are essentially optimal, at least in the worst case.

The focus of this work is the natural average-case version of  $k$ -SUM where the problem instance  $a_1, \dots, a_m$  is chosen independently and uniformly at random from an interval  $[-u, u]$ . We call this the *average-case  $k$ -SUM problem*. In this setting, *deciding* whether a  $k$ -SUM solution exists is in many cases trivial. In particular, if  $\binom{m}{k} \ll u$  then a union bound argument shows that the probability of a solution existing approaches 0. We refer to this as the *sparse* regime of the problem. In contrast, if  $\binom{m}{k}$  is sufficiently larger than  $u$ , then a hashing argument guarantees the existence of many solutions, with high probability over the instance. (See Lemma 14.) As already mentioned above, we refer to this as the *dense* regime.

Notwithstanding this triviality, we notice that in the dense regime one could still consider the *search* problem of *finding* a  $k$ -SUM solution. The search problem seems to retain its hardness even in the dense setting and is the focus of our work. Since we consider the search version of the problem, we also refer to the dense regime as the *total* regime, as the associated search problem has a solution with high probability.

The average-case total problem is *not* quite as hard as the worst-case version (at least assuming SETH), since (slight variants of) Wagner’s generalized birthday algorithm [36] and the Blum-Kalai-Wasserman algorithm [12] show how to solve this problem in time  $u^{O(1/\log k)}$  (when  $m = u^{\Omega(1/\log k)}$ ). This contrasts with the  $u^{1-\varepsilon}$  lower bound of [1] in the worst case. (The BKW/Wagner algorithm was originally stated in a slightly different setting, so we restate it in Section 4.) This leaves the question of *how much easier* the average case is compared to the worst case. Given that the lower bounds from the worst-case setting are not a barrier here, it is a-priori unclear what is the best running time in this setting. Can we improve on [12, 36]?

## 1.1 Our Results

In this work we characterize the hardness of average-case  $k$ -SUM in the total regime by presenting a (conditional) lower bound that matches the  $u^{O(1/\log k)}$  upper bound described above, up to the hidden constant in the exponent.

---

to  $t$ . In the worst-case world, the two versions are equivalent.



In more detail, our main result shows that average-case  $k$ -SUM is indeed hard to solve, under the assumption that *worst-case* lattice problems are hard to approximate. We thus introduce a new family of techniques into the study of the hardness of the  $k$ -SUM problem. Concretely, this lower bound shows that a  $u^{o(1/\log k)}$ -time algorithm for average-case  $k$ -SUM (in the dense regime) implies a  $2^{o(n)}$ -time  $n^{1+\varepsilon}$ -approximation algorithm for the shortest independent vectors problem (SIVP) over an  $n$ -dimensional lattice, a lattice problem for which the best known algorithms run in time  $2^{\Omega(n)}$  [4, 3]. E.g., while Wagner’s algorithm runs in time essentially linear in  $m$  when  $m = u^{\Omega(1/\log k)}$ , we show that such behavior for  $m = u^{o(1/\log k)}$  would imply faster algorithms for SIVP. (One can think of our lower bound as ruling out linear-time algorithms for  $m = u^{o(1/\log k)}$ , or more generally ruling out  $u^{o(1/\log k)}$ -time algorithms for any  $m$ . Indeed, notice that the problem only gets easier as  $m$  becomes larger, so that lower bounds against linear-time algorithms for such large choices of  $m$  immediately implies lower bounds against  $u^{o(1/\log k)}$ -time algorithms for any  $m$ . We therefore switch freely between these two perspectives.)

It is widely believed that no faster algorithm can be found for SIVP. In particular, finding a  $2^{o(n)}$ -time algorithm for SIVP, would have major consequences in lattice-based cryptography both in theory and in practice [32, 6, 7].

We also note in the appendix that some of the connections between  $k$ -SUM and geometric problems from [16, 17] carry over to the dense setting as well. This shows an interesting (and not previously known, as far as we could find) connection between approximate short vectors in lattices, and computational geometry.

## 1.2 Our Techniques

The starting point of our reduction is the well known worst-case to average-case reductions in the lattice world, pioneered by Ajtai [5, 31, 19, 18]. These reductions show that the approximate shortest independent vectors (SIVP) problem, a standard problem in the lattice world, is at least as hard in the worst case as a certain problem called short integer solutions (SIS) on the average. The definition of lattices and the approximate shortest vector problem is not crucial for the current discussion, however we note again that the best algorithms on  $n$ -dimensional lattices that compute any poly( $n$ )-approximation to SIVP run in time  $2^{\Omega(n)}$ . (We refer the curious reader to, e.g., [31, 35], Section 2.3, and the references therein for more background on lattices and lattice problems.)

In the (one-dimensional) *average-case* SIS problem with parameters  $m, Q$  and  $\beta$ , one is given random integers  $a_1, \dots, a_m \in \mathbb{Z}_Q$  and the goal is to find a *non-zero* integer linear combination  $\mathbf{x} = (x_1, \dots, x_m) \in \mathbb{Z}^m$  such that  $\sum_{i \in [m]} a_i x_i = 0 \pmod{Q}$  and  $\mathbf{x}$  is short, namely  $\|\mathbf{x}\|_1 \leq \beta$ . Thus, this is exactly the modular subset sum problem (i.e. subset sum over the group  $\mathbb{Z}_Q$ ), except with weights larger than 1. The parameters of the problem live in the dense/total regime where such solutions are guaranteed to exist with high probability. The worst-case to average-case reductions state that an average-case SIS solver for a sufficiently large  $Q$ , namely  $Q = (\beta n)^{\Omega(n)}$ , gives us an  $\tilde{O}(\sqrt{n \log m} \cdot \beta)$ -approximate algorithm for SIVP. (We refer the reader to Theorem 10 for a more precise statement.)

Our main technical contribution is an average-case to average-case reduction from the SIS problem to the  $k$ -SUM problem. We show this by exhibiting a reduction from SIS to modular  $k$ -SUM (i.e.  $k$ -SUM over the group  $\mathbb{Z}_Q$ ), and one from modular  $k$ -SUM to  $k$ -SUM. The latter is easy to see (in the dense regime): indeed, if you have a  $k$ -subset that sums to 0, it also sums to 0 (mod  $Q$ ) for any  $Q$ . Henceforth in this discussion, when we say  $k$ -SUM, we will mean modular  $k$ -SUM.

To reduce from SIS with parameters  $m, Q, \beta$  to  $k$ -SUM on  $m$  numbers over  $\mathbb{Z}_Q$ , we start with a simple, seemingly trivial, idea. SIS and  $k$ -SUM are so similar that perhaps one could simply run the  $k$ -SUM algorithm on the SIS instance. Unfortunately, this fails. For a  $k$ -SUM solution to exist,  $m$  has to be at least roughly  $Q^{1/k} = n^{\Omega(n/k)}$ . But, this could only possibly give us an approximate SIVP algorithm that runs in time  $n^{\Omega(n/k)}$  (where we are most interested in constant  $k$ ), since the reduction from SIVP has to at least write down the  $m$  samples. This is a meaningless outcome since, as we discussed before, there are algorithms for approximate SIVP that run in time  $2^{O(n)}$ .

Fortunately, ideas from the BKW algorithm [12] for subset sum (and the closely related algorithm from [36] for  $k$ -SUM) come to our rescue. We will start with SIS modulo  $Q = q^L$  for some  $q$  and  $L$  that we will choose later. ([18] showed that worst-case to average-case reductions work for any sufficiently large  $Q$ , including  $Q = q^L$ .)

The BKW algorithm iteratively produces subsets that sum to 0 modulo  $q^i$  for  $i = 1, \dots, L$ , finally producing SIS solutions modulo  $Q$ . To begin with, observe that for a  $k$ -subset-sum to exist modulo  $q$ , it suffices that  $m \approx q^{1/k} \ll Q^{1/k}$ , potentially getting us out of the conundrum from before. In particular, we will set  $q \approx 2^{n \log k}$ ,  $L \approx \log n / \log k$ , therefore  $Q = q^L \approx n^n$  as needed. We will also set  $m \approx q^{\varepsilon / \log k} \approx 2^{\varepsilon n}$  for a large enough  $\varepsilon$  so that solutions exist (since  $m^k \gg q$ ). Furthermore, a hypothetical  $k$ -SUM algorithm mod  $q$  that performs better than BKW/Wagner – that is, runs in time  $q^{o(1/\log k)} = 2^{o(n)}$  – is potentially useful to us.

With this ray of optimism, let us assume that we can run the  $k$ -SUM algorithm many times to get several,  $m$  many, subsets  $S_j$  that sum to 0 modulo  $q$ . (We will return to, and remove, this unrealistic assumption soon.) That is,

$$b_j := \sum_{i \in S_j} a_i = 0 \pmod{q}$$

The BKW/Wagner approach would then be to use the  $(b_1, \dots, b_m)$  to generate  $(c_1, \dots, c_m)$  that are 0 (mod  $q^2$ ), and so on. Note that  $c_i$  are a linear combination of  $a_1, \dots, a_m$  with weight  $k^2$ . At the end of the iterations, we will obtain at least one linear combination of  $(a_1, \dots, a_m)$  of weight  $\beta = k^L$  that sums to 0 modulo  $q^L = Q$ , solving SIS. (We also need to make sure that this is a non-zero linear combination, which follows since the coefficients of all intermediate linear combinations are positive.)

This would finish the reduction, except that we need to remove our unrealistic assumption that we can use the  $k$ -SUM oracle to get many  $k$ -subsets of  $(a_1, \dots, a_m)$  that sum to 0. For one, the assumption is unrealistic because if we feed the  $k$ -SUM oracle with the same  $(a_1, \dots, a_m) \pmod{q}$  twice, we will likely get the same  $k$ -SUM solution. On the other hand, using a fresh random instance for every invocation of the  $k$ -SUM oracle will require  $m$  to be too large (essentially returning to the trivial idea above). A natural idea is to observe that each  $k$ -SUM solution touches a very small part of the instance. Therefore, one could hope to first receive a  $k$ -SUM  $a_{i_1} + \dots + a_{i_k}$  from the oracle, and in the next iteration, use as input  $\{a_1, \dots, a_m\} \setminus \{a_{i_1}, \dots, a_{i_k}\}$ , which is nearly as large as the original set. Unfortunately, continuing like this cannot work in general. The distributions of the successive instances that we feed to the oracle will no longer be uniform, and even worse, the oracle itself can choose which elements to remove from our set. A malicious oracle could therefore prevent us from obtaining many  $k$ -SUMs in this way, even if the oracle has high success probability on uniform input. (One can even imagine that the fastest algorithm for  $k$ -SUM would actually yield such a malicious oracle. E.g., if an algorithm has a “preference” for some types of  $k$ -SUMs over others, then running the algorithm repeatedly in this way could eventually deplete the input of such  $k$ -SUMs, causing the algorithm to fail.)

Instead, our key idea is rather simple, namely to resort to (re)randomization. Given an instance  $(a_1, \dots, a_m) \in \mathbb{Z}_q^m$ , we compute many random subset sums to generate  $(a'_1, \dots, a'_m) \in \mathbb{Z}_q^m$ . That is, we choose  $k$ -subsets  $T'_i \subseteq [m]$  and let

$$a'_i = \sum_{j \in T'_i} a_j \pmod{q}$$

Since  $q \gg m^{1/k}$ , the leftover hash lemma [24] tells us that the  $a'_i$  are (statistically close to) uniformly random mod  $q$ . Furthermore, a  $k$ -subset sum of  $(a'_1, \dots, a'_m)$  will give us a  $k^2$ -subset sum of  $(a_1, \dots, a_m)$  that sums to 0 (mod  $q$ ). To obtain a new subset sum of  $(a_1, \dots, a_m)$ , simply run this process again choosing fresh subsets  $T''_i$  to generate  $(a''_1, \dots, a''_m)$ ; and so on. Eventually, this will give us a  $\beta = k^{2L}$  weight solution to SIS, which is a quadratic factor worse than before, but good enough for us. (We are glossing over an important technical detail here, which is how we ensure that the resulting subset sums yield uniformly random independent elements in  $q\mathbb{Z}/q^2\mathbb{Z}$ .)

To finish the analysis of the reduction, observe that it calls the  $k$ -SUM oracle  $\approx mL$  times. Assuming the oracle runs in time  $q^{o(1/\log k)}$ , this gives us a  $2^{o(n)}$ -time algorithm for approximate SIVP. The approximation factor is  $\tilde{O}(\sqrt{n \log m} \cdot \beta) \approx n^3$ . (In the sequel, we achieve  $n^{1+o(1)}$  by a careful choice of parameters.)

Interestingly, our reduction re-imagines the BKW/Wagner *algorithm as a reduction* from the SIS problem to  $k$ -SUM, where the original algorithm is achieved (in retrospect) by plugging in the trivial algorithm for 2-SUM. Of course, the algorithm is much simpler than the reduction (in particular, there is no need for re-randomization) since we don't need to account for "malicious"  $k$ -SUM solvers. Our main technical contribution can therefore be viewed as making the ideas from the BKW/Wagner *algorithm* (ideas which are now ubiquitous in the study of algorithms for subset sum and lattice problems) work as a *reduction* – i.e., with an arbitrary average-case  $k$ -SUM oracle.

### 1.3 Open Problems and Future Directions

Our work introduces the powerful toolkit of lattice problems into the field of average-case fine-grained complexity, and raises several natural directions for further research.

First is the question of whether a result analogous to what we show holds in the sparse/planted regime as well. A possible theorem here would rule out an  $m^{o(k)}$ -time algorithm for  $k$ -SUM, assuming the hardness of lattice problems. To the best of our knowledge, in the sparse/planted regime it is not known whether the average-case problem is easier than the worst-case problem as in the dense regime.

Second is the question of whether we can obtain average-case hardness of  $k$ -SUM for concrete small constants  $k$ , perhaps even  $k = 3$ . Our hardness result is asymptotic in  $k$ .

Third is the question of whether we can show the average-case hardness of natural distributions over *combinatorial and computational-geometric* problems, given their connection to  $k$ -SUM. In this vein, we show a simple reduction to (perhaps not the most natural distribution on) the  $(Q, m, d)$ -plane problem in Appendix A, but we believe much more can be said. More generally, now that we have shown average-case hardness of  $k$ -SUM, it is natural to try to reduce average-case  $k$ -SUM to other natural average-case problems.

### 1.4 Other Related Works

There are now quite a few works that study average-case fine-grained hardness of problems in  $P$ . We mention a few. First, Ball, Rosen, Sabin, and Vasudevan [9] showed a reduction from SAT to an (average-case) variant of the orthogonal vectors problem. They demonstrated that sub-quadratic algorithms for their problem would refute SETH.

There is also a sequence of works on the average-case hardness of counting  $k$ -cliques. The work of Goldreich and Rothblum [20, 21] shows worst-case to average-case *self*-reductions for the problem of counting  $k$ -cliques (and other problems in  $P$ ). Boix-Adserà, Brennan, and Bresler proved the same result for  $G_{n,p}$  [13], and Hirahara and Shimizu recently showed that it is even hard to count the number of  $k$ -cliques with even a small probability of success [22]. In contrast, our reductions go from the worst-case of one problem (SIVP) to the average-case of another ( $k$ -SUM). We find it a fascinating problem to show a worst-case to average-case *self*-reduction for  $k$ -SUM.

Dalirrooyfard, Lincoln, and Vassilevska Williams [15] recently proved fine-grained average-case hardness for many different problems in  $P$  under various complexity-theoretic assumptions. In particular, they show fine-grained average-case hardness of *counting* the number of solutions of a non-standard *factored* variant of  $k$ -SUM under well-studied fine-grained hardness assumptions. In contrast, we show fine-grained average-case hardness of the standard *search*  $k$ -SUM problem under an assumption that is well-studied in the lattice community but perhaps not previously considered in the fine-grained complexity world. So, our conclusion is more natural – both because it works for a search problem rather than a counting problem and because it works with the standard notion of  $k$ -SUM rather than a factored variant – but we rely on a less-standard assumption.

For the lattice expert, we remark that if one unwraps our reduction from SIVP to SIS and then to  $k$ -SUM, we obtain a structure that is superficially similar to [30]. However, in their setting, they do not need to reuse samples and therefore do not need the re-randomization technique, which is the key new idea in this work.

More generally, there are many works that use BKW/Wagner-style techniques together with a specific solver for  $k$ -SUM or subset sum to solve various lattice problems. (See, for example, [29, 30, 26].) In contrast, we show a generic *reduction* from SIVP to average-case  $k$ -SUM that can be instantiated with *any* average-case  $k$ -SUM oracle.

## 1.5 Organization of the Paper

Section 3 describes the modular variant of  $k$ -SUM as well as the standard  $k$ -SUM (over the integers), shows their totality on average, and reductions between them. For completeness, we describe the BKW/Wagner algorithm in Section 4. We remark that while the standard descriptions of the algorithm refer to finite groups, we need one additional trick (namely, Lemma 15) to obtain the algorithm over the integers. Finally, our main result, the worst-case to average-case reduction is described in Section 5. The connection to computational geometry is provided in Appendix A.

## 2 Preliminaries

We write  $\log$  for the logarithm base two and  $\ln$  for the natural logarithm. We write  $\binom{m}{k} := \frac{m!}{(m-k)!k!}$  for the binomial coefficient.

### 2.1 Probability

We make little to no distinction between random variables and their associated distributions.

For two random variables  $X, Y$  over some set  $S$ , we write  $\Delta(X, Y) := \sum_{z \in S} |\Pr[X = z] - \Pr[Y = z]|$  for the statistical distance between  $X$  and  $Y$ . For a finite set  $S$ , we write  $U_S$  for the uniform distribution over  $S$ .

Recall that a set of functions  $\mathcal{H} \subseteq \{h : X \rightarrow Y\}$  is a *universal family of hash functions* from  $X$  to  $Y$  if for any distinct  $x, x' \in X$

$$\Pr_{h \sim \mathcal{H}} [h(x) = h(x')] \leq 1/|Y|.$$

► **Lemma 1** (Leftover hash lemma, [24]). *If  $\mathcal{H}$  is a universal family of hash functions from  $X$  to  $Y$ , then*

$$\Pr[\Delta(h(U_X), U_Y) \geq \beta] \leq \beta,$$

where the probability is over a random choice of  $h \sim \mathcal{H}$  and  $\beta := (|Y|/|X|)^{1/4}$ .

► **Lemma 2.** *For any positive integers  $Q, m$ , let  $\mathcal{H}$  be the family of hash functions from  $\{0, 1\}^m$  to  $\mathbb{Z}_Q$  given by  $h_{\mathbf{a}}(\mathbf{x}) = \langle \mathbf{a}, \mathbf{x} \rangle \bmod Q$  for all  $\mathbf{a} \in \mathbb{Z}_Q^m$ . Then,  $\mathcal{H}$  is a universal family of hash functions.*

**Proof.** Let  $\mathbf{x}, \mathbf{y} \in \{0, 1\}^m$  be distinct vectors, and suppose without loss of generality that  $x_1 = 1$  and  $y_1 = 0$ . We write  $\mathbf{a}' \in \mathbb{Z}_Q^{m-1}$  for the vector obtained by removing the first coordinate from  $\mathbf{a}$  and  $a_1$  for the first coordinate itself. Similarly, we write  $\mathbf{x}', \mathbf{y}' \in \{0, 1\}^{m-1}$  for the vectors  $\mathbf{x}, \mathbf{y}$  with their first coordinate removed. Then,

$$\begin{aligned} \Pr[\langle \mathbf{a}, \mathbf{x} \rangle = \langle \mathbf{a}, \mathbf{y} \rangle \bmod Q] &= \Pr[\langle \mathbf{a}', \mathbf{x}' \rangle + a_1 = \langle \mathbf{a}', \mathbf{y}' \rangle \bmod Q] \\ &= \Pr[a_1 = \langle \mathbf{a}', \mathbf{y}' - \mathbf{x}' \rangle \bmod Q] = 1/Q, \end{aligned}$$

where the probability is over the random choice of  $\mathbf{a} \in \mathbb{Z}_Q^m$ . The last equality follows from the fact that  $a_1 \in \mathbb{Z}_Q$  is uniformly random and independent of  $\mathbf{a}'$ . ◀

► **Corollary 3.** *For any positive integers  $Q, m$  and any subset  $X \subseteq \{0, 1\}^m$ ,*

$$\Pr_{\mathbf{a} \sim \mathbb{Z}_Q^m} [\Delta(\langle \mathbf{a}, U_X \rangle \bmod Q, U_{\mathbb{Z}_Q}) \geq \beta] \leq \beta,$$

where  $\beta := (Q/|X|)^{1/4}$ .

► **Corollary 4.** *Let  $\mathbf{a} := (a_1, \dots, a_M) \in \mathbb{Z}_Q^M$  be sampled uniformly at random, and let  $S_1, \dots, S_{M'} \subset [M]$  be sampled independently and uniformly at random with  $|S_i| = t$ . Let  $c_i := \sum_{j \in S_i} a_j \bmod Q$ . Then,  $(\mathbf{a}, \mathbf{c}) := (a_1, \dots, a_M, c_1, \dots, c_{M'})$  is within statistical distance  $\delta$  of a uniformly random element in  $\mathbb{Z}_Q^{M+M'}$ , where*

$$\delta := (M' + 1) \cdot Q^{1/4} \cdot \binom{M}{t}^{-1/4} \leq (M' + 1) \cdot \left(\frac{Qt^t}{M^t}\right)^{1/4}.$$

**Proof.** Let  $X_t := \{\mathbf{x} \in \{0, 1\}^M : \|\mathbf{x}\|_1 = t\}$ , and notice that  $|X_t| = \binom{M}{t}$ . Call  $\mathbf{a}$  good if  $\Delta(\langle \mathbf{a}, U_{X_t} \rangle \bmod Q, U_{\mathbb{Z}_Q}) \leq \beta := Q^{1/4}/|X_t|^{1/4}$ . From Corollary 3, we see that  $\mathbf{a}$  is good except with probability at most  $\beta$ .

Finally, notice that the  $c_i$  are distributed exactly as independent samples from  $\langle \mathbf{a}, U_{X_t} \rangle$ . Therefore, if  $\mathbf{a}$  is good, each of the  $c_i$  is within statistical distance  $\beta$  of an independent uniform sample. The result then follows from the union bound. ◀

## 2.2 Hitting probabilities

► **Definition 5.** For  $\mathbf{a} := (a_1, \dots, a_M) \in \mathbb{Z}_Q^M$ ,  $\mathbf{c} := (c_1, \dots, c_{M'}) \in \mathbb{Z}_Q^{M'}$ ,  $I \subset [M]$ ,  $J \subset [M']$ , and a positive integer  $t$ , the  $t$ -hitting probability of  $\mathbf{a}$ ,  $\mathbf{c}$ ,  $I$ , and  $J$  is defined as follows. For each  $j \in J$ , sample a uniformly random  $S_j \in \binom{[M]}{t}$  with  $\sum_{i \in S_j} a_i = c_j$ . (If no such  $S_j$  exists, then we define the hitting probability to be 1.) The hitting probability is then

$$p_{\mathbf{a}, \mathbf{c}, I, J, t} := \Pr[\exists j, j' \in J \text{ such that } S_j \cap I \neq \emptyset \text{ or } S_{j'} \cap I \neq \emptyset].$$

► **Lemma 6.** For any positive integers  $Q, M, t$  and  $0 < \varepsilon < 1$ ,

$$\Pr_{\mathbf{a} \sim \mathbb{Z}_Q^M, \mathbf{c} \sim \mathbb{Z}_Q} \left[ p_{\mathbf{a}, \mathbf{c}, t} \geq \frac{1 + \varepsilon}{1 - \varepsilon} \cdot \frac{t}{M} \right] \leq \frac{4Q^{1/4}}{\varepsilon \cdot \binom{M-1}{t-1}^{1/4}}.$$

where

$$p_{\mathbf{a}, \mathbf{c}, t} := p_{\mathbf{a}, \mathbf{c}, \{1\}, \{1\}, t}.$$

**Proof.** We have

$$p_{\mathbf{a}, \mathbf{c}, t} = \Pr_{\mathbf{x} \sim X_t} [x_1 = 1 \mid \langle \mathbf{a}, \mathbf{x} \rangle = c \bmod Q],$$

where  $X_t := \{\mathbf{x} \in \{0, 1\}^M : \|\mathbf{x}\|_1 = t\}$ . Therefore,

$$\begin{aligned} p_{\mathbf{a}, \mathbf{c}, t} &= \Pr_{\mathbf{x} \sim X_t} [x_1 = 1] \cdot \Pr_{\mathbf{x} \sim X_t} [\langle \mathbf{a}, \mathbf{x} \rangle = c \bmod Q \mid x_1 = 1] / \Pr_{\mathbf{x} \sim X_t} [\langle \mathbf{a}, \mathbf{x} \rangle = c \bmod Q] \\ &= \frac{t}{M} \cdot \Pr_{\mathbf{x}' \sim X'_{t-1}} [\langle \mathbf{a}_{-1}, \mathbf{x}' \rangle = c - a_1 \bmod Q] / \Pr_{\mathbf{x} \sim X_t} [\langle \mathbf{a}, \mathbf{x} \rangle = c \bmod Q], \end{aligned}$$

where  $\mathbf{a}_{-1}$  is  $\mathbf{a}$  with its first coordinate removed and  $X'_{t-1} := \{\mathbf{x} \in \{0, 1\}^{M-1} : \|\mathbf{x}\|_1 = t-1\}$ .

So, let

$$p_{\mathbf{a}, \mathbf{c}} := \Pr_{\mathbf{x} \sim X_t} [\langle \mathbf{a}, \mathbf{x} \rangle = c \bmod Q],$$

and

$$p'_{\mathbf{a}, \mathbf{c}} := \Pr_{\mathbf{x}' \in X'_{t-1}} [\langle \mathbf{a}_{-1}, \mathbf{x}' \rangle = c - a_1 \bmod Q].$$

As in the proof of Corollary 4, we see that

$$\sum_{c \in \mathbb{Z}_Q} |p_{\mathbf{a}, \mathbf{c}} - 1/Q| = \Delta(\langle \mathbf{a}, U_{X_t} \rangle \bmod Q, U_{\mathbb{Z}_Q}) \leq Q^{1/4} / \binom{M}{t}^{1/4} \quad (1)$$

except with probability at most  $Q^{1/4} / \binom{M}{t}^{1/4}$  over  $\mathbf{a}$ . Similarly,

$$\sum_{c \in \mathbb{Z}_Q} |p'_{\mathbf{a}, \mathbf{c}} - 1/Q| = \Delta(\langle \mathbf{a}_{-1}, U_{X'_{t-1}} \rangle \bmod Q, U_{\mathbb{Z}_Q}) \leq Q^{1/4} / \binom{M-1}{t-1}^{1/4} \quad (2)$$

except with probability at most  $Q^{1/4} / \binom{M-1}{t-1}^{1/4}$  over  $\mathbf{a}$ .

So, suppose that  $\mathbf{a}$  satisfies Eq. (1) and Eq. (2). Then, by Markov's inequality,

$$\Pr_{c \in \mathbb{Z}_Q} [p_{\mathbf{a}, \mathbf{c}} \geq (1 - \varepsilon)/Q] \leq \frac{Q^{1/4}}{\varepsilon \cdot \binom{M}{t}^{1/4}}$$

for any  $0 < \varepsilon < 1$ , and similarly,

$$\Pr_{c \in \mathbb{Z}_Q} [p'_{\mathbf{a},c} \leq (1+r)/Q] \leq \frac{Q^{1/4}}{\varepsilon \cdot \binom{M-1}{t-1}^{1/4}}.$$

Therefore, for such  $\mathbf{a}$ ,

$$\Pr[p_{\mathbf{a},c,t} \geq (1+\varepsilon)t/((1-\varepsilon)M)] \leq \frac{2Q^{1/4}}{\varepsilon \cdot \binom{M-1}{t-1}^{1/4}}.$$

The result then follows by union bound.  $\blacktriangleleft$

By repeated applications of union bound, we derive the following corollary.

► **Corollary 7.** *For any positive integers  $Q, M, M', t, v, v'$  and  $0 < \varepsilon < 1$ , let  $\mathbf{a} \sim \mathbb{Z}_Q^M$  and  $\mathbf{c} \sim \mathbb{Z}_Q^{M'}$  be sampled uniformly at random. Then,*

$$p_{\mathbf{a},\mathbf{c},I,J,t} \leq (v + tv') \cdot v' \cdot \frac{1+\varepsilon}{1-\varepsilon} \cdot \frac{t}{M}$$

for all  $I \in \binom{[M]}{\leq v}$ ,  $J \in \binom{[M']}{\leq v'}$  except with probability at most

$$4MM' \frac{Q^{1/4}}{\varepsilon \cdot \binom{M-1}{t-1}^{1/4}}.$$

**Proof.** Let  $\eta := \max_{i,j} p_{\mathbf{a},\mathbf{c},\{i\},\{j\},t}$ . By union bound, for any set  $I$ , we have

$$p_{\mathbf{a},\mathbf{c},I,\{j\},t} \leq \sum_{i \in I} p_{\mathbf{a},\mathbf{c},\{i\},\{j\},t} \leq |I| \cdot \eta.$$

Fix some set  $J$ . Let  $S_j$  be as in the definition of the hitting probability, and let  $I_{-j} := I \cup \bigcup_{j' \in J \setminus \{j\}} S_{j'}$ . Notice that  $|I_{-j}| \leq |I| + t|J|$ . Then,

$$p_{\mathbf{a},\mathbf{c},I,J,t} \leq \sum_{j \in J} p_{\mathbf{a},\mathbf{c},I_{-j},\{j\},t} \leq (|I| + t|J|) \cdot |J| \cdot \eta.$$

Finally, by union bound and Lemma 6, we have

$$\eta \leq \frac{1+\varepsilon}{1-\varepsilon} \cdot \frac{r}{M}$$

except with probability at most

$$4MM' \frac{Q^{1/4}}{\varepsilon \cdot \binom{M-1}{t-1}^{1/4}}.$$

The result follows.  $\blacktriangleleft$

### 2.3 Lattices and Lattice Problems

► **Definition 8** (Shortest Independent Vectors Problem). *For an approximation factor  $\gamma := \gamma(n) \geq 1$ ,  $\gamma$ -SIVP is the search problem defined as follows. Given a lattice  $\mathcal{L} \subset \mathbb{R}^n$ , output  $n$  linearly independent lattice vectors which all have length at most  $\gamma(n)$  times the minimum possible,  $\lambda_n(\mathcal{L})$ .*

► **Definition 9** (Short Integer Solutions). For integers  $m, Q, \beta$ , the (average-case) short integer solutions problem  $\text{SIS}(m, Q, \beta)$  is defined by  $m$  integers  $a_1, \dots, a_m$  drawn uniformly at random and independently from  $\mathbb{Z}_Q$ , and the goal is to come up with a non-zero vector  $\mathbf{x} = (x_1, \dots, x_m)$  where

$$\sum_{i \in [m]} x_i a_i = 0 \pmod{Q} \quad \text{and} \quad \|\mathbf{x}\|_1 := \sum_{i=1}^m |x_i| \leq \alpha$$

Following the seminal work of Ajtai [5], there have been several works that show how to solve the *worst-case*  $\gamma$ -SIVP problem given an algorithm for the *average-case* SIS problem. We will use the most recent one due to Gama et al. [18] (specialized to the case of cyclic groups for simplicity).

► **Theorem 10** (Worst-Case to Average-Case Reduction for SIS [31, 18]). Let  $n, Q, \beta \in \mathbb{N}$  where  $Q = (\beta n)^{\Omega(n)}$ . If there is an algorithm for the average-case SIS problem  $\text{SIS}(m, Q, \beta)$  over  $\mathbb{Z}_Q$  that runs in time  $T$ , then there is an  $(m + T) \cdot \text{poly}(n)$ -time algorithm for worst-case  $\tilde{O}(\sqrt{n \log m} \cdot \beta)$ -SIVP on any  $n$ -dimensional lattice  $L$ .

### 3 Variants of Average-case $k$ -SUM: Totality and Reductions

We define two variants of average-case  $k$ -SUM, one over the integers (which is the standard version of  $k$ -SUM) and one over the finite group  $\mathbb{Z}_Q$  of integers modulo  $Q$ . We show that the hardness of the two problems is tied together, which will allow us to use the modular version for our results down the line.

► **Definition 11** (Average-case  $k$ -SUM). For positive integers  $m, k \geq 2$  and  $u \geq 1$ , the average-case  $k$ -SUM( $u, m$ ) problem is the search problem defined as follows. The input is  $a_1, \dots, a_m \in [-u, u]$  chosen uniformly and independently at random, and the goal is to find  $k$  distinct indices  $i_1, \dots, i_k$  such that  $a_{i_1}, \dots, a_{i_k}$  with  $a_{i_1} + \dots + a_{i_k} = 0$ .

We define the modular version of the problem where the instance consists of numbers chosen at random from the finite additive group  $\mathbb{Z}_Q$  of numbers modulo  $Q$ . This will appear as an intermediate problem in our algorithm in Section 4 and our worst-case to average-case reduction in Section 5.

► **Definition 12** (Average-case Modular  $k$ -SUM). For integers  $m, k \geq 2$  and integer modulus  $Q \geq 2$ , the average-case  $k$ -SUM( $\mathbb{Z}_Q, m$ ) problem is the search problem defined as follows. The input is  $a_1, \dots, a_m \sim \mathbb{Z}_Q$  chosen uniformly and independently at random, and the goal is to find  $k$  distinct indices  $i_1, \dots, i_k$  such that  $a_{i_1}, \dots, a_{i_k}$  with  $a_{i_1} + \dots + a_{i_k} = 0 \pmod{Q}$ .

We highlight the distinction in our notation for the two problems. The former (non-modular version) is denoted  $k$ -SUM( $u, m$ ) (the first parameter is the bound  $u$  on the absolute value of the elements), whereas the latter is denoted  $k$ -SUM( $\mathbb{Z}_Q, m$ ) (the first parameter indicates the group on which the problem is defined). The second parameter always refers to the number of elements in the instance.

We now show that the modular problem is total when  $\binom{m}{k} \gtrsim Q$  and is unlikely to have a solution when  $\binom{m}{k} \lesssim Q$ .

► **Lemma 13.** If  $a_1, \dots, a_m \sim \mathbb{Z}_Q$  are sampled uniformly at random, and  $E_k$  is the event that there exist distinct indices  $i_1, \dots, i_k$  with  $a_{i_1} + \dots + a_{i_k} = 0 \pmod{Q}$ , then

$$1 - Q / \binom{m}{k} \leq \Pr[E_k] \leq \binom{m}{k} / Q.$$



**Proof.** Notice that for fixed indices  $i_1, \dots, i_k$ , the probability that  $a_{i_1} + \dots + a_{i_k} = 0$  is exactly  $1/Q$ . The upper bound then follows from a union bound over all  $\binom{m}{k}$   $k$ -tuples of indices. Furthermore, notice that  $i_1, \dots, i_k$  and  $j_1, \dots, j_k$ , the event that  $a_{i_1} + \dots + a_{i_k} = 0 \pmod{Q}$  is independent of the event that  $a_{j_1} + \dots + a_{j_k} = 0 \pmod{Q}$  as long as  $\{i_1, \dots, i_k\} \neq \{j_1, \dots, j_k\}$ . The lower bound then follows from Chebyshev's inequality.  $\blacktriangleleft$

► **Lemma 14.** *If  $a_1, \dots, a_m \sim [-u, u]$  are sampled uniformly at random, and  $E_k$  is the event that there exist distinct indices  $i_1, \dots, i_k$  with  $a_{i_1} + \dots + a_{i_k} = 0$ , then*

$$1 - e^{-\alpha} \leq \Pr[E_k] \leq \binom{m}{k} / (2u + 1),$$

where

$$\alpha := \frac{1}{4k + 2} \cdot \left\lfloor \frac{m}{k(20u + 10)^{1/k}} \right\rfloor \approx m / (k^2 u^{1/k}).$$

**Proof.** The upper bound follows immediately from the upper bound in Lemma 13 together with the observation that elements that sum to zero over the integers must sum to zero modulo  $Q := 2u + 1$  as well.

Let  $m' := k(10Q)^{1/k}$ . Let  $E'_k$  be the event that there exist distinct indices  $i_1, \dots, i_k \leq m'$  with  $a_{i_1} + \dots + a_{i_k} = 0$ . Notice that

$$\Pr[E_k] \geq 1 - (1 - \Pr[E'_k])^{\lfloor m/m' \rfloor} \geq 1 - \exp(-\lfloor m/m' \rfloor \Pr[E'_k]).$$

So, it suffices to show that

$$\Pr[E'_k] \geq \frac{1 - Q / \binom{m'}{k}}{2k + 1} \geq \frac{1}{4k + 2}.$$

By Lemma 13, we know that with probability at least  $1 - Q / \binom{m'}{k}$ , there exists a  $k$ -SUM that sums to zero modulo  $Q$  in the first  $m'$  elements. I.e.,  $a_{i_1} + \dots + a_{i_k} = \ell Q$  for some  $\ell \in \{-k, -k + 1, \dots, k - 1, k\}$  and  $i_1, \dots, i_k \leq m'$ . We wish to argue that  $\ell = 0$  is at least as likely as  $\ell = i$  for any  $i$ .

Let  $p(k', s) := \Pr[a_1 + \dots + a_{k'} = s]$  for integers  $k', s$ . Notice that for  $s \geq 0$ , we have

$$\begin{aligned} p(k' + 1, s) - p(k' + 1, s + 1) &= (p(k', -(s + u)) - p(k', s + u + 1)) / (2u + 1) \\ &= (p(k', s + u) - p(k', s + u + 1)) / (2u + 1). \end{aligned}$$

It then follows from a simple induction argument that  $p(k, s + 1) \leq p(k, s)$ . In particular,  $p(k, \ell Q) \leq p(k, 0)$  for any  $\ell$ . Therefore, letting  $E'_{k,Q}$  be the event that the first  $m'$  elements contain a  $k$ -SUM modulo  $Q$ , we have

$$\begin{aligned} \Pr[E'_k] &\geq \Pr[E'_{k,Q}] \cdot \Pr[a_{i_1} + \dots + a_{i_k} = 0 \mid a_{i_1} + \dots + a_{i_k} = 0 \pmod{Q}] \\ &\geq \frac{\Pr[E'_{k,Q}]}{2k + 1}. \end{aligned}$$

Finally, by Lemma 13, we have

$$\Pr[E'_{k,Q}] \geq 1 - Q / \binom{m'}{k} \geq 1/2,$$

as needed.  $\blacktriangleleft$

### 3.1 From $k$ -SUM to Modular $k$ -SUM and Back

We first show that an algorithm for the modular  $k$ -SUM problem gives us an algorithm for the  $k$ -SUM problem. A consequence of this is that when we describe the algorithm for  $k$ -SUM in Section 4, we will focus on the modular variant.

► **Lemma 15.** *Let  $u$  be a positive integer and let  $Q = 2u + 1$ . If there is an algorithm for the  $k$ -SUM( $\mathbb{Z}_Q, m$ ) that runs in time  $T$  and succeeds with probability  $p$ , then there is an algorithm for  $2k$ -SUM( $u, 2m$ ) that runs in time  $O(T)$  and succeeds with probability at least  $p^2/k$ .*

**Proof.** Let  $\mathcal{A}$  be the purported algorithm for  $k$ -SUM( $\mathbb{Z}_Q, m$ ). The algorithm for  $2k$ -SUM( $u, 2m$ ) receives  $2m$  integers  $a_1, \dots, a_{2m}$  in the range  $[-u, u]$  and works as follows. We use the natural embedding to associate elements in  $\mathbb{Z}_Q$  with elements in  $[-u, u]$ , so we may think of  $a_1, \dots, a_{2m}$  also as elements in  $\mathbb{Z}_Q$  (simply by considering their coset modulo  $Q$ ).

- Run  $\mathcal{A}$  on  $a_1, \dots, a_m$  to obtain a  $k$ -subset  $S_1$ . If  $\mathcal{A}$  does not succeed, then fail.
- Run  $\mathcal{A}$  on  $-a_{m+1}, \dots, -a_{2m}$  to obtain a  $k$ -subset  $S_2$ . If  $\mathcal{A}$  does not succeed, then fail.
- If  $\sum_{i \in S_1} a_i = -\sum_{i \in S_2} a_i$ , output  $S_1 \cup S_2$  as the  $2k$ -subset. Fail otherwise.

It is clear that the run-time is  $O(T)$  and that if the algorithm does not fail then it indeed outputs a valid  $2k$ -sum. It suffices to bound the probability that the algorithm succeeds.

Since the first two steps run  $\mathcal{A}$  on independent and identically distributed input, we can deduce that the probability that both succeed is  $p^2$ , and in the case that both succeed, their output satisfies

$$\sum_{i \in S_1} a_i = \alpha_1 Q \quad \text{and} \quad \sum_{i \in S_2} a_i = \alpha_2 Q$$

for some integers  $\alpha_1, \alpha_2 \in (-k/2, k/2)$ , which are independent and identically distributed random variables. The probability that  $\alpha_1 = \alpha_2$  is therefore at least  $1/k$ , since the collision probability of a random variable is bounded by the inverse of its support size. If this happens then,  $\sum_{i \in S_1} a_i = \sum_{i \in S_2} a_i$  and the algorithm succeeds. Thus, we conclude that our algorithm succeeds with probability at least  $p^2/k$ . ◀

Finally, we show a proof in the other direction. Namely, that an algorithm for the  $k$ -SUM problem gives us an algorithm for the modular  $k$ -SUM problem. We will use this when we describe the worst-case to average-case reduction in Section 5.

► **Lemma 16.** *For  $m \geq k \cdot u^{2/k}$ , if there is an algorithm for  $k$ -SUM( $u, m$ ) that runs in time  $T$  and succeeds with probability  $p$ , then there is an algorithm for  $k$ -SUM( $\mathbb{Z}_{2u+1}, m$ ) that runs in time  $T$  and succeeds with probability  $p$ .*

**Proof.** Let  $\mathcal{A}$  be the purported algorithm for  $k$ -SUM( $u, m$ ). The algorithm for  $k$ -SUM( $\mathbb{Z}_{2u+1}, m$ ) receives  $m$  integers  $a_1, \dots, a_m \in \mathbb{Z}_{2u+1}$  and works as follows.

As before, identify  $\mathbb{Z}_{2u+1}$  with the interval  $[-u, u]$  and run  $\mathcal{A}$  on  $a_1, \dots, a_m$ . We can then simply output the resulting  $k$ -subset  $S$ . In particular, if  $\sum_{i \in S} a_i = 0$ , then we of course have  $\sum_{i \in S} a_i = 0 \pmod{2u+1}$ .

Clearly, the success probability of the resulting algorithm is at least  $p$ , since the input to  $\mathcal{A}$  is distributed uniformly. ◀

#### 4 The $u^{O(1/\log k)}$ -time Algorithm for Average-case $k$ -SUM

In this section, we describe a variant of the Blum-Kalai-Wasserman algorithm [12] for the average-case  $k$ -SUM problem that runs in time  $u^{O(1/\log k)}$ .

► **Theorem 17.** *There is a  $\tilde{O}(2^\ell q^2)$ -time algorithm that solves average-case  $2^\ell$ -SUM( $\mathbb{Z}_{q^\ell}, m$ ) for  $m = \tilde{\Theta}(2^\ell q^2)$ .*

**Proof.** On input  $a_1, \dots, a_m \in \mathbb{Z}_{q^\ell}$  with  $m := 1000\ell^2 q^2 2^\ell \log q = \tilde{\Theta}(2^\ell q^2)$ , the algorithm behaves as follows. Let  $L_1 := (a_1, \dots, a_m)$ . For  $i = 1, \dots, \ell$ , the algorithm groups the elements in  $L_i$  according to their value modulo  $q^i$ . It then greedily groups them into  $m_{i+1}$  disjoint points  $(a, b)$  with  $a + b = 0 \pmod{q^i}$ . It sets  $L_{i+1}$  to be the list of sums of these pairs (and records the indices of the  $2^i$  input elements that sum to  $a + b$ ). If at any point the algorithm fails to find such pairs, it simply fails; otherwise, the algorithm outputs the elements  $a_{i_1}, \dots, a_{i_{2^\ell}}$  satisfying  $\sum a_{i_j} = 0 \pmod{q^\ell}$  found in the last step.

The running time of the algorithm is clearly  $\text{poly}(\ell, \log q, \log m)m$  as claimed. To prove correctness, we need to show that at each step the algorithm is likely to succeed in populating the list  $L_{i+1}$  with at least  $m_i := (\ell^2 - i^2)/\ell^2 \cdot m/2^{i-1}$  elements, since clearly the algorithm outputs a valid  $2^\ell$ -SUM in this case.

Suppose that the algorithm succeeds up to the point where it populates  $L_i$ . Let  $L_i = (b_1, \dots, b_{m_i})$ , and  $b'_i := (b_i/q^{i-1}) \pmod{q}$ , where the division by  $q^{i-1}$  is possible because  $b_i = 0 \pmod{q^{i-1}}$  by assumption. Notice that the  $b'_i$  are independent and uniformly random. For  $j \in \mathbb{Z}_q$ , let  $c_j := |\{i : b'_i = j \pmod{q}\}|$ . Notice that the algorithm successfully populates  $L_{i+1}$  if and only if

$$\sum_{j \in \mathbb{Z}_q} \min\{c_j, c_{-j}\}/2 \geq m_{i+1}.$$

By the Chernoff-Hoeffding bound, we have that

$$\Pr [c_j < m_i/q - 10\sqrt{m_i \log m_i}] \leq 1/m_i^2$$

It follows that

$$\sum_j \min\{c_j, c_{-j}\}/2 \geq q \min\{c_j\}/2 \geq m_i/2 - 5q\sqrt{m_i \log m_i} \geq m_{i+1}$$

except with probability at most  $1/m_i$ . By union bound, we see that the algorithm succeeds in populating every list except with probability at most  $\sum 1/m_i \ll 1/10$ , as needed. ◀

Combining this with Lemma 15 (the reduction from  $k$ -SUM to modular  $k$ -SUM), we obtain the following corollary.

► **Corollary 18.** *For  $u = (q^\ell - 1)/2$  for odd  $q$  and  $k = 2^{\ell+1}$ , there is a  $u^{O(1/\log k)}$ -time algorithm for  $k$ -SUM( $u, m$ ) for  $m = u^{\Theta(1/\log k)}$ .*

#### 5 From Worst-case Lattice Problems to Average-case $k$ -SUM

In this section, we describe our main result, namely a worst-case to average-case reduction for  $k$ -SUM. We state the theorem below.

► **Theorem 19.** *Let  $k, m, u, n$  be positive integers, and  $0 < \varepsilon < \varepsilon'$  where*

$$u = k^{2(1+\varepsilon')cn/\varepsilon'} \text{ and } m = u^{\varepsilon/(2 \log k)}$$

*for some universal constant  $c > 0$ . If there is an algorithm for average-case  $k$ -SUM( $u, m$ ) that runs in time  $T_{\text{kSUM}} = T_{\text{kSUM}}(k, u, m)$ , then there is an algorithm for the worst-case  $n^{1+\varepsilon'}$ -approximate shortest independent vectors problem (SIVP) that runs in time  $2^{O(\varepsilon n/\varepsilon' + \log n)} \cdot T_{\text{kSUM}}$ .*

When we say that a  $k$ -SUM algorithm succeeds, we mean that it outputs a  $k$ -subset of the input that sums to 0 with probability  $1 - \delta$  for some tiny  $\delta$ . This can be achieved starting from an algorithm that succeeds with (some small) probability  $p$  by repeating, at the expense of a multiplicative factor of  $1/p \cdot \log(1/\delta)$  in the run-time. We ignore such issues for this exposition, and assume that the algorithm outputs a  $k$ -sum with probability  $1 - \delta$  for a tiny  $\delta$ .

Before we proceed to the proof, a few remarks on the parameters of Theorem 19 are in order. First, note that the parameter settings imply that  $m^k \gg u$ , therefore putting us in the total regime of parameters for  $k$ -SUM. Secondly, setting  $\varepsilon' = 100$  (say), we get the following consequence: if there is a  $k$ -SUM algorithm that, on input  $m = u^{\varepsilon/(2 \log k)}$  numbers, runs in time roughly  $m$ , then we have an  $n^{101}$ -approximate SIVP algorithm that runs in time  $\approx 2^{\varepsilon cn}$ . Now,  $\varepsilon$  is the “knob” that one can turn to make the SIVP algorithm run faster, assuming a correspondingly fast  $k$ -SUM algorithm that works with a correspondingly smaller instance.

**Proof.** The theorem follows from the following observations:

- First, by Theorem 10, there is a reduction from  $\tilde{O}(\sqrt{n \log m'} \cdot \beta)$ -approximate SIVP to SIS( $m', Q, \beta$ ), where we take  $m' := \lceil k^{10cn/(k\varepsilon')} n^{10} \rceil$ . The reduction produces SIS instances over  $\mathbb{Z}_Q$  where  $Q \geq (\beta n)^{cn}$  for some constant  $c$ , and works as long as the SIS algorithm produces solutions of  $\ell_1$  norm at most  $\beta$ . If the SIS algorithm runs in time  $T_{\text{SIS}} = T_{\text{SIS}}(m', Q, \beta)$ , the SIVP algorithm runs in time  $(m' + T_{\text{SIS}}) \cdot \text{poly}(n)$ . We take  $\beta := n^{\varepsilon'}$ .
- Second, as our main technical contribution, we show in Lemma 20 how to reduce SIS to  $k$ -SUM. Note that Theorem 10 gives us the freedom to pick  $Q$ , as long as it is sufficiently large. We will set  $Q = q^r$  where  $r = \lfloor \varepsilon' \log n / (2 \log k) \rfloor$  for a prime  $q \approx u \approx (\beta n)^{cn/r} \approx k^{2(1+\varepsilon')cn/\varepsilon'}$ .

Now, Lemma 20 (with  $k = t$ ) shows a reduction from SIS( $m', Q, \beta$ ) to  $2k$ -SUM( $\mathbb{Z}_q, m$ ) (provided that  $m' \gg q^{1/k} k^{4r/k} m^{4/k}$ , which holds in this case). The reduction produces a SIS solution with  $\ell_1$  norm bounded by  $k^{2r} \leq \beta$ .

The running time of the resulting algorithm is

$$rm'(m \cdot \text{poly}(k, \log q) + 10T_{\text{kSUM}}) \approx 2^{O(\varepsilon n/\varepsilon' + \log n)} \cdot T_{\text{kSUM}} .$$

- Finally, by Lemma 16, we know that modular  $2k$ -SUM over  $\mathbb{Z}_q$  can be reduced to  $k$ -SUM over the integers in the interval  $[-u, u]$  for  $u \approx q$  with essentially no overhead.

This finishes the proof. ◀

The following lemma shows our main reduction from SIS to  $k$ -SUM. In particular, taking  $k = t$ ,  $m' \gg (m^4 q)^{1/k} \cdot (10k)^{4r}$  (so that  $\delta$  is small), and  $m \gg q^{1/k}$  (so that  $k$ -SUM( $\mathbb{Z}_q, m$ ) is total) gives a roughly  $rm m'$ -time reduction from SIS over  $\mathbb{Z}_{q^r}$  to  $k$ -SUM over  $\mathbb{Z}_q$  with high success probability.

► **Lemma 20.** *Let  $m, m', k, r, t$  be positive integers and  $q > (tk)^r$  a prime, and let  $Q = q^r$ . If there is an algorithm that solves (average-case)  $k$ -SUM( $\mathbb{Z}_q, m$ ) in time  $T$  with success probability  $p$ , then there is an algorithm that solves SIS( $m', Q, \beta$ ) in time  $r \cdot m'(m \cdot \text{poly}(k, t, \log q) + 10T)/p$  with success probability at least  $1 - \delta$  and produces a solution with  $\ell_1$  norm  $\beta \leq (tk)^r$ , where*

$$\delta := \frac{100rm(m')^2}{p} \cdot \frac{q^{1/4}}{(m'/(10tk)^{2r+1})^{t/4}}.$$

**Proof.** At a high level, the idea is to run a variant of the Blum-Kalai-Wasserman [12] algorithm where in each iteration, we call a  $k$ -SUM oracle. In particular, on input  $a_1, \dots, a_{m'}$ , the algorithm operates as follows.

- In the beginning of the  $i^{\text{th}}$  iteration, the algorithm starts with a sequence of

$$m_i := \lceil m' / (10t^2k^2)^{i-1} \rceil$$

numbers  $a_{i,1}, \dots, a_{i,m_i}$ . The invariant is that  $a_{i,j} = 0 \pmod{q^{i-1}}$  for all  $j$ . It then generates disjoint  $S_{i,1}, \dots, S_{i,m_{i+1}} \subseteq [m_i]$  such that  $|S_{i,\ell}| \leq kt$  and  $\sum_{j \in S_{i,\ell}} a_{i,j} = 0 \pmod{q^i}$ , in a way that we will describe below.

As the base case, for  $i = 1$ ,  $a_{1,j} = a_j$ , the input itself, and the invariant is vacuous.

- In the  $i^{\text{th}}$  iteration, we apply the re-randomization lemma (Corollary 4), computing subsets of  $t$  randomly chosen elements from  $a_{i,1}, \dots, a_{i,m_i}$ , to generate  $m_i^* := 10m \lceil m_{i+1}/p \rceil$  numbers  $c_{i,1}, \dots, c_{i,m_i^*}$ .
- Let  $d_{i,j} = c_{i,j}/q^{i-1} \pmod{q} \in \mathbb{Z}_q$ . Note that this is well-defined because each  $c_{i,j} = 0 \pmod{q^{i-1}}$ .
- Divide the  $d_{i,j}$  into  $10 \lceil m_{i+1}/p \rceil$  disjoint blocks of  $m$  elements each, set  $\ell = 1$ . For each block, feed the block to the  $k$ -SUM algorithm to obtain  $d_{i,j_1}, \dots, d_{i,j_k}$ . This yields corresponding subsets  $S_1^*, \dots, S_k^* \in \binom{[m_i]}{t}$  such that  $\sum_{j \in S_x^*} a_{i,j}/q^{i-1} = d_{i,j_x} \pmod{q}$ . If  $d_{i,j_1} + \dots + d_{i,j_k} = 0 \pmod{q}$  and the sets  $S_1^*, \dots, S_k^*, S_{i,1}, \dots, S_{i,\ell-1}$  are pairwise disjoint, then set  $S_{i,\ell} := \bigcup S_x^*$  and increment  $\ell$ .
- If  $\ell \leq m_{i+1}$ , the algorithm fails. Otherwise, take  $a_{i+1,\ell} := \sum_{j \in S_{i,\ell}} a_{i,j}$  for  $\ell = 1, \dots, m_{i+1}$ .
- At the end of the  $r^{\text{th}}$  iteration we obtain a  $(kt)^r$ -subset of the  $a_1, \dots, a_{m'}$  that sums to 0  $\pmod{Q}$ .

We now analyze the correctness, run-time and the quality of output of this reduction.

The reduction calls the  $k$ -SUM oracle  $\sum m_i^*/m \leq 20rm'/p$  times. The rest of the operations take  $rm m' \text{poly}(k, t, \log q)/p$  time for a total of  $r \cdot m'(m \text{poly}(k, t, \log q) + 10T)/p$  time, as claimed. Furthermore, the  $\ell_1$  norm of the solution is  $\beta \leq (tk)^r$ , as claimed.

Finally, we show that the algorithm succeeds with the claimed probability. Since the sets  $S_{i,\ell}$  are disjoint and do not depend on  $a_{i,j} - (a_{i,j} \pmod{q^i})$ , it follows from a simple induction argument that at each step the  $a_{i,j}$  are uniformly random and independent elements from  $q^{i-1}\mathbb{Z}/q^r\mathbb{Z}$ . Therefore, by Corollary 4, the statistical distance of the collection of all  $d_{i,j}$  (for a given  $i$ ) from uniformly random variables that are independent of the  $a_{i,j}$  is  $\delta_i \leq (m_i^* + 1) \cdot (\frac{qt}{m_i^*})^{1/4}$ . In total, the statistical distance of all samples from uniform is then at most  $\sum \delta_i < \delta/3$  for our choice of parameters. So, up to statistical distance  $\delta/3$ , we can treat the  $d_{i,j}$  as uniformly random and independent elements.

It remains to show that, assuming that the  $d_{i,j}$  are uniformly random and independent, then we will find *disjoint* sets  $S_{i,1}, \dots, S_{i,m_{i+1}}$  with  $\sum_{j \in S_{i,\ell}} d_{i,j} = 0 \pmod{q}$  at each step except with probability at most  $2\delta/3$ . Let  $\mathbf{b}_i := (a_{i,1}/q^{i-1} \pmod{q}, \dots, a_{i,m_i}/q^{i-1} \pmod{q})$ . By Corollary 7, we have

$$p_{\mathbf{b}_i, \mathbf{d}_i, I, J, t} \leq 10t^2k^2m_{i+1}/m_i \leq 1/2$$

for all  $I \in \binom{[m_i]}{\leq v}$  and  $J \in \binom{[m_i^*]}{\leq v}$  except with probability at most  $10m_i m_i^* q^{1/4} / \binom{m_i-1}{t-1}^{1/4} < \delta/3$  for  $v := tkm_{i+1} \geq |T|$ ,  $v' := k$ , and  $\varepsilon := 1/2$ .

So, suppose this holds. Notice that  $\Pr[d_{i,j_1} + \dots + d_{i,j_k} = 0 \pmod{q}] = p$  by definition. And, conditioned on  $d_{i,j_1}, \dots, d_{i,j_k}$ , the  $S_x^*$  are independent and uniformly random subject to the constraint that  $\sum_{j \in S_x^*} a_{i,j} / q^{i-1} = d_{i,j_x} \pmod{q}$ . Therefore, the probability that  $S_1^*, \dots, S_k^*$ ,  $I := S_{i,1} \cup \dots \cup S_{i,\ell-1}$  are pairwise disjoint in this case is exactly

$$pb_{i,d_i,I,J,t} \leq 1/2,$$

where  $J := \{j_1, \dots, j_k\}$ . So, each time we call the oracle, we increment  $\ell$  with probability at least  $p/2$ . It follows from the Chernoff-Hoeffding bound that we increment  $\ell$  at least  $m_{i+1}$  times except with probability at most  $e^{-m_{i+1}/100} \ll \delta/3$ .

Putting everything together, we see that the algorithm fails with probability at most  $\delta$ , as claimed.  $\blacktriangleleft$

---

## References

- 1 Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. SETH-based lower bounds for subset sum and bicriteria path. In *SODA*, 2019.
- 2 Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *FOCS*, 2014.
- 3 Divesh Aggarwal and Eldon Chung. A note on the concrete hardness of the shortest independent vector in lattices. *Information Processing Letters*, 167, 2021.
- 4 Divesh Aggarwal, Jianwei Li, Phong Q. Nguyen, and Noah Stephens-Davidowitz. Slide reduction, revisited—Filling the gaps in SVP approximation. In *CRYPTO*, 2020. URL: <https://arxiv.org/abs/1908.03724>.
- 5 Miklós Ajtai. Generating hard instances of lattice problems. In *STOC*, 1996.
- 6 Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of Learning with Errors. *J. Mathematical Cryptology*, 9(3), 2015. URL: <http://eprint.iacr.org/2015/046>.
- 7 Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange — A new hope. In *USENIX Security Symposium*, 2016.
- 8 Kyriakos Axiotis, Arturs Backurs, Ce Jin, Christos Tzamos, and Hongxun Wu. Fast modular subset sum using linear sketching. In *SODA*, 2019.
- 9 Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Average-case fine-grained hardness. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *STOC*, 2017.
- 10 Anja Becker, Jean-Sébastien Coron, and Antoine Joux. Improved generic algorithms for hard knapsacks. In *CRYPTO*, 2011.
- 11 Richard Bellman. Notes on the theory of dynamic programming iv - maximization over discrete sets. *Naval Research Logistics Quarterly*, 3:67–70, 1956. doi:10.1002/nav.3800030107.
- 12 Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003. doi:10.1145/792538.792543.
- 13 E. Boix-Adserà, M. Brennan, and G. Bresler. The average-case complexity of counting cliques in Erdős-Rényi hypergraphs. In *FOCS*, 2019.
- 14 Karl Bringmann. A near-linear pseudopolynomial time algorithm for subset sum. In Philip N. Klein, editor, *SODA*, 2017.
- 15 Mina Dalirrooyfard, Andrea Lincoln, and V. Vassilevska Williams. New techniques for proving fine-grained average-case hardness. In *FOCS*, 2020.
- 16 Anka Gajentaan and Mark H Overmars. On a class of  $O(n^2)$  problems in computational geometry. *Computational Geometry*, 5(3), 1995. URL: <http://www.sciencedirect.com/science/article/pii/0925772195000222>.

- 17 Anka Gajentaan and Mark H. Overmars. On a class of  $O(n^2)$  problems in computational geometry. *Computational Geometry*, 45(4), 2012. URL: <http://www.sciencedirect.com/science/article/pii/S0925772111000927>.
- 18 Nicolas Gama, Malika Izabachène, Phong Q. Nguyen, and Xiang Xie. Structural lattice reduction: Generalized worst-case to average-case reductions and homomorphic cryptosystems. In Marc Fischlin and Jean-Sébastien Coron, editors, *Eurocrypt*, 2016.
- 19 Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008. URL: <https://eprint.iacr.org/2007/432>.
- 20 Oded Goldreich and Guy N. Rothblum. Counting t-cliques: Worst-case to average-case reductions and direct interactive proof systems. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 77–88. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00017.
- 21 Oded Goldreich and Guy N. Rothblum. Worst-case to average-case reductions for subclasses of P. In Oded Goldreich, editor, *Computational Complexity and Property Testing - On the Interplay Between Randomness and Computation*, volume 12050 of *Lecture Notes in Computer Science*, pages 249–295. Springer, 2020. doi:10.1007/978-3-030-43662-9\_15.
- 22 Shuichi Hirahara and Nobutaka Shimizu. Nearly optimal average-case complexity of counting bicliques under SETH, 2020. arXiv:2010.05822.
- 23 Ellis Horowitz and Sartaj Sahni. Computing partitions with applications to the knapsack problem. *J. ACM*, 21(2):277–292, 1974. doi:10.1145/321812.321823.
- 24 Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In David S. Johnson, editor, *STOC*, 1989.
- 25 Ce Jin and Hongxun Wu. A simple near-linear pseudopolynomial time randomized algorithm for subset sum. In *SOSA*, 2019.
- 26 Paul Kirchner and Pierre-Alain Fouque. Time-memory trade-off for lattice enumeration in a ball, 2016.
- 27 Konstantinos Koiliaris and Chao Xu. A faster pseudopolynomial time algorithm for subset sum. In Philip N. Klein, editor, *SODA*, 2017.
- 28 Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3SUM conjecture. In *SODA*, 2016.
- 29 Ravi Kumar and D. Sivakumar. On polynomial-factor approximations to the shortest lattice vector length. *SIAM J. Discrete Math.*, 16(3):422–425, 2003.
- 30 Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In *CRYPTO*, 2013.
- 31 Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM Journal of Computing*, 37(1), 2007.
- 32 NIST. Post-quantum cryptography standardization. URL: <https://csrc.nist.gov/Projects/Post-Quantum-Cryptography>.
- 33 Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In *STOC*, 2010.
- 34 Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. In *SODA*, 2010.
- 35 Chris Peikert. A decade of lattice cryptography. *Foundations and Trends in Theoretical Computer Science*, 10(4), 2016.
- 36 David A. Wagner. A generalized birthday problem. In *CRYPTO*, 2002.
- 37 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians (ICM 2018)*. 2018.
- 38 Virginia Vassilevska Williams and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013. doi:10.1137/09076619X.

## A Total $k$ -SUM and Computational Geometry

Here, we show that one of the main results in [16, 17] can be extended meaningfully to our setting, i.e., to the case of search  $k$ -sum over  $\mathbb{Z}_Q$  with  $\binom{m}{k} \gg Q$ . (In [16, 17], Gajentaan and Overmars only considered decisional 3-SUM.) Specifically, we will reduce the following problem to  $k$ -SUM in this regime.

► **Definition 21.** For  $d \geq 1$  and  $Q, m \geq 2$  with  $Q$  prime, the  $(Q, m, d)$ -Plane problem is the following search problem. The input is  $\mathbf{a}_1, \dots, \mathbf{a}_m \in \mathbb{Z}_Q^{d+1}$ . The goal is to find distinct  $\mathbf{a}_{i_1}, \dots, \mathbf{a}_{i_{d+2}}$  that lie in a  $d$ -dimensional affine hyperplane over the field  $\mathbb{Z}_Q$ . (In other words,  $\mathbf{a}_{i_{d+1}} - \mathbf{a}_{i_{d+2}}$  can be written as a linear combination of  $\mathbf{a}_{i_1} - \mathbf{a}_{i_{d+2}}, \dots, \mathbf{a}_{i_d} - \mathbf{a}_{i_{d+2}}$  over  $\mathbb{Z}_Q$ .)

► **Lemma 22.** For  $d \geq 1$  and  $Q, m \geq 2$  with  $Q$  prime, there is a reduction from  $(d+2)$ -SUM( $\mathbb{Z}_Q, m$ ) to  $(Q, m, d)$ -Plane.

**Proof.** Let  $f_d : \mathbb{Z}_Q \rightarrow \mathbb{Z}_Q^{d+1}$  be the map  $f_d(a) := (a, a^2, a^3, \dots, a^d, a^{d+2})$ . E.g.,  $f_1(a) = (a, a^3)$ ,  $f_2(a) = (a, a^2, a^4)$ , etc. On input  $a_1, \dots, a_m \in \mathbb{Z}_Q$ , the reduction simply calls its  $(Q, m, d)$ -Plane oracle on  $f_d(a_1), \dots, f_d(a_m) \in \mathbb{F}_Q^{d+1}$ , receiving as output distinct indices  $i_1, \dots, i_{d+2}$  such that  $f_d(a_{i_1}), \dots, f_d(a_{i_{d+2}})$  lie in a  $d$ -dimensional affine hyperplane (assuming that such indices exist). The reduction simply outputs these indices, i.e., it claims that  $a_{i_1} + \dots + a_{i_{d+2}} = 0 \pmod{Q}$ .

Notice that  $d+2$  points  $\mathbf{b}_1, \dots, \mathbf{b}_{d+2} \in \mathbb{Z}_Q^{d+1}$  lie in a  $d$ -dimensional affine hyperplane if and only if the matrix  $(\mathbf{b}_1 - \mathbf{b}_{d+2}, \mathbf{b}_2 - \mathbf{b}_{d+2}, \dots, \mathbf{b}_{d+1} - \mathbf{b}_{d+2}) \in \mathbb{Z}_Q^{(d+1) \times (d+1)}$  has determinant zero. (Here, we have used the fact that  $\mathbb{Z}_Q$  is a field.) So, we consider the matrix

$$\mathbf{M} := \mathbf{M}(b_1, \dots, b_{d+2}) := \begin{pmatrix} b_1 - b_{d+2} & b_2 - b_{d+2} & \cdots & b_{d+1} - b_{d+2} \\ b_1^2 - b_{d+2}^2 & b_2^2 - b_{d+2}^2 & \cdots & b_{d+1}^2 - b_{d+2}^2 \\ \vdots & \vdots & \ddots & \vdots \\ b_1^d - b_{d+2}^d & b_2^d - b_{d+2}^d & \cdots & b_{d+1}^d - b_{d+2}^d \\ b_1^{d+2} - b_{d+2}^{d+2} & b_2^{d+2} - b_{d+2}^{d+2} & \cdots & b_{d+1}^{d+2} - b_{d+2}^{d+2} \end{pmatrix} \in \mathbb{F}_Q^{(d+1) \times (d+1)}.$$

We claim that

$$\det(\mathbf{M}) = (-1)^d (b_1 + \dots + b_{d+2}) \cdot \prod_{i < j} (b_j - b_i),$$

The result then follows, since this is zero if and only if  $b_i = b_j$  for some  $i \neq j$  or  $b_1 + \dots + b_{d+2} = 0$ . Since by definition the  $(Q, m, d)$ -Plane oracle only outputs distinct vectors on a hyperplane, this means that its output must correspond to distinct elements with  $a_{i_1} + \dots + a_{i_{d+2}} = 0 \pmod{Q}$ .

To prove that the determinant has the appropriate form, we first notice that without loss of generality we may take  $b_{d+2} = 0$ . Next, we define

$$\mathbf{M}' := \mathbf{M}'(b_1, \dots, b_{d+1}) := \begin{pmatrix} b_1 & b_2 & \cdots & b_{d+1} \\ b_1^2 & b_2^2 & \cdots & b_{d+1}^2 \\ \vdots & \vdots & \ddots & \vdots \\ b_1^d & b_2^d & \cdots & b_{d+1}^d \\ b_1^{d+1} & b_2^{d+1} & \cdots & b_{d+1}^{d+1} \end{pmatrix} \in \mathbb{F}_Q^{(d+1) \times (d+1)}$$

This is just a Vandermonde matrix with columns scaled up by  $b_i$ . So, its determinant is a scaling of the Vandermonde determinant,

$$\det(\mathbf{M}') = b_1 \cdots b_{d+1} \cdot \prod_{i < j} (b_j - b_i).$$



Finally, we recall Cramer's rule, which in particular tells us that

$$\det(\mathbf{M}) = p_{d+1} \det(\mathbf{M}')$$

for the unique  $\mathbf{p} := (p_1, \dots, p_{d+1}) \in \mathbb{Z}_Q^{d+1}$  satisfying  $\mathbf{p}^T \mathbf{M}' = (b_1^{d+2}, \dots, b_{d+1}^{d+2})$ . I.e., the coordinates of  $\mathbf{p}$  form the polynomial  $p(x) := p_1 + p_2x + \dots + p_{d+1}x^d$  such that  $p(b_i) = b_i^{d+1}$ . The result follows by noting that  $p_i = (-1)^{i-1} \sum_{S \in \binom{[d+2-i]}{d+2-i}} \prod_{j \in S} b_j$ . In particular,  $p_{d+1} = (-1)^d (b_1 + \dots + b_{d+1})$ , as needed.  $\blacktriangleleft$



# Improved Hitting Set for Orbit of ROABPs

Vishwas Bhargava  

Department of Computer Science, Rutgers University, Piscataway, NJ, USA

Sumanta Ghosh  

Department of Computer Science, IIT Bombay, India

---

## Abstract

The orbit of an  $n$ -variate polynomial  $f(\mathbf{x})$  over a field  $\mathbb{F}$  is the set  $\{f(A\mathbf{x}+\mathbf{b}) \mid A \in \text{GL}(n, \mathbb{F}) \text{ and } \mathbf{b} \in \mathbb{F}^n\}$ , and the orbit of a polynomial class is the union of orbits of all the polynomials in it. In this paper, we give improved constructions of hitting-sets for the orbit of read-once oblivious algebraic branching programs (ROABPs) and a related model. Over fields with characteristic zero or greater than  $d$ , we construct a hitting set of size  $(ndw)^{O(w^2 \log n \cdot \min\{w^2, d \log w\})}$  for the orbit of ROABPs in unknown variable order where  $d$  is the individual degree and  $w$  is the width of ROABPs. We also give a hitting set of size  $(ndw)^{O(\min\{w^2, d \log w\})}$  for the orbit of polynomials computed by  $w$ -width ROABPs in any variable order. Our hitting sets improve upon the results of Saha and Thankey [43] who gave an  $(ndw)^{O(d \log w)}$  size hitting set for the orbit of commutative ROABPs (a subclass of *any-order* ROABPs) and  $(nw)^{O(w^6 \log n)}$  size hitting set for the orbit of multilinear ROABPs. Designing better hitting sets in large individual degree regime, for instance  $d > n$ , was asked as an open problem by [43] and this work solves it in small width setting.

We prove some new rank concentration results by establishing *low-cone concentration* for the polynomials over vector spaces, and they strengthen some previously known *low-support* based rank concentrations shown in [17]. These new low-cone concentration results are crucial in our hitting set construction, and may be of independent interest. To the best of our knowledge, this is the first time when low-cone rank concentration has been used for designing hitting sets.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Algebraic complexity theory; Computing methodologies  $\rightarrow$  Algebraic algorithms

**Keywords and phrases** Hitting Set, Low Cone Concentration, Orbits, PIT, ROABP

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.30

**Category** RANDOM

**Related Version** *Full Version:* <https://ecc.weizmann.ac.il/report/2021/062>

**Funding** *Vishwas Bhargava:* Research supported in part by the Simons Collaboration on Algorithms and Geometry and NSF grant CCF-1909683.

**Acknowledgements** The authors would like to thank the anonymous referees for useful comments that improved the presentation of the results.

## 1 Introduction

Polynomial identity testing (PIT) problem is a fundamental problem in the area of algebraic circuit complexity. PIT is the problem of deciding whether a given multivariate polynomial is identically zero, where the input is given as an algebraic formula, circuit or other computational models like algebraic branching program. One way of testing zeroness of a polynomial is to check whether the coefficients of all the monomials are zero. However, the polynomial computed by a circuit or a branching program may have, in the worst-case, an exponential number of monomials compared to its size. Hence, by computing the explicit polynomial from the input, we cannot solve PIT problem in polynomial time. However, evaluating the polynomial at a point can be done in polynomial time of the input size. This helps us to



© Vishwas Bhargava and Sumanta Ghosh;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 30; pp. 30:1–30:23



Leibniz International Proceedings in Informatics

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

get a polynomial time randomized algorithm for PIT by evaluating the input circuit at a random point, since any nonzero polynomial evaluated at a random point gives a nonzero value with high probability [10, 57, 49]. However, finding a deterministic polynomial time algorithm for PIT is a long-standing open question in algebraic complexity theory.

PIT captures several problems in algebra and combinatorics. For example, parallel algorithms for perfect matching [55, 35, 14, 54], primality testing [2], multivariate polynomial factorization [31], and many other problems [50, 11, 22]. PIT also has strong connection to circuit lower bounds [25, 26, 13, 7, 21]. See [45, 53, 48] for surveys on PIT.

PIT problem is studied in two different settings: 1) *whitebox*, where we are allowed to access the internal structure of the circuit, and 2) *blackbox*, where only evaluation of the circuit at points is allowed. Deterministic blackbox PIT for an  $n$ -variate circuit class is equivalent to efficiently finding a set of points  $\mathcal{H} \subseteq \mathbb{F}^n$ , called a *hitting-set*, such that for any nonzero  $P$  in that circuit class, the set  $\mathcal{H}$  contains a point at which  $P \neq 0$ <sup>1</sup>. In this work, we only focus on the blackbox model.

Despite a lot of effort, little progress has been made on the PIT problem in general. However, efficient deterministic PIT algorithms are known for many special circuit models. For example, blackbox PIT for depth-2 circuits (or sparse polynomials) [6, 30, 34], PIT algorithms for depth-3 circuits with bounded top fan-in [12, 29, 28, 27, 46, 47, 48], depth-3 diagonal circuits [44, 17, 16] and various other subclasses of depth-3 circuits [42, 1, 9], PIT for the subclasses of depth-4 circuits [3, 5, 15, 32, 40] and certain types of symbolic determinants [14, 54, 24].

The focus of this work is on the model of *read-once oblivious algebraic branching programs (ROABPs)*. An ROABP is a product of matrices

$$f = \mathbf{a}^T \cdot M_1(x_{\pi(1)})M_2(x_{\pi(2)}) \cdots M_n(x_{\pi(n)}) \cdot \mathbf{c},$$

where  $\mathbf{a}, \mathbf{c} \in \mathbb{F}^{w \times 1}$  and for some permutation  $\pi$  on  $[n]$  for each  $i \in [n]$ ,  $M_i(x_{\pi(i)}) \in \mathbb{F}^{w \times w}[x_{\pi(i)}]$  can be viewed as a polynomial over the matrix algebra. The permutation  $\pi$  is called the variable order of the ROABP. One reason to be interested in ROABP is that derandomizing blackbox PIT for ROABP can be viewed as an algebraic analogue of the RL vs. L question. Besides that, the ROABP model is surprisingly rich and powerful. It captures several other interesting circuit classes such as sparse polynomials or depth-two circuits, depth-three powering circuits (symmetric tensors), set-multilinear depth-three circuits (tensors), and semi-diagonal depth-3 circuits [19]. Some notable polynomials such as the iterated matrix multiplication polynomial, the elementary and the power symmetric polynomials, and the sum-product polynomials can be computed by linear size ROABPs. Hitting sets for ROABPs have also led to the derandomization of an interesting case of the Noether Normalization Lemma [38, 18], and to hitting sets for non-commutative algebraic branching programs [19].

PIT question for ROABPs and its variants has been widely studied. There are three parameters associated with an ROABP: the number of variables  $n$ , the size of the matrices  $w$  called *width* and the individual degree  $d$  which is the maximum possible degree of any variable. First, [41] gave a polynomial time whitebox PIT algorithm for this model. [19] first gave  $(ndw)^{O(\log n)}$  size hitting set for ROABPs when the variable order is known. Later, [17] gave an  $(ndw)^{O(d \log w \cdot \log n)}$  size hitting set for ROABPs with unknown variable order, and subsequently, [1] gave an improved hitting set of size  $(ndw)^{O(\log n)}$  for this model. For zero or large characteristic fields, [22] gave an  $ndw^{\log n}$  size hitting sets for the known order ROABPs

---

<sup>1</sup> When  $\mathbb{F}$  is a finite field, we are allowed to go some suitable extension  $\mathbb{K}$  of  $\mathbb{F}$  and pick points from  $\mathbb{K}^n$ .

and the size becomes polynomially large when the width is constant. Better hitting set is known for a special class of ROABPs, called *any-order* ROABP. A polynomial  $f$  is computable by a  $w$ -width *any-order* ROABP, if for every permutation  $\pi$  on  $[n]$ ,  $f$  is computable by a  $w$ -width ROABP. The notion of *any-order* ROABP subsumes the notion of commutative ROABP. An ROABP is called commutative ROABP if the polynomial computed by it remains unchanged under any permutation of the matrices involved in the product. [17] gave two different constructions of hitting sets of size  $(ndw)^{O(\log w)}$  and  $d^{O(\log w)} \cdot (nw)^{O(\log \log w)}$  for *any-order* ROABPs<sup>2</sup>. Later, [22] gives an improved hitting set of size  $(ndw)^{O(\log \log w)}$  for this model. Recently, [20] gives improved hitting sets for both ROABPs and *any-order* ROABPs. Compared to the previous constructions, the size of hitting sets in [20] have finer dependence on the parameters of ROABPs. However, the construction of polynomial size hitting sets for ROABPs and its variants is still open.

In this work, we study the PIT question for the orbit of ROABPs. The *orbit* of an  $n$ -variate polynomial  $f(\mathbf{x})$  over a field  $\mathbb{F}$ , denoted by  $\text{orbit}(f)$ , is the set of polynomials obtained by applying invertible affine transformations on the variables of  $f$ , that is,  $\text{orbit}(f) = \{f(A\mathbf{x} + \mathbf{b}) \mid A \in \text{GL}(n, \mathbb{F}), \text{ and } \mathbf{b} \in \mathbb{F}^n\}$ . The orbit of a polynomial class  $\mathcal{C}$ , denoted by  $\text{orbit}(\mathcal{C})$ , is the union of the orbits of the polynomials in the class. Apart from being a natural question to study the sturdiness of the known techniques (and improving them), designing hitting sets for the orbits of polynomial families and circuit classes is interesting for the following reasons:

- As observed by [43], the affine projections of “simple” polynomials have great expressive power. The set of affine projections of an  $n$ -variate polynomial  $f(\mathbf{x})$  over a field  $\mathbb{F}$  is  $\text{aproj}(f) := \{f(A\mathbf{x} + \mathbf{b}) \mid A \in \mathbb{F}^{n \times n} \text{ and } \mathbf{b} \in \mathbb{F}^n\}$ . Formally, they show that if the characteristic of  $\mathbb{F}$  is zero, the set of affine projections of an  $n$ -variate polynomial  $f(\mathbf{x})$  over a field  $\mathbb{F}$  lies inside the Zariski closure of the orbit of  $f$  (denoted by  $\overline{\text{orbit}(f)}$ ), that is  $\text{aproj}(f) \subseteq \overline{\text{orbit}(f)}$ . This observation has some interesting implications. For instance, using the above observation one can show that, the entire class of depth-3 circuits  $\Sigma\Pi\Sigma$  with top fan-in  $s$  and degree  $d$  is contained in  $\text{aproj}(\text{SP}_{s,d})$ , where  $\text{SP}_{s,d} := \sum_{i \in [s]} \prod_{j \in [d]} x_{i,j}$  is a very structured  $s$ -sparse polynomial. The orbit closure of ROABPs is also very powerful, in fact they are as powerful as general ABPs. This can be seen by observing, the iterated matrix multiplication polynomial  $\text{IMM}_{w,d}$  is computable by a linear-size ROABP, yet every polynomial computable by a size- $s$  general algebraic branching program is in  $\text{aproj}(\text{IMM}_{s,s})$ . For more polynomial families whose orbit closures contain interesting circuit classes, see [36].
- For an  $n$ -variate polynomial  $f$  over a field  $\mathbb{F}$ , let  $\mathbb{V}(f)$  denotes the variety (that is, zero locus) of  $f$ . Hitting set construction for an  $n$ -variate polynomial class  $\mathcal{C}$  is the problem of picking a set of points  $\mathcal{H}$  such that for each polynomial  $f \in \mathcal{C}$ ,  $\mathcal{H}$  is not entirely contained in  $\mathbb{V}(f)$ . On the other hand, Constructing hitting sets for the orbits of a polynomial class  $\mathcal{C}$  is the task of finding a small set of points  $\mathcal{H}$  such that for every  $f \in \mathcal{C}$ ,  $\mathcal{H}$  is not entirely contained in the set  $\{A\mathbf{a} + \mathbf{b} \mid \mathbf{a} \in \mathbb{V}(f), A \in \text{GL}(n, \mathbb{F}) \text{ and } \mathbf{b} \in \mathbb{F}^n\}$ . This ensures that  $\mathcal{H}$  will be independent to the choice of coordinate system, making it mathematically and geometrically robust.

For a more detailed discussion on the reasons for studying hitting set of orbits, see [43].

Hitting set construction for orbits of circuit classes is very recent, somewhat simultaneously Medini and Shpilka [36] and Saha and Thankey [43] started exploring PIT for the orbit of various polynomial classes. Medini and Shpilka [36] gave a quasi-polynomial size hitting

<sup>2</sup> In [17], *any-order* ROABPs are referred by “commutative ROABPs”.

set for the orbits of sparse polynomials ( $\sum \prod$  circuits) and read-once formulas (ROFs). Saha and Thankey [43] gave hitting sets for the orbits of ROABPs and constant-read (more generally, constant-occur) formulas. Concretely, [43] gave an  $(ndw)^{d \log w}$  size hitting set for the orbit of  $n$ -variate individual degree  $d$  width  $w$  commutative ROABPs. They also gave an  $(nw)^{O(w^6 \log n)}$  size hitting set for the orbit of  $n$ -variate multilinear polynomials computed by width  $w$  ROABPs. Building on this, they also gave quasi-polynomial size hitting set for constant-depth constant-occur formulas whose leaves are labeled by  $s$ -sparse polynomials with constant individual degree. In this work, we design hitting sets for the orbit of ROABPs and *any-order* ROABPs. Our results significantly improve the dependence on individual degree in the size of hitting sets in comparison to [43], from exponential to polynomial.

## 1.1 Our Results

First, we define the models studied in this paper. Algebraic branching programs (ABPs) were defined by Nisan in [39]. In this paper, we study a variant of ABPs known as read-once oblivious ABPs (ROABPs). While Nisan defined ABPs using directed graphs, we use a more conventional definition using product of matrices. Let  $f(x_1, \dots, x_n)$  be an  $n$ -variate individual degree  $d$  polynomial over a field  $\mathbb{F}$ . Let  $\pi$  be a permutation on  $[n]$ . We say  $f$  is computed by a width  $w$  ROABP with variable order  $\pi$ , if  $f$  can be written as

$$f = \mathbf{a}^T \cdot M_1(x_{\pi(1)})M_2(x_{\pi(2)}) \cdots M_n(x_{\pi(n)}) \cdot \mathbf{c},$$

where  $\mathbf{a}, \mathbf{c} \in \mathbb{F}^{w \times 1}$  and for all  $i \in [n]$ ,  $M_i(x_{\pi(i)}) \in \mathbb{F}^{w \times w}[x_{\pi(i)}]$  can be viewed as a polynomial in  $x_{\pi(i)}$  over the matrix algebra with degree at most  $d$ . We say  $f$  is computable by a  $w$ -width *any order* ROABP, if for every permutation  $\pi$  on  $[n]$ ,  $f$  is computable by a width  $w$  ROABP. We say  $f$  is computed by a width  $w$  *commutative* ROABP, if all  $M_i(x_{\pi(i)})$ 's are polynomials over a commutative sub-algebra of the matrix algebra. For example, consider the coefficients of each  $M_i$  are diagonal matrices. One can observe that the set of polynomials computed by  $w$ -width commutative ROABPs are also computable by  $w$ -width *any-order* ROABPs. However, the converse direction is unknown to us. All PIT algorithms for ROABPs are designed by analyzing the coefficient space of  $M_1(x_{\pi(1)})M_2(x_{\pi(2)}) \cdots M_n(x_{\pi(n)})$ .

In this paper, we design hitting sets for the orbits of ROABPs and *any-order* ROABPs. Let  $f(\mathbf{x})$  be an  $n$ -variate polynomial over a field  $\mathbb{F}$ . The orbit of  $f$ , denoted by  $\text{orbit}(f)$ , is the set  $\{f(A\mathbf{x} + \mathbf{b}) \mid A \in \text{GL}(n, \mathbb{F}) \text{ and } \mathbf{b} \in \mathbb{F}^n\}$ . For a polynomial class  $\mathcal{C}$ , the orbit of  $\mathcal{C}$ , denoted by  $\text{orbit}(\mathcal{C})$ , is the union of orbits of all the polynomials in  $\mathcal{C}$ . Now, we describe our result for the orbit of *any-order* ROABPs.

► **Theorem 1.** *Let  $\mathbb{F}$  be a field of characteristic zero or greater than  $d$ . Let  $\mathcal{C}$  be the set of  $n$ -variate polynomials over  $\mathbb{F}$  with individual degree at most  $d$  and computable by a width  $w$  *any-order* ROABP. Then, there exists a hitting set for  $\text{orbit}(\mathcal{C})$  computable in time  $(ndw)^{O(\ell)}$  where  $\ell = \min\{w^2, 2d \log w\}$ .*

### Comparison with previous works

As far as we know, this is the first result addressing the orbit of *any-order* ROABPs, and it subsumes the commutative ROABP result of Saha and Thankey [43]. They gave an  $(ndw)^{O(d \log w)}$  size hitting set for the orbit of commutative ROABPs. In fact, our result strengthens [43] in “low width” setting. Concretely, if the individual degree is  $\text{poly}(\log n)$ , [43] gives quasi-polynomial time PIT for the orbit of commutative ROABPs. However, when  $d \geq n$ , their algorithm does not give any non-trivial PIT for the orbit of commutative

ROABPs. On the other hand, our result gives quasi-polynomial time PIT for the orbit of *any-order* ROABPs when  $\min\{d, w\} = \text{poly}(\log n)$ . Also, for constant width *any-order* ROABPs with unbounded individual degree, our result gives a polynomial time PIT for its orbit. However, [43] gives polynomial time PIT for the orbit commutative ROABPs when both  $d$  and  $w$  are constants. Thus, our result has much better dependence on the individual degree in comparison with [43].

Now, we describe our result regarding the orbit of ROABPs.

► **Theorem 2.** *Let  $\mathbb{F}$  be a field of characteristic zero or greater than  $d$ . Let  $\mathcal{C}$  be the set of  $n$ -variate polynomials over  $\mathbb{F}$  with individual degree at most  $d$  and computable by a width  $w$  ROABP. Then there exists a hitting set for  $\text{orbit}(\mathcal{C})$  computable in time  $(ndw)^{O(\ell)}$  where  $\ell = (w^2 \log n) \cdot \min\{w^2, 2d \log w\}$ .*

### Comparison with previous works

Saha and Thankey [43] gave an  $(nw)^{O(w^6 \log n)}$  time PIT for the orbit of multilinear polynomials computed by ROABPs. Therefore, our result can be seen as the first one which gives PIT for the orbit of ROABPs with unbounded individual degree. Irrespective of the value of the individual degree, our result gives a quasi-polynomial time PIT for the orbit of ROABPs when the width  $w = \text{poly}(\log n)$ . Also, the time complexity of our algorithm has better dependence on the width of ROABPs in comparison with [43].

### Remark

Our results in this paper continue to hold even if we consider a more generalized definition for the orbit of an  $n$ -variate polynomial  $f(\mathbf{x})$ , that is  $\text{orbit}(f) = \{f(A\mathbf{y} + \mathbf{b}) \mid m \geq n, A \in \mathbb{F}^{n \times m} \text{ with rank } n \text{ and } \mathbf{b} \in \mathbb{F}^n\}$  where  $\mathbf{y} = (y_1, \dots, y_m)$ . However, we work with the conventional definition of the orbit of polynomials for the simplicity of exposition, and because the proofs of the results with the generalized definition of orbit is almost the same as the proofs given in this paper.

## 1.2 Proof techniques

First, we briefly sketch the abstract framework followed by the proofs of our results. Let  $\mathcal{C}$  be a set of  $n$ -variate polynomials in  $\mathbf{y} = (y_1, \dots, y_n)$  with individual degree at most  $d$ . Then  $\text{orbit}(\mathcal{C})$  is the set of  $n$ -variate polynomials in  $\mathbf{x} = (x_1, \dots, x_n)$  is defined as follows: for all  $f(\mathbf{x}) \in \text{orbit}(\mathcal{C})$  there exists a polynomial  $h(\mathbf{y}) \in \mathcal{C}$ , an invertible linear transformation  $L(\mathbf{x}) = (\ell_1, \dots, \ell_n)$  from  $\mathbb{F}^n$  to  $\mathbb{F}^n$  and a point  $\mathbf{b} \in \mathbb{F}^n$  such that

$$f(\mathbf{x}) = h(L(\mathbf{x}) + \mathbf{b}).$$

In this paper, we design hitting sets for the orbits of ROABPs and *any-order* ROABPs. Hitting sets for ROABPs are constructed by designing a “smartly” chosen shift  $\mathbf{g}(\mathbf{t})$  (a low variate polynomial map) such that when we shift any polynomial  $h(\mathbf{y})$  computable by a small size ROABP, then there exists a “low-support” monomial (with nonzero coefficient) in  $h(\mathbf{x} + \mathbf{g})$ . Note that, it is straightforward to construct hitting sets when such a low-support monomial (with nonzero coefficient) exists. However, this approach does not *directly* work for a polynomial  $f(\mathbf{x}) = h(L(\mathbf{x}) + \mathbf{b})$  in the orbit of ROABPs as shifting  $f$  has a slightly different effect. Note,

$$f(\mathbf{x} + \mathbf{g}) = h(L(\mathbf{x} + \mathbf{g}) + \mathbf{b}) = h(L(\mathbf{x}) + L \circ \mathbf{g} + \mathbf{b}).$$

That is, the shift gets composed with the affine transformation  $L(\mathbf{x}) + \mathbf{b}$ . The main idea in our construction is to choose a shift such that the transformed shift (for *any* affine transformation) is also “smart”. That is, for any invertible linear transformation  $L(\mathbf{x})$  and  $\mathbf{b} \in \mathbb{F}^n$ , there exists a “low-support” monomial (with nonzero coefficient) in  $f(\mathbf{x} + \mathbf{g}) = h(L(\mathbf{x}) + L \circ \mathbf{g} + \mathbf{b})$ .

Let  $\mathbf{g}(\mathbf{t}) = (g_1, \dots, g_n)$  be a polynomial map from  $\mathbb{F}^m$  to  $\mathbb{F}^n$  and  $h'(\mathbf{y}) = h(\mathbf{y} + L \circ \mathbf{g} + \mathbf{b})$ . Note that,  $f'(\mathbf{x}) := f(\mathbf{x} + \mathbf{g}) = h'(L(\mathbf{x}))$ . Our abstract format to design hitting sets for the orbits of ROABPs and *any-order* ROABPs has the following two steps.

**Step 1:** First we find some suitable low degree polynomial map  $\mathbf{g}$  in few variables (compare to  $n$ ) such that for *all* invertible linear transformation  $L(\mathbf{x})$  and  $\mathbf{b} \in \mathbb{F}^n$ , after shifting  $h(\mathbf{y}) \in \mathcal{C}$  by  $L \circ \mathbf{g} + \mathbf{b}$ , the new polynomial  $h'(\mathbf{y}) = h(\mathbf{y} + L \circ \mathbf{g} + \mathbf{b})$  has the following property: for some small positive integer  $k$ ,  $\text{hom}_{\leq k}(h'(\mathbf{y}))$  is a nonzero polynomial in  $\mathbf{y}$  over the field  $\mathbb{F}(\mathbf{t})$ , where  $\text{hom}_{\leq k}(\cdot)$  denotes the degree up to  $k$  part of the input polynomial. This step, more specifically the construction of  $\mathbf{g}(\mathbf{t})$ , heavily relies on the structure of  $\mathcal{C}$ .

**Step 2:** Since  $L(\mathbf{x})$  is an invertible linear transformation, all  $\ell_i$ 's are algebraically independent. Also,  $\text{hom}_{\leq k}(f') = \text{hom}_{\leq k}(h')(L(\mathbf{x}))$ . Therefore,  $\text{hom}_{\leq k}(f')$  is a nonzero polynomial in  $\mathbf{x}$  over the field  $\mathbb{F}(\mathbf{t})$ . This implies that there exists a monomial  $\mathbf{x}^{\mathbf{e}} = \prod_{i=1}^n x_i^{e_i}$  such that the support of  $\mathbf{e}$  is at most  $k$  and the coefficient of  $\mathbf{x}^{\mathbf{e}}$  in  $f'$  is a nonzero polynomial in  $\mathbf{t}$ . There are well known constructions of hitting sets for polynomials like  $f'(\mathbf{x})$ . For example, combining Lemma 23 and Observation 17 we get a hitting set for  $f'$  of size around  $(nd)^{O(m+k)}$ . Thus, we design a hitting set for orbit( $\mathcal{C}$ ). This step is independent of the polynomial class  $\mathcal{C}$ .

For instance, assume that  $\mathcal{C}$  is the set of  $n$ -variate polynomials with individual degree and sparsity are at most  $d$  and  $s$ , respectively. Then, from [15], after shifting any polynomial  $h(\mathbf{y}) \in \mathcal{C}$  by an  $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)$  with all  $\alpha_i$ 's are nonzero the following holds: there exists a monomial  $\mathbf{y}^{\mathbf{e}}$  such that the support of  $\mathbf{e}$  is at most  $\log s$  and its coefficient in  $h(\mathbf{y} + \boldsymbol{\alpha})$  is nonzero. Let  $\mathbf{g}(t)$  be the polynomial map from  $\mathbb{F}$  to  $\mathbb{F}^n$  defined as  $(t, t^2, \dots, t^n)$  and  $\mathbf{b} = (b_1, \dots, b_n)$ . Then, each  $\ell_i(\mathbf{g}) + b_i$  is a nonzero polynomial. Therefore, there exists a monomial  $\mathbf{y}^{\mathbf{e}}$  of support-size at most  $\log s$  such that its coefficient in  $h'(\mathbf{y})$  is a nonzero polynomial in  $t$ . Since the individual degree is at most  $d$ , the degree of  $\mathbf{y}^{\mathbf{e}}$  is at most  $\leq d \log s$ . Now from the step 2, there exists a monomial in  $\mathbf{x}$  of support-size at most  $d \log s$  such that its coefficient in  $f'$  is a nonzero polynomial in  $t$ . Thus, we have a hitting set for orbit( $\mathcal{C}$ ) of size  $(nd)^{O(d \log s)}$ . This gives a different (and much simpler) hitting set construction than [43, Theorem 7] for the orbit of sparse polynomials with low individual degree.

### Stronger rank concentration results

We describe some stronger rank concentration results, which will be very useful in designing our hitting sets for the orbits of ROABPs and *any-order* ROABPs. Let  $G(\mathbf{x})$  be an  $n$ -variate polynomial over the vector space  $\mathbb{F}^k$ . The *coefficient space* of  $G$  is the vector space spanned by the coefficients (from  $\mathbb{F}^k$ ) in  $G$ . In general, the coefficient space of  $G$  can be spanned by the coefficients of any arbitrary set of monomials. In rank concentration, our goal is to construct a polynomial map  $\mathbf{g}(\mathbf{t})$  such that after shifting  $G(\mathbf{x})$  by  $\mathbf{g}(\mathbf{t})$ , the coefficient space of the new polynomial  $G'(\mathbf{x}) = G(\mathbf{x} + \mathbf{g})$  is spanned the coefficients of a “small” set of monomials  $S$ . For example,

1. if  $S$  is the set of monomials whose support-size is  $\leq \ell$ , we say  $G'$  has  $\ell$ -support concentration. The support-size of a monomial is the number of variables appearing in it.



2. if  $S$  is the set of monomials whose cone-size is  $\leq \ell$ , we say  $G'$  has  $\ell$ -cone concentration. The cone-size of a monomial is the number of monomials dividing it.
3. if  $S$  is the set of monomials which is closed under sub-monomials, we say  $G'$  has a cone-closed basis.

The notion of rank-concentration was introduced in [4]. Subsequently, many PIT results are obtained based on “low-support” rank concentration [4, 17, 23, 22, 43]. Later, [16] introduced the notion of cone concentration and cone-closed basis. Among the three notions of rank concentrations, cone-closed basis is stronger than the other two, then comes cone concentration and after that support concentration. More specifically, cone-closed basis of  $G'$  implies that it has also  $k$ -cone concentration, and  $k$ -concentration for  $G'$  implies it has also  $\log k$ -support concentration. For more details about the relation between these three notions of rank concentrations see Lemma 26. The notion of cone concentration is important for designing our improved hitting sets over [43]. Although the notion of cone concentration was first introduced in [16] and they showed some low-cone concentration result, we are not aware of any “non-trivial” application of them in designing PIT algorithms. Therefore, to the best of our knowledge, this is the first time when the notion of cone concentration is used in designing PIT algorithms.

In this work, we strengthen some of the rank concentration results shown in [17, 16]. [17] showed that if  $G(\mathbf{x})$  is shifted by  $\mathbf{t} = (t_1, \dots, t_n)$ , the new polynomial  $G(\mathbf{x} + \mathbf{t})$  has  $\log k$ -support concentration over the field  $\mathbb{F}(\mathbf{t})$ . Moreover, they showed that if  $G$  is shifted by a  $n$ -wise independent monomial map  $\mathbf{g}'(\mathbf{s}, \mathbf{t})$ , then the new shifted polynomial has  $\log k$ -support concentration. A polynomial map  $\mathbf{g}'(\mathbf{s}, \mathbf{t})$  from  $\mathbb{F}^m \times \mathbb{F}^{m'}$  to  $\mathbb{F}^n$  is called  $\ell$ -wise independent monomial map if for every  $S \subseteq [n]$  of size  $\leq \ell$  there exists an  $\boldsymbol{\alpha} \in \mathbb{F}^m$  such that polynomials  $\{\mathbf{g}'(\boldsymbol{\alpha}, \mathbf{t})^e\}_{\text{supp}(\mathbf{e}) \subseteq S}$  are distinct monomials in  $\mathbf{t}$ . Later, [16] showed that  $G(\mathbf{x} + \mathbf{t})$  has a cone-closed basis. Their result can also be extended to show that  $G(\mathbf{x} + \mathbf{g}')$  has a cone-closed basis when  $\mathbf{g}'$  is an  $n$ -wise independent monomial map. However, when we take composition of  $\mathbf{g}'$  with an invertible affine transformation, that is  $\mathbf{b} + L \circ \mathbf{g}'$  where  $\mathbf{b} \in \mathbb{F}^n$  and  $L(\mathbf{x})$  is an invertible linear transformation from  $\mathbb{F}^n$  to  $\mathbb{F}^n$ , the  $n$ -wise independence property of  $\mathbf{g}'$  breaks down. Therefore, the previous rank concentration results are not helpful in designing hitting sets for the orbits of circuit classes. We strengthen the rank concentration results of [17, 16] in the following way: After shifting  $G$  by a polynomial map  $\mathbf{g}' = (g_1, \dots, g_n)$  such that all  $g_i$ 's are *algebraically independent*, the new polynomial has a cone-closed basis, hence  $k$ -cone concentration. Observe that the  $n$ -wise independence property implies the algebraic independence property needed in our hypothesis. Therefore, our hypothesis is weaker than the hypothesis used in [17, 16]. Also, algebraic independence property of  $\mathbf{g}'$  preserves even after composing it with invertible affine transformations. For details see Lemma 4. This rank concentration result will be helpful in designing the hitting sets for the orbit of *any-order* ROABPs.

We show one more rank concentration result which will help in designing PIT algorithms for the orbit of ROABPs. Assume that the coefficients of the monomials of total degree up to  $D$  spans the coefficient space of  $G$ . Let  $\mathbf{g}'(\mathbf{s}, \mathbf{t})$  be a *total degree  $D$  independent monomial map* from  $\mathbb{F}^m \times \mathbb{F}^{m'}$  to  $\mathbb{F}^n$ , that is, there exists an  $\boldsymbol{\alpha} \in \mathbb{F}^m$  such that the polynomials  $\{\mathbf{g}'(\boldsymbol{\alpha}, \mathbf{t})^e\}_{|\mathbf{e}|_1 \leq D}$  are distinct monomials in  $\mathbf{t}$ . Then [17] showed that if  $G(\mathbf{x})$  is shifted by  $u\mathbf{g}'$ , then the new shifted polynomial has  $\log k$ -support concentration over the field  $\mathbb{F}(u, \mathbf{s}, \mathbf{t})$ . Our rank concentration result differs from [17] in the following ways:

1. Our hypothesis is slightly stronger than [17]. Instead of total degree  $D$  independent monomial map, we assume that  $\mathbf{g}'(\mathbf{s}, \mathbf{t})$  is a total degree  $Dk$  independent monomial map.

2. On the other hand, we strengthen the conclusion as follows: for every invertible linear transformation  $L(\mathbf{x})$  from  $\mathbb{F}^n$  to  $\mathbb{F}^n$ , if we shift  $G$  by  $uL \circ \mathbf{g}'$ , then the new shifted polynomial has a cone-closed basis over the field  $\mathbb{F}(u, \mathbf{s}, \mathbf{t})$ .

For details see Lemma 5.

### Proof idea of Theorem 1

Suppose that  $\mathcal{C}$  is the set of all  $n$ -variate polynomials in  $\mathbf{y}$  with individual degree at most  $d$  and computed by width  $w$  *any-order* ROABPs. Let  $f(\mathbf{x})$  be an  $n$ -variate polynomial in  $\text{orbit}(\mathcal{C})$ . Then there exists a polynomial  $h(\mathbf{y}) \in \mathcal{C}$ , an invertible linear transformation  $L(\mathbf{x})$  and a point  $\mathbf{b} \in \mathbb{F}^n$  such that

$$f(\mathbf{x}) = h(L(\mathbf{x}) + \mathbf{b}).$$

Since  $h(\mathbf{y}) \in \mathcal{C}$ , there exists a polynomial  $G(\mathbf{y}) \in \mathbb{F}[\mathbf{y}]^{w \times w}$  with individual degree at most  $d$  and computed by a width  $w$  *any-order* ROABP such that

$$h(\mathbf{y}) = \mathbf{a}^T \cdot G(\mathbf{y}) \cdot \mathbf{c},$$

where  $\mathbf{a}, \mathbf{c} \in \mathbb{F}^w$ .

Now we will describe the first step of aforementioned abstract format. First, we show how to achieve  $w^2$ -cone concentration in  $G(\mathbf{y})$ . Let  $\mathbf{g}(\mathbf{t}) = (g_1, \dots, g_n)$  be a polynomial map from  $\mathbb{F}^m$  to  $\mathbb{F}^n$  such that for any  $S \subseteq [n]$  of size  $k := \lceil 2 \log w + 1 \rceil$ , the set of polynomials  $\{g_i \mid i \in S\}$  are algebraically independent. Then, in Lemma 6, we prove that  $G(\mathbf{y} + \mathbf{g})$  has  $w^2$ -cone concentration over the field  $F(\mathbf{t})$ . It strengthens the rank-concentration result for *any-order* ROABPs shown in [17, Theorem 4.1]. They showed that if we shift  $G$  by a  $k$ -wise independent monomial map, then the new polynomial has  $2 \log w$ -support concentration. Next, in Lemma 7, we show that for any invertible linear transformation  $L(\mathbf{x})$  and  $\mathbf{b} \in \mathbb{F}^n$ , the polynomial map defined as the composition of  $L(\mathbf{x}) + \mathbf{b}$  and Shpilka-Volkovich generator  $\mathcal{G}_{n,k}^{SV}$  (see Definition 21, or [51]), that is  $L \circ \mathcal{G}_{n,k}^{SV} + \mathbf{b}$ , satisfies the property required for achieving  $w^2$ -cone concentration in  $G(\mathbf{y})$ . Therefore,  $G(\mathbf{y} + L \circ \mathcal{G}_{n,k}^{SV} + \mathbf{b})$  has  $w^2$ -cone concentration. This implies that there exists a monomial  $\mathbf{y}^e$  of cone-size  $\leq w^2$  such that the coefficient of  $\mathbf{y}^e$  in  $h'(\mathbf{y}) = h(\mathbf{y} + L \circ \mathcal{G}_{n,k}^{SV} + \mathbf{b})$  is nonzero. For any monomial of cone-size  $\leq w^2$ , its degree is less than  $w^2$  and the support set is of size at most  $2 \log w$ . Since the individual degree is at most  $d$ , the degree of  $\mathbf{y}^e$  is at most  $\ell$  where  $\ell := \min\{w^2, d \log w\}$ . Therefore,  $\text{hom}_{\leq \ell}(h')$  is nonzero. Now we apply the step two of the abstract format, which is independent of  $\mathcal{C}$ , and get our desired hitting set for  $\text{orbit}(\mathcal{C})$ .

### Proof idea of Theorem 2

Suppose that  $\mathcal{C}$  is the set of all  $n$ -variate polynomials in  $\mathbf{y}$  with individual degree at most  $d$  and computed by width  $w$  ROABPs. Let  $f(\mathbf{x})$  be an  $n$ -variate polynomial in  $\text{orbit}(\mathcal{C})$ . Then there exists a polynomial  $h(\mathbf{y}) \in \mathcal{C}$ , an invertible linear transformation  $L(\mathbf{x})$  and  $\mathbf{b} \in \mathbb{F}^n$  such that

$$f(\mathbf{x}) = h(L(\mathbf{x}) + \mathbf{b}).$$

Since  $h(\mathbf{y}) \in \mathcal{C}$ , there exists a polynomial  $G(\mathbf{y}) \in \mathbb{F}[\mathbf{y}]^{w \times w}$  and a permutation  $\pi$  on  $[n]$  such that

$$h(\mathbf{y}) = \mathbf{a}^T \cdot G(\mathbf{y}) \cdot \mathbf{c} \text{ and } G(\mathbf{y}) = \prod_{i=1}^n M_i(x_{\pi(i)})$$

where  $\mathbf{a}, \mathbf{c} \in \mathbb{F}^w$  and for all  $i \in [n]$ ,  $M_i(x_{\pi(i)})$  is a polynomial in  $\mathbb{F}[x_{\pi(i)}]^{w \times w}$ .

Now like *any-order* ROABPs, we want to achieve  $w^2$ -cone concentration in  $G(\mathbf{y})$ . However, our approach here will be different from *any-order* ROABPs. Here, we strengthen the “merge-and-reduce” approach of [17] in the following ways:

1. In [17], the polynomial maps  $\mathbf{h}_j$  (for  $j = 0, 1, \dots, \lceil \log n \rceil$ ) were inductively constructed such that after shifting  $G$  by  $\mathbf{h}_j$ , in the new polynomial  $G(\mathbf{x} + \mathbf{h}_j)$ , the product of any  $2^j$  consecutive matrices have  $2 \log w$ -support concentration. We strengthen this result by showing  $w^2$ -cone concentration at each inductive step.
2. At each induction step, since we are dealing with polynomials in orbit (of ROABPs), we not only need to construct a polynomial map which helps to achieve  $w^2$ -cone concentration, but its composition with any invertible affine transformation also helps to achieve the same property.

In [17],  $\mathbf{h}_j$  was constructed as follows:  $\mathbf{h}_0 = \mathbf{0}$  and for all  $j \in [\lceil \log n \rceil]$ ,  $\mathbf{h}_j = \mathbf{h}_{j-1} + u_j \mathbf{g}(\mathbf{s}_j, \mathbf{t}_j)$  where  $\mathbf{g}(\mathbf{s}_j, \mathbf{t}_j)$  is a total degree  $4d \log w$  independent monomial map from  $\mathbb{F}^m \times \mathbb{F}^{m'}$  to  $\mathbb{F}^n$ . They showed that the product of any  $2^j$  consecutive matrices in  $G(\mathbf{y} + \mathbf{h}_j)$  has  $2 \log w$ -support concentration over the field  $\mathbb{F}((u_k, \mathbf{s}_k, \mathbf{t}_k)_{k \in [j]})$ .

Our definition of  $\mathbf{h}_j$  is very close to the definition used in [17]. For  $j = 0$ ,  $\mathbf{h}_j = (t, t^2, \dots, t^n)$  and for all  $j \in [\lceil \log n \rceil]$ ,  $\mathbf{h}_j = \mathbf{h}_{j-1} + u_j \mathbf{g}(\mathbf{s}_j, \mathbf{t}_j)$  where  $\mathbf{g}(\mathbf{s}_j, \mathbf{t}_j)$  is a total degree  $D$  independent monomial map from  $\mathbb{F}^m \times \mathbb{F}^{m'}$  to  $\mathbb{F}^n$  where  $D = 2w^2 \cdot \min\{w^2, 2d \log w\}$ . We show that for every invertible linear transformation  $L(\mathbf{x})$  from  $\mathbb{F}^n$  to  $\mathbb{F}^n$  and  $\mathbf{b} \in \mathbb{F}^n$ , the product of any  $2^j$  consecutive matrices in  $G(\mathbf{y} + L \circ \mathbf{h}_j + \mathbf{b})$  has a cone-closed basis, hence has  $w^2$ -cone concentration, over the field  $\mathbb{F}(t, (u_k, \mathbf{s}_k, \mathbf{t}_k)_{k \in [j]})$ . Our rank concentration results play an important role in proving this property of  $\mathbf{h}_j$ . For more details see Lemma 9 and 10. There are many known constructions of total degree  $D$  independent monomial map with  $m = m' = O(D)$ . For example see Lemma 20. After constructing a polynomial map which gives  $w^2$ -cone concentration in  $G(\mathbf{y})$ , the rest of the proof will be similar to what we did for the *any-order* ROABP case.

## Notations

By  $\mathbb{N}$  we denote the set of natural numbers. For any positive integer  $n$ ,  $[n]$  denotes the set  $\{1, 2, \dots, n\}$ . For a variable tuple  $\mathbf{x} = (x_1, \dots, x_n)$  and a tuple  $\mathbf{e} = (e_1, \dots, e_n) \in \mathbb{N}^n$ ,  $\mathbf{x}^{\mathbf{e}}$  denotes the monomial  $\prod_{i=1}^n x_i^{e_i}$ . The *degree*, or *total degree*, of  $\mathbf{x}^{\mathbf{e}}$  is  $|\mathbf{e}|_1 = \sum_{i=1}^n e_i$  and the *individual degree* of  $\mathbf{x}^{\mathbf{e}}$  is  $|\mathbf{e}|_\infty = \max_{i \in [n]} e_i$ . The *support* of  $\mathbf{x}^{\mathbf{e}}$  is the subset  $S$  of  $[n]$  such that  $i \in S$  if and only if  $e_i > 0$ , and the *support-size* denotes the cardinality of  $S$ . The *cone* of  $\mathbf{x}^{\mathbf{e}}$  is the set of monomials which divide it and the *cone-size* is the cardinality of that set, that is  $\prod_{i=1}^n (e_i + 1)$ . A monomial  $\mathbf{x}^{\mathbf{f}}$  is called a *sub-monomial* of  $\mathbf{x}^{\mathbf{e}}$ , if  $\mathbf{x}^{\mathbf{e}}$  divides  $\mathbf{x}^{\mathbf{f}}$ , that is  $e_i \leq f_i$  for all  $i \in [n]$ . A set of monomials  $B$  is called *cone-closed* if for every monomial in  $B$  all its sub-monomials are also in  $B$ . For a polynomial  $f$  in  $\mathbf{x}$  and a monomial  $\mathbf{x}^{\mathbf{e}}$ ,  $\text{coef}_f(\mathbf{x}^{\mathbf{e}})$  denotes the coefficient of  $\mathbf{x}^{\mathbf{e}}$  in  $f$ .

## 2 Achieving Cone-closed basis by shift

In this section, we show our rank concentration results for polynomials over the vector space  $\mathbb{F}^k$ . By  $M_{n,d}$ , we denote the set of  $n$ -variate monomials with individual degree at most  $d$ . We also use  $M_{n,d}$  to denote the exponent vectors for those monomials since there is one-to-one correspondence between monomials and their exponent vectors. For any  $\mathbf{a}, \mathbf{b} \in \mathbb{N}^n$  with  $\mathbf{a} = (a_1, \dots, a_n)$  and  $\mathbf{b} = (b_1, \dots, b_n)$ ,  $\binom{\mathbf{a}}{\mathbf{b}}$  denotes  $\prod_{i=1}^n \binom{a_i}{b_i}$ .

### 30:10 Improved Hitting Set for Orbit of ROABPs

Let  $G(\mathbf{x})$  be an  $n$ -variate polynomial over  $\mathbb{F}^k$  with individual degree at most  $d$ . After shifting  $G(\mathbf{x})$  by  $\mathbf{z}$ , the coefficients of the shifted polynomial  $G'(\mathbf{x}) = G(\mathbf{x} + \mathbf{z})$  can be written as follows: for all  $\mathbf{e} \in M_{n,d}$ ,

$$\text{coef}_{\mathbf{x}^{\mathbf{e}}}(G') = \sum_{\mathbf{f} \in M_{n,d}} \binom{\mathbf{f}}{\mathbf{e}} \text{coef}_{\mathbf{x}^{\mathbf{f}}}(G) \mathbf{z}^{\mathbf{f}-\mathbf{e}}.$$

The above equation can be written in matrix form as follows:

$$F'(\mathbf{z}) = W^{-1}(\mathbf{z})TW(\mathbf{z})F, \quad (1)$$

where

- $F$  and  $F'(\mathbf{z})$  are the matrices with entries from  $\mathbb{F}$  and  $\mathbb{F}[\mathbf{z}]$ , respectively. The rows of both the matrices are indexed by the elements of  $M_{n,d}$ , and for any monomial  $\mathbf{e} \in M_{n,d}$ , the rows indexed by  $\mathbf{e}$  in  $F$  and  $F'$  are  $\text{coef}_{\mathbf{x}^{\mathbf{e}}}(G)$  and  $\text{coef}_{\mathbf{x}^{\mathbf{e}}}(G')$ , respectively.
- $W(\mathbf{z})$  be the diagonal matrix whose rows and columns are indexed by the elements of  $M_{n,d}$  and for all  $\mathbf{e} \in M_{n,d}$ ,  $W(\mathbf{z})_{\mathbf{e},\mathbf{e}} = \mathbf{z}^{\mathbf{e}}$ .
- $T$  is a square matrix such that the rows and columns are indexed by  $M_{n,d}$  and for all  $\mathbf{e}, \mathbf{f} \in M_{n,d}$ ,  $T_{\mathbf{e},\mathbf{f}} = \binom{\mathbf{f}}{\mathbf{e}}$ . In the literature,  $T$  is known as *transfer matrix*.

In the following lemma, we recall a property of transfer matrix from [16].

► **Lemma 3** (Lemma 17 [16]). *Let  $\mathbb{F}$  be a field of characteristic 0 or greater than  $d$ . Then, for every  $B \subseteq M_{n,d}$ , there exists a cone-closed set  $A \subseteq M_{n,d}$  with  $|A| = |B|$  such that  $T_{A,B}$  is full rank over  $\mathbb{F}$ .*

Next, we show our first rank concentration result. Informally, we prove that if  $G(\mathbf{x})$  is shifted by algebraically independent polynomials, the new polynomial has a cone-closed basis.

► **Lemma 4.** *Let  $\mathbb{F}$  be a field of characteristic 0 or greater than  $d$ . Let  $G(\mathbf{x}) \in \mathbb{F}^k[\mathbf{x}]$  be an  $n$ -variate polynomial with individual degree at most  $d$ . Let  $\mathbf{g}(\mathbf{z}) = (g_1, \dots, g_n)$  be a polynomial map from  $\mathbb{F}^n$  to  $\mathbb{F}^n$  such that all  $g_i$ 's are algebraically independent. Then  $G(\mathbf{x} + \mathbf{g})$  has a cone-closed basis over  $\mathbb{F}(\mathbf{z})$ .*

**Proof.** First we show that  $G'(\mathbf{x}) = G(\mathbf{x} + \mathbf{z})$  has a cone-closed basis over  $\mathbb{F}(\mathbf{z})$ . This part of our proof closely follows the proof outline of [16, Theorem 2]. From Equation 1, we know that the shifted polynomial  $G(\mathbf{x} + \mathbf{z})$  yields the following matrix equation:

$$F'(\mathbf{z}) = W(\mathbf{z})^{-1}TW(\mathbf{z})F.$$

Let  $k'$  be the rank of the matrix  $F$ . Then we divide our proof in two cases:

**Case 1 ( $k' < k$ ).** We reduce this case to the other one where  $k' = k$ . Since the rank of  $F$  is  $k'$ , there exists a  $S \subseteq [k]$  of size  $k'$  such that  $F_{M,S}$  is full rank where  $M = M_{n,d}$ . Let  $G_S(\mathbf{x})$  and  $G'_S(\mathbf{x})$  be the projections of  $G(\mathbf{x})$  and  $G'(\mathbf{x})$  on the coordinates indexed by  $S$ . Then  $G'_S(\mathbf{x}) = G_S(\mathbf{x} + \mathbf{z})$ . One can observe that for any set of monomials  $A$ , if their coefficients in  $G_S(\mathbf{x})$  forms a basis for its coefficient space, then their coefficients in  $G(\mathbf{x})$  also forms a basis for the coefficients space of  $G(\mathbf{x})$ . Similarly, this is also true between  $G'_S(\mathbf{x})$  and  $G'(\mathbf{x})$ . Now from the case 2,  $G'_S(\mathbf{x})$  has a cone-closed basis over  $\mathbb{F}(\mathbf{z})$ , that is, there exists a cone-closed set of monomials  $A$  such that their coefficients in  $G'_S(\mathbf{x})$  forms a basis for its coefficient space. This implies that  $G'(\mathbf{x})$  also has a cone-closed basis over  $\mathbb{F}(\mathbf{z})$ .

**Case 2 ( $k' = k$ ).** The rows of  $F$  are indexed by the monomials in  $M_{n,d}$ . Fix a monomial ordering  $\prec$  on the monomials in  $\mathbf{z}$ . For example, assume  $\prec$  is the lexicographic monomial ordering. Then, from Lemma 13, we have a unique subset  $B$  of  $M_{n,d}$  with the following properties:  $\text{rank}(F_{B,[k]}) = k$ , and for every other subset  $C$  of  $M_{n,d}$  with  $\text{rank}(F_{C,[k]}) = k$ ,

$$\prod_{\mathbf{e} \in B} \mathbf{z}^{\mathbf{e}} \prec \prod_{\mathbf{e}' \in C} \mathbf{z}^{\mathbf{e}'}$$

Using Lemma 3, we have a cone-closed subset  $A$  of  $M_{n,d}$  such that  $T_{A,B}$  has full rank. Now

$$\det(F'(\mathbf{z})_{A,[k]}) = \det(W(\mathbf{z})_{A,A})^{-1} \cdot \det((TW(\mathbf{z})F)_{A,[k]}). \tag{2}$$

Applying Lemma 14, we get that

$$\det((TW(\mathbf{z})F)_{A,[k]}) = \sum_{C \in \binom{M_{n,d}}{k}} \det(T_{A,C}) \det(F_{C,[k]}) \prod_{\mathbf{e} \in C} \mathbf{z}^{\mathbf{e}}. \tag{3}$$

For every  $C \in \binom{M_{n,d}}{k} \setminus \{B\}$  such that  $F_{C,[k]}$  is a full rank matrix, the following holds:  $\prod_{\mathbf{e} \in B} \mathbf{z}^{\mathbf{e}} \prec \prod_{\mathbf{e}' \in C} \mathbf{z}^{\mathbf{e}'}$ . Therefore, the coefficient of  $\prod_{\mathbf{e} \in B} \mathbf{z}^{\mathbf{e}}$  in the above polynomial does not get cancelled by other monomials. Also, the coefficient of  $\prod_{\mathbf{e} \in B} \mathbf{z}^{\mathbf{e}}$ ,  $\det(T_{A,B}) \det(F_{B,[k]}) \neq 0$ . Therefore, the polynomial  $\det((TW(\mathbf{z})F)_{A,[k]})$  is a nonzero polynomial in  $\mathbf{z}$ . Also,  $\det(W(\mathbf{z})_{A,A})^{-1}$  is a nonzero element in  $\mathbb{F}(\mathbf{z})$  since  $\det(W(\mathbf{z})_{A,A})$  is a nonzero polynomial in  $\mathbf{z}$ . Therefore,  $\det(F'(\mathbf{z})_{A,[k]})$  is nonzero in  $\mathbb{F}(\mathbf{z})$ . This implies that  $G'(\mathbf{x}) = G(\mathbf{x} + \mathbf{z})$  has a cone-closed basis over  $\mathbb{F}(\mathbf{z})$ .

Now we show that  $G(\mathbf{x} + \mathbf{g})$  has a cone-closed basis over  $\mathbb{F}(\mathbf{z})$ . In Equation 2, since both  $\det(W(\mathbf{z})_{A,A})$  and  $\det((TW(\mathbf{z})F)_{A,[k]})$  are nonzero polynomials in  $\mathbf{z}$ . Therefore, after evaluating them on any  $n$  algebraically independent polynomials, they will remain nonzero. Thus,  $\det(F'(\mathbf{g})_{A,[k]})$  remains nonzero. This implies that for the polynomial  $G(\mathbf{x} + \mathbf{g})$ , the coefficients of the monomials in  $A$  form a cone-closed basis (over  $\mathbb{F}(\mathbf{z})$ ) for its coefficient space. ◀

The above lemma combined Lemma 26 implies that the polynomial  $G(\mathbf{x} + \mathbf{g})$  also has  $k$ -cone concentration over  $\mathbb{F}(\mathbf{t})$ . Here, we would like to mention that although the above rank concentration result is described in terms of cone-closed basis, to design our hitting sets, proving  $k$ -cone concentration property of  $G(\mathbf{x} + \mathbf{g})$  is sufficient. The similar thing is also true for our next rank concentration result.

► **Lemma 5.** *Let  $\mathbb{F}$  be a field of characteristic zero or greater than  $d$ . Let  $G(\mathbf{x})$  be an  $n$ -variate individual degree  $\leq d$  polynomial over  $\mathbb{F}^k$  such that the coefficients of all the monomials of total degree up to  $D$  spans the coefficient space of  $G$ . For some  $N \geq n$ , let  $L(\mathbf{y}) = (\ell_1, \dots, \ell_n)$  be a linear transformation from  $\mathbb{F}^N$  to  $\mathbb{F}^n$  such that all  $\ell_i$ 's are linearly independent. Let  $\mathbf{g}(\mathbf{s}, \mathbf{t})$  be a total degree  $Dk$  independent monomial map from  $\mathbb{F}^m \times \mathbb{F}^{m'}$  to  $\mathbb{F}^N$ . Then  $G(\mathbf{x} + \mathbf{g}')$ , where  $\mathbf{g}' = uL \circ \mathbf{g}$ , has a cone-closed basis over  $F(u, \mathbf{s}, \mathbf{t})$ .*

For proof of the above lemma see Section B.

### 3 Hitting set for orbit of any-order ROABPs

In this section, we describe our hitting set for the orbit of any-order ROABPs. As mentioned earlier, the notion of low-cone concentration plays an important role is designing our hitting sets. We begin by showing that for  $w$ -width  $n$ -variate any-order ROABPs,  $w^2$ -cone concentration can be established by showing  $w^2$ -cone concentration for every  $\Omega(\log w)$ -size subset of variables.

### 30:12 Improved Hitting Set for Orbit of ROABPs

► **Lemma 6.** *Let  $\mathbb{F}$  be a field of characteristic 0 or greater than  $d$ . Let  $G(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]^{w \times w}$  be an  $n$ -variate polynomial over  $\mathbb{F}$  with individual degree at most  $d$  and computed by a  $w$ -width any-order ROABP. Let  $\ell = \lfloor 2 \log w \rfloor + 1$ . Let  $\mathbf{g}(\mathbf{t}) = (g_1, \dots, g_n)$  be a polynomial map such that for all  $S \subseteq [n]$  of size  $\ell$ , the polynomials  $\{g_i \mid i \in S\}$  are algebraically independent. Then  $G(\mathbf{x} + \mathbf{g})$  has  $w^2$ -cone concentration over  $\mathbb{F}(\mathbf{t})$ .*

For proof of the above lemma see the full version. Our next lemma, using Shpilka-Volkovich generator (Definition 21), gives the construction of a polynomial map which satisfies the condition of the above lemma.

► **Lemma 7.** *Let  $L(\mathbf{x}) = (\ell_1, \dots, \ell_n)$  be an invertible linear transformation from  $\mathbb{F}^n$  to  $\mathbb{F}^n$ . Let  $\mathbf{b}$  be a point in  $\mathbb{F}^n$ . For some  $k \leq n$ , let  $\mathbf{g}(\mathbf{s}, \mathbf{t}) = (g_1, \dots, g_n)$  be the polynomial map from  $\mathbb{F}^k \times \mathbb{F}^k$  to  $\mathbb{F}^n$ , defined as  $\mathbf{g} = L \circ \mathcal{G}_{n,k}^{SV} + \mathbf{b}$ . Then for all  $S \subseteq [n]$  of size  $k$ , the polynomials  $\{g_i \mid i \in S\}$  are algebraically independent.*

For proof of the above lemma the full version. Combining the above two lemmas, we get the following.

► **Corollary 8.** *Let  $\mathbb{F}$  be a field of characteristic 0 or greater than  $d$ . Let  $G(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]^{w \times w}$  be an  $n$ -variate polynomial with individual degree at most  $d$  and computed by a width  $w$  any-order ROABP. Let  $L(\mathbf{x})$  be an invertible linear transformation from  $\mathbb{F}^n$  to  $\mathbb{F}^n$  and  $\mathbf{b}$  be a point in  $\mathbb{F}^n$ . Let  $k = \lfloor 2 \log w \rfloor + 1$  and  $\mathbf{g} = L \circ \mathcal{G}_{n,k}^{SV} + \mathbf{b}$ . Then  $G(\mathbf{x} + \mathbf{g})$  has  $w^2$ -cone concentration over  $\mathbb{F}(\mathbf{s}, \mathbf{t})$ .*

**Proof.** Let  $\mathbf{g}(\mathbf{s}, \mathbf{t}) = (g_1, \dots, g_n)$ . From Lemma 7, for every subset  $S \subseteq [n]$  of size  $k$ , the polynomials  $\{g_i \mid i \in S\}$  are algebraically independent. Therefore, using Lemma 6, we get that  $G(\mathbf{x} + \mathbf{g})$  has  $w^2$ -cone concentration over  $\mathbb{F}(\mathbf{s}, \mathbf{t})$ . ◀

Now we describe the construction of our hitting set for the orbit of any-order ROABPs.

**Proof of Theorem 1.** Let  $f(\mathbf{x})$  be an  $n$ -variate individual degree  $\leq d$  polynomial which is in the orbit of width  $w$  any-order ROABPs. Then, there exists an  $n$ -variate individual degree  $\leq d$  polynomial  $G(\mathbf{y}) \in \mathbb{F}^{w \times w}[\mathbf{y}]$  computed by a width  $w$  any-order ROABP, an invertible linear transformation  $L(\mathbf{x})$  from  $\mathbb{F}^n$  to  $\mathbb{F}^n$  and a point  $\mathbf{b} \in \mathbb{F}^n$  such that

$$f(\mathbf{x}) = \mathbf{a}^T \cdot G(L\mathbf{x} + \mathbf{b}) \cdot \mathbf{c},$$

where  $\mathbf{a}, \mathbf{c} \in \mathbb{F}^n$ . Let  $\mathbf{g}(\mathbf{s}, \mathbf{t}) = L \circ \mathcal{G}_{n,k}^{SV} + \mathbf{b}$  where  $k = \lfloor 2 \log w \rfloor + 1$ , and let

$$h(\mathbf{y}) = \mathbf{a}^T \cdot G(\mathbf{y} + \mathbf{g}) \cdot \mathbf{c}.$$

This implies that

$$f'(\mathbf{x}) = f(\mathbf{x} + \mathcal{G}_{n,k}^{SV}) = h(L(\mathbf{x})). \quad (4)$$

From Corollary 8,  $G(\mathbf{y} + \mathbf{g})$  has  $w^2$ -cone concentration over  $\mathbb{F}(\mathbf{s}, \mathbf{t})$ . This implies that there exists a monomial  $\mathbf{y}^e$  in  $h$  with cone-size  $\leq w^2$  such that  $\text{coef}_{\mathbf{y}^e}(h)$  is nonzero. For a monomial of cone-size  $\leq w^2$ , its total degree is less than  $w^2$  and the support-size is  $\leq \log w^2$ . Since the individual degree of each variable in  $G(\mathbf{y})$  is at most  $d$ , Therefore, the degree of  $\mathbf{y}^e$  is  $\leq \ell$  where  $\ell = \min\{w^2, 2d \log w\}$ . Hence,  $\text{hom}_{\leq \ell}(h(\mathbf{y}))$  is a nonzero polynomial in  $\mathbf{y}$ . Since

$$\text{hom}_{\leq \ell}(h(L(\mathbf{x}))) = (\text{hom}_{\leq \ell}(h))(L(\mathbf{x})),$$

from Lemma 24,  $\text{hom}_{\leq \ell}(h(L(\mathbf{x})))$  is a nonzero polynomial. Therefore, from Equation 4,  $\text{hom}_{\leq \ell}(f'(\mathbf{x}))$  is a nonzero polynomial over  $\mathbb{F}(\mathbf{s}, \mathbf{t})$ . This implies that there exists a monomial  $\mathbf{x}^e$  of support-size  $\leq \ell$  such that its coefficient in  $f'$  is nonzero. Thus, from Lemma 23,  $f'(\mathcal{G}_{n,\ell}^{SV}) = f'(\mathcal{G}_{n,k+\ell}^{SV})$  is a  $k + \ell$ -variate nonzero polynomial over  $\mathbb{F}$ . The total degree of  $f$  is at most  $nd$ , and from Observation 22, the individual degree of each coordinate of  $\mathcal{G}_{n,k+\ell}^{SV}$  is at most  $n$ . Also,  $\mathcal{G}_{n,k+\ell}^{SV}$  is  $\text{poly}(ndw)$ -explicit. Thus, from Observation 17,  $f$  has a hitting set computable in time  $(ndw)^{O(\ell)}$ . ◀

#### 4 Hitting Set for orbit of ROABPs

Here, we discuss the construction of our hitting set for the orbit of ROABPs. Towards that, first we need to construct some polynomial map which helps us in achieving low-cone concentration for ROABPs. At this step, we also have to be more careful as we are dealing with the orbit of ROABPs. Lemma 10 describes inductive construction of a polynomial map, by taking sum of logarithmically many variable disjoint copies of total degree  $D$  independent monomial maps (Definition 19) for some small  $D$ , such that the following holds: by shifting its composition with any invertible affine transformation we can achieve low-cone concentration for ROABPs. We begin by showing how to achieve cone-closed basis for the product of two polynomials in a disjoint set of variables, with the property that each polynomial also has a cone-closed basis.

► **Lemma 9.** *Let  $\mathbf{y}$  and  $\mathbf{z}$  be two disjoint sets of variables. Let  $G(\mathbf{y}) \in \mathbb{F}[\mathbf{y}]^{w \times w}$  and  $H(\mathbf{z}) \in \mathbb{F}[\mathbf{z}]^{w \times w}$  be two  $n$ -variate individual degree  $\leq d$  polynomials such that both have cone-closed bases. Let  $L(\mathbf{x}) = (\ell_1, \dots, \ell_{|\mathbf{y}|+|\mathbf{z}|})$  be a linear transformation from  $\mathbb{F}^{|\mathbf{x}|}$  to  $\mathbb{F}^{|\mathbf{y}|} \times \mathbb{F}^{|\mathbf{z}|}$  such that all  $\ell_i$ s are linearly independent. Let  $D = 2w^2 \cdot \min\{w^2, 2d \log w\}$ ,  $\mathbf{g}(\mathbf{s}, \mathbf{t})$  be a total degree  $D$  independent monomial map from  $\mathbb{F}^{|\mathbf{s}|} \times \mathbb{F}^{|\mathbf{t}|}$  to  $\mathbb{F}^{|\mathbf{x}|}$ , and  $\mathbf{g}' = uL \circ \mathbf{g}$ . Then  $G(\mathbf{y} + \mathbf{g}'|_{\mathbf{y}})H(\mathbf{z} + \mathbf{g}'|_{\mathbf{z}})$  has a cone-closed basis over  $F(u, \mathbf{s}, \mathbf{t})$ , where  $\mathbf{g}'|_{\mathbf{y}}$  and  $\mathbf{g}'|_{\mathbf{z}}$  are the restrictions of  $\mathbf{g}'$  over  $\mathbf{y}$  and  $\mathbf{z}$ , respectively.*

For proof of the above lemma see the full version. Applying the above lemma repeatedly, the next one gives the construction of a polynomial map which helps us to achieve low-cone concentration for ROABPs.

► **Lemma 10.** *Let  $n \geq 0$ ,  $N = 2^n$  and  $d, w \geq 1$ . Let  $D = 2w^2 \cdot \min\{w^2, 2d \log w\}$ . Let  $\mathbf{g}(\mathbf{s}, \mathbf{t})$  be a total degree  $D$  independent monomial map from  $\mathbb{F}^m \times \mathbb{F}^{m'}$  to  $\mathbb{F}^N$ . Let  $\mathbf{t}_0 = (t, t^2, \dots, t^N)$ . Let*

$$\mathcal{G}_{n,d,w} = \mathbf{t}_0 + \sum_{i=1}^n u_i \mathbf{g}(\mathbf{s}_i, \mathbf{t}_i),$$

where all  $\mathbf{s}_i$ 's and  $\mathbf{t}_i$ 's are disjoint set of variables.

Let  $\pi$  be permutation on  $[N]$ . Let  $F(\mathbf{x}) = \prod_{i=1}^N M_i(x_{\pi(i)})$  such that each  $M_i(x_{\pi(i)})$  is a polynomial in  $\mathbb{F}^{w \times w}[x_{\pi(i)}]$  with individual degree at most  $d$ . Then for every invertible linear transformation  $L(\mathbf{x})$  from  $\mathbb{F}^N$  to  $\mathbb{F}^N$  and  $\mathbf{b} \in \mathbb{F}^N$ ,  $F(\mathbf{x} + \mathbf{b} + L \circ \mathcal{G}_{n,d,w})$  has a cone-closed basis over the field  $\mathbb{F}(t, (u_i, \mathbf{s}_i, \mathbf{t}_i)_{i \in [n]})$ .

**Proof.** Let  $L(\mathbf{x}) = (\ell_1, \dots, \ell_N)$ . Let  $\mathbf{h}_0 = \mathbf{b} + L(\mathbf{t}_0)$ , and for all  $k \in [n]$ ,

$$\mathbf{h}_k = \mathbf{h}_{k-1} + u_k L \circ \mathbf{g}(\mathbf{t}_k, \mathbf{s}_k).$$

Then  $\mathbf{h}_n = \mathbf{b} + L \circ \mathcal{G}_{n,d,w}$ . For all  $1 \leq i \leq j \leq N$ , let

$$F_{i,j}[\mathbf{x}] = \prod_{r=i}^j M_r(x_{\pi(r)}).$$

### 30:14 Improved Hitting Set for Orbit of ROABPs

Using induction, we show that for all  $k \in \{0, 1, \dots, n\}$  and  $i, j \in [n]$  with  $j - i + 1 = 2^k$ ,  $F_{ij}[\mathbf{x} + \mathbf{h}_k]$  has a cone-closed basis over  $\mathbb{F}(t, (u_i, \mathbf{s}_i, \mathbf{t}_i)_{i \in [k]})$ .

**For  $k = 0$ .** Let  $\mathbf{b} = (b_1, \dots, b_N)$ . We need to show that for all  $i \in [N]$ ,  $M_i(x_{\pi(i)} + \ell_{\pi(i)}(\mathbf{t}_0) + b_{\pi(i)})$  has a cone-closed basis over  $\mathbb{F}(t)$ . Since  $L(\mathbf{x})$  is an invertible linear transformation, each  $\ell_i$  is a nonzero linear polynomial over  $\mathbf{x}$ . Therefore,  $\ell_i(\mathbf{t}_0)$  is a non-constant polynomial in  $t$ . Hence, using Lemma 3, for all  $i \in [N]$ ,  $M_i(x_{\pi(i)} + \ell_{\pi(i)}(\mathbf{t}_0) + b_{\pi(i)})$  has a cone-closed basis over  $\mathbb{F}(t)$ .

**For  $k > 0$ .** Let  $i, j \in [N]$  such that  $j - i + 1 = 2^k$ . Let  $\mathbf{y}$  and  $\mathbf{z}$  be a partition of the variables  $(x_{\pi(i)}, \dots, x_{\pi(j)})$  into two equal halves such that they respect the permutation  $\pi$ . Then  $F_{ij}[\mathbf{x}]$  can be written as  $G(\mathbf{y})H(\mathbf{z})$  where  $G(\mathbf{y}) \in \mathbb{F}[\mathbf{y}]^{w \times w}$  and  $H(\mathbf{z}) \in \mathbb{F}[\mathbf{z}]^{w \times w}$ . From the induction hypothesis, we know that both

$$G'(\mathbf{y}) = G(\mathbf{y} + \mathbf{h}_{k-1}|\mathbf{y}) \text{ and } H'(\mathbf{z}) = H(\mathbf{z} + \mathbf{h}_{k-1}|\mathbf{z})$$

have cone-closed bases over  $\mathbb{F}(t, (u_i, \mathbf{s}_i, \mathbf{t}_i)_{i \in [k-1]})$ . Let  $F'_{ij}(\mathbf{x}) = G'(\mathbf{y})H'(\mathbf{z})$ . Then, using Lemma 9,

$$F_{ij}(\mathbf{x} + \mathbf{h}_k) = F'_{ij}(\mathbf{x} + u_k L \circ \mathbf{g}(\mathbf{s}_k, \mathbf{t}_k))$$

has a cone-closed basis over  $\mathbb{F}(t, (u_i, \mathbf{s}_i, \mathbf{t}_i)_{i \in [k]})$ . This completes our proof.  $\blacktriangleleft$

From Lemma 20, using Klivans-Spielman generator (Lemma 18), we can construct a total degree  $D$  independent monomial map. Therefore, Klivans-Spielman generator combined with the above lemma we get the following corollary.

**► Corollary 11.** *Let  $n \geq 0$ ,  $N = 2^n$  and  $d, w \geq 1$ . Let  $D = 2w^2 \cdot \min\{w^2, 2d \log w\}$ . Let*

$$\mathcal{G}'_{n,d,w} = \mathbf{t}_0 + \sum_{i=1}^n u_i \mathcal{G}_{N,d,ND}^{KS}(\mathbf{s}_i, \mathbf{t}_i). \quad (5)$$

*Let  $\pi$  be permutation on  $[N]$ . Let  $F(\mathbf{x}) = \prod_{i=1}^N M_i(x_{\pi(i)})$  such that each  $M_i(x_{\pi(i)})$  is a polynomial in  $\mathbb{F}[x_{\pi(i)}]^{w \times w}$  with individual degree at most  $d$ . Then,*

1. *for every invertible linear transformation  $L(\mathbf{x})$  from  $\mathbb{F}^N$  to  $\mathbb{F}^N$  and  $\mathbf{b} \in \mathbb{F}^N$ , the polynomial  $F(\mathbf{x} + \mathbf{b} + L \circ \mathcal{G}'_{n,d,w})$  has a cone-closed basis over the field  $\mathbb{F}(t, (u_i, \mathbf{s}_i, \mathbf{t}_i)_{i \in [n]})$ .*
2.  *$\mathbf{b} + \mathcal{G}'_{n,d,w}$  is a polynomial map from  $\mathbb{F} \times (\mathbb{F} \times \mathbb{F}^m \times \mathbb{F}^m)^n$  to  $\mathbb{F}^N$  where  $m = O(D)$ .*
3.  *$\mathcal{G}'_{n,d,w}$  is poly( $dND$ )-explicit polynomial map and its each coordinate is a polynomial of individual degree at most poly( $dN$ ).*

**Proof.** From Lemma 20,  $\mathcal{G}_{N,d,ND}^{KS}(\mathbf{s}, \mathbf{t})$  is a poly( $NDd$ )-explicit total degree  $D$  independent monomial map from  $\mathbb{F}^m \times \mathbb{F}^m$  to  $\mathbb{F}^N$ , where  $m = O(D)$ . Also, each coordinate of  $\mathcal{G}_{N,d,ND}^{KS}$  is a polynomial of individual degree at most poly( $dN$ ). Now this combined with Lemma 10 prove the above corollary.  $\blacktriangleleft$

Now we describe the construction of hitting set for orbit of ROABPs.

**Proof of Theorem 2.** Let  $f(\mathbf{x})$  be an  $n$ -variate individual degree  $\leq d$  polynomial which is in the orbit of width  $w$  ROABPs. Then, there exists an  $n$ -variate individual degree  $\leq d$  polynomial  $G(\mathbf{y}) \in \mathbb{F}[\mathbf{y}]^{w \times w}$  computed by a  $w$ -width ROABP, an invertible linear transformation  $L(\mathbf{x})$  from  $\mathbb{F}^n$  to  $\mathbb{F}^n$  and  $\mathbf{b} \in \mathbb{F}^n$  such that

$$f(\mathbf{x}) = \mathbf{a}^T \cdot G(L(\mathbf{x}) + \mathbf{b}) \cdot \mathbf{c},$$



where  $\mathbf{a}, \mathbf{c} \in \mathbb{F}^n$ . Let  $D = 2w^2 \cdot \min\{w^2, d \log w^2\}$ . Let  $\mathcal{G}'_{\lceil \log n \rceil, d, w}$  be defined as Equation 5 in Corollary 11, that is

$$\mathcal{G}'_{\lceil \log n \rceil, d, w} = \mathbf{t}_0 + \sum_{i=1}^{\lceil \log n \rceil} u_i \mathcal{G}_{n, d, n^D}^{KS}(\mathbf{s}_i, \mathbf{t}_i),$$

where  $\mathbf{t}_0 = (t, t^2, \dots, t^n)$ . Then,  $\mathcal{G}'_{\lceil \log n \rceil, d, w}$  is a polynomial map from  $\mathbb{F} \times (\mathbb{F} \times \mathbb{F}^m \times F^m)^{\lceil \log n \rceil}$  to  $\mathbb{F}^n$  where  $m = O(D)$ . This implies that the number of variables used in  $\mathcal{G}'_{\lceil \log n \rceil, d, w}$  is  $O(D \log n)$ . Let

$$g(\mathbf{y}) = \mathbf{a}^T \cdot G(\mathbf{y} + \mathbf{b} + L \circ \mathcal{G}'_{\lceil \log n \rceil, d, w}) \cdot \mathbf{c}.$$

Then

$$f'(\mathbf{x}) = f(\mathbf{x} + \mathcal{G}'_{\lceil \log n \rceil, d, w}) = g(L(\mathbf{x})). \quad (6)$$

From Corollary 11,

$$G'(\mathbf{y}) = G(\mathbf{y} + \mathbf{b} + L \circ \mathcal{G}'_{\lceil \log n \rceil, d, w})$$

has a cone-closed basis over  $\mathbb{F}(t, (u_i, \mathbf{s}_i, \mathbf{t}_i)_{i \in [\lceil \log n \rceil]})$ . Therefore, from Lemma 26,  $G'(\mathbf{y})$  has also  $w^2$ -cone concentration. This implies that  $g(\mathbf{y})$  has a monomial of nonzero coefficient and its cone-size is at most  $w^2$ . For every monomial of cone-size at most  $w^2$ , its degree is also at most  $w^2$  and its support-size is at most  $2 \log w$ . Therefore, for every monomial of cone-size  $\leq w^2$  and individual degree  $\leq d$ , its degree is at most  $k = \min\{w^2, 2d \log w\}$ . Therefore,  $\text{hom}_{\leq k}(g(\mathbf{y}))$  is a nonzero polynomial in  $\mathbf{y}$  over  $\mathbb{F}(t, (u_i, \mathbf{s}_i, \mathbf{t}_i)_{i \in [\lceil \log n \rceil]})$ . Since

$$\text{hom}_{\leq k}(g(L(\mathbf{x}))) = (\text{hom}_{\leq k}(g))(L(\mathbf{x})),$$

from Lemma 24,  $\text{hom}_{\leq k}(g(L(\mathbf{x})))$  is also nonzero polynomial. Therefore, from Equation 6,  $\text{hom}_{\leq k}(f'(\mathbf{x}))$  is also a nonzero polynomial. This implies that there exists a monomial  $\mathbf{x}^e$  of support-size at most  $k$  such that  $\text{coef}_{\mathbf{x}^e}(f')$  is nonzero. Thus, from Lemma 23,

$$f'(\mathcal{G}_{n, k}^{SV}) = f(\mathcal{G}_{n, k}^{SV} + \mathcal{G}'_{\lceil \log n \rceil, d, w})$$

is a nonzero polynomial. Let  $\mathcal{G} = \mathcal{G}_{n, k}^{SV} + \mathcal{G}'_{\lceil \log n \rceil, d, w}$ . Then,  $\mathcal{G}$  is a polynomial map in  $O(kw^2 \log n)$  many variables and the individual degree of each coordinate is at most  $\text{poly}(ndw)$ . Since both  $\mathcal{G}_{n, k}^{SV}$  and  $\mathcal{G}'_{\lceil \log n \rceil, d, w}$  both are  $\text{poly}(ndw)$ -explicit,  $\mathcal{G}$  is also  $\text{poly}(ndw)$ -explicit. Thus, applying Observation 17, we have a hitting set for  $f$  computable in time  $(ndw)^{O(\ell)}$  where  $\ell = (w^2 \log n) \cdot \min\{w^2, d \log w^2\}$ . ◀

## 5 Conclusion

In this paper, we studied the hitting set problem for the orbits of ROABPs and *any-order* ROABPs. We have designed improved hitting sets for these two polynomial classes. In low-width but high-individual-degree setting, our hitting sets are more efficient than the previous ones given by Saha and Thankey. On the technical front, we have shown some stronger rank concentration results by establishing low-cone concentration for polynomials over vector spaces. These new rank concentration results have played a significant role in designing our hitting sets. However, our hitting sets for the orbits of ROABPs and *any-order* ROABPs are yet to match the time complexity of hitting sets known for ROABPs and its variants. Therefore, it is an interesting open question to close this gap.

## References

- 1 Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Hitting-sets for ROABP and sum of set-multilinear circuits. *SIAM Journal on Computing*, 44(3):669–697, 2015.
- 2 Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of mathematics*, pages 781–793, 2004.
- 3 Manindra Agrawal, Chandan Saha, Ramprasad Satharishi, and Nitin Saxena. Jacobian hits circuits: hitting-sets, lower bounds for depth-d occur-k formulas & depth-3 transcendence degree-k circuits. In *STOC*, pages 599–614, 2012.
- 4 Manindra Agrawal, Chandan Saha, and Nitin Saxena. Quasi-polynomial hitting-set for set-depth- $\Delta$  formulas. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 321–330, 2013.
- 5 M. Beekun, J. Mittmann, and N. Saxena. Algebraic Independence and Blackbox Identity Testing. *Inf. Comput.*, 222:2–19, 2013. (Conference version in ICALP 2011).
- 6 Michael Ben-Or and Prasoos Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 301–309, 1988. doi:10.1145/62212.62241.
- 7 Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. Hardness vs randomness for bounded depth arithmetic circuits. In *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, pages 13:1–13:17, 2018. doi:10.4230/LIPIcs.CCC.2018.13.
- 8 David A. Cox, John Little, and Donal O’Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer Publishing Company, Incorporated, 4th edition, 2015.
- 9 Rafael Mendes de Oliveira, Amir Shpilka, and Ben Lee Volk. Subexponential size hitting sets for bounded depth multilinear formulas. In *30th Conference on Computational Complexity, CCC 2015, June 17-19, 2015, Portland, Oregon, USA*, pages 304–322, 2015. doi:10.4230/LIPIcs.CCC.2015.304.
- 10 Richard A. Demillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, 1978.
- 11 Zeev Dvir, Rafael Mendes de Oliveira, and Amir Shpilka. Testing Equivalence of Polynomials under Shifts. In *41st International Colloquium on Automata, Languages, and Programming, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 417–428, 2014. doi:10.1007/978-3-662-43948-7\_35.
- 12 Zeev Dvir and Amir Shpilka. Locally decodable codes with two queries and polynomial identity testing for depth 3 circuits. *SIAM J. Comput.*, 36(5):1404–1434, 2007. doi:10.1137/05063605X.
- 13 Zeev Dvir, Amir Shpilka, and Amir Yehudayoff. Hardness-randomness tradeoffs for bounded depth arithmetic circuits. *SIAM J. Comput.*, 39(4):1279–1293, 2009. doi:10.1137/080735850.
- 14 Stephen A. Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite perfect matching is in quasi-NC. In *48th Annual ACM Symposium on Theory of Computing*, pages 754–763, 2016.
- 15 Michael A Forbes. Deterministic divisibility testing via shifted partial derivatives. In *56th Annual Symposium on Foundations of Computer Science*, pages 451–465, 2015.
- 16 Michael A. Forbes, Sumanta Ghosh, and Nitin Saxena. Towards blackbox identity testing of log-variate circuits. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 54:1–54:16, 2018.
- 17 Michael A. Forbes, Ramprasad Satharishi, and Amir Shpilka. Pseudorandomness for multilinear read-once algebraic branching programs, in any order. *Electron. Colloquium Comput. Complex.*, 20:132, 2013. Conference version is accepted in STOC 2014. URL: <http://eccc.hpi-web.de/report/2013/132>.
- 18 Michael A Forbes and Amir Shpilka. Explicit noether normalization for simultaneous conjugation via polynomial identity testing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 527–542. Springer, 2013.

- 19 Michael A. Forbes and Amir Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 243–252. IEEE Computer Society, 2013. doi:10.1109/FOCS.2013.34.
- 20 Zeyu Guo and Rohit Gurjar. Improved explicit hitting-sets for roabps. In Jaroslav Byrka and Raghu Meka, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*, volume 176 of *LIPICs*, pages 4:1–4:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.APPROX/RANDOM.2020.4.
- 21 Zeyu Guo, Mrinal Kumar, Ramprasad Satharishi, and Noam Solomon. Derandomization from algebraic hardness. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:65, 2019. Preliminary version in FOCS 2019. URL: <https://ecc.weizmann.ac.il/report/2019/065/>.
- 22 Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Identity testing for constant-width, and any-order, read-once oblivious arithmetic branching programs. *Theory of Computing*, 13(2):1–21, 2017.
- 23 Rohit Gurjar, Arpita Korwar, Nitin Saxena, and Thomas Thierauf. Deterministic identity testing for sum of read-once oblivious arithmetic branching programs. In *30th Conference on Computational Complexity, CCC 2015, June 17-19, 2015, Portland, Oregon, USA*, pages 323–346, 2015. doi:10.4230/LIPICs.CCC.2015.323.
- 24 Rohit Gurjar and Thomas Thierauf. Linear matroid intersection is in quasi-nc. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 821–830, 2017.
- 25 Joos Heintz and Claus-Peter Schnorr. Testing polynomials which are easy to compute (extended abstract). In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA*, pages 262–272, 1980.
- 26 Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004. Preliminary version in the 35th Annual ACM Symposium on Theory of Computing (STOC), 2003. doi:10.1007/s00037-004-0182-6.
- 27 Zohar Shay Karnin and Amir Shpilka. Black box polynomial identity testing of generalized depth-3 arithmetic circuits with bounded top fan-in. *Combinatorica*, 31(3):333–364, 2011. doi:10.1007/s00493-011-2537-3.
- 28 Neeraj Kayal and Shubhangi Saraf. Blackbox polynomial identity testing for depth 3 circuits. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 198–207, 2009. doi:10.1109/FOCS.2009.67.
- 29 Neeraj Kayal and Nitin Saxena. Polynomial identity testing for depth 3 circuits. *Computational Complexity*, 16(2):115–138, 2007.
- 30 Adam R. Klivans and Daniel A. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 216–223, 2001. doi:10.1145/380752.380801.
- 31 Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. Equivalence of polynomial identity testing and deterministic multivariate polynomial factorization. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*, pages 169–180, 2014.
- 32 Mrinal Kumar and Shubhangi Saraf. Arithmetic circuits with locally low algebraic rank. *Theory of Computing*, 13(1):1–33, 2017. Preliminary version in the 31st Conference on Computational Complexity (CCC), 2016. doi:10.4086/toc.2017.v013a006.
- 33 Mrinal Kumar and Ben Lee Volk. A polynomial degree bound on equations for non-rigid matrices and small linear circuits. 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 9:1–9:9. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.ITCS.2021.9.

- 34 Richard J. Lipton and Nisheeth K. Vishnoi. Deterministic identity testing for multivariate polynomials. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA.*, pages 756–760, 2003.
- 35 László Lovász. On determinants, matchings, and random algorithms. In *FCT*, volume 79, pages 565–574, 1979.
- 36 Dori Medini and Amir Shpilka. Hitting sets and reconstruction for dense orbits in vpe and  $\Sigma\Pi\Sigma$  circuits. *CoRR*, abs/2102.05632, 2021. [arXiv:2102.05632](https://arxiv.org/abs/2102.05632).
- 37 Daniel Minahan and Ilya Volkovich. Complete derandomization of identity testing and reconstruction of read-once formulas. In Ryan O’Donnell, editor, *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, volume 79 of *LIPICs*, pages 32:1–32:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. [doi:10.4230/LIPICs.CCC.2017.32](https://doi.org/10.4230/LIPICs.CCC.2017.32).
- 38 Ketan D. Mulmuley. Geometric complexity theory V: Equivalence between blackbox derandomization of polynomial identity testing and derandomization of Noether’s normalization lemma. In *FOCS*, pages 629–638, 2012.
- 39 Noam Nisan and Avi Wigderson. Lower bounds on arithmetic circuits via partial derivatives. *Computational Complexity*, 6(3):217–234, 1997. Preliminary version in the 36th Annual Symposium on Foundations of Computer Science (FOCS), 1995. [doi:10.1007/BF01294256](https://doi.org/10.1007/BF01294256).
- 40 Anurag Pandey, Nitin Saxena, and Amit Sinhababu. Algebraic independence over positive characteristic: New criterion and applications to locally low-algebraic-rank circuits. *Computational Complexity*, 27(4):617–670, 2018. Preliminary version in the 41st International Symposium on Mathematical Foundations of Computer Science (MFCS), 2016. [doi:10.1007/s00037-018-0167-5](https://doi.org/10.1007/s00037-018-0167-5).
- 41 Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005. [doi:10.1007/s00037-005-0188-8](https://doi.org/10.1007/s00037-005-0188-8).
- 42 Chandan Saha, Ramprasad Satharishi, and Nitin Saxena. A case of depth-3 identity testing, sparse factorization and duality. *Computational Complexity*, 22(1):39–69, 2013.
- 43 Chandan Saha and Bhargav Thankey. Hitting sets for orbits of circuit classes and polynomial families. *Electron. Colloquium Comput. Complex.*, 28:15, 2021. URL: <https://eccc.weizmann.ac.il/report/2021/015>.
- 44 Nitin Saxena. Diagonal circuit identity testing and lower bounds. In *ICALP*, volume 5125 of *Lecture Notes in Computer Science*, pages 60–71. Springer, 2008.
- 45 Nitin Saxena. Progress on polynomial identity testing. *Bulletin of the EATCS*, 99:49–79, 2009.
- 46 Nitin Saxena and C. Seshadhri. An almost optimal rank bound for depth-3 identities. *SIAM J. Comput.*, 40(1):200–224, 2011. [doi:10.1137/090770679](https://doi.org/10.1137/090770679).
- 47 Nitin Saxena and C. Seshadhri. Blackbox identity testing for bounded top-fanin depth-3 circuits: The field doesn’t matter. *SIAM Journal on Computing*, 41(5):1285–1298, 2012.
- 48 Nitin Saxena and C. Seshadhri. From sylvester-gallai configurations to rank bounds: Improved blackbox identity test for depth-3 circuits. *J. ACM*, 60(5):33:1–33:33, 2013. [doi:10.1145/2528403](https://doi.org/10.1145/2528403).
- 49 J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.
- 50 Adi Shamir.  $\text{Ip}=\text{pspace}$ . In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 11–15, 1990. [doi:10.1109/FSCS.1990.89519](https://doi.org/10.1109/FSCS.1990.89519).
- 51 Amir Shpilka and Ilya Volkovich. Improved polynomial identity testing for read-once formulas. In Irit Dinur, Klaus Jansen, Joseph Naor, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop, APPROX 2009, and 13th International Workshop, RANDOM 2009, Berkeley, CA, USA, August 21-23, 2009. Proceedings*, volume 5687 of *Lecture Notes in Computer Science*, pages 700–713. Springer, 2009. [doi:10.1007/978-3-642-03685-9\\_52](https://doi.org/10.1007/978-3-642-03685-9_52).
- 52 Amir Shpilka and Ilya Volkovich. Read-once polynomial identity testing. *Comput. Complex.*, 24(3):477–532, 2015. [doi:10.1007/s00037-015-0105-8](https://doi.org/10.1007/s00037-015-0105-8).

- 53 Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010.
- 54 Ola Svensson and Jakub Tarnawski. The matching problem in general graphs is in quasi-nc. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 696–707, 2017. doi:10.1109/FOCS.2017.70.
- 55 W. T. Tutte. The factorization of linear graphs. *Journal of the London Mathematical Society*, s1-22(2):107–111, 1947.
- 56 Jiang Zeng. A bijective proof of Muir’s identity and the Cauchy-Binet formula. *Linear Algebra and its Applications*, 184:79–82, 1993.
- 57 Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation, EUROSAM ’79*, pages 216–226, 1979.

## A Preliminaries

We start with the following observation.

► **Observation 12.** *For a monomial of cone-size at most  $k$ , its degree is less than  $k$  and the support-size is at most  $\log k$ .*

A *monomial ordering* is a total ordering on the set of all monomials in  $\mathbf{x}$  with following properties:

1. for all  $\mathbf{a} \in \mathbb{N}^n \setminus \{\mathbf{0} = (0, \dots, 0)\}$ ,  $\mathbf{1} \prec \mathbf{x}^{\mathbf{a}}$ .
2. for all  $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{N}^n$ , if  $\mathbf{x}^{\mathbf{a}} \prec \mathbf{x}^{\mathbf{b}}$  then  $\mathbf{x}^{\mathbf{a}+\mathbf{c}} \prec \mathbf{x}^{\mathbf{b}+\mathbf{c}}$ .

For more on monomial ordering, see [8, Chapter 2].

Suppose that  $M$  is a matrix whose rows and columns are indexed by  $A$  and  $B$ , respectively. Then for every  $S \subseteq A$  and  $T \subseteq B$ ,  $M_{S,T}$  denotes the submatrix of  $M$  with rows and columns indexed by  $S$  and  $T$ , respectively. The next lemma is a well known phenomenon in matroid theory which, informally, says that given distinct weights to the elements of a matroid there exists a unique minimum weight base. Here, we describe it in a language which is suitable for our context.

► **Lemma 13.** *Let  $k$  be a positive integer and  $M_{n,d}$  be the set of all  $n$ -variate monomials in  $\mathbf{x}$  with individual degree  $\leq d$ . Let  $M$  be a matrix over  $\mathbb{F}$  of rank  $r$  such that its rows are indexed by  $[k]$  and the columns are indexed by  $M_{n,d}$ . Let  $\prec$  be a monomial ordering on the set of monomials in  $\mathbf{x}$ . Then there exists a unique subset  $B \subseteq M_{n,d}$  of size  $r$  such that  $\text{rank}(M_{[k],B}) = r$  and for every other subset  $B' \subseteq M_{n,d}$  with  $\text{rank}(M_{[k],B'}) = r$ ,  $\prod_{\mathbf{e} \in B} \mathbf{x}^{\mathbf{e}} \prec \prod_{\mathbf{e}' \in B'} \mathbf{x}^{\mathbf{e}'}$ .*

Here we give a very brief sketch of the proof. Using the monomial ordering  $\prec$ , greedily choose  $r$  linearly independent columns of  $M$  as follows: at each step pick the least  $\prec$ -indexed column of  $M$  such that it increases the rank of the chosen vectors, and denote that set by  $B = \{m_1, \dots, m_r\}$  with  $m_1 \prec \dots \prec m_r$ . Let  $B'$  be another subset of  $M_{n,d}$  with  $r$  linearly independent columns of  $M$ , and  $B' = \{m'_1, \dots, m'_r\}$  with  $m'_1 \prec \dots \prec m'_r$ . Then one can show that  $B \preceq B'$  point-wise, that is  $m_i \preceq m'_i$  for all  $i \in [r]$ , and there exists an  $i_0 \in [r]$  such that  $m_{i_0} \prec m'_{i_0}$ . This implies that  $\prod_{i \in [r]} m_i \prec \prod_{i \in [r]} m'_i$ . For more details one can see [17, Lemma 5.2 and 5.3].

Next, we give an expression for the product of a “fat” matrix with a “tall” matrix. It is known as Cauchy-Binet formula. It will be useful to prove the rank concentration results in Section 2.

## 30:20 Improved Hitting Set for Orbit of ROABPs

► **Lemma 14** (Cauchy-Binet formula, [56]). *Let  $n \geq m$  be two positive integers. Let  $M$  and  $N$  two  $m \times n$  and  $n \times m$  matrices, respectively, over  $\mathbb{F}$ . Then*

$$\det(AB) = \sum_{S \in \binom{[n]}{m}} \det(M_{[m],S}) \cdot \det(M_{S,[m]}).$$

### A.1 Hitting sets

► **Definition 15.** *Let  $\mathcal{C}$  be a set of  $n$ -variate polynomials over a field  $\mathbb{F}$ . A set of points  $\mathcal{H} \subseteq \mathbb{F}^n$  is called a hitting set for  $\mathcal{C}$  if for every polynomial  $f \in \mathcal{C}$ ,  $f$  is nonzero if and only if there exists a point  $\alpha \in \mathcal{H}$  such that  $f(\alpha) \neq 0$ .*

We say a hitting set  $\mathcal{H}$  is *computable in time  $T$*  if there exists an algorithm which computes all the points in the set  $\mathcal{H}$  in time  $T$ . When  $\mathbb{F}$  is a finite field, we are allowed to pick points from  $\mathbb{K}^n$  where  $\mathbb{K}$  is a polynomially large extension of  $\mathbb{F}$ . In PIT literature, a common method of designing hitting sets is via hitting set generator.

► **Definition 16.** *Let  $\mathcal{C}$  be a set of  $n$ -variate polynomial class over a field  $\mathbb{F}$ . A polynomial map  $\mathbf{g}(\mathbf{t})$  from  $\mathbb{F}^m$  to  $\mathbb{F}^n$  is called hitting set generator for  $\mathcal{C}$  if for every  $f \in \mathcal{C}$ ,  $f$  is nonzero if and only if  $f(\mathbf{g}) \neq 0$ .*

*Furthermore,  $\mathbf{g}(\mathbf{t})$  is called  $t(m, n)$ -explicit if there exists an  $n$ -output circuit which computes  $\mathbf{g}(\mathbf{t})$  and the circuit is computable in  $t(m, n)$  time.*

Hitting set generators immediately give us hitting sets.

► **Observation 17.** *Let  $\mathcal{C}$  be an  $n$ -variate polynomial class over a field  $\mathbb{F}$  such that the degree of each polynomial is at most  $d$ . Let  $\mathbf{g}(\mathbf{t}) : \mathbb{F}^m \leftarrow \mathbb{F}^n$  be a hitting set generator for  $\mathcal{C}$  such that the individual degree of each coordinate of  $\mathbf{g}$  is at most  $r$ . Let  $S$  be a subset of  $\mathbb{F}$  of size  $dr + 1$ . Then  $\mathcal{H} := \mathbf{g}(S^m)$  is a hitting set for  $\mathcal{C}$ . Moreover, if  $\mathbf{g}(\mathbf{t})$  is  $t$ -explicit then the hitting set  $\mathcal{H}$  is computable in  $\text{poly}(t(dr)^m)$  time.*

**Proof.** Since  $\mathbf{g}$  is a hitting set generator for  $\mathcal{C}$  and each coordinate of  $\mathbf{g}$  is a  $m$ -variate polynomial, for every nonzero  $f \in \mathcal{C}$ ,  $f(\mathbf{g})$  is a nonzero  $m$ -variate polynomial. Also, the individual degree of  $f(\mathbf{g})$  is at most  $dr$ . Thus, there exists a point  $\alpha \in S^m$  such that  $f(\mathbf{g}(\alpha)) \neq 0$ . Therefore,  $\mathcal{H}$  is a hitting set for  $\mathcal{C}$ . Since  $\mathbf{g}$  is  $t$ -explicit, each point in  $\mathcal{H}$  is computable in time  $\text{poly}(t)$ . Therefore,  $\mathcal{H}$  is computable in time  $\text{poly}(t(dr)^m)$ . ◀

### A.2 Some useful polynomial maps

Suppose that  $\mathbf{g}(\mathbf{t}) = (g_1, \dots, g_n)$  be a polynomial map from  $\mathbb{F}^m$  to  $\mathbb{F}^n$ . Then, we say  $\mathbf{g}$  is a  $t(m, n)$ -explicit polynomial map if there exists an  $n$ -output circuit  $C$  which computes the polynomials  $(g_1, \dots, g_n)$  and the circuit  $C$  is computable in time  $t(m, n)$ . Let  $\mathbf{g}(\mathbf{y})$  be a polynomial map from  $\mathbb{F}^m$  to  $\mathbb{F}^n$  and  $\mathbf{h}(\mathbf{x}) = (h_1, \dots, h_k)$  be a polynomial map from  $\mathbb{F}^n$  to  $\mathbb{F}^k$ . Then  $\mathbf{h} \circ \mathbf{g}$  denotes the composition of  $\mathbf{g}$  with  $\mathbf{h}$ , that is  $\mathbf{h}(\mathbf{g}) = (h_1(\mathbf{g}), \dots, h_k(\mathbf{g}))$ . A polynomial map  $L(\mathbf{x}) = (\ell_1, \dots, \ell_n)$  from  $\mathbb{F}^n$  to  $\mathbb{F}^n$  is called an *invertible linear transformation* if each  $\ell_i$  is a linear polynomial of form  $\ell_{i1}x_1 + \dots + \ell_{in}x_n$  and all  $\ell_i$ 's are linearly independent. An *invertible affine transformation* is a polynomial map of form  $L(\mathbf{x}) + \mathbf{b}$  where  $L(\mathbf{x})$  is an invertible linear transformation and  $\mathbf{b} \in \mathbb{F}^n$ . Next, we describe some well known polynomial maps and their properties which are frequently used in designing PIT algorithms, and they also will be useful for us. First, we describe the generator for sparse polynomial due to Klivans and Spielman [30].

► **Lemma 18** (Klivans-Spielman generator [30]). *Let  $n, d, s, m$  be positive integers such that  $m = \Theta(\log_{nd} s)$ . Let  $\mathbb{F}$  be a field of size  $\geq \text{poly}(nd)$ . Then there exists a  $\text{poly}(nd)$ -explicit polynomial map  $\mathcal{G}_{n,d,s}^{KS}(\mathbf{s}, \mathbf{t})$  from  $\mathbb{F}^m \times \mathbb{F}^m$  to  $\mathbb{F}^n$  such that*

1. *for all  $i \in [n]$ ,  $(\mathcal{G}_{n,d,s}^{KS})_i$  is a polynomial of individual degree  $\leq \text{poly}(nd)$ .*
2. *for every subset  $S$  of at most  $s$  monomials in  $n$ -variables with individual degree at most  $d$ , there exists an  $\boldsymbol{\alpha} \in \mathbb{F}^m$  such that the polynomials  $\{(\mathcal{G}_{n,d,s}^{KS}(\boldsymbol{\alpha}, \mathbf{t}))^e\}_{e \in S}$  are nonzero, distinct monomials in  $\mathbf{t}$ .*

The above generator is a slight variation of the construction given in [30], but it can be constructed from their techniques. For a proof-sketch see [17, Theorem 2.3]. Next, we define total degree  $D$  independent monomial map from [17].

► **Definition 19.** *For some positive integers  $n$  and  $D$ , a polynomial map  $\mathbf{g}(\mathbf{s}, \mathbf{t})$  from  $\mathbb{F}^m \times \mathbb{F}^{m'}$  to  $\mathbb{F}^n$  is called total degree  $D$  independent monomial map if there exists an  $\boldsymbol{\alpha} \in \mathbb{F}^m$  such that the polynomials  $\{\mathbf{g}(\boldsymbol{\alpha}, \mathbf{t})^e\}_{|e|_1 \leq D}$  are nonzero, distinct monomials in  $\mathbf{t}$ .*

In the following lemma, we describe a construction of total degree  $D$  independent monomial map using Klivans-Spielman generator.

► **Lemma 20.** *Let  $n, d, D$  be positive integers. Let  $|\mathbb{F}| \geq \text{poly}(nd)$ . Then,  $\mathcal{G}_{n,d,n^D}^{KS}$  is a  $\text{poly}(ndD)$ -explicit total degree  $D$  independent monomial map from  $\mathbb{F}^m \times \mathbb{F}^m$  to  $\mathbb{F}^n$  where  $m = O(D)$ .*

For proof see [17, Lemma 6.4]. Next, we describe a polynomial map introduced by Shpilka and Volkovich [51]. It is a widely used tool in PIT and other related results [51, 17, 52, 37, 33, 36, 43], and also crucial for proving our results.

► **Definition 21** (Shpilka-Volkovich generator [51]). *Fix a positive integer  $n$  and a set of  $n$  distinct elements  $\mathcal{A} = \{\alpha_1, \dots, \alpha_n\} \subseteq \mathbb{F}$ . Let  $L_i(t)$  be the  $i$ th Lagrange interpolation polynomial for the set  $\mathcal{A}$ . That is,  $L_i(t)$  is a univariate polynomial of degree  $n - 1$  such that  $L_i(\alpha_j) = \delta_{ij}$ . Let  $\mathbf{s} = (s_1, \dots, s_k)$  and  $\mathbf{t} = (t_1, \dots, t_k)$ . Then  $\mathcal{G}_{n,k}^{SV}(\mathbf{s}, \mathbf{t})$  is the polynomial map from  $\mathbb{F}^k \times \mathbb{F}^k$  to  $\mathbb{F}^n$  defined as follows: for all  $i \in [n]$*

$$(\mathcal{G}_{n,k}^{SV})_i = \sum_{j=1}^k L_i(s_j) t_j.$$

The above definition gives the following properties of Shpilka-Volkovich generator.

► **Observation 22.** *Fix a set of  $k$  distinct elements  $S = \{i_1, \dots, i_k\} \subseteq [n]$ . Let  $\boldsymbol{\alpha} = (\alpha_{i_1}, \dots, \alpha_{i_k})$ . Then, for all  $j \in [k]$ ,  $(\mathcal{G}_{n,k}^{SV}(\boldsymbol{\alpha}, \mathbf{t}))_{i_j} = t_j$ , and the other coordinates of  $\mathcal{G}_{n,k}^{SV}(\boldsymbol{\alpha}, \mathbf{t})$  are zero. Furthermore, for all  $i \in [n]$ , the degree of the polynomial  $(\mathcal{G}_{n,k}^{SV})_i$  is at most  $n$ .*

Using Shpilka-Volkovich generator, the following lemma describes a nonzeroness preserving variable reduction for polynomials having a “low-support” monomial with nonzero coefficient.

► **Lemma 23.** *Let  $f(\mathbf{x})$  be an  $n$ -variate polynomial over  $\mathbb{F}$  such that there exists a monomial  $\mathbf{x}^e$  with nonzero coefficient in  $f$  and the support-size of  $\mathbf{e}$  is at most  $\ell$ . Then  $f \circ \mathcal{G}_{n,\ell}^{SV} \neq 0$ .*

**Proof.** Let  $\{x_{i_1}, \dots, x_{i_\ell}\}$  be the support set of the monomial  $\mathbf{x}^e$ . Then, from Observation 22, there exists an  $\boldsymbol{\alpha} \in \mathbb{F}^\alpha$  such that for all  $j \in [\ell]$ ,  $(\mathcal{G}_{n,\ell}^{SV}(\boldsymbol{\alpha}, \mathbf{t}))_{i_j} = t_j$  and the other coordinates of  $\mathcal{G}_{n,\ell}^{SV}(\boldsymbol{\alpha}, \mathbf{t})$  are zero. This implies that  $f(\mathcal{G}_{n,\ell}^{SV}(\boldsymbol{\alpha}, \mathbf{t})) \neq 0$ , and therefore  $f \circ \mathcal{G}_{n,\ell}^{SV} \neq 0$ . ◀

### A.3 Algebraic independence

Suppose that  $\mathcal{A} = \{g_1, \dots, g_k\}$  is a set of  $n$ -variate polynomials over a field  $\mathbb{F}$ . We say that the set of polynomials  $\mathcal{A}$  are *algebraically dependent* over  $\mathbb{F}$  if there exists a nonzero  $k$ -variate polynomial  $A(z_1, \dots, z_k)$  over  $\mathbb{F}$  such that  $A(g_1, \dots, g_k) = 0$ . Otherwise, they are called *algebraically independent* (over  $\mathbb{F}$ ). In the following lemma, we describe a well known criteria regarding algebraic independence of a set of linear polynomials.

► **Lemma 24.** *Let  $m \geq n$  be two positive integers. Let  $L(\mathbf{x}) = (\ell_1, \dots, \ell_n)$  be a linear transformation from  $\mathbb{F}^m$  to  $\mathbb{F}^n$  such that all  $\ell_i$ 's are linearly independent. Then, all  $\ell_i$ 's are also algebraically independent.*

**Proof.** For the sake of contradiction, assume that all  $\ell_i$ 's are not algebraically independent. Then there exists a nonzero polynomial  $A(z_1, \dots, z_n)$  such that  $A(L(\mathbf{x})) = A(\ell_1, \dots, \ell_n) = 0$ . Let  $\mathbf{x} = (x_1, \dots, x_m)$  and  $A'(\mathbf{x}) = A(L(\mathbf{x}))$ . Since all  $\ell_i$ 's are linearly independent, there exists a tuple of linear polynomials  $U(\mathbf{x}) = (u_1, \dots, u_m)$  and a subset  $\{i_1, \dots, i_n\}$  of  $[m]$  such that for all  $j \in [n]$ ,

$$\ell_j(U(\mathbf{x})) = x_{i_j}.$$

This implies that  $A'(U(\mathbf{x})) = A(x_{i_1}, \dots, x_{i_n}) = 0$  which is a contradiction. Therefore, all  $\ell_i$ 's are algebraically independent. ◀

### A.4 Various notions of rank concentration

We define various notions of rank concentration and show the relation between them. Suppose that  $G(\mathbf{x})$  be an  $n$ -variate polynomial over the vector space  $\mathbb{F}^k$ . The *coefficient space* of  $G$  is the vector space spanned by the coefficient vectors of  $G$ .

► **Definition 25** (Rank Concentration). *We say that  $G$  has*

1.  $\ell$ -support concentration if there exists a set of monomials  $B$  such that the support-size of each monomial in  $B$  is at most  $\ell$  and their coefficients form a basis for the coefficient space of  $G$ .
2.  $\ell$ -cone concentration if there exists a set of monomials  $B$  such that the cone-size of each monomial in  $B$  is at most  $\ell$  and their coefficients form a basis for the coefficient space of  $G$ .
3. a cone-closed basis if there is a cone-closed set of monomials  $B$  whose coefficients in  $G$  form a basis of the coefficient space of  $G$ .

In the next lemma, we show that cone-closed basis notion subsumes the other two notions of rank concentration.

► **Lemma 26.** *Let  $G(\mathbf{x})$  be a polynomial in  $\mathbb{F}[\mathbf{x}]^k$ . Suppose that  $G(\mathbf{x})$  has a cone-closed basis. Then,  $G(\mathbf{x})$  has  $k$ -cone concentration and  $\log k$ -support concentration.*

**Proof.** Let  $B$  be a cone-closed set of monomials whose coefficients in  $G$  form a basis for the coefficient space of  $G$ . Since the cardinality of  $B$  is at most  $k$  and it is closed under submonomials, the cone-size of each monomial  $B$  is at most  $k$ . Therefore,  $G$  has  $k$ -cone concentration.

Let  $m \in B$  and  $S$  be the support set of  $m$ . Let  $m'$  be the monomial defined as  $m' = \prod_{i \in S} x_i$ . Since  $B$  is cone-closed, every sub-monomial  $m'$  is also in  $B$ . Thus the cardinality of  $S$  can be at most  $\log k$ . Therefore,  $G$  has  $\log k$ -support concentration. ◀



## B Proof of Lemma 5

**Proof of Lemma 5.** First we study the shifted polynomial  $G'(\mathbf{x}) = G(\mathbf{x} + u\mathbf{z})$ . To do so, we revisit the proof of our Lemma 4. There we considered the lexicographic monomial ordering over the monomials in  $\mathbf{z}$ . Here we consider the *deg-lex* monomial ordering, that is, first order the monomials from lower degree to higher degree and then within each degree arrange them in lexicographic order. Like Equation 1, the matrix equation for the shifted polynomial  $G'(\mathbf{x})$  will be

$$F'(u\mathbf{z}) = W^{-1}(u\mathbf{z})TW(u\mathbf{z})F, \quad (7)$$

that is scaling of each variable in Equation 1 by  $u$ . Applying Lemma 13, let  $B$  be the unique subset of  $M_{n,d}$  such that the rows of  $F$  indexed by  $B$  form the least basis for the row-space of  $F$  with respect to the deg-lex monomial ordering. From the hypothesis of the lemma, there exists a subset  $C \subseteq M_{n,d}$  such that the rows in  $F$  indexed by  $C$  forms a basis for the row-space of  $F$  (same as the coefficient space of  $G$ ) and  $\deg(C) = \sum_{\mathbf{e} \in C} |\mathbf{e}|_1 \leq Dk$ . Therefore,  $\deg(B)$  is also  $\leq Dk$  since the rows indexed by  $B$  forms the least basis (with respect to deg-lex monomial ordering) for the row-space of  $F$ . As promised by Lemma 3, let  $A$  be a cone-closed subset of  $M_{n,d}$  such that  $T_{A,B}$  is full rank. Now we see how Equation 2 and 3 in the proof of Lemma 4 change here. Like Equation 2, we get

$$\det(F'(u\mathbf{z})_{A,[k]}) = \det(W(u\mathbf{z})_{A,A})^{-1} \cdot \det((TW(u\mathbf{z})F)_{A,[k]}) \quad (8)$$

and Equation 3 changes as follows:

$$\det((TW(u\mathbf{z})F)_{A,[k]}) = \sum_i \left( \sum_{C \in \binom{M_{n,d}}{k} : \deg(C)=i} \det(T_{A,C}) \det(F_{C,[k]}) \prod_{\mathbf{e} \in C} \mathbf{z}^{\mathbf{e}} \right) u^i. \quad (9)$$

Since  $B$  is the least basis (with respect to deg-lex monomial ordering), the coefficient of  $u^{\deg(B)}$  is a nonzero degree  $\deg(B)$  homogeneous polynomial in  $\mathbf{z}$ . Thus,  $\det(F'(u\mathbf{z})_{A,[k]})$  is a nonzero-polynomial in  $(u, \mathbf{z})$ . This implies the coefficients of the monomials in  $A$  is a cone-close basis for  $G(\mathbf{x} + u\mathbf{z})$ . For  $G(\mathbf{x} + uL)$ , the polynomial  $\det((TW(uL)F)_{A,[k]})$  looks like the following:

$$\det((TW(uL)F)_{A,[k]}) = \sum_i \left( \sum_{C \in \binom{M_{n,d}}{k} : \deg(C)=i} \det(T_{A,C}) \det(F_{A,C}) \prod_{\mathbf{e} \in C} L^{\mathbf{e}} \right) u^i.$$

Since all  $\ell_i$ 's are linearly independent, from Lemma 24, they are also algebraically independent. Therefore, the coefficient of  $u^{\deg(B)}$  in  $\det((TW(uL)F)_{A,[k]})$  is also a nonzero degree  $\deg(B)$  homogeneous polynomial in  $\mathbf{y}$ . Also,  $\deg(B) \leq Dk$ . Therefore, after substituting  $\mathbf{z}$  by  $L \circ \mathbf{g}$  in Equation 9, we get  $\det((TW(\mathbf{g}')F)_{A,[k]})$  which is a nonzero polynomial in  $(u, \mathbf{s}, \mathbf{t})$ . Since  $\det(W(\mathbf{g}'))$  is also a nonzero polynomial in  $(u, \mathbf{s}, \mathbf{t})$ ,  $\det(F'(\mathbf{g}')_{A,[k]})$  is nonzero in  $\mathbb{F}(u, \mathbf{s}, \mathbf{t})$ . This implies that  $G(\mathbf{x} + \mathbf{g}')$  has a cone-closed basis over  $\mathbb{F}(u, \mathbf{s}, \mathbf{t})$ .  $\blacktriangleleft$



# A New Notion of Commutativity for the Algorithmic Lovász Local Lemma

David G. Harris ✉

University of Maryland, College Park, MD, USA

Fotis Iliopoulos ✉

Institute for Advanced Study, Princeton, NJ, USA

Vladimir Kolmogorov ✉

Institute of Science and Technology, Klosterneuburg, Austria

---

## Abstract

The Lovász Local Lemma (LLL) is a powerful tool in probabilistic combinatorics which can be used to establish the *existence* of objects that satisfy certain properties. The breakthrough paper of Moser & Tardos and follow-up works revealed that the LLL has intimate connections with a class of stochastic local search algorithms for finding such desirable objects. In particular, it can be seen as a sufficient condition for this type of algorithms to converge fast.

Besides conditions for convergence, many other natural questions can be asked about algorithms; for instance, “are they parallelizable?”, “how many solutions can they output?”, “what is the expected “weight” of a solution?”. These questions and more have been answered for a class of LLL-inspired algorithms called commutative. In this paper we introduce a new, very natural and more general notion of commutativity (essentially matrix commutativity) which allows us to show a number of new refined properties of LLL-inspired local search algorithms with significantly simpler proofs.

**2012 ACM Subject Classification** Mathematics of computing → Probabilistic algorithms; Mathematics of computing → Combinatorics

**Keywords and phrases** Lovász Local Lemma, Resampling, Moser-Tardos algorithm, latin transversal, commutativity

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.31

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2008.05569>

**Funding** *Fotis Iliopoulos*: This material is based upon work directly supported by the IAS Fund for Math and indirectly supported by the National Science Foundation Grant No. 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 National Science Foundation. This work is also supported by the National Science Foundation Grant No. CCF-1815328.

*Vladimir Kolmogorov*: Supported by the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement no 616160.

## 1 Introduction

The Lovász Local Lemma (LLL) is a powerful tool in probabilistic combinatorics which can be used to establish the *existence* of objects that satisfy certain properties [9]. At a high level, it states that given a collection of bad events in a probability space  $\mu$ , if each bad-event is not too likely and, further, is independent of most other bad events, then the probability of avoiding all of them is strictly positive.



© David G. Harris, Fotis Iliopoulos, and Vladimir Kolmogorov;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 31; pp. 31:1–31:25



Leibniz International Proceedings in Informatics

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

In its simplest, “symmetric” form, it states that if each bad-event has probability at most  $p$  and is dependent with at most  $d$  others, where  $epd \leq 1$ , then with positive probability no bad-events become true. In particular, a configuration avoiding all the bad-events exists. Although the LLL applies to general probability spaces, most constructions in combinatorics use a simpler setting we refer to as the *variable version LLL*. Here, the probability space  $\mu$  is a cartesian product with  $n$  independent variables, and each bad-event is determined by a subset of the variables. Two bad-events may conflict if they depend on a common variable.

For example, consider a CNF formula with  $n$  variables where each clause has  $k$  literals and each variable appears in at most  $L$  clauses. For each clause  $c$  we can define the bad event  $B_c$  that  $c$  is violated in a chosen assignment of the variables. For a uniformly random variable assignment, each bad-event has probability  $p = 2^{-k}$  and affects at most  $d \leq kL$  others. So when  $L \leq \frac{2^k}{ek}$ , the formula is satisfiable; crucially, this criterion does not depend on the number of variables  $n$ .

A generalization known as the Lopsided LLL (LLLL) allows bad-events to be *positively* correlated with others; this is as good as independence for the purposes of the LLL. Some notable probability spaces satisfying the LLLL include the uniform distribution on permutations and the variable setting, where two bad-events  $B, B'$  are dependent only if they *disagree* on the value of a common variable.

In a seminal work, Moser & Tardos [25] presented a simple local search algorithm to make the variable-version LLLL constructive. This algorithm can be described as follows:

■ **Algorithm 1** The Moser-Tardos (MT) resampling algorithm.

- 
- 1: Draw the state  $\sigma$  from distribution  $\mu$
  - 2: **while** some bad-event is true on  $\sigma$  **do**
  - 3:     Select, arbitrarily, a bad-event  $B$  true on  $\sigma$
  - 4:     Resample, according to distribution  $\mu$ , all variables in  $\sigma$  affecting  $B$
- 

Moser & Tardos showed that if the symmetric LLL criterion (or more general asymmetric criterion) is satisfied, then this algorithm quickly converges. Following this work, a large effort has been devoted to making different variants of the LLL constructive. This research has taken many directions, including further analysis of Algorithm 1 and its connection to different LLL criteria [6, 22, 26].

One line of research has been to use variants of Algorithm 1 for general probability spaces beyond the variable LLL. These include applications of the LLL to permutations and matchings of the clique [1, 2, 17, 21, 19] as well as settings not directly connected to the LLL itself [3, 7, 18]. At a high level of generality, we summarize this in the following framework. There is a discrete state space  $\Omega$ , with a collection  $\mathcal{F}$  of subsets (which we call *flaws*) of  $\Omega$ . There is also some problem-specific randomized procedure called the *resampling oracle*  $\mathfrak{R}_f$  for each flaw  $f$ ; it takes some random action to attempt to “fix” that flaw, resulting in a new state  $\sigma' \leftarrow \mathfrak{R}_f(\sigma)$ . With these ingredients, we define the general local search algorithm as follows:

■ **Algorithm 2** The general local search algorithm.

- 
- 1: Draw the state  $\sigma$  from some distribution  $\mu$
  - 2: **while** some flaw holds on  $\sigma$  **do**
  - 3:     Select a flaw  $f$  of  $\sigma$ , according to some rule  $S$ .
  - 4:     Update  $\sigma \leftarrow \mathfrak{R}_f(\sigma)$ .
-

We refer to Algorithm 2 throughout as the *Search Algorithm*. The most important question about its behavior is whether it converges to a flawless object. But, there are other important questions to ask; for instance, “is it parallelizable?”, “how many solutions can it output?”, “what is the expected “weight” of a solution?”. These questions and more have been answered for the Moser-Tardos algorithm in a long series of papers [6, 8, 11, 12, 15, 16, 22, 25]. As a prominent example, the result of Haeupler, Saha and Srinivasan [12], as well as follow-up work of Harris and Srinivasan [14, 16], allows one to argue about the dynamics of Algorithm 1, resulting in several new applications such as estimating the entropy of the output distribution, partially avoiding bad events and dealing with super-polynomially many bad events.

There is one important difference between Algorithm 1 and Algorithm 2: the choice of which flaw to resample, if multiple flaws are simultaneously true. The flaw selection rule  $S$  in the Search Algorithm should select a flaw  $f \ni \sigma$  at each time  $t$ ; it may depend on the prior states and may be randomized. The original MT algorithm allows nearly complete freedom for this. For general resampling oracles,  $S$  is much more constrained; only a few relatively rigid rules are known to guarantee convergence, such as selecting the flaw of least index [19]. However, in [23], Kolmogorov identified a general property called *commutativity* that allows a free choice for  $S$ . This free choice, seemingly a minor detail, turns out to play a key role in extending the additional properties of the MT algorithm to the general Search Algorithm. For instance, it leads to parallel algorithms [23] and to bounds on the output distribution [20].

At a high level, our goal is to provide a more conceptual, algebraic explanation for the commutativity properties of resampling oracles and their role in the Search Algorithm. We do this by introducing a notion of commutativity, essentially matrix commutativity, that is both more general and simpler than the definition in [23]. Most of our results had already been shown, in slightly weaker forms, in prior works [23, 20, 14]. However, the proofs were computationally heavy and narrowly targeted to certain probability spaces, with numerous technical side conditions and restrictions.

Before we provide formal definitions, let us give some intuition. For each flaw  $f$ , consider an  $|\Omega| \times |\Omega|$  transition matrix  $A_f$ . Each row of  $A_f$  describes the probability distribution obtained by resampling  $f$  at a given state  $\sigma$ . We call the resampling oracle *commutative* if the transition matrices commute for any pair of flaws which are “independent” (in the LLL sense). We show a number of results for such oracles:

1. We obtain bounds on the distribution of the state at the termination of the Search Algorithm. These bounds are comparable to the *LLL-distribution*, i.e., the distribution induced by conditioning on avoiding all bad events. Similar results, albeit with a number of additional technical restrictions, had been shown in [20] for the original definition of commutativity.
2. For some probability spaces, stronger and specialized distributional bounds are available, beyond the “generic” LLL bounds [14]. Previously, these had been shown with ad-hoc arguments specialized to each probability space. Our construction recovers most of these results automatically.
3. We develop a generic parallel implementation of the Search Algorithm. This extends results of [23, 15], with simpler and more general proofs.
4. In many settings, flaws are formed from smaller “atomic” events [15]. We show that, if the atomic events satisfy the generalized commutativity definition, then so do the larger “composed” events. This natural property did not seem to hold for the original commutativity definition of [23].

## 1.1 Example application: latin transversals

As a motivating example, let us examine a classic combinatorial problem of latin transversals. Consider an  $n \times n$  array  $C$ , wherein each entry of  $C$  has a color. A latin transversal of  $C$  is a permutation  $\pi$  over  $\{1, \dots, n\}$  such that all the colors  $C(i, \pi i)$  are distinct for  $i = 1, \dots, n$ .

The “lopsided” variant of the LLL was first developed by Erdős & Spencer [10] for this problem. The underlying probability space is the uniform distribution on permutations. For each pair of cells  $(x_1, y_1), (x_2, y_2)$  of the same color, there is a corresponding flaw defined by  $\pi x_1 = y_1 \wedge \pi x_2 = y_2$ . They showed that if each color appears at most  $\Delta = \frac{n}{4e}$  times, then the LLL criteria are satisfied and a transversal exists. The cluster-expansion criterion [5] tightens this to  $\Delta = \frac{27n}{256}$ , which is the strongest bound currently known.

This construction has been a motivating example for much of the research on the algorithmic LLL, particularly for “exotic” probability spaces (beyond the variable setting). In particular, [17] showed that the Search Algorithm generates a latin transversal  $\pi$  under the same conditions as the existential LLLL. One of the main applications in this paper is to show that  $\pi$  has nice distributional properties. In particular, we show the following:

► **Theorem 1.** *If each color appears at most  $\frac{27}{256}n$  times in the array, then the Search Algorithm generates a latin transversal  $\pi$  where, for every pair  $(x, y)$ , we have  $0.53/n \leq \Pr(\pi x = y) \leq 1.36/n$ .*

The upper bound improves quantitatively over a similar bound of [14]; to the best of our knowledge, no non-trivial lower bound of any kind could previously be shown. Intriguingly, such bounds are not known to hold for the LLL-distribution itself.

To better situate Theorem 1, note that Alon, Spencer, & Tetali [4] showed that there is a (minuscule) universal constant  $\beta > 0$  with the following property. If each color appears at most  $\Delta = \beta n$  times in the array and  $n$  is a power of two, then the array can be *partitioned* into  $n$  independent transversals. In this case, if we randomly select one transversal from this list, we would have  $\Pr(\pi x = y) = 1/n$ . Theorem 1 can be regarded as a simplified *fractional* analogue of their result, i.e. we fractionally decompose the given array into  $O(n)$  transversals, such that  $\Pr(\pi x = y) = \Theta(1/n)$  for all pairs  $x, y$ . Furthermore, we achieve this guarantee automatically, merely by running the Search Algorithm.

## 1.2 Overview of our approach

Although it will require significant definitions and technical development to state our results formally, let us try to provide a high-level summary here. As a starting point, consider the MT algorithm. Moser & Tardos [25] used a construction referred to as a *witness tree* for the analysis: for each resampling of a bad-event  $B$  at a given time, there is a corresponding witness tree which records an “explanation” of why  $B$  was true at that time. More properly, it provides a history of all the prior resamplings which affected the variables involved in  $B$ .

The main technical lemma governing the behavior of the MT algorithm is the “Witness Tree Lemma,” which states that the probability of producing a given witness tree is at most the product of the probabilities of the corresponding events. The bounds on the algorithm runtime, as well as parallel algorithms and distributional properties, then follow from a union bound over witness trees.

Versions of this Witness Tree Lemma have been shown for some variants of the MT algorithm [13, 18] Iliopoulos [20] further showed that it held for general spaces which satisfy the commutativity property; this, in turn, leads to the nice algorithmic properties such as parallel algorithms.

Our main technical innovation is to generalize the Witness Tree Lemma. Instead of keeping track of a *scalar* product of probabilities in a witness tree, we instead consider a *matrix product*. We bound the probability of a given witness tree (or, more properly, a slight generalization known as the witness DAG) in terms of the products of the transition matrices of the corresponding flaws. Commutativity can thus be rephrased and simplified in terms of *matrix commutativity* for the transition matrices.

At the end, we obtain the scalar form of the Witness Tree Lemma by projecting everything to a one-dimensional space. For this, we take advantage of some methods of [3] for viewing the evolution of the Search Algorithm in terms of spectral bounds.

## 2 Background and Basic Definitions

Throughout the paper we consider implementations of the Search Algorithm. For each flaw  $f$ , state  $\sigma \in f$ , and state  $\sigma' \in \Omega$ , we define  $A_f[\sigma, \sigma']$  to be the probability that applying the resampling oracle  $\mathfrak{R}_f$  to  $\sigma$  yields state  $\sigma'$ , i.e.  $A_f[\sigma, \sigma'] = \Pr(\mathfrak{R}_f(\sigma) = \sigma')$ . For  $\sigma \notin f$ , we define  $A_f[\sigma, \sigma'] = 0$ . We sometimes write  $\sigma \xrightarrow{f} \sigma'$  to denote that the algorithm resamples flaw  $f$  at  $\sigma$  and moves to  $\sigma'$ . Observe that, for any vector  $\theta$  over  $\Omega$ , there holds  $\|\theta^\top A_f\|_1 = \sum_{\sigma \in f} \theta[\sigma] \leq \|\theta^\top\|_1$ . Thus, the matrix  $A_f$  is substochastic.

We define a *trajectory*  $T$  to be a finite or countably infinite sequence of the states and flaws of the form  $(\sigma_0, f_1, \sigma_1, f_2, \dots)$ , and  $\text{len}(T)$  is its *length* (possibly  $\text{len}(T) = \infty$ ). We define the *shift* of  $T$  to be  $(\sigma_1, f_2, \sigma_2, f_3, \dots)$ . We define  $\hat{T}$  to be the sequence states and flaws resampled during the Search Algorithm, i.e.  $\sigma_i$  is the state at time  $i$  and flaw  $f_i \in \sigma_i$  is the flaw resampled.

For our purposes, we use an *undirected* notion of dependence. Formally, we suppose that we have fixed some symmetric relation  $\sim$  on  $\mathcal{F}$ , with the property that  $f \sim f$  for all  $f$  and for every distinct pair of flaws  $f \not\sim g$ , we are guaranteed that resampling flaw  $f$  cannot introduce  $g$  or vice-versa, i.e.  $\mathfrak{R}_f$  never maps a state  $\Omega - g$  into  $g$  and likewise  $\mathfrak{R}_g$  never maps a state from  $\Omega - f$  into  $f$ . We define  $\bar{\Gamma}(f)$  to be the set of flaws  $g$  with  $f \sim g$ , and we also define  $\Gamma(f) = \bar{\Gamma}(f) \setminus \{f\}$ .

We say that a set  $I \subseteq \mathcal{F}$  is *stable* if  $f \not\sim g$  for all distinct pairs  $f, g \in I$ .

For an arbitrary event  $E \subseteq \Omega$ , we define  $e_E$  to be the indicator vector for  $E$ , i.e.  $e_E[\sigma] = 1$  if  $\sigma \in E$  and  $e_E[\sigma] = 0$  otherwise. For a state  $\sigma \in \Omega$ , we write  $e_\sigma$  as shorthand for  $e_{\{\sigma\}}$ , i.e. the basis vector which has a 1 in position  $\sigma$  and zero elsewhere.

For vectors or matrices  $u, v$  we write  $u \preceq v$  if  $u[i] \leq v[i]$  for all entries  $i$ . We write  $u \propto v$  if there is some scalar value  $c$  with  $u = cv$ .

**Regenerating oracles.** The Moser-Tardos algorithm and extensions to other probability spaces can be viewed in terms of *regenerating oracles* [19], i.e. each resampling action  $\mathfrak{R}_f$  should convert the distribution of  $\mu$  conditioned on  $f$  into the unconditional distribution  $\mu$ . We provide more detail later in Section 4, but, we can summarize this crisply with our matrix notation: the resampling oracle  $\mathfrak{R}$  is regenerating if  $\mu$  is a left-eigenvector of each matrix  $A_f$ , with associated eigenvalue  $\mu(f)$ , i.e.

$$\forall f \quad \mu^\top A_f = \mu(f) \cdot \mu^\top \tag{1}$$

### 2.1 The new commutativity definition

The original definition of commutativity given by Kolmogorov [23] required that for every  $f \approx g \in \mathcal{F}$ , there is an injective mapping from state transitions  $\sigma_1 \xrightarrow{f} \sigma_2 \xrightarrow{g} \sigma_3$  to state transitions  $\sigma_1 \xrightarrow{g} \sigma'_2 \xrightarrow{f} \sigma_3$ , so that  $A_f[\sigma_1, \sigma_2]A_g[\sigma_2, \sigma_3] = A_g[\sigma_1, \sigma'_2]A_f[\sigma'_2, \sigma_3]$ .

This definition is cumbersome, as well as lacking important symmetry and invariance properties. As one of the major contributions of this paper, we introduce a more natural notion of algorithmic commutativity that is also more general than the notion of [23].

► **Definition 2** (Transition matrix commutativity). *We say that the resampling oracle is transition matrix commutative with respect to dependence relation  $\sim$  if  $A_f A_g = A_g A_f$ , for every  $f, g \in \mathcal{F}$  such that  $f \approx g$ .*

► **Observation 3.** *If the resampling oracle is commutative in the sense of [23], then it is transition matrix commutative.*

**Proof.** Consider  $f \not\approx g$  and states  $\sigma, \sigma'$ . By symmetry, we need to show that  $A_f A_g[\sigma, \sigma'] \leq A_g A_f[\sigma, \sigma']$ . Since  $f \not\approx g$ , we can see that both the LHS and RHS are zero unless  $\sigma \in f \cap g$ .

Let  $V$  denote the set of states  $\sigma''$  with  $A_f[\sigma, \sigma''] A_g[\sigma'', \sigma'] > 0$ . By definition, there is an injective function  $F : V \rightarrow \Omega$  such that  $A_f[\sigma, \sigma''] A_g[\sigma'', \sigma'] = A_g[\sigma, F(\sigma'')] A_f[F(\sigma''), \sigma']$ . Therefore, we have  $(A_f A_g)[\sigma, \sigma'] = \sum_{\sigma'' \in V} A_f[\sigma, \sigma''] A_g[\sigma'', \sigma'] = \sum_{\sigma'' \in V} A_g[\sigma, F(\sigma'')] A_f[F(\sigma''), \sigma']$ .

Since  $F$  is injective, each term  $A_g[\sigma, \tau] A_f[\tau, \sigma']$  is counted at most once in this sum with  $\tau = F(\sigma'')$ . So  $(A_f A_g)[\sigma, \sigma'] \leq \sum_{\tau \in f} A_g[\sigma, \tau] A_f[\tau, \sigma'] = (A_g A_f)[\sigma, \sigma']$ . ◀

For brevity, we say *commutative* to mean transition matrix commutative throughout this paper; by contrast, we refer to the previous notion as *commutative in the sense of [23]*.

When this definition applies, we define  $A_I$  to be the matrix  $\prod_{f \in I} A_f$  for a stable set  $I$ ; note that this product is well-defined (without specifying ordering of  $I$ ) since the matrices  $A_f$  all commute.

**For the remainder of this paper, we assume that the resampling oracle  $\mathfrak{R}$  is transition-matrix commutative unless explicitly stated otherwise.**

### 3 Witness DAGs and matrix bounds

In this section, we study *witness DAGs*, a key graph structure developed in [11] for analyzing the evolution of commutative resampling oracles. The main result of this section is Lemma 5, which is a generalization of the Witness Tree Lemma described in the introduction. Notably, while our result is more general, its proof is significantly simpler. At a high level, the role of a witness DAG is to give an “explanation” of why a certain flaw appeared during the execution of the algorithm. To bound the probability that flaw  $f$  appears during the algorithm execution, we simply add up the probabilities of all the witness DAGs that explain it.

Formally, we define a *witness DAG* (abbreviated *wdag*) to be a directed acyclic graph  $G$ , where each vertex  $v \in G$  has a label  $L(v)$  from  $\mathcal{F}$ , and such that for all pairs of vertices  $v, w \in G$ , there is an edge between  $v$  and  $w$  (in either direction) if and only if  $L(v) \sim L(w)$ . For a wdag  $G$  with sink nodes  $v_1, \dots, v_k$ , note that  $L(v_1), \dots, L(v_k)$  are all distinct and  $\{L(v_1), \dots, L(v_k)\}$  is a stable set which we denote by  $\text{sink}(G)$ . We say that a flaw  $f$  is *unrelated* to a wdag  $G$  if there is no node  $v \in G$  with  $L(v) \sim f$ .

We define  $\mathfrak{H}$  to be the collection of all wdaGs, and we define  $\mathfrak{H}(I)$  to be the collection of all such wdaGs with  $\text{sink}(H) = I$ . With some abuse of notation, we also write  $\mathfrak{H}(f)$  as shorthand for  $\mathfrak{H}(\{f\})$ .

There is a key connection between wdaGs and the transition matrices. For any wdag  $H$ , we define an associated  $|\Omega| \times |\Omega|$  matrix  $A_H$  inductively as follows. If  $H = \emptyset$ , then  $A_H$  is the identity matrix on  $\Omega$ . Otherwise, we choose an arbitrary source node  $v$  of  $H$  and set  $A_H = A_{L(v)} A_{H-v}$ .



► **Proposition 4.** *The definition of  $A_H$  does not depend on the chosen source node  $v$ . Furthermore, there is an enumeration of the nodes of  $H$  as  $v_1, \dots, v_t$  such that  $A_H = A_{L(v_1)} \cdots A_{L(v_t)}$ .*

**Proof.** Let us show the first property by induction on  $|H|$ . When  $|H| = 0$  this is vacuously true. For induction case, suppose  $H$  has two source nodes  $v_1, v_2$ . We need to show that we get the same value by recursing on  $v_1$  or  $v_2$ , i.e.  $A_{L(v_1)}A_{H-v_1} = A_{L(v_2)}A_{H-v_2}$ .

We can apply the induction hypothesis to  $H - v_1$  and  $H - v_2$ , noting that  $v_2$  is a source node of  $H - v_1$  and  $v_1$  is a source node of  $H - v_2$ . We get  $A_{H-v_1} = A_{L(v_2)}A_{H-v_1-v_2}$ ,  $A_{H-v_2} = A_{L(v_1)}A_{H-v_1-v_2}$ . Thus, in order to show  $A_{L(v_1)}A_{H-v_1} = A_{L(v_2)}A_{H-v_2}$ , it suffices to show that  $A_{L(v_1)}A_{L(v_2)} = A_{L(v_2)}A_{L(v_1)}$ . Since  $v_1, v_2$  are both source nodes, we have  $L(v_1) \not\sim L(v_2)$ . Thus, this follows from commutativity.

For the second property, we have  $A_H = A_{L(v_1)}A_{H-v_1}$  for a source node  $v$ . Recursively peeling away vertices gives  $A_H = A_{L(v_1)}A_{L(v_2)} \cdots A_{L(v_t)}$ . ◀

As a warm-up, we first show how to use wdags to bound the number of resamplings performed in commutative algorithms. This will allow us to show bounds on the expected runtime of the Search Algorithm as well as allowing parallel implementations. The main point here is to demonstrate how the new commutativity definition helps with the crucial task of bounding the probability of appearance of a given wdag.

As in the original proof of Moser & Tardos [25], we will estimate the expected number of times each flaw  $f \in F$  is resampled. Consider an execution of the Search Algorithm with trajectory  $T$ . For each time  $t \leq \text{len}(T)$ , we generate a corresponding wdag  $G_t^T$  which provides the history of the  $t^{\text{th}}$  resampling. Initially, we set  $G_t^T$  to consist of a singleton node labeled  $f_t$ . Then, for  $s = t - 1, \dots, 1$ , there are two cases:

1. if wdag  $G_t^T$  has any node with label  $g$  where  $g \sim f_s$ , then we add a vertex labeled  $f_s$ , with a sink node to all nodes  $w$  such that  $L(w) \sim f_s$ ;
2. Otherwise, if  $G_t^T$  is unrelated to  $f_s$ , then we do not modify  $G_t$ .

We define  $G_{[s,t]}^T$  to be the partial wdag formed only by considering times  $t, \dots, s$ ; then  $G_t^T = G_{[1,t]}^T$  and  $G_{[t,t]}^T$  is a singleton node labeled  $f_t$  and  $G_{[s,t]}^T$  is formed by copying  $G_{[s+1,t]}^T$  and adding, or not, a node labeled  $f_s$ . We say that a wdag  $H$  appears if  $H$  is isomorphic to  $G_t$  for any value  $t$ ; with a slight abuse of notation, we write this as  $G_t = H$ .

To calculate the expected running time of the Search Algorithm, we sum the wdag appearance probabilities. One of the main ingredients in the original proof of Moser & Tardos is that any wdag  $G$  appears with probability at most  $\prod_{v \in G} \mu(L(v))$ , i.e., the product of probabilities of the flaws that label its vertices. Their proof depends on properties of the variable setting and does not extend to other probability spaces.

Our key message is that commutativity allows us to bound the probability of a wdag appearing by considering the product of transition matrices for flaws that label its vertices. Specifically, we show the following. (Recall that  $\mu$  denotes the initial state distribution.)

► **Lemma 5.** *For any wdag  $H$ , the probability that  $H$  appears is at most  $\mu^\top A_H \vec{1}$ .*

**Proof.** We first show that if the Search Algorithm runs for at most  $t_{\max}$  steps starting with state  $\sigma$ , where  $t_{\max}$  is an arbitrary integer, then  $H$  appears with probability at most  $e_\sigma^\top A_H \vec{1}$ . We prove this claim by induction on  $t_{\max}$ . If  $t_{\max} = 0$ , or  $\sigma$  is flawless, the claim can be easily seen to hold vacuously.

So suppose that  $t_{\max} \geq 1$  and  $S$  selects a flaw  $g$  to resample in  $\sigma$ , and define  $\mathcal{E}_H$  to be the event that  $H$  appears when running the search algorithm  $\mathcal{A}$ . By conditioning on the random seed used by the flaw choice strategy  $S$  (if any), we may assume that the search strategy  $S$  is deterministic.

We can now view the evolution of  $\mathcal{A}$  as a two-part process: we first resample  $g$ , reaching state  $\sigma'$  with probability  $A_g[\sigma, \sigma']$ . We then execute a new search algorithm  $\mathcal{A}'$  starting at state  $\sigma'$ , wherein the flaw selection rule  $S'$  on history  $(\sigma', \sigma_2, \dots, \sigma_t)$  is the same as the choice of  $S$  on history  $(\sigma, \sigma', \sigma_2, \dots, \sigma_t)$ . Let us denote by  $G'_s$  the wdags produced for this new search algorithm  $\mathcal{A}'$ .

Suppose that  $H$  appears, so that  $G_s = H$  for some value  $s \leq t_{\max}$ . In this case, we claim that one of the two conditions must hold: (i)  $H$  has a unique source node  $v$  labeled  $g$  and  $G'_{s-1} = H - v$ ; or (ii)  $g$  is unrelated to  $H$  and  $G'_{s-1} = H$ . For, suppose that  $H$  has another node  $w$  with  $L(w) \sim L(v)$ ; in this case, when forming the wdag  $G_s$ , the rule would be to add a new node labeled  $g$ , which is perforce a source node.

In case (i), then in order for event  $\mathcal{E}_H$  to occur on the original search algorithm  $\mathcal{A}$ , we must also have  $\mathcal{E}_{H-v}$  hold on  $\mathcal{A}'$  within  $t-1$  timesteps. By induction hypothesis, this has probability at most  $e_{\sigma'}^\top A_{H-v} \vec{1}$  for a fixed  $\sigma'$ . Summing over  $\sigma'$  gives a total probability of  $\sum_{\sigma'} A_g[\sigma, \sigma'] e_{\sigma'}^\top A_{H-v} \vec{1} = e_\sigma^\top A_g A_{H-v} \vec{1} = e_\sigma^\top A_H \vec{1}$  as required.

In case (ii), then in order for event  $\mathcal{E}_H$  to occur for  $\mathcal{A}$ , we must also have  $\mathcal{E}_H$  occur for  $\mathcal{A}'$  within  $t-1$  timesteps. By induction hypothesis, this has probability at most  $e_{\sigma'}^\top A_H \vec{1}$  for a fixed  $\sigma'$ . Summing over  $\sigma'$  gives a total probability of  $\sum_{\sigma'} A_g[\sigma, \sigma'] e_{\sigma'}^\top A_H \vec{1} = e_\sigma^\top A_g A_H \vec{1}$ . Since  $A_g$  commutes with  $A_H$ , this is at most  $e_\sigma^\top A_H A_g \vec{1}$ . Since  $A_g$  is substochastic, this in turn is at most  $e_\sigma^\top A_H \vec{1}$ , which completes the induction.

By countable additivity, we can compute the probability that  $H$  ever appears from starting state  $\sigma$ , as  $\Pr(\bigvee_{t=1}^{\infty} G_t^{\hat{T}} = H) = \lim_{t_{\max} \rightarrow \infty} \Pr(\bigvee_{t=1}^{t_{\max}} G_t^{\hat{T}} = H) \leq \lim_{t_{\max} \rightarrow \infty} e_\sigma^\top A_H \vec{1} = e_\sigma^\top A_H \vec{1}$ .

Finally, integrating over  $\tau$ , gives  $\sum_{\tau} \mu[\tau] e_\tau^\top A_H \vec{1} = \mu^\top A_H \vec{1}$ .  $\blacktriangleleft$

This can be used to show a generalization of the key Witness Tree Lemma of Moser & Tardos:

► **Corollary 6.** *Suppose the resampling oracle is regenerating. Then, for a given wdag  $H$ , the probability that  $H$  appears is at most  $\prod_{v \in H} \mu(L(v))$ .*

**Proof.** Let  $f_1, \dots, f_t$  be the labels of the vertices in  $H$ , ordered from source nodes to sink nodes. We can write  $A_H = A_{f_1} \cdots A_{f_t}$ . Since  $\mu$  is a left-eigenvector of every transition matrix (see Eq. (1)), we have  $\mu^\top A_H \vec{1} = \mu^\top A_{f_1} \cdots A_{f_t} \vec{1} = \mu(f_1) \cdots \mu(f_t) \mu^\top \vec{1} = \mu(f_1) \cdots \mu(f_t)$ .  $\blacktriangleleft$

As we have already discussed, this gives the following important corollary:

► **Corollary 7.** *The expected number of steps of the Search Algorithm is at most  $\sum_{\substack{f \in \mathcal{F} \\ H \in \mathfrak{H}(f)}} \mu^\top A_H \vec{1}$ .*

For example, if the resampling oracle is regenerating, then, the expected number of steps of the algorithm is at most  $\sum_{f \in \mathcal{F}} \sum_{H \in \mathfrak{H}(f)} \prod_{v \in H} \mu(L(v))$ , i.e. the usual Witness Tree Lemma.

We emphasize that we are not aware of any direct proof of Corollary 6; it seems necessary to first show the matrix bound of Lemma 5, and then project down to scalars. As we show in Appendix A, under some natural conditions the matrix commutativity property is necessary to obtain Lemma 5.

#### 4 Estimating weights of wdags

The statement of Lemma 5 in terms of matrix products is very general and powerful, but difficult for calculations. To use it effectively, we need to bound the sums of the form

$$\sum_{H \in \mathfrak{H}} \mu^\top A_H \vec{1}$$

There are two, quite distinct, issues that arise in this calculation. First, for a given fixed wdag  $H$ , we need to estimate  $\mu^\top A_H \vec{1}$ ; second, we need to bound the sum of these quantities over  $H$ . The second issue is well-studied and is at the heart of the probabilistic and algorithmic conditions for the LLL. The first issue is not as familiar. Following [3], we can bound the matrix product by using a heuristic based on spectral bounds of the matrices  $A_f$ . Let us define a quantity called the *charge*  $\gamma_f$  for each flaw  $f$  as follows.

$$\gamma_f = \max_{\tau \in \Omega} \sum_{\sigma \in f} \frac{\mu(\sigma)}{\mu(\tau)} A_f[\sigma, \tau] \quad (2)$$

The following result of [21] illustrates the connection between this measure and the Lopsided Lovász Local Lemma (LLLL):

► **Theorem 8** ([21]). *For each set  $S \subseteq \mathcal{F} - \Gamma(f)$ , there holds  $\mu(f \mid \bigcap_{g \in S} \bar{g}) \leq \gamma_f$ .*

Moreover, as shown in [2], the charge  $\gamma_f$  captures the compatibility between resampling oracle for  $f$  and the measure  $\mu$ . A resampling oracle  $\mathfrak{R}$  with  $\gamma_f = \mu(f)$  for all  $f$ , is called a *regenerating oracle* [19], as it perfectly removes the conditional of the resampled flaw. (This is equivalent to satisfying Eq. (1).)

For a wdag  $H$ , let us define the scalar value  $w(H) = \prod_{v \in H} \gamma_{L(v)}$ . We get the following estimate for  $\mu^\top A_H \vec{1}$  in terms of  $w(H)$ :

► **Theorem 9.** *For any event  $E \subseteq \Omega$  we have  $\mu^\top A_H e_E \leq \mu(E) \cdot w(H)$ . In particular, with  $E = \Omega$ , we have  $\mu^\top A_H \vec{1} \leq w(H)$ .*

**Proof.** From definition of  $\gamma_f$ , it can be observed that  $\mu^\top A_f \preceq \gamma_f \mu^\top$  for any  $f$ . In particular,  $\mu^\top A_f \cdot \theta \leq \gamma_f \theta$  for any vector  $\theta$ . Now, by Proposition 4, we can write  $A_H = A_{f_1} \dots A_{f_t}$  where  $f_1, \dots, f_t$  are the labels of the nodes of  $H$ . We thus have:

$$\mu^\top A_H e_E = \mu^\top A_{f_1} \dots A_{f_t} e_E \leq \mu^\top \gamma_{f_1} A_{f_2} \dots A_{f_t} \leq \dots \leq \gamma_{f_1} \dots \gamma_{f_t} \mu^\top e_E = w(H) \mu(E) \blacktriangleleft$$

In light of Theorem 9, we define for any stable set  $I$  the key quantity  $\Psi(I) = \sum_{H \in \mathfrak{H}(I)} w(H)$ . We also define  $\Psi(f) = \Psi(\{f\})$  for brevity.

► **Corollary 10.**

1. *Any given wdag  $H$  appears with probability at most  $w(H)$ .*
2. *The expected number of resamplings of any flaw  $f$  is at most  $\Psi(f)$ .*
3. *The expected runtime of the Search Algorithm is at most  $\sum_f \Psi(f)$ .*

We summarize a few well-known bounds on these quantities.

► **Proposition 11.**

1. *(Symmetric criterion) Suppose that  $\gamma_f \leq p$  and  $|\bar{\Gamma}(f)| \leq d$  for parameters  $p, d$  with  $epd \leq 1$ . Then  $\Psi(f) \leq e\gamma_f \leq ep$  for all  $f$ .*
2. *(Asymmetric criterion) Suppose there is some function  $x : \mathcal{F} \rightarrow [0, 1)$  with the property that  $\gamma_f \leq x(f) \prod_{g \in \Gamma(f)} (1 - x(g))$  for all  $f$ . Then  $\Psi(f) \leq \frac{x(f)}{1-x(f)}$  for all  $f$ .*

3. (*Cluster-expansion criterion*) Suppose there is some function  $\eta : \mathcal{F} \rightarrow [0, \infty)$  with the property that  $\eta(f) \geq \gamma_f \cdot \sum_{\substack{I \subseteq \bar{\Gamma}(f) \\ I \text{ stable}}} \prod_{g \in I} \eta(g)$  for all  $f$ . Then  $\Psi(f) \leq \eta(f)$  for all  $f$ .

A related quantity is  $\bar{\Psi}(I) = \sum_{J \subseteq I} \Psi(J)$ . A useful and standard formula (see e.g., [19, Claim 59]) is that for any set  $I$  we have  $\Psi(I) \leq \prod_{f \in I} \Psi(f)$  and  $\bar{\Psi}(I) \leq \prod_{f \in I} (1 + \Psi(f))$ . We also write  $\Psi_{\mathcal{F}}, \bar{\Psi}_{\mathcal{F}}$  to indicate the role of the flaw set  $\mathcal{F}$ , if it is relevant.

As an illustration, consider latin transversals. Here, we have a flaw  $f$  for each pair of cells  $(x_1, y_1), (x_2, y_2)$  of the same color, i.e.  $\pi x_1 = y_1 \wedge \pi x_2 = y_2$ . We denote this by flaw  $[(x_1, y_1), (x_2, y_2)]$ . We define the dependency graph by setting  $f \sim f'$  if and only if  $f$  and  $f'$  are mutually incompatible, i.e.  $f = [(x_1, y_1), (x_2, y_2)], f' = [(x'_1, y'_1), (x'_2, y'_2)]$  where either  $x_1 = x'_1, y_1 \neq y'_1$  or  $x_1 \neq x'_1, y_1 = y'_1$ . We will examine this construction in more detail later in Section 6.

► **Proposition 12.** *Suppose that each color appears at most  $\Delta = \frac{27}{256}n$  times in the array. Then the expected number of steps of the Search Algorithm is  $O(n)$ . Furthermore,  $\Psi(f) \leq \frac{256}{81n^2}$  for each  $f$ .*

**Proof.** We apply the cluster-expansion criterion with  $\eta(f) = \frac{256}{81n^2}$  for each flaw  $f$ . Consider a flaw  $f$  corresponding to cells  $(x_1, y_1), (x_2, y_2)$ , and a stable set  $I$  of neighbors of  $f$ . There can be one or zero elements  $g$  of  $I$  of the form  $[(x_1, y'_1), (x'_2, y'_2)]$ . There are  $n$  choices for  $x_1$ ; given that pair  $(x_1, y'_1)$  is determined, there are at most  $\Delta - 1$  other cells with the same color. Each such  $g$  has  $\eta(g) = \frac{256}{81n^2}$ . Similar arguments apply to elements in  $I$  of the form  $[(x'_1, y_1), (x_2, y'_2)]$  etc. Overall, the sum over stable neighbor sets  $I$  is at most  $(1 + n(\Delta - 1)\frac{256}{81n^2})^4$ .

So we need to show that

$$\frac{256}{81n^2} \geq \frac{1}{n^2} \cdot (1 + n(\Delta - 1)\frac{256}{81n^2})^4$$

and simple calculations show that this holds for  $n \geq 2$ . (The case  $n = 1$  holds trivially).

Also, the total number of flaws is at most  $n^2(\Delta - 1)/2 = O(n^3)$ . Thus, the expected number of steps is at most  $|\mathcal{F}| \cdot \frac{256}{81n^2} \leq O(n)$ . ◀

## 5 Parallel algorithms

Moser & Tardos [25] described a simple parallel version of their resampling algorithm. A variety of parallel resampling algorithms have also been developed for other probability spaces [17, 13]. One main benefit of the commutativity property is that it enables much more general parallel implementations of the Search Algorithm. As a starting point, [23] discussed a generic framework for parallelization which we summarize as follows:

■ **Algorithm 3** Generic parallel resampling framework.

- 
- 1: Draw state  $\sigma$  from distribution  $\mu$
  - 2: **while** some flaw holds on  $\sigma$  **do**
  - 3:     Set  $V \neq \emptyset$  to be the set of flaws currently holding on  $\sigma$
  - 4:     **while**  $V \neq \emptyset$  **do**
  - 5:         Select, arbitrarily, a flaw  $f \in V$ .
  - 6:         Update  $\sigma \leftarrow \mathfrak{R}_\sigma(\sigma)$ .
  - 7:         Remove from  $V$  all flaws  $g$  such (i)  $\sigma \notin g$ ; or (ii)  $f \sim g$
-

Each iteration of the main loop (lines 2 – 7) is called a *round*. We emphasize this is a *sequential* algorithm, which can be viewed as a version of the Search Algorithm with an unusual flaw-selection choice. Most known parallel local search algorithms, including the original parallel algorithm of Moser & Tardos, fall into this framework. One main result of [23] is that, when the resampling oracle is commutative (in the sense of [23]), then the total number of rounds in Algorithm 3 is polylogarithmic with high probability.

Harris [15] further showed a general method for simulating each round in parallel, for resampling oracles which satisfy a property called *obliviousness* (see Section 7 for a formal definition). These two results combine to give an overall RNC search algorithm. We will now extend these results to our commutative resampling oracles, via bounding the weights of certain classes of wdags.

We define  $V_k$  to be the set of flaws  $V$  in round  $k$ , and we define the stable set  $I_k$  to be the set of flaws which are actually resampled at round  $k$  (i.e. a flaw  $f$  selected at some iteration of line 5). Let  $b_k = \sum_{i < k} |I_i|$  be the total number of resamplings made before round  $k$ ; thus  $b_1 = 0$ , and when “serialize” Algorithm 3 and view it as an instance of the Search Algorithm, the resamplings in round  $k$  of Algorithm 3 correspond to the resamplings at iterations  $b_k + 1, \dots, b_{k+1}$  of the Search Algorithm.

► **Proposition 13.** *For all  $f \in V_k$  there exists  $g \in I_{k-1}$  with  $f \sim g$ .*

► **Proposition 14.** *Consider running Algorithm 3 obtaining trajectory  $\hat{T}$ . Then, for each  $t$  in the range  $b_k + 1, \dots, b_k$  the wdag  $G_t^{\hat{T}}$  has depth precisely  $k$ .*

The proof of Propositions 13 and 14 are nearly identical to results for the variable LLLL shown in [15]; we omit them here.

► **Proposition 15.** *For any  $f \in \mathcal{F}$  and index  $k \geq 1$ , we have  $\Pr(f \in V_k) \leq \sum_{\substack{H \in \mathfrak{H}(f) \\ \text{depth}(H)=k}} \mu^\top A_H \vec{1}$ .*

**Proof.** As we have discussed, Algorithm 3 can be viewed as an instantiation of the Search Algorithm with a flaw selection rule  $S$ . For a fixed  $f$ , let us define a new flaw selection rule  $S_f$  as follows: it agrees with  $S$  up to round  $k$ ; it then selects  $f$  to resample at round  $k$  if it is true. The behavior for  $S$  and  $S_f$  is identical up through the first  $b_{k-1}$  resamplings. Furthermore, Algorithm 3 has  $f \in V_k$  if and only if the Search Algorithm selects  $f$  for resampling at iteration  $b_k + 1$ .

Consider the resulting wdag  $G_t^{\hat{T}}$ ; by Proposition 14 it has depth  $k$ . Furthermore, it has a sink node labeled  $f$ . Thus, if  $f \in V_k$ , then there is some  $H \in \mathfrak{H}(f)$  with  $\text{depth}(H) = k$  which appears. To bound the probability of  $f \in V_k$ , we take a union bound over all such  $H$  and apply Lemma 5. ◀

The usual strategy to bound the sum over wdags  $H$  with  $\text{depth}(H) \geq t$  is to use an “inflated” weight function defined as  $w_\epsilon(H) = w(H)(1 + \epsilon)^{|H|}$ , and corresponding sum  $W_\epsilon = \sum_{H \in \mathfrak{H}} w_\epsilon(H)$  for some  $\epsilon > 0$ . Using standard calculations as well as the bounds of Propositions 13, 14, 15, one can show the following results:

► **Proposition 16.** *With probability at least  $1 - \delta$ , Algorithm 3 terminates in  $O(\frac{\log(1/\delta + \epsilon W_\epsilon)}{\epsilon})$  rounds and has  $\sum_k |V_k| \leq O(W_\epsilon/\delta)$ . Furthermore, if the resampling oracle is regenerating and satisfies the computational requirements given in [15] for input length  $n$ , then with probability  $1 - 1/\text{poly}(n)$  the algorithm of [15] terminates in  $O(\frac{\log^4(n + \epsilon W_\epsilon)}{\epsilon})$  time on an EREW PRAM.*

## 31:12 A New Notion of Commutativity for the Algorithmic Lovász Local Lemma

Bounding  $W_\epsilon$  is very similar to bounding  $\sum_H w(H) = W_0$ , except with a small “slack” in the charges. For example, using standard estimates (see [11, 23, 3]) we get the following bounds:

► **Proposition 17.**

1. Suppose that the resampling oracle is regenerating and that the vector of probabilities  $p(1+\epsilon)$  still satisfies the LLLL criterion. Then  $W_{\epsilon/2} \leq O(m/\epsilon)$ . In particular, Algorithm 3 terminates after  $O(\frac{\log(m/\delta)}{\epsilon})$  rounds with probability  $1 - \delta$ .
2. Suppose that  $\gamma_f \leq p$  and  $|\bar{\Gamma}(f)| \leq d$  such that  $\text{epd}(1 + \epsilon) \leq 1$ . Then  $W_{\epsilon/2} \leq O(m/\epsilon)$ . Algorithm 3 terminates after  $O(\frac{\log(m/\delta)}{\epsilon})$  rounds with probability at least  $1 - \delta$ .

## 6 Distributional properties

The most important consequence of commutativity is that it leads to good bounds on the distribution of objects generated by the Search Algorithm. Consider an event  $E$  in  $\Omega$ , and define  $P(E)$  to be the probability that  $E$  holds in the output of the Search Algorithm. Also define  $N(E)$  to be the expected number of times  $t$  such that  $E$  is caused to be true at time  $t$ ; this includes both the original sampling at time  $t = 0$ , or if resampling flaw  $f$  at time  $t$  moved the state from  $\bar{E}$  to  $E$ . Clearly, there holds  $P(E) \leq N(E)$ . We also write  $P_{\mathcal{F}}(E)$  and  $N_{\mathcal{F}}(E)$  to emphasize the dependence on flaw set  $\mathcal{F}$ .

To obtain the tightest bounds on  $N(E)$ , and thereby  $P(E)$ , we will use a more refined construction of wdags. For this we need several definitions.

We say that a stable set  $I \subseteq \mathcal{F}$  is *orderable* for  $E$  if there is an enumeration  $I = \{g_1, \dots, g_r\}$  such that

$$\forall i = 1, \dots, r \quad A_{g_i} A_{g_{i+1}} \dots A_{g_r} e_E \not\subseteq A_{g_{i+1}} \dots A_{g_r} e_E \quad (3)$$

We define  $\mathfrak{D}(E)$  to be the collection of stable sets orderable for  $E$ . Also, we define  $\tilde{\Gamma}(E)$  to be the set of flaws  $f \in \mathcal{F}$  which can cause  $E$  to occur, i.e.  $f$  maps some state  $\sigma' \notin E$  to  $\sigma \in E$ .

► **Observation 18.** If  $I \in \mathfrak{D}(E)$ , then  $I \subseteq \tilde{\Gamma}(E)$ .

With this notation, our main distributional bound will be to show that

$$N(E) \leq \mu(E) \sum_{I \in \mathfrak{D}(E)} \Psi(I)$$

For a flaw  $f$  and wdag  $H$ , we say that  $f$  is *dominated by a wdag  $H$  for  $E$*  if  $A_f A_H \vec{1} \preceq A_H \vec{1}$ . Consider a trajectory  $T = (\sigma_0, f_1, \dots)$ . For each time  $t$  where  $E$  holds (including possibly  $t = 0$ ), we will generate a corresponding wdag  $J_t^T$ , however, the rule for adding nodes is slightly more restrictive. See Algorithm 4 for the precise construction.

■ **Algorithm 4** Forming  $J_t^T$ .

- 
- 1: Initialize  $J_t^T = \emptyset$
  - 2: **for**  $s = t, \dots, 1$  **do**
  - 3:     **if**  $f_s$  is not dominated by  $J_t^T$  or if  $J_t^T$  has a source node labeled  $f_s$  **then**
  - 4:         Add to  $J_t^T$  a node  $v_s$  labeled  $f_s$ , with an edge from  $v_s$  to each  $v_j$  such that  $f_j \sim f_s$
- 

We write  $J_{[s,t]}^T$  for the wdag  $J_t^T$  after iteration  $s$ , so that  $J_{[s,t]}^T$  is derived from  $J_{[s+1,t]}^T$  by adding (or not) a vertex labeled  $f_s$ . We have  $J_t^T = J_{[1,t]}^T$  and  $J_{[t+1,t]}^T = \emptyset$ . Also, if  $E$  is not true at time  $t$ , we define  $J_t^T = \perp$ . We define  $\mathfrak{J}'$  to be the collection of all wdags that can be produced in this process.

► **Proposition 19.** *Consider a wdag  $G \in \mathfrak{S}'$ . If  $J_t^T = G$  for a trajectory  $T$  with  $t \geq 1$  and  $f_1$  resampled at time 1, then the wdag  $G' = J_{t-1}^{T'}$  for the shifted trajectory  $T'$  is uniquely determined according to the following rule:*

- *If  $G$  contains a unique source node  $v$  labeled  $f_1$ , then  $G' = G - v$*
- *Otherwise,  $G' = G$  and  $f_1$  is dominated by  $G_t^T$*

**Proof.** Since  $t \geq 1$ , Algorithm 4 obtains  $J_t^T$  by possibly adding a node  $v_1$  labeled  $f_1$  to  $J_{t-1}^{T'}$ . If Algorithm 4 adds node  $v_1$  to  $G'$ , then  $f_1$  is the label of a source node  $v$  of  $J_t^T = G$ , and  $G' = J_t^T - v$ . If Algorithm 4 does not add such node, then  $J_t^T = G'$ . By the rule for adding nodes, it must be that  $G'$  does not have a source node labeled  $f_1$ , and also  $f_1$  must be dominated by  $J_{t-1}^{T'}$ . Since  $G' = J_t^T$ , these imply that  $f_1$  is dominated by  $J_t^T = G$  as well. ◀

Our main result for this construction will be the following:

► **Lemma 20.** *For any wdag  $H \in \mathfrak{S}'$ , there holds  $\Pr(\bigvee_{t=0}^{\infty} J_t = H) \leq \mu^\top A_H e_E$*

**Proof.** Define  $\mathcal{E}_{H,t_{\max}}$  to be the event that  $J_t^{\hat{T}} = H$  holds for some  $t \leq t_{\max}$  during execution of the Search Algorithm  $\mathcal{A}$ . By a limiting argument, it suffices to show that  $\Pr(\mathcal{E}_{H,t_{\max}}) \leq \mu^\top A_H e_E$  for any integer  $t_{\max} \geq 1$ . We will prove by induction on  $t_{\max}$  that, if we start at any state  $\sigma$ , then  $\Pr(\mathcal{E}_{H,t_{\max}}) \leq e_\sigma^\top A_H e_E$ ; the Lemma then follows by integrating over starting state  $\sigma$ .

If  $t_{\max} = 0$  or  $\sigma$  is flawless, then  $\mathcal{E}_{H,t_{\max}}$  is impossible and the desired bound. So suppose that  $t_{\max} \geq 1$ , and that  $S$  selects a flaw  $g$  to resample in  $\sigma$ . We can now view the evolution of  $\mathcal{A}$  as a two-part process: we first resample  $g$ , reaching state  $\sigma'$  with probability  $A_g[\sigma, \sigma']$ . We then execute a new search algorithm  $\mathcal{A}'$  starting at state  $\sigma'$ , wherein the flaw selection rule  $S'$  on history  $(\sigma', \sigma_2, \dots, \sigma_r)$  is the same as the choice of  $S$  on history  $(\sigma, \sigma', \sigma_2, \dots, \sigma_r)$ .

Suppose now that  $\mathcal{E}_{H,t_{\max}}$  holds for  $\mathcal{A}$ , i.e.  $J_t^{\hat{T}} = H$  for some  $t \leq t_{\max}$ . Note that the actual trajectory  $\hat{T}'$  for  $\mathcal{A}'$  is given by  $\hat{T}'$  which is the shift of  $\hat{T}$ . Thus, by Proposition 19, one of the two conditions must hold: (i) either  $H$  has a unique source node  $v$  labeled  $g$  and  $J_{t-1}^{T'} = H - v$ ; or (ii)  $H$  has no such node and  $J_{t-1}^{T'} = H$  and  $g$  is dominated by  $H$ .

In the first case, there must also hold  $\mathcal{E}_{H-v,t_{\max}-1}$  for  $\mathcal{A}'$ . By induction hypothesis, this has probability at most  $e_{\sigma'}^\top A_H e_E$  conditional on a fixed  $\sigma'$ . Summing over  $\sigma'$ , we get a total probability of  $\sum_{\sigma'} A_g[\sigma, \sigma'] e_{\sigma'}^\top A_H e_E = e_\sigma^\top A_g A_H e_E = e_\sigma^\top A_H e_E$ .

In the second case, there must also hold  $\mathcal{E}_{H,t_{\max}-1}$  for  $\mathcal{A}'$ . By induction hypothesis, this has probability at most  $e_{\sigma'}^\top A_H e_E$  conditional on a fixed  $\sigma'$ . Summing over  $\sigma'$ , we get a total probability of  $\sum_{\sigma'} A_g[\sigma, \sigma'] e_{\sigma'}^\top A_H e_E = e_\sigma^\top A_g A_H e_E$ . Since  $g$  is dominated by  $H$  for  $E$ , this is at most  $e_\sigma^\top A_H e_E$ , again completing the induction. ◀

► **Proposition 21.** *Suppose that event  $E$  is true at times  $s$  and  $t$  with  $s < t$ , but false at some intermediate time. Then  $J_s^T \neq J_t^T$ .*

**Proof.** We prove this by induction on  $s$ . For the base case  $s = 0$ , we have  $J_s^T = \emptyset$ . Suppose for contradiction that  $J_t^T = \emptyset$  as well. Since  $E$  is false at an intermediate time but true at time  $t$ , it must become true due to resampling some  $g$  at time  $t' < t$ . Clearly also  $J_{[t'+1,t]}^T = \emptyset$ . Since  $g$  makes  $E$  be true where it was false, we have  $g \in \tilde{\Gamma}(E)$ . As a result,  $g$  is not dominated for the empty wdag. So the rule for forming  $J_t^T$  would add a node to  $J_{[t',t]}^T$ , contradicting that  $J_t^T = \emptyset$ .

For the induction step, suppose  $s > 0$  and  $J_t^T = J_s^T$ . Let  $T'$  be the shift of  $T$ . By Proposition 19, both  $J_{s-1}^{T'}$  and  $J_{t-1}^{T'}$  are updated in the same manner depending on the flaw  $f_1$ . Thus,  $J_{s-1}^{T'} = J_{t-1}^{T'}$ . But this contradicts the induction hypothesis. ◀

► **Proposition 22.**  $N(E) \leq \sum_{H \in \mathfrak{H}'} \mu^\top A_H e_E$ .

**Proof.** By Proposition 21, for each time  $t$  that  $E$  switches from false to true, the corresponding wdag  $J_t^T$  must be distinct. Thus, the total number of times that  $E$  becomes true is at most  $\sum_{H \in \mathfrak{H}'} [\sum_{t \geq 0} \mathbb{1}_{J_t^T = H}]$ . Now take expectations and apply Lemma 20. ◀

► **Proposition 23.** For a wdag  $H \in \mathfrak{H}'$ , there holds  $\text{sink}(H) \in \mathfrak{D}(E)$ .

**Proof.** Let us fix some value  $t$  where  $E$  holds; for  $s = 1, \dots, t$  let  $H_s = J_{[s,t]}^T$  and  $I_s = \text{sink}(H_s)$ , and so that  $H_1 = H$ . We claim by induction on  $s$  that each wdag  $H_s$  has the stated property. The base case is  $s = t$ ; this holds since  $I_t = \emptyset \in \mathfrak{D}(E)$ .

Now consider some  $s < t$ , where  $H_s$  is formed from  $H_{s+1}$  by possibly adding a new node labeled  $g$ . The induction step is obvious if  $I_s = I_{s+1}$ . Thus, we may assume that  $g$  is unrelated to  $H_{s+1}$  (else it would not form a new sink node.) So the only relevant case is if  $g$  is not dominated by wdag  $G_{[s,t]}^T$ .

By induction hypothesis,  $I_{s+1} \in \mathfrak{D}(E)$ . Then it can be enumerated as  $I_{s+1} = \{g_1, \dots, g_r\}$  to satisfy Eq. (3). Suppose, for contradiction, that  $I_s \notin \mathfrak{D}(E)$ . If we enumerate  $I_s = \{g, g_1, \dots, g_r\}$ , then there would hold  $\theta^\top A_g A_{g_1} \dots A_{g_r} e_E \leq \theta^\top A_{g_1} \dots A_{g_r} e_E$  for all distributions  $\theta$  over the states.

Letting  $V$  denote the sink nodes of  $H_{s+1}$ , we have  $A_{H_{s+1}} = A_{H_{s+1}-V} A_V$ . Then, for any state  $\sigma$ , we have  $e_\sigma^\top A_g A_{H_{s+1}} e_E = e_\sigma^\top A_g A_{H_{s+1}-V} A_V e_E = e_\sigma A_{H_{s+1}-V} A_g A_{g_1} \dots A_{g_r} e_E$ , where in the last inequality we use the facts that  $g$  is unrelated to  $H_{s+1}$  and that  $A_V = A_{g_1} \dots A_{g_r}$ . By our bound with  $\theta = e_\sigma A_{H_{s+1}-V}$ , we see that this is at most  $e_\sigma A_{H_{s+1}-V} A_{g_1} \dots A_{g_r} e_E = e_\sigma A_{H_{s+1}} e_E$ . Hence  $g$  is dominated by  $G_{s+1}$  and we would not add the new node to  $H_s$ , a contradiction. ◀

We now get our desired distributional bounds:

► **Corollary 24.**  $N(E) \leq \mu(E) \sum_{I \in \mathfrak{D}(E)} \Psi(I)$ .

**Proof.** We have shown  $N(E) \leq \sum_{H \in \mathfrak{H}'} \mu^\top A_H e_E$ . Since each  $H \in \mathfrak{H}'$  has  $\text{sink}(H) \in \mathfrak{D}(E)$ , this is at most  $\sum_{I \in \mathfrak{D}(E)} \sum_{H \in \mathfrak{H}(I)} \mu^\top A_H e_E$ . By Theorem 9, this is at most  $\sum_{I \in \mathfrak{D}(E), H \in \mathfrak{H}(I)} w(H) \mu(E) = \mu(E) \sum_{I \in \mathfrak{D}(E)} \Psi(I)$ . ◀

► **Corollary 25.**  $P(E) \leq \mu(E) \sum_{I \in \mathfrak{D}(E)} \Psi_{\mathcal{G}}(I)$  where  $\mathcal{G} = \{f \in \mathcal{F} : f \not\subseteq E\}$ .

**Proof.** Consider the first time that  $E$  becomes true, if any. Then, only flaws in  $\mathcal{G}$  can be resampled up to that point; if some other flaw with  $f \subseteq E$  was resampled, then necessarily  $E$  was true earlier. Up to this time, the behavior of the Search Algorithm is identical to if had restricted to the flaw set  $\mathcal{G}$ . So  $P(E) \leq N_{\mathcal{G}}(E)$ ; by Corollary 24 this is at most  $\sum_{I \in \mathfrak{D}(E)} \Psi_{\mathcal{G}}(I)$ . ◀

Since any set  $I \in \mathfrak{D}(E)$  is also a subset of  $\tilde{\Gamma}(E)$ , we have the following crisp corollary:

► **Corollary 26.**  $P(E) \leq \mu(E) \bar{\Psi}(\tilde{\Gamma}(E))$ .

We note that Iliopoulos [20] had previously shown a bound similar to Corollary 26, but it had three additional technical restrictions: (i) it only worked for commutative resampling oracles in the sense of [23]; (ii) it additionally required the construction of a commutative resampling oracle for the event  $E$  itself; and (iii) if the resampling oracle is not regenerating, it gives a strictly worse bound.

The following result shows how to apply these bounds with common LLL criteria.



► **Proposition 27.** *Under the criteria of Proposition 11, we have the following estimates for  $P(E)$ :*

1. *If the symmetric criterion holds, then  $P(E) \leq \mu(E) \cdot e^{e^{|\tilde{\Gamma}(E)|p}}$ .*
2. *If function  $x$  satisfies the asymmetric criterion, then  $P(E) \leq \mu(E) \cdot \prod_{f \in \tilde{\Gamma}(E)} \frac{1}{1-x(f)}$ .*
3. *If function  $\eta$  satisfies the cluster-expansion criterion, then  $P(E) \leq \mu(E) \cdot \sum_{I \in \mathcal{D}(E)} \prod_{g \in I} \eta(g)$ .*

One weakness of distributional bounds such as Corollary 26 is that the definition of  $\tilde{\Gamma}(E)$  is binary: either flaw  $f$  cannot possibly cause  $E$ , or every occurrence of  $f$  must be tracked to determine if it caused  $E$ . The next results allow us to take account of flaws which can “partially” cause  $E$ .

For flaw  $f$  and event  $E$ , let us define

$$\kappa(f, E) = \max_{\sigma \in f \cap \bar{E}} \frac{e_{\sigma}^{\top} A_f e_E}{e_{\sigma}^{\top} A_f e_{E \cup \bar{f}}}$$

Note that  $\kappa(f, E) = 0$  for  $f \notin \tilde{\Gamma}(E)$ , and  $\kappa(f, E) \leq 1$  always. Thus,  $\kappa(f, E)$  is a weighted measure of the extent to which  $f$  causes  $E$ . Also note that usually  $e_{\sigma}^{\top} A_f e_f$  is small, and the denominator in the definition of  $\kappa(f, E)$  is close to one.

► **Theorem 28.**  $P(E) \leq \mu(E) + \sum_{f \in \mathcal{G}} \kappa(f, E) \cdot \min_{F \supseteq \bar{E}} N_{\mathcal{G}}(F \cap f)$  where  $\mathcal{G} = \{f : f \not\subseteq E\}$ .

**Proof.** See Appendix B. ◀

We remark that to obtain Theorem 28, we needed to bound  $N(F \cap f)$ ; bounds on  $P(F \cap f)$  alone would not have been enough. This explains why we analyzed the more general quantity. By applying Theorem 28 to the event  $\bar{E}$ , we can obtain a *lower bound* on the probability of  $E$ :

► **Corollary 29.**  $P(E) \geq \mu(E) - \sum_{f \in \mathcal{G}} \kappa(f, \bar{E}) \cdot \min_{F \subseteq E} N_{\mathcal{G}}(f - E)$  where  $\mathcal{G} = \{f : f \cap E \neq \emptyset\}$ .

For example, consider the permutation setting, where the probability space  $\Omega$  is the uniform distribution on permutations on  $n$  letters, and each flaw has the form  $g_1 \cap \dots \cap g_k$ , where each  $g_i$  is an atomic event of the form  $\pi x_i = y_i$ . We then get the following distributional result:

► **Theorem 30** ([14]). *In the permutation setting, consider an event  $E = g_1 \cap \dots \cap g_k$  where each  $g_i$  is an atomic event. We have  $N(E) \leq \frac{(n-k)!}{n!} \prod_{i=1}^k \left(1 + \sum_{f \in \mathcal{F}: f \sim g_i} \Psi(f)\right)$ .*

As another example, consider the setting where the underlying probability space  $\Omega$  is the uniform distribution on perfect matchings on the clique  $K_n$ , and each flaw has the form  $g_1 \cap \dots \cap g_k$ , where each  $g_i$  is an atomic event of the form  $\{x_i, y_i\} \subseteq M$ . We then get the following distributional result:

► **Theorem 31.** *In the settings of perfect matchings of the clique, consider an event  $E = g_1 \cap \dots \cap g_k$  where each  $g_i$  is an atomic event. We have  $N(E) \leq \frac{(n-2k-1)!!}{(n-1)!!} \prod_{i=1}^k \left(1 + \sum_{f \in \mathcal{F}: f \sim g_i} \Psi(f)\right)$ .*

The work [14] showed (what is essentially) Theorem 30 using a complicated and ad-hoc analysis based on a variant of witness trees, while Theorem 31 is new. The proofs are deferred to Appendix C.

Using these bounds, we can show the following estimates on individual entries of  $\pi$ :

► **Theorem 32.** *If each color appears at most  $\Delta = \frac{27}{256}n$  times in the array, then the Search Algorithm generates a latin transversal where, for each cell  $x, y$ , there holds*

$$\frac{17}{32n} \leq P(\pi x = y) \leq \frac{173}{128n}$$

**Proof.** Define the event  $E$  that  $\pi x = y$ . For the upper bound, Theorem 28 (with  $F = \Omega$ ) gives  $P(\pi x = y) \leq \mu(E) + \sum_{f \in \mathcal{F}} N_{\mathcal{F}}(f) \kappa(f, E)$ . Now, consider some flaw  $f = [(x_1, y_1), (x_2, y_2)] \in \mathcal{F}$ . The flaw  $f$  must involve cell  $(x, y')$  or  $(x', y)$ , else  $\eta(f, E, \Omega) = 0$ . For a flaw that does so, we can see that there is a probability of at most  $1/(n-1)$  that the resampling causes  $E$ , since there is at most one possible choice that can cause  $\pi x = y$ . Thus  $\eta(f, E, \Omega) \leq \frac{1/(n-1)}{1-1/n(n-1)}$ . By Theorem 30, we have  $N_{\mathcal{F}}(f) \leq \frac{(n-2)!}{n!} (1 + \sum_{f' \in \mathcal{G}: f' \sim f} \Psi(f'))$  where  $g_1, g_2$  are the two atoms in  $f$ . Since  $\Psi(f') \leq \gamma = \frac{256}{81n^2}$ , and there are at most  $2n(\Delta - 1)$  choices for  $f'$ , this is overall at most  $\frac{1}{n(n-1)}(1 + 2n(\Delta - 1)\gamma)$ .

Since either  $x_1 = x$  or  $y_1 = y$ , there are  $2n(\Delta - 1)$  choices for  $f$ . Summing over these, we get

$$P(E) \leq \frac{1}{n} + 2n(\Delta - 1) \cdot \frac{1}{n(n-1)} (1 + 2n(\Delta - 1)\gamma) \cdot \frac{1/(n-1)}{1-1/n(n-1)} \leq \frac{173}{128n}$$

For the lower bound, we use Corollary 29. Letting  $\mathcal{G}$  denote the flaws which do not involve cells  $(x, y')$  for  $y' \neq y$ , or  $(x', y)$  for  $x' \neq x$  and setting  $F = E$ , we have  $P(E) \geq \mu(E) - \sum_{f \in \mathcal{F}} N(E \cap f) \kappa(f, \bar{E})$ . Now, consider some such flaw  $f = [(x_1, y_1), (x_2, y_2)]$ . If  $(x_1, y_2) = (x, y)$ , then in this case,  $f \cap E = f$  and so Theorem 30 implies that  $N_{\mathcal{G}}(f \cap E) \leq \frac{1}{n(n-1)}(1 + (\Delta - 1)\gamma)(1 + 2n(\Delta - 1)\gamma)$ . (We emphasize that, because we are restricting to  $\mathcal{G}$ , there are no neighbors which involve cells  $(x, y')$  etc.) Otherwise, if  $(x, y)$  is distinct from  $(x_1, y_1), (x_2, y_2)$ , then  $f \cap E = [(x_1, y_1), (x_2, y_2), (x, y)]$  and Theorem 30 implies that  $N_{\mathcal{G}}(f \cap E) \leq \frac{1}{n(n-1)(n-2)}(1 + (\Delta - 1)\gamma)(1 + 2n(\Delta - 1)\gamma)^2$ .

There are at most  $(\Delta - 1)$  flaws in the first category, and each trivially has  $\kappa(f, \bar{E}) \leq 1$ . There are at most  $n^2(\Delta - 1)/2$  flaws in the second category; each such flaw  $f$  has  $\kappa(f, \bar{E}) \leq \frac{2/n}{1-1/n(n-1)}$ , since there are two choices for the cell to swap and in each case there is at most one way to get  $\pi x = y$  in a swap.

Putting all terms together, and with some algebraic simplifications, we get  $P(E) \geq \frac{17}{32n}$ . ◀

An analogous result can be shown for perfect matchings of the clique; we omit the proof here.

► **Theorem 33.** *Consider an edge-coloring  $C$  of the clique  $K_n$ , for  $n$  an even integer, such that each color appears on at most  $\Delta = \frac{27}{256}n$  edges. Then the Search Algorithm generates a perfect matching  $M$  such that  $C(e) \neq C(e')$  for all distinct edges  $e, e'$  of  $M$ . Moreover, for each edge  $e$ , the probability there holds  $\frac{17}{32(n-1)} \leq P(e \in M) \leq \frac{173}{128(n-1)}$ .*

## 7 Compositional properties for resampling oracles

The flaws and their resampling oracles are often built out of a collection of simpler, “atomic” events. For example, in the permutation LLL setting, these would be events of the form  $\pi x = y$ . In [15], Harris described a generic construction when the atomic events satisfy an additional property referred to as *obliviousness*. Let us now review this construction, and how it works with commutativity.

Consider a set  $\mathcal{A}$  of events, along with a resampling oracle  $\mathfrak{R}$  and a dependency relation  $\sim$ . It is allowed, but not required, to have  $f \sim f$  for  $f \in \mathcal{A}$ . For the compositional construction, we must define explicitly how the resampling oracle  $\mathfrak{R}_f$  uses the random seed. Specifically, to resample  $\sigma' \leftarrow \mathfrak{R}_f(\sigma)$ , we first draw a random seed  $r$  from some probability space  $R_f$ , and then set  $\sigma' = F(\sigma, r)$  for some *deterministic* function  $F$ . For brevity, we write this as  $\sigma' = r\sigma$ .

We refer to the elements of  $\mathcal{A}$  as *atoms*. These should be thought of as “pre-flaws”, that is, they have the *structural* algebraic properties of a resampling oracle, but do not necessarily satisfy any convergence condition such as the LLLL. We have the following key definition:

► **Definition 34** (Oblivious resampling oracle [15]). *The resampling oracle  $\mathfrak{R}$  is called oblivious if for every pair  $f, g \in \mathcal{A}$  with  $f \not\sim g$  and for each  $r \in R_f$ , one of the following two properties holds:*

- For all  $\sigma \in f \cap g$  we have  $r\sigma \in g$
- For all  $\sigma \in f \cap g$  we have  $r\sigma \notin g$

We assume throughout this section that  $\mathfrak{R}$  is oblivious. For each  $f \in \mathcal{A}$  and  $g_1, \dots, g_s \in \mathcal{A}$  with  $g_i \not\sim f$ , we define  $R_{f;g_1,\dots,g_s}$  to be the set of values  $r \in R_f$  such that  $r\sigma \in g_1 \cap \dots \cap g_t$ . With some abuse of notation, we also use  $R_{f;g_1,\dots,g_s}$  to refer to the probability distribution of drawing  $r$  from  $R_f$ , conditioned on having  $r$  in the set  $R_{f;g_1,\dots,g_s}$ . Note that in light of Definition 34 this is well-defined irrespective of  $\sigma$ .

For a stable set  $C \subseteq \mathcal{A}$ , we define  $\langle C \rangle$  to be the intersection of the events in  $C$ , i.e.,  $\langle C \rangle = \bigcap_{f \in C} f$ . From  $\mathcal{A}$ , one can construct an enlarged set of events  $\overline{\mathcal{A}} = \{\langle C \rangle \mid C \text{ a stable subset of } \mathcal{A}\}$ . We define the relation  $\sim$  on  $\overline{\mathcal{A}}$  by setting  $\langle C \rangle \sim \langle C' \rangle$  iff either (i)  $C = C'$  or (ii) there exist  $f \in C, f' \in C'$  with  $f \sim f'$ . We also define a corresponding resampling oracle  $\mathfrak{R}$  on  $\overline{\mathcal{A}}$  which will satisfy all its required structural properties. The intent is to choose the flaw set  $\mathcal{F}$  to be some arbitrary subset of  $\overline{\mathcal{A}}$ ; as before,  $\overline{\mathcal{A}}$  does not necessarily satisfy any LLLL convergence criterion.

To determine  $\mathfrak{R}$ , consider some  $g = \langle C \rangle$  for a stable set  $C$ , with some arbitrary enumeration  $C = \{f_1, \dots, f_t\}$ . We define  $R_g$  to be the probability distribution on tuples  $r = (r_1, \dots, r_t)$  wherein each  $r_i$  is drawn independently from  $R_{f_i;f_{i+1},\dots,f_t}$ , and we set  $r\sigma = r_t \dots r_1 \sigma$ .

► **Theorem 35** ([15]). *Suppose that  $\mathfrak{R}$  is an oblivious resampling oracle for  $\mathcal{A}$ , which is not necessarily commutative. Then:  $\mathfrak{R}$  with dependency relation  $\sim$  provides an oblivious resampling oracle for  $\overline{\mathcal{A}}$ . If  $\mathfrak{R}$  is regenerating on  $\mathcal{A}$ , then the resampling oracle on  $\overline{\mathcal{A}}$  is also regenerating.*

It would seem reasonable that if  $\mathcal{A}$  is commutative, then  $\overline{\mathcal{A}}$  would be as well. Unfortunately, we do not know how to show this for the commutativity definition of [23]. For our commutativity definition, this is easy to show; in addition,  $\overline{\mathcal{A}}$  will inherit a number of other nice properties. This is a good illustration of how the new definition of commutativity is easier to work with, beyond its advantage of greater generality.

► **Proposition 36.** *Suppose that  $\mathcal{A}$  is oblivious but not necessarily commutative. For a flaw  $g = \langle C \rangle$ , suppose that we have fixed an enumeration  $C = \{f_1, \dots, f_t\}$  to define  $\mathfrak{R}_g$ . Then  $A_g \propto A_{f_1} \dots A_{f_t}$ .*

**Proof.** By definition of  $\mathfrak{R}_g$ , we have  $A_g[\sigma, \sigma'] = \Pr(r_t \dots r_1 \sigma = \sigma')$ , where each  $r_i$  is drawn independently from  $R_{f_i;f_{i+1},\dots,f_t}$ . Let us define  $R'_i = R_{f_i;f_{i+1},\dots,f_t}$  and  $\sigma_i = r_i \dots r_1 \sigma$  for  $i = 0, \dots, t$  (where  $\sigma_0 = \sigma$ ). By enumerating over possible values for  $\sigma_1, \dots, \sigma_t$ , we get  $A_g[\sigma, \sigma'] = \sum_{\substack{\sigma_1, \dots, \sigma_t \\ \sigma_t = \sigma'}} \prod_{i=1}^t \Pr_{r_i \sim R'_i}(r_i \sigma_{i-1} = \sigma_i)$ .

Note that if  $\sigma_i \notin f_j$  for some  $j > i$ , then the term  $\Pr_{r_i \sim R'_i}(r_i \sigma_{i-1} = \sigma_i)$  must be zero, since  $r_i \in R'_i \subseteq R_{f_i; f_j}$ . So we may restrict the sum to terms with  $\sigma_i \in f_{i+1} \cap \dots \cap f_t$  for all  $i = 0, \dots, t$ . For each such term, we have  $\Pr_{r_i \sim R'_i}(r_i \sigma_{i-1} = \sigma_i) = \frac{\Pr_{r_i \sim R_i}(r_i \sigma_{i-1} = \sigma_i)}{\Pr_{r_i \sim R_i}(r_i \in R'_i)} = \frac{A_{f_i}[\sigma_{i-1}, \sigma]}{\Pr_{r_i \sim R_i}(r_i \in R'_i)}$ . So:

$$\begin{aligned} A_g[\sigma, \sigma'] &= \sum_{\substack{\sigma_1, \dots, \sigma_t \\ \sigma_t = \sigma'}} \frac{A_{f_1}[\sigma_0, \sigma_1] \dots A_{f_t}[\sigma_{t-1}, \sigma_t]}{\prod_{i=1}^t \Pr_{r_i \sim R_i}(r_i \in R'_i)} = \frac{\sum_{\sigma_t = \sigma'}^{\sigma_1, \dots, \sigma_t} A_{f_1}[\sigma_0, \sigma_1] \dots A_{f_t}[\sigma_{t-1}, \sigma_t]}{\prod_{i=1}^t \Pr_{r_i \sim R_i}(r_i \in R'_i)} \\ &= (cA_{f_1} \dots A_{f_t})[\sigma, \sigma'] \end{aligned} \quad \blacktriangleleft$$

► **Proposition 37.** *If  $\mathcal{A}$  is commutative, then the transition matrix  $A_g$  for a flaw  $g = \langle C \rangle$  does not depend on the chosen enumeration  $C = \{f_1, \dots, f_t\}$ .*

**Proof.** By Proposition 36, we have  $A_g = cA'_g$  for  $A'_g = A_{f_1} \dots A_{f_t}$ . Since the matrices  $A_{f_i}$  all commute,  $A'_g$  does not depend on the enumeration of  $C$ . Furthermore, the constant  $c$  can be determined from  $A'_g$  by choosing an arbitrary state  $\sigma \in g$  and setting  $c = \frac{1}{\sum_{\sigma'} A'_g[\sigma, \sigma']}$ . ◀

► **Theorem 38.** *If the resampling oracle is commutative on  $\mathcal{A}$ , then it is also commutative on  $\overline{\mathcal{A}}$ .*

**Proof.** Let  $g = \langle C \rangle$  and  $g' = \langle C' \rangle$  for stable sets  $C, C'$  such that  $g \not\sim g'$ . So  $f \not\sim f'$  for all  $f \in C$  and  $f' \in C'$ . By Proposition 36 we have

$$A_g A_{g'} = c_g c_{g'} \left( \prod_{f \in C} A_f \prod_{f' \in C'} A_{f'} \right), \quad A_{g'} A_g = c_{g'} c_g \left( \prod_{f' \in C'} A_{f'} \prod_{f \in C} A_f \right)$$

for scalar constants  $c_g, c_{g'}$ . All these matrices  $A_f, A_{f'}$  commute, so both quantities are equal. ◀

Another useful property for such resampling oracles is *idempotence*. We say that  $\mathcal{A}$  is idempotent if  $A_f^2 \propto A_f$  for all  $f \in \mathcal{A}$ . Most of the known commutative resampling oracles, resampling oracles have this property, including the variable LLLL and the permutation LLL.

► **Proposition 39.** *If the resampling oracle is commutative and idempotent on  $\mathcal{A}$ , then it is also idempotent on  $\overline{\mathcal{A}}$ . Furthermore, for any stable set  $I = \{\langle C_1 \rangle, \dots, \langle C_k \rangle\}$  of  $\overline{\mathcal{A}}$  and stable set  $J = C_1 \cup \dots \cup C_k$  of  $\mathcal{A}$ , there holds  $A_I \propto A_J$ .*

**Proof.** First, let  $f = \langle C \rangle$  for stable set  $C = \{g_1, \dots, g_k\}$ . Proposition 36 gives  $A_f^2 \propto (A_{g_1} \dots A_{g_k})^2$ . Since the matrices  $A_{g_i}$  commute with each other, this gives  $A_f^2 \propto A_{g_1}^2 \dots A_{g_k}^2$ . Since  $\mathcal{A}$  is idempotent, this is proportional to  $A_{g_1} \dots A_{g_k}$ , which again by Proposition 36 is proportional to  $A_f$ .

For the second result, Proposition 36 gives  $A_I \propto \prod_{i=1}^k \prod_{g \in C_i} A_g = \prod_{g \in J} A_g^{n_g}$  where  $n_g \geq 1$  is the number of copies of  $g$  appearing in  $C_1, \dots, C_k$ . Since  $\mathcal{A}$  is idempotent, each term  $A_g^{n_g}$  is proportional to  $A_g$ . Hence we have  $A_I \propto \prod_{g \in J} A_g = A_J$ . ◀

---

## References

- 1 Dimitris Achlioptas and Fotis Iliopoulos. Random walks that find perfect objects and the Lovász local lemma. *Journal of the ACM*, 63(3):Article #22, 2016. doi:10.1145/2818352.

- 2 Dimitris Achlioptas, Fotis Iliopoulos, and Vladimir Kolmogorov. A local lemma for focused stochastic algorithms. *SIAM Journal on Computing*, 48(5):1583–1602, 2019. doi:10.1137/16M109332X.
- 3 Dimitris Achlioptas, Fotis Iliopoulos, and Alistair Sinclair. Beyond the Lovász local lemma: Point to set correlations and their algorithmic applications. In *Proc. 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 725–744, 2019.
- 4 Noga Alon, Joel Spencer, and Prasad Tetali. Covering with latin transversals. *Discrete applied mathematics*, 57(1):1–10, 1995.
- 5 Rodrigo Bissacot, Roberto Fernández, Aldo Procacci, and Benedetto Scoppola. An improvement of the Lovász local lemma via cluster expansion. *Combinatorics, Probability & Computing*, 20(5):709–719, 2011. doi:10.1017/S0963548311000253.
- 6 Karthekeyan Chandrasekaran, Navin Goyal, and Bernhard Haeupler. Deterministic algorithms for the Lovász local lemma. *SIAM Journal on Computing*, 42(6):2132–2155, 2013. doi:10.1137/100799642.
- 7 Antares Chen, David G. Harris, and Aravind Srinivasan. Partial resampling to approximate covering integer programs. *Random Structures & Algorithms*, pages 69–93, 2021.
- 8 Kai-Min Chung, Seth Pettie, and Hsin-Hao Su. Distributed algorithms for the Lovász local lemma and graph coloring. *Distributed Computing*, 30(4):261–280, 2017.
- 9 Paul Erdős and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In *Infinite and finite sets (Colloq., Keszthely, 1973; dedicated to P. Erdős on his 60th birthday), Vol. II*, pages 609–627. Colloq. Math. Soc. János Bolyai, Vol. 10. 1975.
- 10 Paul Erdős and Joel Spencer. Lopsided Lovász local lemma and latin transversals. *Discrete Applied Mathematics*, 30(2-3):151–154, 1991. doi:10.1016/0166-218X(91)90040-4.
- 11 Bernhard Haeupler and David G. Harris. Parallel algorithms and concentration bounds for the Lovász local lemma via witness DAGs. *ACM Transactions on Algorithms*, 13(4):Article #25, 2017.
- 12 Bernhard Haeupler, Barna Saha, and Aravind Srinivasan. New constructive aspects of the Lovász local lemma. *Journal of the ACM*, 58(6):Article #28, 2011. doi:10.1145/2049697.2049702.
- 13 David G. Harris. Lopsidedependency in the Moser-Tardos framework: Beyond the lopsided Lovász local lemma. *ACM Transactions on Algorithms*, 13(1):Article #17, 2016. doi:10.1145/3015762.
- 14 David G. Harris. New bounds for the Moser-Tardos distribution. *Random Structures & Algorithms*, 57(1):97–131, 2020.
- 15 David G. Harris. Oblivious resampling oracles and parallel algorithms for the Lopsided Lovász Local Lemma. *ACM Transactions on Algorithms*, 17(1):Article #1, 2021.
- 16 David G. Harris and Aravind Srinivasan. Algorithmic and enumerative aspects of the Moser-Tardos distribution. *ACM Transactions on Algorithms*, 13(3):Article #33, 2017.
- 17 David G. Harris and Aravind Srinivasan. A constructive Lovász Local Lemma for permutations. *Theory of Computing*, 13(1):Article #17, 2017.
- 18 David G. Harris and Aravind Srinivasan. The Moser–Tardos framework with partial resampling. *Journal of the ACM*, 66(5):Article #36, 2019.
- 19 Nicholas J. A. Harvey and Jan Vondrák. An algorithmic proof of the Lovász local lemma via resampling oracles. *SIAM Journal on Computing*, 49(2):394–428, 2020.
- 20 Fotis Iliopoulos. Commutative algorithms approximate the LLL-distribution. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 2018.
- 21 Fotis Iliopoulos and Alistair Sinclair. Efficiently list-edge coloring multigraphs asymptotically optimally. In *Proc. 14th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2319–2336, 2020.
- 22 Kashyap Kolipaka and Mario Szegedy. Moser and Tardos meet Lovász. In *Proc. 43rd annual ACM Symposium on Theory of Computing (STOC)*, pages 235–244, 2011.

- 23 Vladimir Kolmogorov. Commutativity in the algorithmic Lovász local lemma. *SIAM Journal on Computing*, 47(6):2029–2056, 2018.
- 24 László Lovász. Submodular functions and convexity. In *Mathematical Programming: the State of the Art*, pages 235–257. Springer, 1983.
- 25 Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász local lemma. *Journal of the ACM*, 57(2):Article #11, 2010. doi:10.1145/1667053.1667060.
- 26 Wesley Pegden. An extension of the Moser-Tardos algorithmic local lemma. *SIAM Journal on Discrete Mathematics*, 28(2):911–917, 2014. doi:10.1137/110828290.

## A Necessity of transition matrix commutativity for Lemma 5

Consider a set of events  $\mathcal{B}^*$  with a dependency relation  $\sim$ . We say that  $\mathcal{B}^*$  is *complete* if for each  $\sigma \in \Omega$  there exists a flaw  $h_\sigma = \{\sigma\} \in \mathcal{B}^*$ , and with  $h_\sigma \sim g$  for all  $g \in \mathcal{B}^*$ . Note that this definition is satisfied if  $\mathcal{B}^*$  is generated by atomic events corresponding to permutations, perfect matchings of hypergraphs, or spanning trees.

We show now that if transition matrix commutativity fails in a complete set of events, even for a single pair of flaws, then some wdags may appear with probability arbitrarily higher than their weight. It can be checked that preconditions of the lemma (and thus its conclusion) apply to some existing resampling oracles, such as the oracle for spanning trees from [19] and the oracle for perfect matchings of complete  $s$ -uniform hypergraphs (for  $s \geq 3$ ) from [15].

► **Theorem 40.** *Suppose that  $\mathcal{B}^*$  is complete, regenerating, and contains a pair  $f, g \in \mathcal{B}^*$  with  $f \approx g$  and  $A_f A_g \neq A_g A_f$ . Then for any  $C > 0$  there exists a set of flaws  $\mathcal{B} \subseteq \mathcal{B}^*$  with  $|\mathcal{B}| = 3$ , wdag  $H$  with a single sink and a flaw resampling strategy  $S$  such that the probability that  $H$  appears in the execution of the algorithm is at least  $C \cdot w(H) = C \cdot \prod_{v \in H} \mu(L(v))$ .*

**Proof.** Consider states  $\sigma, \tau$  with  $A_f A_g[\sigma, \tau] \neq A_g A_f[\sigma, \tau]$ . Denote  $x = A_f A_g e_\tau$  and  $y = A_g A_f e_\tau$ , and assume w.l.o.g. that  $x[\sigma] < y[\sigma]$ . Note that  $\mu^\top A_f A_g = \mu^\top A_g A_f = \gamma_f \gamma_g \cdot \mu^\top$  since the oracles are regenerating, and therefore  $\mu^\top x = \mu^\top y = \gamma_f \gamma_g \cdot \mu[\tau] = \gamma_f \gamma_g \gamma_h$ .

Consider the following strategy  $S$  given a current state  $\sigma_1$ : (i) if  $\sigma_1 \neq \sigma$  then prioritize flaws  $f, g, h$  at steps 1,2,3 respectively; (ii) if  $\sigma_1 = \sigma$  then prioritize flaws  $g, f, h$  at steps 1,2,3 respectively. We say that the run *succeeds* if the sequence of addressed flaws is  $(f, g, h)$  in the first case and  $(g, f, h)$  in the second case. Clearly, the probability of success equals  $e_{\sigma_1}^\top A_f A_g e_\tau = e_{\sigma_1}^\top x$  in the first case and  $e_{\sigma_1}^\top A_g A_f e_\tau = e_{\sigma_1}^\top y$  in the second case. If  $\sigma_1$  is distributed according to  $\mu$  then the probability of success is

$$\begin{aligned} p &= \mu[\sigma] \cdot e_\sigma^\top y + \sum_{\sigma_1 \in \Omega - \{\sigma\}} \mu[\sigma_1] \cdot e_{\sigma_1}^\top x = \mu[\sigma] \cdot (e_\sigma^\top y - e_\sigma^\top x) + \sum_{\sigma_1 \in \Omega} \mu[\sigma_1] \cdot e_{\sigma_1}^\top x \\ &= \mu^\top x + \mu[\sigma] \cdot (y[\sigma] - x[\sigma]) > \gamma_f \gamma_g \gamma_h \end{aligned}$$

Furthermore, if the run succeeds then the last state is distributed according to  $\mu$  (since step 3 resamples  $h$  at state  $\tau$ , and the oracles are regenerating).

Now consider the trajectory which repeats the sequence  $f, g, h$  for  $n$  times, and the corresponding wdag  $H = G_{3n}^T$  which has a single sink node labeled  $h$ . Let  $S^n$  be the strategy  $S$  repeated cyclically. From the previous paragraph, the probability that the run starting with some distribution  $\mu$  produces  $H$  is given by  $c_\mu \cdot p^{n-1}$ , where  $c_\mu$  depends only on the initial distribution. Note that  $w(H) = (\gamma_f \gamma_g \gamma_h)^n$ . Choosing  $n$  sufficiently large now gives the claim. ◀

## B Proof of Theorem 28

First, there is a probability of  $\mu(E)$  that  $E$  is true at time 0. To bound the probability the  $E$  becomes true later, let us say that a pair  $(f, t)$  is *good* if (i)  $E$  is false at time  $t$ ; (ii)  $f$  is resampled at time  $t$ ; and (iii) either this is the first resampling of  $f$ , or if the most recent resampling of  $f$  had occurred at time  $t' < t$ , then  $f$  had been false at some intermediate time between  $t' + 1$  and  $t$ . If  $(f, t)$  is good, we say it is *finalized at time*  $s \geq t$  if (i)  $f$  is resampled at time  $s$ ; (ii)  $E$  has been false at all times prior to  $s$ ; and (iii)  $f$  is true at times  $t, \dots, s$ .

We claim that if  $E$  becomes true, then there is some pair  $(f, t)$  which is good and is finalized at some time  $s \geq t$ . For, suppose that  $E$  first becomes true at time  $s \geq 1$  and that  $f$  is resampled at time  $s$ . Going backward in time, look for the earliest time  $t$  such that the same flaw  $f$  is resampled at time  $t$  and that  $f$  remained true between  $t$  and  $s$  (possibly  $t = s$ ). Then  $(f, t)$  is good and is finalized at time  $s$ .

Now let us fix good pair  $(f, t)$ . For each  $s \geq t$ , let  $\mathcal{E}_s$  be the event that  $(f, t)$  is finalized by some  $s' \geq s$ . We claim that  $\Pr(\mathcal{E}_s) \leq \kappa(f, E)$ ; furthermore, this probability bound holds conditional on the full state of the system at times up to  $s$ .

By a limiting argument, it suffices to show this bound if we restrict to  $s \leq s_{\max}$  for arbitrary integer  $s_{\max}$ . For fixed  $s_{\max}$ , we show it by induction backward on  $s$ . The claim follows immediately from induction if  $f$  is not being resampled at time  $s$  or if  $s = s_{\max}$ . If  $E$  is true at time  $s$ , then  $\mathcal{E}_s$  is impossible (since  $(f, t)$  would have needed to be finalized at some earlier time  $s' < s$ ). Likewise, if there was an intervening time between  $t$  and  $s$  where  $f$  was false, then  $\mathcal{E}_s$  is impossible.

So, suppose we resample  $f$  at time  $s$  while  $E$  is false and  $f$  has remained true for times  $t, \dots, s$ . Let  $\tau \in f \cap \bar{E}$  be the state at time  $s$ . There are three things that can happen when resampling  $f$ :

- $E$  becomes true. This has probability  $e_{\tau}^{\top} A_f e_E$ . In this case, event  $\mathcal{E}_s$  may have occurred.
- $E$  remains false, and  $f$  becomes false. This has probability  $e_{\tau}^{\top} A_f e_{\bar{E} \cap \bar{f}}$ . In this case, event  $\mathcal{E}_s$  is impossible.
- $E$  becomes false, and  $f$  remains true. This has probability  $e_{\tau}^{\top} A_f e_{\bar{E} \cap f}$ . In this case, in order to  $\mathcal{E}_s$  to occur, it must be that  $\mathcal{E}_{s+1}$  holds after resampling  $f$ . By induction hypothesis, this has probability at most  $\kappa(f, E)$ .

Overall, we have  $\Pr(\mathcal{E}_s) \leq e_{\tau}^{\top} A_f e_E \cdot 1 + e_{\tau}^{\top} A_f e_{\bar{E} \cap f} \cdot \kappa(f, E)$ . This is at most  $\kappa(f, E) e_{\tau}^{\top} A_f e_{E \cup \bar{f}} + e_{\tau}^{\top} A_f e_{\bar{E} \cap f} \cdot \kappa(f, E) \leq \kappa(f, E)$ . This concludes the induction.

Now consider any good pair  $(f, t)$ . Because of the claim, we know that the probability that  $(f, t)$  is finalized (by any time  $s \geq t$ ) is at most  $\kappa(f, E)$ , conditional on all other state at time  $t$ . Since this is a necessary condition for  $E$  to become true, the overall probability that  $E$  becomes true is at most  $\mathbf{E}[L_f] \cdot \kappa(f, E)$ , where  $L$  is the number of good pairs  $(f, t)$ . Note that, between any good pairs  $(f, t)$  and  $(f, t')$  for  $t' > t$ , the event  $F \cap f$  is false at least once, where  $F$  is an arbitrary event with  $F \supseteq \bar{E}$ . However, at times  $t$  and  $t'$ , the event  $F \cap f$  is true. Thus,  $F \cap f$  is caused to become true at least  $L_f$  times, and so by Proposition 22, we have  $\mathbf{E}[L_f] \leq N_{\mathcal{G}}(F \cap f)$ .

## C Proof of Theorem 30 and Theorem 31

We begin by considering the setting where  $\Omega$  is the uniform distribution on the permutations  $\pi$  on  $[n]$ . The set  $\mathcal{A}$  is defined as follows: for each pair  $(x, y) \in [n] \times [n]$ , there is atom  $\pi x = y$ , which we denote by  $[x, y]$ . The resampling oracle here, for such an event, is to update the state  $\pi \leftarrow (y z)\pi$ , where  $z$  is uniformly drawn from  $[n]$ . (Here and throughout the section,  $(y z)$  denotes the permutation which swaps  $y$  and  $z$ .) We have  $[x, y] \sim [x', y']$  if exactly one

of the following holds: (i)  $x = x'$  or (ii)  $y = y'$ . Equivalently, this holds iff  $[x, y] \cap [x', y'] = \emptyset$ . See [15] for further details, including a proof that the resampling oracle is commutative and oblivious.

We begin with a basic observation on how atoms of  $\mathcal{A}$  interact with events in  $\bar{\mathcal{A}}$ .

► **Proposition 41.** *Let  $f = [x, y]$  be an atom and let  $E = \langle C \rangle$  where  $C$  is a stable set of  $\mathcal{A}$ . Then  $A_f e_E \propto e_{E'}$ , where  $E' = \langle C' \rangle$  and stable set  $C'$  is obtained from  $C$  as follows:*

- If  $C$  contains exactly two atoms  $f_1, f_2$  which are neighbors of  $f$ , i.e.  $f_1 = [x, y_1]$  and  $f_2 = [x_2, y]$ , then  $C' = C - \{f_1, f_2\} \cup \{[x, y], [x_2, y_1]\}$ .
- If  $C$  contains exactly one atom  $f_1$  which is a neighbor of  $f$ , then  $C' = C - f_1 \cup \{f\}$ .
- Otherwise, if  $C$  contains no neighbors of  $f$ , then  $C' = C \cup \{f\}$ .

**Proof.** Consider state  $\pi$ , and suppose we resample  $f$  to obtain  $\pi' = (y z)\pi$ . If  $\pi \notin f$ , then  $e_\pi^\top A_f e_E = 0 = e_\pi^\top e_{E'}$ . Similarly, for each  $f' \in C$  which is not a neighbor of  $f$ , we must have  $\pi \in f'$  as otherwise  $e_\pi^\top A_f e_E = 0 = e_\pi^\top e_{E'}$ . In these cases, we also automatically have  $\pi' \in f'$  for all such  $f'$ . Thus, we suppose that  $\pi \in f$  and also  $\pi \in f'$  for all  $f' \in C - \Gamma(f)$ . We consider the following cases in turn:

- If  $C$  contains two atoms  $f_1, f_2$ , then we claim that  $\pi' \in E$  precisely when  $z = y_1$  and  $\pi x_2 = y$ . For, in order to have  $\pi' \in f_1$ , we must have  $\pi' x = y_1$ . Since  $\pi x = y$ , this implies that  $(y z)y = y_1$ , i.e.  $z = y_1$ . Thus,  $\pi' = (y y_1)\pi$ . To satisfy  $f_2$ , we must have  $y = \pi' x_2 = (y y_1)\pi x_2$ , i.e.  $\pi x_2 = y$ . In this case, we see that  $e_\pi^\top A_f e_E = 1/n$  for all  $\pi$  and also  $e_\pi^\top e_{E'} = 1$ .
- Suppose that  $C$  contains a neighbor  $f_1 = [x, y_1]$ . In this case, we have  $\pi' \in f_1$  precisely if  $y_1 = z$ . Similarly, suppose that  $C$  contains a neighbor  $f_2 = [x_2, y]$ . In this case, we have  $\pi' \in f_2$  precisely if  $z = \pi x_2$ . Thus, we have  $e_\pi^\top A_f e_E = 1/n$  for all such  $\pi$  and also  $e_\pi^\top e_{E'} = 1$ .
- If  $C$  has no neighbors of  $f$ , then  $\pi'$  is in  $E$  iff  $z \notin \{y_1, \dots, y_k\}$  where  $C = \{[x_1, y_1], \dots, [x_k, y_k]\}$ . Thus  $e_\pi^\top A_f e_E = \frac{n-k}{n}$  and  $e_\pi^\top e_{E'} = 1$  ◀

To understand more complex, multi-atom interactions, let us fix event  $E = \langle C \rangle$  for a stable set  $C$ . For a stable set  $I \subseteq \mathcal{A}$ , we can form an associated bipartite graph  $G_I$ , as follows: the left vertices correspond to  $C$  (we call these  $C$ -nodes), and the right vertices correspond to  $I$  (we call these  $I$ -nodes). It has an edge between  $f$  and  $f'$  iff  $f \sim f'$ . Observe that since  $C$  and  $I$  are stable, the graph  $G_I$  has degree at most two – each node  $[x, y]$  can have one neighbor of the form  $[x', y]$  and another neighbor of the form  $[x, y']$ . So,  $G_I$  decomposes into paths and cycles.

We define  $\tau(I)$  to be the size of a maximum matching in  $G_I$ . We also define the *active conditions* for  $I$ , denoted  $\text{Active}(I) \subseteq \mathcal{A}$ , as follows. First, for each  $f \in I$ , we also place  $f$  into  $\text{Active}(I)$ . Second, consider some maximal path of  $G_I$  starting and ending at  $C$ -nodes (which we call a  $C$ -path). The path can be written (in one of its two orientations) as

$$[x_1, y_1], [x_1, y_2], [x_2, y_2], \dots, [x_k, y_{k-1}], [x_k, y_k].$$

In this case, we also put  $[x_k, y_1]$  into  $\text{Active}(I)$ . (It is possible that  $k = 1$ , in which case  $[x_1, y_1]$  is an isolated  $C$ -node.)

For brevity, we define  $\alpha(I)$  to be the event  $\langle \text{Active}(I) \rangle$  in  $\bar{\mathcal{A}}$ . The active conditions determine the vector  $A_I e_E$ , and also have a number of nice combinatorial properties.

► **Proposition 42.** *Let  $I$  be a stable set of  $\mathcal{A}$ . Then the following properties hold:*

1.  $A_I e_E \propto e_{\alpha(I)}$ .
2.  $|\text{Active}(I)| = |C| + |I| - \tau(I)$ .
3. Any  $f \in I$  with  $\tau(I) = \tau(I - f)$  has  $\text{Active}(I) = \text{Active}(I - f) \cup \{f\}$ .



**Proof.**

1. We show this by induction on  $I$ . The base case  $I = 0$  is clear, since then  $\text{Active}(I) = C$ . For the induction step, consider  $f \in I$ . We have  $A_I e_E = A_f A_{I-f} e_E$ ; by induction hypothesis, this is proportional to  $A_f e_{\langle U \rangle}$  where  $U = \text{Active}(I - f)$ . By Proposition 41, this in turn is proportional to  $e_{\langle U' \rangle}$ , where  $U'$  is formed according to the specific given rules. We thus need to show that  $U' = \text{Active}(I)$ . There are three cases.  
 If  $U$  has no neighbors of  $f$ , then  $U' = U \cup \{f\}$ . Furthermore,  $G_I$  has no changes in its  $C$ -paths compared to  $G_{I-f}$ , so  $\text{Active}(I) = \text{Active}(I - f) \cup \{f\} = U'$ .  
 Next suppose  $U$  has one neighbor  $f' = [x', y']$  of  $f$ . Since  $I$  is a stable set, it must have  $f' \notin I$ , i.e.  $G_{I-f}$  contains a  $C$ -path with endpoints  $x', y'$ . This  $C$ -path now terminates in a degree-one  $I$ -node  $[x, y]$  in  $G_I$ , and hence it is removed from  $G_I$ . So  $\text{Active}(I) = \text{Active}(I - f) - f' \cup \{f\} = U'$ .  
 Finally, If  $U$  has two neighbors  $f_1 = [x, y_1], f_2 = [x_2, y]$ , then again since  $I$  is a stable set these must correspond to  $C$ -paths in  $G_{I-f}$ . Thus, there are two  $C$ -paths with endpoints  $x, y_1$  and  $x_2, y$  respectively. Now in  $G_I$ , there is a new degree-two  $I$ -node  $[x, y]$ . This merges the two  $C$ -paths into a single new  $C$ -path with endpoints  $x_2, y_1$ . Thus again  $\text{Active}(I) = \text{Active}(I - f) - \{f_1, f_2\} \cup \{f, [x_2, y_1]\} = U'$ .
2. Consider some connected component of  $G_I$ ; it is a path or cycle with  $i$  distinct  $I$ -nodes and  $c$  distinct  $C$ -nodes, where  $c \in \{i - 1, i, i + 1\}$ . If  $c = i - 1$ , then it has maximum matching size  $t = i - 1$  else it has maximum matching size  $t = i$ . If  $c = i + 1$ , then it has one additional active condition corresponding to the  $C$ -path on its nodes, and thus has  $a = i + 1$  active conditions; else it has  $a = i$  active conditions. In all cases, it can be checked that  $a = c + i - t$ . The claimed formula is obtained by summing over all components.
3. By part (2), we have  $\tau(I) = \tau(I - f)$  precisely if  $|\text{Active}(I)| = |\text{Active}(I - f)| + 1$ , i.e.  $G_I$  has the same number of  $C$ -paths as  $G_{I-f}$ . We claim in this case that  $G_I$  has the same  $C$ -paths as  $G_{I-f}$  as well, which will show the claim.  
 For, if not, then  $G_I$  would need to gain, and lose, some  $C$ -paths compared to  $G_{I-f}$ . The new  $I$ -node  $[x, y]$  would need to participate in a new  $C$ -path. This can only occur if  $G_{I-f}$  has two  $C$ -paths with endpoints  $[x, y']$  and  $[x', y]$ . But in this case, these two existing  $C$ -paths get destroyed in  $G_I$ , and thus in fact  $G_I$  has strictly fewer  $C$ -paths compared to  $G_{I-f}$ .  $\blacktriangleleft$

► **Proposition 43.** *Let  $I = \{f_1, \dots, f_k\}$  be a stable set in  $\bar{\mathcal{A}}$ , where  $f_i = \langle F_i \rangle$  for each  $i$ . Consider the stable sets  $J' = F_1 \cup \dots \cup F_{k-1}$  and  $J = J' \cup F_k$  of  $\mathcal{A}$ . If  $\tau(J') = \tau(J)$ , then  $f_k$  is dominated by  $I - f_k$  in  $\bar{\mathcal{A}}$ .*

**Proof.** Let  $I' = \{f_1, \dots, f_{k-1}\}$ . It is easily seen that the permutation LLL setting is idempotent. Thus, by Proposition 39, we have  $A_{I'} \propto A_{J'}$ . Combined with Proposition 42(1), this implies that there is some scalar value  $p \geq 0$  such that  $A_{I'} e_E = p e_{\alpha(J')}$ . We want to show that

$$e_{\pi}^{\top} A_I e_E \leq e_{\pi}^{\top} A_{I'} e_E \tag{4}$$

for any state  $\pi$ .

By Proposition 39, we have  $A_I \propto A_J$ . Thus, the LHS of Eq. (4) is zero if  $\pi \notin \alpha(J)$ , in which case the inequality clearly holds. So suppose that  $\pi \in \alpha(J)$ . By Proposition 42(3), we have  $\text{Active}(J') \subseteq \text{Active}(J)$  since  $\tau(J') = \tau(J)$ . In this case, also  $\pi \in \alpha(J')$  so the RHS of Eq. (4) is equal to  $p$ . The LHS can be factored as  $e_{\pi}^{\top} A_I e_E = \sum_{\sigma} A_f[\pi, \sigma] \cdot e_{\sigma}^{\top} A_{I'} e_E = \sum_{\sigma \in \alpha(J')} A_f[\pi, \sigma] p$ . Since matrix  $A_f$  is substochastic, this is at most  $p$ . This establishes the desired inequality.  $\blacktriangleleft$

► **Proposition 44.** *For any  $I \in \mathfrak{D}(E)$ , there is an injective function  $\phi_I : I \rightarrow C$  with  $g \sim \phi_I(g)$  for all  $g \in I$ .*

**Proof.** By definition,  $I$  can be ordered as  $I = \{f_1, \dots, f_k\}$ , where  $f_i = \langle F_i \rangle$  and such that each  $f_i$  is not dominated by  $\{f_1, \dots, f_{i-1}\}$ . Let us define  $J_i = F_1 \cup \dots \cup F_i$  for each  $i$ . By Proposition 43, we must have  $\tau(J_i) > \tau(J_{i-1})$  for each  $i$ . Thus, for each  $i$ , there is some  $g_i \in F_i - J_{i-1}$  and some  $F'_i \subseteq F_i - \{g_i\}$  with  $\tau(J_{i-1} \cup F'_i \cup \{g_i\}) > \tau(J_{i-1} \cup F'_i)$ . It is known (see, e.g. [24, Example 1.4]) that  $\tau$  is a submodular set function. Hence, we have

$$1 = \tau(J_{i-1} \cup F'_i \cup \{g_i\}) - \tau(J_{i-1} \cup F'_i) \leq \tau(\{g_1, \dots, g_{i-1}\} \cup \{g_i\}) - \tau(\{g_1, \dots, g_{i-1}\})$$

since  $\{g_1, \dots, g_{i-1}\} \subseteq J_{i-1}$ .

This implies that  $\tau(\{g_1, \dots, g_k\}) = k$  and  $G_{\{g_1, \dots, g_k\}}$  has a matching  $M$  of size  $k$ . We define the function  $\phi$  by setting  $\phi(f_i) = c_i$  where  $g_i$  is matched to  $c_i$  in  $M$ . ◀

We can now obtain Theorem 30.

**Proof of Theorem 30.** Clearly  $\mu(E) = \frac{(n-k)!}{n!}$ . To enumerate a set  $I \in \mathfrak{D}(E)$ , by Proposition 44, we choose, for each  $g \in C$ , either zero or one preimages  $f = \phi_I^{-1}(g)$  in  $I$ . If we write  $I_g$  for the set of preimages of  $g$ , then  $|I_g| \leq 1$  for all  $g$  and  $I = \bigcup_{g \in C} I_g$ . Overall, this shows that

$$\sum_{I \in \mathfrak{D}(E)} \Psi(I) \leq \sum_{I_{g_1}, \dots, I_{g_k}} \Psi(I_{g_1} \cup \dots \cup I_{g_k}) \leq \sum_{I_{g_1}, \dots, I_{g_k}} \Psi(I_{g_1}) \cdots \Psi(I_{g_k})$$

where the last inequality follows from log-subadditivity of  $\Psi$ . This can be written as  $\prod_{i=1}^k \sum_{I_{g_i}} \Psi(I_{g_i})$ . The case of  $I_{g_i} = \emptyset$  contributes 1, and the case of  $I_{g_i} = \{f\}$  contributes  $\Psi(f)$ . ◀

We next consider the setting where  $\Omega$  is the set of perfect matchings  $M$  on the clique on vertex set  $[n]$ , where  $n$  is an even integer. The set  $\mathcal{A}$  is defined as follows: for each pair  $(x, y) \in [n] \times [n]$  with  $x \neq y$ , there is an atomic event that  $M \supseteq \{x, y\}$ . We denote this event by  $[x, y]$ ; note that  $[x, y] = [y, x]$ , which is different from the permutation setting. The resampling oracle, for such an event with  $x < y$ , is to update the state by drawing  $z$  uniformly from  $[n] - x$  and setting  $M \leftarrow (y z)M$ . Here, we are using the natural left-group action of permutations on matchings, i.e.  $\sigma M = \{\{\sigma x', \sigma y'\} \mid \{x', y'\} \in M\}$ .

We have  $[x, y] \sim [x', y']$  if  $|\{x, y\} \cap \{x', y'\}| = 1$ . Equivalently, this holds iff  $[x, y] \cap [x', y'] = \emptyset$ . See [15] for further details, including a proof that this resampling oracle is commutative and oblivious.

As before, we begin with a basic observation on how atoms of  $\mathcal{A}$  interact with events in  $\overline{\mathcal{A}}$ .

► **Proposition 45.** *Let  $f = [x, y]$  be an atom and let  $E = \langle C \rangle$  where  $C$  is a stable set of  $\mathcal{A}$ . Then  $A_f e_E \propto e_{E'}$ , where  $E' = \langle C' \rangle$  and stable set  $C'$  is obtained from  $C$  as follows:*

- *If  $C$  contains exactly two atoms  $f_1, f_2$  which are neighbors of  $f$ , i.e.  $f_1 = [x, y_1]$  and  $f_2 = [x_2, y]$ , then  $C' = C - \{f_1, f_2\} \cup \{[x, y], [x_2, y_1]\}$ .*
- *If  $C$  contains exactly one atom  $f_1$  which is a neighbor of  $f$ , then  $C' = C - f_1 \cup \{f\}$ .*
- *Otherwise, if  $C$  contains no neighbors of  $f$ , then  $C' = C \cup \{f\}$ .*

**Proof.** Consider state  $M \in f$ , and suppose we resample  $f$  to  $M' = (y z)M$  where  $z$  is drawn from  $[n] - x$ . For each  $f' \in C$  which is not a neighbor of  $f$ , we must have  $M \in f'$  as otherwise  $e_M^\top A_f e_E = 0 = e_M^\top e_{E'}$ . In these cases, we also automatically have  $M' \in f'$  for all such  $f'$ . Thus, we suppose that  $M \in f$  and also  $M \in f'$  for all  $f' \in C - \Gamma(f)$ . We consider the following cases:

- Suppose that  $C$  contains two atoms  $f_1, f_2$ . Then we claim that  $M' \in E$  precisely when  $z = y_1$  and  $\{x_2, y\} \in M$ . First, we have  $\{x, y_1\} \in M'$  with  $M' = (y z)M$  iff  $z = y_1$ . Thus,  $M' = (y y_1)M$ . To satisfy  $f_2$ , we must have  $\{x_2, y_1\} \in M$ . In this case,  $e_M^\top A_f e_E = 1/(n-1)$  and also  $e_M^\top e_{E'} = 1$ .
- Suppose that  $C$  contains a neighbor  $f_1 = [x, y_1]$ . In this case, we have  $M' \in f_1$  precisely if  $y_1 = z$ . Thus, we have  $e_M^\top A_f e_E = 1/(n-1)$  and also  $e_M^\top e_{E'} = 1$ .
- Suppose  $C$  has no neighbors of  $f$ . Let  $C = \{[x_1, x_2], \dots, [x_{2k-1}, x_{2k}]\}$ ; we have  $M \in E'$  precisely if  $z \notin \{x_1, \dots, x_{2k}\}$ . Since  $z \neq x$ , we thus have  $e_M^\top A_f e_E = \frac{n-2k+1}{n-1}$ . We also have  $e_M^\top e_{E'} = 1$ . ◀

To understand multi-atom interactions, let us fix event  $E = \langle C \rangle$  for a stable set  $C$ . For a stable set  $I \subseteq \mathcal{A}$ , we can form an associated bipartite graph  $G_I$ , whose left vertices correspond to  $C$  and whose right vertices correspond to  $I$ . It has an edge between  $f$  and  $f'$  iff  $f \sim f'$ . Observe that since  $C$  and  $I$  are stable, the graph  $G_I$  has degree at most two – each node  $[x, y]$  can have one neighbor of the form  $[x', y]$  and another neighbor of the form  $[x, y']$ . So,  $G_I$  decomposes into paths and cycles.

We define  $\tau(I)$  to be the size of a maximum matching in  $G_I$ . We also define  $\text{Active}(I) \subseteq \mathcal{A}$  as follows. First, for each  $f \in I$ , we also place  $f$  into  $\text{Active}(I)$ . Second, consider some maximal path of  $G_I$  starting and ending at  $C$ -nodes; the path can be written as  $[x_1, x_2], [x_2, x_3], \dots, [x_{k-1}, x_k]$  for even  $k$ . In this case, we also put  $[x_1, x_k]$  into  $\text{Active}(I)$ .

► **Proposition 46.** *Let  $I$  be a stable set of  $\mathcal{A}$ . Then the following properties hold:*

1.  $A_I e_E \propto e_{\alpha(I)}$  where  $\alpha(I) = \langle \text{Active}(I) \rangle$ .
2.  $|\text{Active}(I)| = |C| + |I| - \tau(I)$ .
3. Any  $f \in I$  with  $\tau(I) = \tau(I - f)$  has  $\text{Active}(I) = \text{Active}(I - f) \cup \{f\}$ .

We omit the the proof of Proposition 44, as well as the remainder of the proof of Theorem 31, as they are precisely analogous to the proof of Theorem 30.



# From Coupling to Spectral Independence and Blackbox Comparison with the Down-Up Walk

Kuikui Liu ✉

University of Washington, Seattle, WA, USA

---

## Abstract

---

We show that the existence of a “good” coupling w.r.t. Hamming distance for any local Markov chain on a discrete product space implies rapid mixing of the Glauber dynamics in a blackbox fashion. More specifically, we only require the expected distance between successive iterates under the coupling to be summable, as opposed to being one-step contractive in the worst case. Combined with recent local-to-global arguments [16], we establish asymptotically optimal lower bounds on the standard and modified log-Sobolev constants for the Glauber dynamics for sampling from spin systems on bounded-degree graphs when a curvature condition [44] is satisfied. To achieve this, we use Stein’s method for Markov chains [10, 46] to show that a “good” coupling for a local Markov chain yields strong bounds on the spectral independence of the distribution in the sense of [6].

Our primary application is to sampling proper list-colorings on bounded-degree graphs. In particular, combining the coupling for the flip dynamics given by [49, 13] with our techniques, we show optimal  $O(n \log n)$  mixing for the Glauber dynamics for sampling proper list-colorings on any bounded-degree graph with maximum degree  $\Delta$  whenever the size of the color lists are at least  $(\frac{11}{6} - \epsilon) \Delta$ , where  $\epsilon \approx 10^{-5}$  is small constant. While  $O(n^2)$  mixing was already known before, our approach additionally yields Chernoff-type concentration bounds for Hamming Lipschitz functions in this regime, which was not known before. Our approach is markedly different from prior works establishing spectral independence for spin systems using spatial mixing [6, 14, 15, 30], which crucially is still open in this regime for proper list-colorings.

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

**Keywords and phrases** Markov chains, Approximate counting, Spectral independence

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.32

**Category** RANDOM

**Funding** *Kuikui Liu*: The author is supported by NSF grants CCF-1552097, CCF-1907845.

**Acknowledgements** The author would like to thank their advisor Shayan Oveis Gharan for comments on a preliminary draft of this paper. We also thank Nima Anari and Pierre Youssef for informing us that a conjecture posed in a preliminary draft of the paper was already known to be false. We finally thank the anonymous reviewers for delivering valuable feedback on this paper.

## 1 Introduction

Given a probability distribution  $\mu$  on a collection of subsets of a finite universe  $U$  with a fixed size  $n$ , one would like to generate (approximate) samples from  $\mu$ . This problem is widely encountered in machine learning, statistical physics, and theoretical computer science, and encompasses many problems as special cases, including distributions over bases of matroids, discrete probabilistic graphical models, etc. A popular approach used in practice is to run a Markov chain on  $\text{supp}(\mu)$  whose stationary distribution is  $\mu$ . The main question then becomes how quickly does the distribution of the Markov chain converge to stationarity, i.e. does it mix rapidly?

A particularly natural Markov chain known as the “down-up walk” (or “high-order walk”) originally studied in the high-dimensional expander community [39, 21, 40, 45, 1] has recently received a lot of attention due to applications to sampling from discrete log-concave



© Kuikui Liu;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 32; pp. 32:1–32:21



Leibniz International Proceedings in Informatics

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

distributions [4, 18, 3, 5] and spin systems in statistical physics [1, 6, 14, 15, 30, 2, 16]. For sampling bases of matroids, the down-up walk recovers exactly the bases exchange walk first studied in [29, 43], and for sampling from discrete graphical models, the down-up walk recovers exactly the classical Glauber dynamics. One of the main insights in this area is that to prove rapid mixing of the down-up walk, it suffices to look only at pairwise correlations between elements, albeit for all conditional distributions of  $\mu$  (see Definition 13). One additional advantage behind this approach is that one can prove local-to-global results not just for the spectral gap [40, 1], but also for the rate of entropy decay [18, 34, 16] and even for the rate of decay for arbitrary  $f$ -divergences [2]. This has led to asymptotically optimal mixing times for many problems [18, 5, 16] as well as Chernoff-type concentration bounds for Lipschitz functions.

However, establishing sufficiently strong bounds on pairwise correlations (or, more precisely, pairwise influences; see Definition 13), remains a challenging problem. Prior works typically rely on one of three techniques: Oppenheim’s trickle-down theorem [45, 4], spatial mixing (or correlation decay) [6, 14, 15, 30, 16], or the absence of roots for the multivariate generating polynomial of  $\mu$  in a sufficiently large region of the complex plane [2, 17]. However, there are settings, such as proper list-colorings when the number of colors is less than twice the maximum degree of the graph, where the trickle-down theorem fails, and where spatial mixing and the existence of a nice root-free region are not known.

In this work, we show that the classical technique of (path) coupling can be used to bound these pairwise correlations. In fact, we will show that the existence of a “good” coupling for **any** sufficiently “local” dynamics with stationary distribution  $\mu$  implies spectral independence for  $\mu$  in the sense of [6], and hence, rapid mixing of the down-up walk. Hence, one can view our main result as a blackbox comparison result between any local Markov chain and the down-up walk.

As our main concrete application, we use the variable-length path coupling devised by [13] for the flip dynamics, building off work of Vigoda [49], to show  $O(n \log n)$  mixing of the Glauber dynamics for sampling proper list-colorings on graphs of maximum degree  $\Delta \leq O(1)$  whenever the number of available colors is at least  $(\frac{11}{6} - \epsilon) \Delta$ , where  $\epsilon \approx 10^{-5}$  is a small constant. This mixing time is asymptotically optimal [36]. While  $O(n^2)$  mixing was known earlier [31] (see also [49, 13]) by using a spectral gap comparison argument [19], our approach yields optimal bounds on the rate of entropy decay as well as Chernoff-type concentration inequalities. As mentioned earlier, strong spatial mixing and the existence of sufficiently large root-free regions are not known in this regime for proper list-colorings. Along the way, we make an additional conceptual contribution by answering the natural question of if Dobrushin-type mixing conditions imply spectral independence.

## 1.1 Our Contributions

To state our blackbox comparison result, let us first define the down-up walk for sampling from distributions over homogeneous set systems. We eschew the use of much of the terminology of high-dimensional expanders so as to simplify the exposition. Let  $\mu$  be a distribution over  $\binom{U}{n} = \{S \subseteq U : |S| = n\}$  for a finite set  $U$  and a positive integer  $n \geq 1$ <sup>1</sup>. The down-up walk

---

<sup>1</sup> One can view  $\mu$  as being a distribution over the facets of a “pure simplicial complex” weighted by  $\mu$ . These are a generalization of usual graphs which are studied in geometry, topology, and combinatorics. The notion of spectral independence (Definition 13) was derived from a high-dimensional notion of “expansion” for simplicial complexes known as “local spectral expansion” first discovered by [21, 40, 45].

is described by the following two-step process. If the current state of the chain is  $S^{(t)}$ , then we select the next state  $S^{(t+1)}$  as follows:

1. Select a uniformly random element  $i \in S$ .
2. Sample a set  $S \in \text{supp}(\mu)$  satisfying  $S \supseteq S^{(t)} \setminus \{i\}$  with probability proportional to  $\mu(S)$  and transition to  $S^{(t+1)} = S$ .

As special cases, this class of Markov chains includes the bases exchange walk for matroids [43] and the Glauber dynamics for distributions over discrete product spaces.

We also define our notion of a “good” coupling and locality precisely here.

► **Definition 1 (Amortized Convergent Coupling).** Fix an irreducible transition probability matrix  $P$  which is reversible w.r.t. a distribution  $\pi$  on a finite state space  $\Omega$ . Further endow  $\Omega$  with a metric  $d(\cdot, \cdot)$ . We say a coupling of two faithful copies of the chain  $(X^{(t)})_{t \geq 0}, (Y^{(t)})_{t \geq 0}$  is  **$C$ -amortized convergent w.r.t.  $d(\cdot, \cdot)$**  if the following holds for all  $x, y \in \Omega$ :

$$\sum_{t=0}^{\infty} \mathbb{E}_{X^{(t)}, Y^{(t)}} \left[ d(X^{(t)}, Y^{(t)}) \mid \begin{matrix} X^{(0)}=x \\ Y^{(0)}=y \end{matrix} \right] \leq C \cdot d(x, y).$$

► **Definition 2 (Locality of Dynamics).** Fix an irreducible transition probability matrix  $P$  which is reversible w.r.t. a distribution  $\pi$  on a finite state space  $\Omega$ . Further endow  $\Omega$  with a metric  $d(\cdot, \cdot)$ . For a positive real number  $\ell > 0$ , we say the dynamics  $P$  is  **$\ell$ -local w.r.t.  $d(\cdot, \cdot)$**  if

$$\max_{x, y \in \Omega: P(x, y) > 0} d(x, y) \leq \ell.$$

Throughout the paper, unless stated otherwise, we work with Hamming distance. With these notions in hand, we now state our blackbox comparison result.

► **Theorem 3 (Blackbox Comparison with Down-Up Walk).** Let  $\mu$  be a distribution on  $\binom{U}{n}$ , where  $U$  is a finite universe and  $n \geq 1$  is a positive integer. For each  $A \subseteq U$  with  $|A| \leq n-2$  and  $A \subseteq S$  for some  $S \in \text{supp}(\mu)$ , let  $P_{\mu|A}$  be some Markov chain on  $\text{supp}(\mu \mid A)$  with stationary distribution  $\mu \mid A$ . Assume the family of Markov chains  $\{P_{\mu|A}\}$  satisfy the following:

1. *Locality:* For some  $\ell \geq 0$ ,  $P_{\mu|A}$  is  $\ell$ -local w.r.t. Hamming distance for all  $A$ .
2. *Good Coupling:* For some  $C_{n-k} > 0$ ,  $P_{\mu|A}$  admits a  $C_{n-k}$ -amortized convergent coupling w.r.t. Hamming distance for all  $k$  and  $A$  with  $|A| = k$ .
3. *Bounded Differences Between Chains:* For some  $C'_{n-k} > 0$ , we have the bound

$$\max_{S \in \text{supp}(\mu \mid (A \cup i))} \left\{ \sum_{T \neq S} |P_{\mu|A}(S \rightarrow T) - P_{\mu|(A \cup i)}(S \rightarrow T)| \right\} \leq C'_{n-k},$$

for all  $k, i$  and  $A$  with  $|A| = k$ .

If  $\ell \cdot C_{n-k} \cdot C'_{n-k} \leq O(1)$  for all  $k$ , then the down-up walk has spectral gap at least  $n^{-O(1)}$ . If, in addition,  $\mu$  is the Gibbs distribution of a spin system (see Section 2.1) on a bounded-degree graph, then the spectral gap, standard and modified log-Sobolev constants Equation (4) for the down-up walk are all  $\Omega(1/n)$ .

We refer the reader to Section A and references therein for the importance of lower bounding the spectral gap, standard and modified log-Sobolev constants, and in particular, their relation to mixing and concentration.

► **Remark 4.** While initially it may seem inconvenient to first build an entire family of Markov chains, one for each conditional distribution, this is very natural for many classes of distributions, in particular those which are closed under conditioning. As we will see, in practice, it is easy to obtain bounded differences between chains with  $C'_{n-k} \lesssim \frac{1}{n-k}$  simply via brute force calculation. While  $C_{n-k} \gtrsim n-k$  is often unavoidable, particularly for  $\ell$ -local chains with  $\ell \leq O(1)$ , we will see that in many settings, we have  $C_{n-k} \lesssim n-k$  as well. If additionally our dynamics are  $\ell$ -local with  $\ell \leq O(1)$ , then the above yields a  $n^{-O(1)}$  spectral gap for the down-up walk. It will turn out that our notion of  $\ell$ -locality can also be relaxed; see Remark 10.

Our primary concrete application is to sampling proper list-colorings on graphs via the Glauber dynamics, which may be realized as a down-up walk. In this setting, we compare with another useful Markov chain known as the flip dynamics. The flip dynamics is  $\ell$ -local w.r.t. unweighted Hamming distance with  $\ell \leq 12$ , and was analyzed in [49], who gave a greedy coupling which is one-step contractive whenever the number of available colors is at least  $\frac{11}{6}\Delta$ , implying it is  $C$ -amortized convergent with  $C \leq O(n)$ . [13] tweaked the parameters of the flip dynamics slightly while preserving locality, and further constructed a variable-length coupling which contracts by a constant factor every expected  $O(n)$  steps whenever the number of available colors is at least  $(\frac{11}{6} - \epsilon)\Delta$  for a small constant  $\epsilon \approx 10^{-5}$ . We will show this variable-length coupling is also  $C$ -amortized convergent with  $C \leq O(n)$ , and deduce optimal mixing for list-colorings in this regime.

► **Theorem 5.** *Let  $(G, \mathcal{L})$  be a list-coloring instance where  $G = (V, E)$  is a graph of maximum degree  $\Delta \leq O(1)$  and  $\mathcal{L} = (L(v))_{v \in V}$  is a collection of color lists. Then for some absolute constant  $\epsilon \approx 10^{-5}$ , if  $|L(v)| \geq (\frac{11}{6} - \epsilon)\Delta$  for all  $v \in V$ , then the uniform distribution over proper list-colorings for  $(G, \mathcal{L})$  is  $(\eta_0, \dots, \eta_{n-2})$ -spectrally independent where  $\eta_k \leq O(1)$  for all  $k$ . Furthermore, the spectral gap, standard and modified log-Sobolev constants Equation (4) for the Glauber dynamics are all  $\Omega(1/n)$ , and the mixing time is  $O(n \log n)$ .*

► **Remark 6.** Our running time dependence on  $\Delta$  is roughly  $\Delta^{O(\Delta^c)}$  for a mild constant  $c$ , which is rather poor. The main bottleneck in improving this dependence lies in the local-to-global result of [16], although our spectral independence bound, which depends polynomially on  $\Delta$ , can also be significantly improved.

To prove Theorem 3, we leverage recent local-to-global results [1, 16] (see Theorems 14 and 15 for formal statements), which show that if one has sufficiently strong upper bounds on the total pairwise correlation  $\sum_{j \in U} |\Pr_{S \sim \mu}[j \in S] - \Pr_{S \sim \mu|i}[j \in S]|$ , then one can deduce rapid mixing for the down-up walk [39, 21, 40, 45]. To upper bound these correlations, we considerably generalize a result simultaneously due to [10, 46], which was discovered in the context of bounding the Wasserstein 1-distance between Ising models, or more generally, two measures on the discrete hypercube  $\{-1, +1\}^n$ . More specifically, we extend their results in several different directions:

1. We replace the Glauber dynamics by any local dynamics.
2. We allow the dynamics to admit a coupling which in a sense “contracts on average”, as opposed to a step-wise contraction in the worst-case.

► **Theorem 7.** *Let  $\mu$  be a distribution on  $\binom{U}{n}$ , where  $U$  is some finite universe and  $n \geq 1$  is a positive integer. Fix an arbitrary  $i \in U$ . Let  $P_\mu$  (resp.  $P_{\mu|i}$ ) be the transition kernel of any irreducible Markov chain on  $\text{supp}(\mu)$  (resp.  $\text{supp}(\mu | i)$ ) which is reversible w.r.t.  $\mu$  (resp.  $\text{supp}(\mu | i)$ ). Suppose that  $P_\mu$  is  $\ell$ -local and admits a  $C$ -amortized convergent coupling, both*



w.r.t. the Hamming metric  $d_H(\cdot, \cdot)$ . Then we have the bound

$$\sum_{j \in U} \left| \Pr_{S \sim \mu} [j \in S] - \Pr_{S \sim \mu|i} [j \in S] \right| \leq C \cdot \ell \cdot \max_{S \in \text{supp}(\mu|i)} \left\{ \sum_{T \neq S} |P_\mu(S \rightarrow T) - P_{\mu|i}(S \rightarrow T)| \right\}.$$

## 1.2 Main Technical Result

We now state our main technical result, which provides the most general bound on the difference between marginals of two distributions  $\mu, \nu$ . We immediately use it to deduce Theorem 7.

► **Theorem 8 (Main Technical).** *Let  $\mu, \nu$  be any two distributions on  $2^U$  for a finite set  $U$  with  $\text{supp}(\nu) \subseteq \text{supp}(\mu)$ , where  $U$  is a finite universe and  $n \geq 1$  is a positive integer. Further, let  $P_\mu$  (resp.  $P_\nu$ ) be the transition kernel of any Markov chain on  $\text{supp}(\mu)$  (resp.  $\text{supp}(\nu)$ ) with stationary distribution  $\mu$  (resp.  $\nu$ ). Assume  $P_\mu$  is irreducible and reversible w.r.t.  $\mu$ . Then we may bound both  $\sum_{j \in U} |\Pr_{S \sim \mu}[j \in S] - \Pr_{S \sim \nu}[j \in S]|$  and the 1-Wasserstein distance  $W_1(\mu, \nu)$  (see Definition 11) by the following quantity:*

$$\mathbb{E}_{S \sim \nu} \left[ \sum_{T \neq S} |P_\mu(S \rightarrow T) - P_\nu(S \rightarrow T)| \cdot \sum_{t=0}^{\infty} \mathbb{E}_{X^{(t)}, Y^{(t)}} \left[ d_H(X^{(t)}, Y^{(t)}) \mid \begin{matrix} X^{(0)}=S \\ Y^{(0)}=T \end{matrix} \right] \right],$$

where  $(X^{(t)}, Y^{(t)})_{t=0}^{\infty}$  is a coupling of the Markov chain  $P_\mu$ .

► **Remark 9.** The technical condition  $\text{supp}(\nu) \subseteq \text{supp}(\mu)$  is just for convenience, as it ensures the transition probability  $P_\mu(S \rightarrow T)$  also makes sense when  $S \sim \nu$ . This assumption is certainly satisfied in our application where  $\nu$  is a conditional distribution of  $\mu$ .

**Proof of Theorem 7.** We use Theorem 8 with  $\nu = \mu | i$  to obtain the upper bound

$$\begin{aligned} & \mathbb{E}_{S \sim \mu|i} \left[ \sum_{T \neq S} |P_\mu(S \rightarrow T) - P_{\mu|i}(S \rightarrow T)| \cdot \sum_{t=0}^{\infty} \mathbb{E}_{X^{(t)}, Y^{(t)}} \left[ d_H(X^{(t)}, Y^{(t)}) \mid \begin{matrix} X^{(0)}=S \\ Y^{(0)}=T \end{matrix} \right] \right] \\ & \leq \max_{S \in \text{supp}(\mu|i)} \left\{ \sum_{T \neq S} |P_\mu(S \rightarrow T) - P_{\mu|i}(S \rightarrow T)| \right\} \\ & \quad \cdot \underbrace{\mathbb{E}_{S \sim \mu|i} \left[ \max_{T: P_\mu(S \rightarrow T) > 0} \sum_{t=0}^{\infty} \mathbb{E}_{X^{(t)}, Y^{(t)}} \left[ d_H(X^{(t)}, Y^{(t)}) \mid \begin{matrix} X^{(0)}=S \\ Y^{(0)}=T \end{matrix} \right] \right]}_{(*)}. \end{aligned}$$

It suffices to bound  $(*)$  by  $C \cdot \ell$ . Since  $P_\mu$  admits a  $C$ -amortized convergent coupling, we have that

$$\sum_{t=0}^{\infty} \mathbb{E}_{X^{(t)}, Y^{(t)}} \left[ d_H(X^{(t)}, Y^{(t)}) \mid \begin{matrix} X^{(0)}=S \\ Y^{(0)}=T \end{matrix} \right] \leq C \cdot d_H(S, T).$$

Hence,

$$(*) \leq C \cdot \mathbb{E}_{S \sim \mu|i} \left[ \max_{T: P_\mu(S \rightarrow T) > 0} d_H(S, T) \right] \leq C \cdot \ell. \quad \blacktriangleleft$$

► Remark 10. One can see from the proof that we only needed that

$$\mathbb{E}_{S \sim \mu | i} \left[ \max_{T: P_\mu(S \rightarrow T)} d_H(S, T) \right] \leq \ell,$$

as opposed to the stronger notion of  $\ell$ -locality, where we have  $\max_{S, T: P_\mu(S \rightarrow T)} d_H(S, T) \leq \ell$ . Thus, in some sense, we only need the dynamics to make local moves “on average”. We leave it to future work to exploit this additional flexibility.

### 1.3 Independent Work

The results we obtain here were also independently discovered in [7].

### 1.4 Organization of the Paper

We state preliminaries on spin systems, spectral independence, etc. in Section 2. We then move to the proof of our main technical result (Theorem 8) in Section 3. In Section 4, we apply our techniques to distributions on discrete product spaces. In Section 5, we combine our techniques with couplings constructed in prior works to obtain spectral independence for proper list-colorings.

## 2 Preliminaries

For a positive integer  $n \geq 1$ , we write  $[n] = \{1, \dots, n\}$ . For a distribution  $\mu$  on some finite state space  $\Omega$ , we write  $\text{supp}(\mu) = \{x \in \Omega : \mu(x) > 0\}$  for the support of  $\mu$ . For a matrix  $A$ , we write  $\|A\|_\infty = \max_i \sum_j |A(i, j)|$  for the maximum absolute row sum, and if  $A$  has real eigenvalues, we write  $\lambda_{\max}(A)$  for the largest eigenvalue of  $A$ .

Throughout, we write  $G = (V, E)$  for an undirected graph, and we will write  $\Delta$  for the maximum degree of  $G$ . For a finite universe  $U$  and  $S \subseteq U$ , we write  $\mathbb{I}_S$  for the  $\{0, 1\}$ -indicator function of  $S$ ; for an element  $j \in U$ , we write  $\mathbb{I}_j$  as opposed to  $\mathbb{I}_{\{j\}}$ . If  $\mu$  is a distribution over  $\binom{U}{n}$  and  $S \subseteq U$ , then we write  $\mu | S$  for the conditional distribution of  $\mu$  on  $\binom{U \setminus S}{n - |S|}$ , where  $(\mu | S)(T) \propto \mu(S \cup T)$  whenever  $S \cup T \in \text{supp}(\mu)$ ,  $S \cap T = \emptyset$ , and  $(\mu | S)(T) = 0$  otherwise.

We will measure convergence of our Markov chains using total variation distance, defined as

$$d_{\text{TV}}(\mu, \nu) = \frac{1}{2} \sum_x |\mu(x) - \nu(x)| = \sup_{S \subseteq \Omega} |\mu(S) - \nu(S)|$$

for two distributions  $\mu, \nu$  on a common state space  $\Omega$ . We define the  $\epsilon$ -mixing time of a Markov chain  $P$  on a state space  $\Omega$  with stationary distribution  $\pi$  as

$$t_{\text{mix}}(\epsilon) \stackrel{\text{def}}{=} \max_{x \in \Omega} \min\{t \geq 0 : d_{\text{TV}}(\mathbb{I}_x P^t, \pi) \leq \epsilon\}.$$

The mixing time of the chain is defined as  $t_{\text{mix}}(1/4)$ . For the reader’s convenience, in Section A, we record the relation between mixing, spectral gap, modified and standard log-Sobolev constants. Finally, we also define the 1-Wasserstein distance.

► **Definition 11** (1-Wasserstein Distance). *Given two probability measures  $\mu, \nu$  on a common state space  $\Omega$  endowed with a metric  $d(\cdot, \cdot)$ , we define the 1-Wasserstein distance  $W_1(\mu, \nu)$  w.r.t.  $d(\cdot, \cdot)$  by*

$$W_1(\mu, \nu) = \sup_f |\mathbb{E}_\mu f - \mathbb{E}_\nu f|,$$

where the supremum is over functions  $f : \Omega \rightarrow \mathbb{R}$  which are 1-Lipschitz w.r.t.  $d(\cdot, \cdot)$  (i.e.  $|f(x) - f(y)| \leq d(x, y)$  for all  $x, y \in \Omega$ ).

► **Remark 12.** By Kantorovich duality, one may equivalently define the 1-Wasserstein distance as

$$W_1(\mu, \nu) = \inf_{\gamma} \mathbb{E}_{(x,y) \sim \gamma} [d(x, y)],$$

where the infimum is over all couplings  $\gamma$  of  $\mu, \nu$  on  $\Omega \times \Omega$ .

## 2.1 Spin Systems

Fix an undirected graph  $G = (V, E)$ , and a positive integer  $q \geq 2$ . We view  $[q]$  as a collection of possible “spin assignments” for the vertices of  $G$ . We also fix a symmetric nonnegative matrix  $A \in \mathbb{R}_{\geq 0}^{q \times q}$  of “edge interaction activities” and a positive vector  $h \in \mathbb{R}_{> 0}^q$  of “external fields”. The Gibbs distribution of the spin system on  $G = (V, E)$  with parameters  $A, h$  is the distribution  $\mu = \mu_{G,A,h}$  over configurations  $\sigma : V \rightarrow [q]$  given by

$$\mu(\sigma) \propto \prod_{\{u,v\} \in E} A(\sigma(u), \sigma(v)) \prod_{v \in V} h(\sigma(v)),$$

where the constant of proportionality is the partition function of the system, given by

$$Z_G(A, h) = \sum_{\sigma: V \rightarrow [q]} \prod_{\{u,v\} \in E} A(\sigma(u), \sigma(v)) \prod_{v \in V} h(\sigma(v)).$$

Many classical models in statistical physics as well as distributions over often-studied combinatorial objects on graphs may be found as special cases. Notable examples include the Ising model on cuts, the hardcore gas model on independent sets, the monomer-dimer model on matchings, and the zero-temperature antiferromagnetic Potts model on proper colorings.

We call a configuration  $\sigma : V \rightarrow [q]$  feasible if  $\mu(\sigma) > 0$ . For instance, if  $A$  has all positive entries, then all configurations  $\sigma : V \rightarrow [q]$  are feasible. We call a partial configuration  $\xi : S \rightarrow [q]$ , where  $S \subseteq V$  is a subset of vertices, a boundary condition. For such a boundary condition, we write  $\mu \mid \xi$  for the conditional Gibbs distribution on  $V \setminus S$  given by taking  $\mu$  and conditioning on the event that the sampled  $\sigma \sim \mu$  satisfies  $\sigma(v) = \xi(v)$  for all  $v \in S$ .

## 2.2 Discrete Product Spaces and Homogeneous Set Systems

Fix a collection of finite sets  $(\Omega(v))_{v \in V}$ , where  $V$  is some finite index set with  $|V| = n$ , and consider a measure  $\mu$  on the product space  $\prod_{v \in V} \Omega(v)$ . For instance, if  $\Omega(v) = \{-1, +1\}$  for each  $v \in V$ , then  $\mu$  is just a measure on the discrete hypercube  $\{-1, +1\}^V$ . An important subclass of examples which we will discuss at length include discrete probabilistic graphical models, where the index set  $V$  is the set of vertices of a (hyper)graph, and the measure  $\mu$  designed in such a way that the (hyper)edges represent local interactions between vertices of the model; see Section 2.1 for more details.

As done in [6, 14, 15, 30], we view  $\mu$  as a measure on  $\binom{U}{n}$  where

$$U = \{(v, \omega(v)) : v \in V, \omega(v) \in \Omega(v)\}.$$

Note the usual Hamming distance  $d_H(\cdot, \cdot)$  on  $\binom{U}{n}$  is twice the usual Hamming distance typically associated with a discrete product space.

We will often write a single vertex-assignment pair  $(v, c)$ , where  $v \in V$  and  $c \in \Omega(v)$ , as simply  $vc$ . Here, each configuration  $\sigma \in \prod_{v \in V} \Omega(v)$  corresponds to the set  $\{(v, \sigma(v)) : v \in V\}$ . In this setting, the down-up walk is precisely the **Glauber dynamics** (or **Gibbs sampler**) for sampling from  $\mu$ . For each configuration  $\sigma \in \prod_{v \in V} \Omega(v)$ , we transition to the next configuration by the following process:

1. Select a uniformly random coordinate  $v \in V$ .
2. Resample  $\sigma(v)$  according to  $\mu$  conditioned on  $\sigma_{-v}$ .

Let us make this more concrete. For each  $\sigma \in \prod_{v \in V} \Omega(v)$  and  $v \in V$ , we write  $\sigma_{-v}$  for the partial subconfiguration of  $\sigma$  which only excludes  $\sigma(v)$ . For  $c \in \Omega(v)$ , we also write  $\sigma_{vc}$  for the configuration obtained by flipping the coordinate of  $v$  from  $\sigma(v)$  to  $c$ . We may then write  $\mu^v(\cdot | \sigma_{-v})$  for the marginal distribution of  $\sigma(v)$  under  $\mu$  conditioned on  $\sigma_{-v}$ . The transition kernel of the Glauber dynamics may then be written as

$$P_\mu(\sigma \rightarrow \sigma_{vc}) = \frac{1}{n} \cdot \mu^v(c | \sigma_{-v}).$$

### 2.3 Spectral Independence and The Down-Up Walk

Here, we formalize spectral independence and its connection with rapid mixing of the down-up walk. Throughout the paper, we will assume the following connectivity/nondegeneracy condition. Assuming Theorem 7 holds, we will also give a proof of Theorem 3.

**Assumption:** The down-up walk for  $\mu$  and all of its conditional distributions is connected. In the context of spin systems, this condition is guaranteed by “total connectivity” of the system parameters  $(A, h)$  [16]. For instance, this is satisfied by all “soft-constraint” models (i.e. those with  $A > 0$ ), and many “hard-constraint” models such as the hardcore model and the uniform distribution over proper colorings when  $q \geq \Delta + 2$ .

Let us now formalize spectral independence.

► **Definition 13** (Pairwise Influence and Spectral Independence [6]). *Fix a finite universe  $U$  and a positive integer  $n \geq 1$ . Fix a distribution  $\mu$  on  $\binom{U}{n} = \{S \subseteq U : |S| = n\}$ . We define the **pairwise influence** of an element  $i$  on another element  $j$  by*

$$\mathcal{I}_\mu(i \rightarrow j) \stackrel{\text{def}}{=} \Pr_{S \sim \mu} [j \in S | i \in S] - \Pr_{S \sim \mu} [j \in S].$$

We write  $\mathcal{I}_\mu \in \mathbb{R}^{U \times U}$  defined by  $\mathcal{I}_\mu(i, j) = \mathcal{I}_\mu(i \rightarrow j)$  for the **pairwise influence matrix** of  $\mu$ . We say the distribution  $\mu$  is  **$\eta$ -spectrally independent** if  $\lambda_{\max}(\mathcal{I}_\mu) \leq \eta + 1$ . We say the distribution  $\mu$  is  **$(\eta_0, \dots, \eta_{n-2})$ -spectrally independent** if  $\mu$  is  $\eta_0$ -spectrally independent,  $\mu | i$  is  $\eta_1$ -spectrally independent for all  $i \in U$ , and so on.

Often in practice, and in this paper, instead of bounding  $\lambda_{\max}(\mathcal{I}_\mu)$ , we will bound

$$\|\mathcal{I}_\mu\|_\infty = \max_{i \in U} \sum_{j \in U} |\mathcal{I}_\mu(i \rightarrow j)|,$$

which is sufficient since it is well-known that  $\lambda_{\max}(A) \leq \|A\|_\infty$  for any matrix  $A$  with real eigenvalues. The main usefulness of spectral independence is that it implies rapid mixing of the down-up walk, while only requiring bounds on pairwise correlations. We state the main local-to-global results most relevant to us here. In the most general setting, we may deduce an inverse polynomial spectral gap from sufficiently strong spectral independence [21, 40, 1].

► **Theorem 14** ([1], [6]). *Let  $U$  be a finite universe, and  $n \geq 1$  a positive integer. Let  $\mu$  be a distribution on  $\binom{U}{n}$  which is  $(\eta_0, \dots, \eta_{n-2})$ -spectrally independence. Then the down-up walk on  $\binom{U}{n}$  for sampling from  $\mu$  has spectral gap at least*

$$\frac{1}{n} \prod_{k=0}^{n-2} \left(1 - \frac{\eta_k}{n - k - 1}\right).$$

*In particular, if  $\eta_k \leq O(1)$  for all  $k = 0, \dots, n-2$ , then we have  $n^{O(1)}$ -mixing of the down-up walk.*

Subject to a certain mild technical condition on the marginals of the distribution  $\mu$ , one can transfer spectral independence bounds to “local entropy decay” bounds, and then employ versions of the local-to-global result for entropy decay [34, 16, 2]. In the setting of spin systems, one can further take advantage of the bounded-degree assumption to obtain  $O(n \log n)$  mixing time upper bounds [16], which are asymptotically optimal [36]. We state the current state-of-the-art for spin systems on bounded-degree graphs here, as we will need it in our application to proper list-colorings.

► **Theorem 15** ([16]). *Let  $(A, h)$  be the parameters of a spin system, and let  $G = (V, E)$  be a graph with maximum degree at most  $\Delta \leq O(1)$ . If the Gibbs distribution  $\mu = \mu_{G, A, h}$  is  $(\eta_0, \dots, \eta_{n-2})$ -spectrally independent where  $\eta_k \leq O(1)$  for all  $k$ , then both the standard and modified log-Sobolev constants of the Glauber dynamics (i.e. the down-up walk) for sampling from  $\mu$  are at least  $\Omega(1/n)$ .*

We conclude this section with a proof of Theorem 3.

**Proof of Theorem 3.** By Theorems 14 and 15, it suffices to establish  $(\eta_0, \dots, \eta_{n-2})$ -spectral independence for  $\eta_k \leq O(1)$  for all  $k$ . Fix an arbitrary  $A \subseteq U$  with  $|A| = k \leq n - 2$  and  $A \subseteq S$  for some  $S \in \text{supp}(\mu)$ . With the locality and coupling assumptions, Theorem 7 shows that for each  $i \in U$ , the absolute row sum of  $\mathcal{I}_{\mu|A}$  for row  $i$  is upper bounded by

$$\ell \cdot C_{n-k} \cdot \max_{S \in \text{supp}(\mu|_i)} \left\{ \sum_{T \neq S} |P_\mu(S \rightarrow T) - P_{\mu|_i}(S \rightarrow T)| \right\}.$$

Bounded differences between chains then yields the upper bound  $\lambda_{\max}(\mathcal{I}_{\mu|A}) \leq \|\mathcal{I}_{\mu|A}\|_\infty \leq \ell \cdot C_{n-k} \cdot C'_{n-k}$ , which is  $O(1)$  by assumption. As this holds for all such  $A$ , it follows that  $\eta_k \leq O(1)$ . ◀

### 3 Stein’s Method for Markov Chains

Our goal in this section is to prove Theorem 8. We follow [10, 46], using what is known as **Stein’s method for Markov chains**. Historically, Stein’s method [48] was developed as a method to bound distances between probability measures, with the primary motivation being to prove quantitative central limit theorems. [10, 46] adapted this method to bound the distance between two probability measures  $\mu, \nu$  on the discrete hypercube  $\{-1, +1\}^n$  assuming the Glauber dynamics of either measure admits a contractive coupling. Our main intuition lies in viewing spectral independence (see Definition 13) as a measure of distance between different conditionings of the same distribution. Thus, one can try to apply this method to bound the spectral independence of a distribution. Let us now elucidate this method.

For a fixed function  $f : \Omega \rightarrow \mathbb{R}$ , we will construct an auxiliary function  $h : \Omega \rightarrow \mathbb{R}$  which satisfies the **Poisson equation**

$$h - P_\mu h = f - \mathbb{E}_\mu f.$$

Questions concerning  $\mathbb{E}_\mu f$  may then be studied by looking at  $P_\mu h$ . The following lemma constructs  $h$  more explicitly.

► **Lemma 16** (see Lemma 2.1 [10], Lemma 2.3 [46]). *Fix an irreducible transition probability matrix  $P$  which is reversible w.r.t. a distribution  $\pi$  on a finite state space  $\Omega$ . Let  $(X^{(t)})_{t=0}^\infty$  be the Markov chain generated by  $P$ , and for a fixed function  $f : \Omega \rightarrow \mathbb{R}$ , define  $h : \Omega \rightarrow \mathbb{R}$  by*

$$h(x) = \sum_{t=0}^\infty \mathbb{E} \left[ f(X^{(t)}) - \mathbb{E}_\pi f \mid X^{(0)} = x \right].$$

## 32:10 Coupling to Spectral Independence

Then  $h$  is well-defined as a function, and further satisfies the Poisson equation

$$h - Ph = f - \mathbb{E}_\pi f.$$

With this lemma in hand, we can immediately prove Theorem 8.

**Proof of Theorem 8.** Fix a function  $f : 2^U \rightarrow \mathbb{R}$ , and let  $h$  be the solution to the Poisson equation  $h - P_\mu h = f - \mathbb{E}_\mu f$  given in Lemma 16. Then since  $\nu$  is stationary w.r.t.  $P_\nu$ , we have  $\mathbb{E}_\nu P_\nu h = \mathbb{E}_\nu h$ , so that using the Poisson equation yields

$$\mathbb{E}_\nu(P_\nu - P_\mu)h = \mathbb{E}_\nu h - \mathbb{E}_\nu[h - f + \mathbb{E}_\mu f] = \mathbb{E}_\nu f - \mathbb{E}_\mu f.$$

Hence, by the Triangle Inequality, we have that  $|\mathbb{E}_\mu f - \mathbb{E}_\nu f| \leq \mathbb{E}_\nu |(P_\nu - P_\mu)h|$ .

Now, let us bound  $|(P_\nu - P_\mu)h|$  entrywise. For each  $S \in \text{supp}(\nu)$ , using that  $P_\mu(S \rightarrow S) = 1 - \sum_{T \neq S} P_\mu(S \rightarrow T)$  (and analogously for  $P_\nu$ ),

$$\begin{aligned} & (P_\nu - P_\mu)h(S) \\ &= \sum_T (P_\nu(S \rightarrow T) - P_\mu(S \rightarrow T)) \cdot h(T) \\ &= \sum_{T \neq S} (P_\nu(S \rightarrow T) - P_\mu(S \rightarrow T)) \cdot (h(T) - h(S)) \\ &= \sum_{T \neq S} (P_\nu(S \rightarrow T) - P_\mu(S \rightarrow T)) \cdot \sum_{t=0}^{\infty} \mathbb{E}_{X^{(t)}, Y^{(t)}} \left[ f(Y^{(t)}) - f(X^{(t)}) \mid \begin{matrix} X^{(0)}=S \\ Y^{(0)}=T \end{matrix} \right]. \end{aligned} \tag{Lemma 16}$$

It follows by the Triangle Inequality that

$$\begin{aligned} |(P_\nu - P_\mu)h(S)| &\leq \sum_{T \neq S} |P_\mu(S \rightarrow T) - P_\nu(S \rightarrow T)| \\ &\quad \cdot \sum_{t=0}^{\infty} \mathbb{E}_{X^{(t)}, Y^{(t)}} \left[ \left| f(X^{(t)}) - f(Y^{(t)}) \right| \mid \begin{matrix} X^{(0)}=S \\ Y^{(0)}=T \end{matrix} \right]. \end{aligned} \tag{1}$$

Taking expectations w.r.t.  $\nu$  finally yields a bound on  $|\mathbb{E}_\mu f - \mathbb{E}_\nu f|$ . The bound on the 1-Wasserstein distance follows immediately by taking  $f$  to be an arbitrary function which is 1-Lipschitz the metric  $d_H(\cdot, \cdot)$ . To obtain the bound on the total difference between marginals  $\sum_{j \in U} |\Pr_{S \sim \mu}[j \in S] - \Pr_{S \sim \nu}[j \in S]|$ , we apply the above inequality to  $f = \mathbb{I}_j$  for each  $j \in U$  and sum over all  $j \in U$ , noting that  $d_H(S, T) = \sum_{j \in U} |\mathbb{I}_j(S) - \mathbb{I}_j(T)|$  and  $\mathbb{E}_\mu f = \mathbb{E}_\mu \mathbb{I}_j = \Pr_{S \sim \mu}[j \in S]$  (and analogously for  $\nu$ ). ◀

## 4 Discrete Ricci Curvature on Product Spaces

In this section, we discuss applications of our results to general distributions on discrete product spaces. We show that the existence of a contractive coupling w.r.t. Hamming distance for the Glauber dynamics implies  $O(1)$ -spectral independence. Such a condition is known as a discrete Ricci curvature condition for the dynamics in the sense of [44]. This also shows that the Dobrushin uniqueness condition implies  $O(1)$ -spectral independence. When combined with the local-to-global result of [16], we resolve an unpublished conjecture due Peres and Tetali for spin systems on bounded-degree graphs; see [28] and references therein for recent progress on this conjecture on general graphs. We also give an alternative proof of the  $\Omega(1/n)$  lower bound on the standard and modified log-Sobolev constants of the Glauber dynamics in this setting when a Dobrushin-type condition is satisfied, recovering a result of [42].

Classical work on Dobrushin-type conditions [25, 22, 23, 24, 35, 26] yield relatively simple and direct criteria for rapid mixing of the Glauber dynamics [11, 12]. The main idea here is intuitively similar to that of spectral independence (although the notion of Dobrushin influence here historically precedes spectral independence): so long as some measure of “total influence” is small, then  $\mu$  is close in some sense to a product distribution, for which rapid mixing holds. However, prior to our work, the precise relationship between Dobrushin influence and the notion of pairwise influence used in spectral independence was unclear. This is an additional conceptual contribution of our work.

► **Definition 17** (Discrete Ricci Curvature [44]). *Fix an irreducible transition probability matrix  $P$  which is reversible w.r.t. a distribution  $\pi$  on a finite state space  $\Omega$ . Further, endow  $\Omega$  with a metric  $d(\cdot, \cdot)$ . We define the **discrete Ricci curvature** of the Markov chain  $P$  w.r.t. the metric space  $(\Omega, d)$  by*

$$\alpha = \inf_{x, y \in \Omega: x \neq y} \left\{ 1 - \frac{W_1(P(x \rightarrow \cdot), P(y \rightarrow \cdot))}{d(x, y)} \right\},$$

where  $W_1(\cdot, \cdot)$  is again the 1-Wasserstein distance w.r.t.  $d(\cdot, \cdot)$ . In other words, for every pair  $x, y \in \Omega$ , there is a coupling of the transitions  $P(x \rightarrow \cdot), P(y \rightarrow \cdot)$  such that the expected distance  $d(\cdot, \cdot)$  under the coupling contracts by a  $(1 - \alpha)$ -multiplicative factor. In this case, we will say  $P$  admits a  **$(1 - \alpha)$ -contractive coupling w.r.t.  $d(\cdot, \cdot)$** .

► **Fact 18.** *Suppose  $P$  admits a  $(1 - \alpha)$ -contractive coupling w.r.t.  $d(\cdot, \cdot)$ . Then this coupling is  $C$ -amortized convergent with  $C = \frac{1}{\alpha}$ .*

The following is an immediate application of Theorem 7, and yields a positive resolution to the Peres-Tetali conjecture for spin systems on bounded-degree graphs.

► **Theorem 19** (Curvature Implies Spectral Independence on Product Spaces). *Let  $\mu$  be a measure on a discrete product space  $\Omega = \prod_{v \in V} \Omega(v)$ , where  $V$  is a finite index set and  $\Omega(v)$  is finite for all  $v \in V$ . Endow  $\Omega$  with the Hamming metric  $d_H(\cdot, \cdot)$ , and let  $\alpha$  be the discrete Ricci curvature of the Glauber dynamics w.r.t.  $(\Omega, d_H)$ . Then, the distribution is  $(\eta_0, \dots, \eta_{n-2})$ -spectrally independent where  $\eta_k \leq \frac{4}{\alpha n} - 1$  for all  $k$ . In particular, if  $\alpha \geq \Omega(1/n)$ , then the Glauber dynamics has spectral gap  $n^{-O(1)}$ . If additionally the measure  $\mu$  is the Gibbs distribution of a spin system on a bounded-degree graph, then the spectral gap, standard and modified log-Sobolev constants for the Glauber dynamics are all  $\Omega(1/n)$ .*

Note that since the Glauber dynamics only updates the assignment to a single  $v \in V$  in each step, it must be that  $\alpha \leq O(1/n)$ . Theorem 19 follows almost immediately from Fact 18 and a straightforward calculation involving the entries of the transition matrix of the Glauber dynamics. In the interest of space, we provide the proof in Section B.

## 4.1 Dobrushin Uniqueness and Spectral Independence

We now use Theorem 19 to show that Dobrushin’s uniqueness condition implies spectral independence.

► **Definition 20** (Dobrushin Influence). *Fix a probability measure on a finite product space  $\prod_{v \in V} \Omega(v)$ , where  $V$  is a finite indexing set. For each  $u \in V$ , let  $D_u$  be the collection of pairs  $\tau, \sigma \in \prod_{v \in V} \Omega(v)$  such that  $\tau_{-u} = \sigma_{-u}$  while  $\tau(u) \neq \sigma(u)$ . For distinct  $u, v \in V$ , we may then define the **Dobrushin influence** of  $u$  on  $v$  by*

$$\rho_\mu(u \rightarrow v) = \max_{(\tau, \sigma) \in D_u} d_{TV}(\mu^v(\cdot \mid \tau_{-v}), \mu^v(\cdot \mid \sigma_{-v})).$$

## 32:12 Coupling to Spectral Independence

We write  $\rho_\mu = (\rho_\mu(u \rightarrow v))_{u,v} \in \mathbb{R}^{V \times V}$  for the **Dobrushin influence matrix**. We say the distribution  $\mu$  satisfies the **Dobrushin uniqueness condition** if

$$\|\rho_\mu\|_1 \stackrel{\text{def}}{=} \max_{u \in V} \sum_{v \in V} \rho_\mu(u \rightarrow v) < 1.$$

A straightforward application of the path coupling technique of [11, 12] shows that if  $\|\rho_\mu\|_1 < 1$ , then there is a coupling for the Glauber dynamics which is one-step contractive w.r.t. Hamming distance. We state this well-known implication formally here, and refer to [26] for the proof.

► **Fact 21.** *Let  $\mu$  be a distribution on some finite product space  $\prod_{v \in V} \Omega(v)$ , where  $V$  is a finite index set. If  $\|\rho_\mu\|_1 \leq \gamma < 1$ , then the Glauber dynamics is  $(1 - \alpha)$ -contractive w.r.t. Hamming distance with  $\alpha = \frac{1}{n}(1 - \gamma)$ .*

In particular, combining Theorem 19 and Fact 21 immediately yields spectral independence under the Dobrushin uniqueness condition. Combined with Theorem 15, this additionally recovers a version of a result due to [42], which says that a weaker  $\ell_2$ -version of the Dobrushin uniqueness condition (see also [35, 26]) implies a  $\Omega(1/n)$  log-Sobolev constant for the Glauber dynamics.

► **Corollary 22 (Dobrushin Uniqueness Implies Spectral Independence).** *Let  $\mu$  be a distribution on some finite product space  $\prod_{v \in V} \Omega(v)$ , where  $V$  is a finite index set. If  $\|\rho_\mu\|_1 \leq \gamma < 1$ , then  $\mu$  is  $(\eta_0, \dots, \eta_{n-2})$ -spectrally independent with  $\eta_k \leq \frac{4}{1-\gamma} - 1$  for all  $k$ . If additionally the measure  $\mu$  is the Gibbs distribution of a spin system on a bounded-degree graph, then the spectral gap, standard and modified log-Sobolev constants for the Glauber dynamics are all  $\Omega(1/n)$ .*

## 5 Spectral Independence for Proper List-Colorings

We now specialize to the setting of proper list-colorings of a graph. Formally, we fix a graph  $G = (V, E)$ , a collection of color lists  $(L(v))_{v \in V}$ . We call a configuration  $\sigma \in \prod_{v \in V} L(v)$  a list-coloring of  $G$ . We say a list-coloring  $\sigma$  is proper if  $\sigma(u) \neq \sigma(v)$  whenever  $u \neq v$  are neighbors. Throughout, we will let  $\Delta$  denote the maximum degree of  $G$ , and we assume  $\Delta \leq O(1)$ . We also assume there is a positive integer  $q \geq \Delta + 2$  such that  $L(v) \subseteq [q]$  for all  $v \in V$ .

A well-known result due to [38] using path coupling shows that if  $|L(v)| > 2\Delta$  for all  $v \in V$ , then there is a contractive one-step coupling for the Glauber dynamics which yields  $O(n \log n)$  mixing. As noted in [16], one can adapt the argument of [32] to obtain strong spatial mixing when  $|L(v)| > 2\Delta$ , and use the arguments of [15, 30] to deduce spectral independence in this regime. However, it is still open whether one can obtain strong spatial mixing below the  $2\Delta$  threshold; see [32, 27] for results going below  $2\Delta$  on special classes of graphs.

In the seminal work of Vigoda [49], it was shown that there is a contractive one-step coupling for a different local Markov chain known as the flip dynamics whenever  $|L(v)| \geq \frac{11}{6}\Delta$ . This threshold was further improved to  $|L(v)| \geq (\frac{11}{6} - \epsilon)\Delta$  in a recent breakthrough by [13], this time using a more sophisticated variable-length coupling. Both works further showed that Glauber dynamics mixes in  $O(n^2)$  time in this regime using a spectral gap comparison argument [19].

Our goal is to use these coupling results along with Theorem 7 to obtain spectral independence for the uniform distribution over proper list-colorings in the regime  $|L(v)| \geq$



$(\frac{11}{6} - \epsilon) \Delta$ . Combined with Theorem 15, we improve the previous  $O(n^2)$  mixing time bound to the optimal  $O(n \log n)$ , as well as show Chernoff-type concentration bounds for Lipschitz functions, which were not known before.

### 5.1 The Flip Dynamics

We follow the presentation in [13], which generalizes the flip dynamics analyzed in [49] to list-colorings. Fix a list-coloring  $\sigma$ . We say a path  $u = w_1, \dots, w_\ell = v$  in  $G$  is an alternating path from  $u$  to  $v$  using colors  $\sigma(u), c$  if for all  $i$ , we have  $\sigma_{w_i} \in \{\sigma(u), c\}$  and  $\sigma_{w_i} \neq \sigma_{w_{i+1}}$ . For a fixed list-coloring  $\sigma$ ,  $v \in V$  and color  $c$ , we define the **Kempe component for  $\sigma, v, c$**  by the following subset of vertices.

$$S_\sigma(u, c) = \left\{ v \in V : \exists \text{ alternating path from } u \text{ to } v \text{ using } \sigma(u), c \right\}.$$

Given  $\sigma$  and a Kempe component  $S = S_\sigma(u, c)$ , we define  $\sigma_S$  to be the coloring obtained by “flipping” the color assigned to vertices in  $\{v \in S : \sigma(v) = \sigma(u)\}$  to  $c$ , and the color assigned to vertices in  $\{v \in S : \sigma(v) = c\}$  to  $\sigma(u)$ . Note that  $\sigma_S$  need not be a proper list-coloring; we say a Kempe component  $S = S_\sigma(u, c)$  is **flippable** if the coloring  $\sigma_S$  is a proper list-coloring.

For each  $j \in \mathbb{N}$ , let  $0 \leq p_j \leq 1$  be a tunable parameter to be determined later. We define the flip dynamics with flip parameters  $\{p_j\}_{j \in \mathbb{N}}$  for sampling proper list-colorings as follows: Given the current list-coloring  $\sigma^{(t-1)}$ , we generate the next list-coloring  $\sigma^{(t)}$  by the following two-step process:

1. Select a uniformly random vertex  $v^{(t)} \in V$ , and a uniformly random color  $c^{(t)} \in L(v^{(t)})$ .
2. If the Kempe component  $S = S_{\sigma^{(t-1)}}(v^{(t)}, c^{(t)})$  is flippable, set  $\sigma^{(t)} = \sigma_S^{(t-1)}$  with probability  $\frac{p_j}{j}$  and  $\sigma^{(t)} = \sigma^{(t-1)}$  otherwise, where  $j = |S|$ .

We write  $P_{\mu, \text{flip}}$  for the transition probability matrix of the flip dynamics. It is straightforward to verify that the stationary distribution of the flip dynamics is uniform over proper list-colorings, regardless of the choice of the flip parameters. One can recover the Wang-Swendsen-Kotecký Markov chain by setting  $p_j = j$  for all  $j \in \mathbb{N}$  [50].

[49] showed that with flip parameters

$$p_1 = 1 \quad p_2 = \frac{13}{42} \quad p_3 = \frac{1}{6} \quad p_4 = \frac{2}{21} \quad p_5 = \frac{1}{21} \quad p_6 = \frac{1}{84} \quad p_j = 0, \forall j \geq 7, \tag{2}$$

there is a one-step coupling which is contractive w.r.t. Hamming distance whenever  $|L(v)| \geq \frac{11}{6} \Delta$ . [13] showed using linear programming arguments that this is optimal in the sense that when  $|L(v)| < \frac{11}{6}$ , there is no choice of the flip parameters which has a one-step contractive coupling w.r.t. Hamming distance. They additionally construct an explicit family of hard instances witnessing optimality.

One of the key insights of [13] is that the optimal choice of flip parameters comes out of the solution to a linear program, with the objective value of the program governing the contraction properties of the coupling. By solving this linear program, they show that for the following choice of flip parameters

$$\begin{aligned} \hat{p}_1 = 1 \quad \hat{p}_2 \approx 0.296706 \quad \hat{p}_3 \approx 0.166762 \quad \hat{p}_4 \approx 0.101790 \\ \hat{p}_5 \approx 0.058475 \quad \hat{p}_6 = 0.025989 \quad p_j = 0, \forall j \geq 7, \end{aligned} \tag{3}$$

there is a variable-length coupling such that the Hamming distance contracts by a constant factor every  $O(n)$  steps in expectation. One can thus expect that the coupling is  $C$ -amortized convergent with  $C \leq O(n)$ .

We formalize their main coupling result in the following subsection. For the moment, we state two intermediate lemmas, and show how they imply Theorem 5.

► **Lemma 23.** *Assume the input graph  $G = (V, E)$  has maximum degree  $\Delta \leq O(1)$ . Then, the flip dynamics with parameters given in Equation (3) satisfy the following:*

$$\max_{\tau \in \text{supp}(\mu|_{uc})} \left\{ \sum_{\sigma \neq \tau} |P_{\mu, \text{flip}}(\tau \rightarrow \sigma) - P_{\mu|_{uc}, \text{flip}}(\tau \rightarrow \sigma)| \right\} \leq O(1/n).$$

► **Lemma 24.** *Let  $(G, \mathcal{L})$  be a list-coloring instance, where  $\Delta \leq O(1)$  and  $|L(v)| \geq \lambda^* \Delta$  for all  $v \in V$ , where  $\lambda^* = \frac{11}{6} - \epsilon$  and  $\epsilon \approx 10^{-5}$  is a small constant. Then the flip dynamics with parameters given in Equation (3) admits a  $C$ -amortized convergent coupling w.r.t. Hamming distance where  $C \leq O(n)$ .*

**Proof of Theorem 5.** The flip dynamics is clearly  $O(1)$ -local w.r.t. Hamming distance since only Kempe components of size at most 6 can be flipped.  $(\eta_0, \dots, \eta_{n-2})$ -spectral independence where  $\eta_k \leq O(1)$  for all  $k$  then follows immediately by combining Lemma 23 and Lemma 24 with Theorem 3. The lower bounds on the spectral gap, standard and modified log-Sobolev constants then follow from Theorem 15. ◀

We provide the proof of Lemma 23 in Section B. At this point, all that remains is to prove Lemma 24, which we do using the variable-length path coupling constructed in [13].

## 5.2 Variable-Length Path Coupling: Proof of Lemma 24

To begin, we first define the notion of variable-length coupling following [37, 13].

► **Definition 25 (Path-Generating Set).** *For a finite state space  $\Omega$ , a **path generating set** is a subset  $S \subseteq \binom{\Omega}{2}$  such that the undirected graph  $(\Omega, S)$  is connected. We let  $d_S(\cdot, \cdot)$  denote the induced shortest-path metric on  $\Omega$ , and write  $d(\cdot, \cdot)$  when the path generating set  $S$  is clear from context. We also write  $x \sim y$  whenever  $\{x, y\} \in S$ .*

► **Definition 26 (Variable-Length Path Coupling [37]).** *Fix an irreducible transition probability matrix  $P$  which is reversible w.r.t. a distribution  $\pi$  on a finite state space  $\Omega$ , and let  $d(\cdot, \cdot)$  be a metric on  $\Omega$  induced by a path generating set  $S \subseteq \binom{\Omega}{2}$ . For every pair of starting states  $x^{(0)}, y^{(0)} \in \Omega$  with  $x^{(0)} \sim y^{(0)}$ , we let  $(\bar{x}, \bar{y}, T) = (\bar{x}(x^{(0)}, y^{(0)}), \bar{y}(x^{(0)}, y^{(0)}), T(x^{(0)}, y^{(0)}))$  denote a random variable where  $T$  is a (potentially random) nonnegative integer and  $\bar{x} = (x^{(0)}, x^{(1)}, \dots, x^{(T)})$ ,  $\bar{y} = (y^{(0)}, y^{(1)}, \dots, y^{(T)})$  are length- $T$  sequences of states in  $\Omega$ .*

*For every integer  $t \geq 0$  and every pair of neighboring states  $x^{(0)} \sim y^{(0)}$ , define random variables  $x_t, y_t$  by the following experiment. Sample  $(\bar{x}, \bar{y}, T)$ , and set  $x_t = x^{(t)}, y_t = y^{(t)}$  if  $t \leq T$ , and sample  $x_t \sim P^{t-T}(x^{(T)}, \cdot), y_t \sim P^{t-T}(y^{(T)}, \cdot)$  if  $t > T$ . We say the random variable  $(\bar{x}, \bar{y}, T)$  is a **variable-length path coupling for  $P$**  if  $x_t \sim P^t(x^{(0)}, \cdot), y_t \sim P^t(y^{(0)}, \cdot)$  for every integer  $t \geq 0$  and every pair of neighboring states  $x^{(0)} \sim y^{(0)}$ . In this case, we say that  $\bar{x}, \bar{y}$  are individually **faithful copies**. If  $T = t$  with probability 1 for some nonnegative integer  $t \geq 0$ , we say that  $(\bar{x}, \bar{y}, T)$  is a  **$t$ -step path coupling**.*

► **Remark 27.** In our application to colorings, the random time  $T$  will be a stopping time in the sense that its value only depends on the past, i.e.  $x^{(0)}, y^{(0)}, \dots, x^{(t)}, y^{(t)}$  for  $t \leq T$ .

Given a variable-length path coupling, [37] showed one can construct a full coupling, generalizing the original path coupling theorem of [11, 12]. Furthermore, the contraction properties of the full coupling are inherited from the path coupling. While the original statement in [37] merely states rapid mixing given a variable-length path coupling, its proof implies the following.

► **Theorem 28** (Proof of Corollary 4 from [37]). *Let  $(\bar{x}, \bar{y}, T)$  be a variable-length path coupling w.r.t. a path generating set  $S$  for a reversible Markov chain  $P$  on a state space  $\Omega$  with stationary distribution  $\pi$ . Let*

$$\begin{aligned}\alpha &\stackrel{\text{def}}{=} 1 - \max_{\{x^{(0)}, y^{(0)}\} \in S} \mathbb{E}[d_H(x^{(T)}, y^{(T)})] \\ W &\stackrel{\text{def}}{=} \max_{\{x^{(0)}, y^{(0)}\} \in S, t \leq T} d_H(x^{(t)}, y^{(t)}) \\ \beta &\stackrel{\text{def}}{=} \max_{\{x^{(0)}, y^{(0)}\} \in S} \mathbb{E}[T].\end{aligned}$$

Assume  $0 < \alpha < 1$ . Then there is a full  $M$ -step coupling with  $M = \lceil \frac{2\beta W}{\alpha} \rceil$  such that for all pairs  $x^{(0)}, y^{(0)}$ , which need not be neighbors in  $S$ , we have the inequality

$$\mathbb{E}[d_H(x^{(M)}, y^{(M)}) \mid x^{(0)}, y^{(0)}] \leq \left(1 - \frac{\alpha}{2}\right) \cdot d_H(x^{(0)}, y^{(0)}).$$

Given this, all we need now is a good variable-length path coupling. This is given by the following result due to [13].

► **Theorem 29** ([13]). *Let  $(G, \mathcal{L})$  be a list-coloring instance, where  $G = (V, E)$  is a graph with maximum degree  $\Delta \leq O(1)$ , and  $\mathcal{L} = (L(v))_{v \in V}$  is a collection of color lists. Let the path generating set  $S$  be given by the set of pairs  $\{\tau, \sigma\}$  such that  $\tau, \sigma$  differ on the coloring of exactly one vertex. Assume  $|L(v)| \geq \lambda^* \Delta$  for all  $v \in V$  where  $\lambda^* = \frac{11}{6} - \epsilon$  for an absolute constant  $\epsilon \approx 10^{-5}$ . Then there exists a variable-length path coupling  $(\bar{\tau}, \bar{\sigma}, T)$  for the flip dynamics w.r.t.  $S$  with flip parameters given in Equation (3), where  $T$  is the first time such that the Hamming distance changes, such that  $\alpha = \frac{q - \lambda^* \Delta}{q - \Delta - 2} = \Theta(1)$ ,  $W = 13$  and  $\beta \leq \frac{qn}{q - \Delta - 2} \leq O(n)$*

With these tools in hand, we may now finally prove Lemma 24 and complete the proof of Theorem 5.

**Proof of Lemma 24.** First, note that the path generating set  $S$  generates the Hamming metric  $d_H(\cdot, \cdot)$  on proper list-colorings. Now, given the variable-length path coupling furnished by Theorem 29, we use Theorem 28 to construct an  $M$ -step coupling with  $M = \lceil \frac{2\beta W}{\alpha} \rceil \leq O(n)$  which contracts with rate  $1 - \alpha$  every  $M$  steps, where  $\alpha$  is a constant independent of  $n$ . Under this coupling, for every  $k = 0, \dots, M - 1$  and every positive integer  $j$ , we have that

$$\begin{aligned}&\mathbb{E}_{\tau^{(jM+k)}, \sigma^{(jM+k)}} \left[ d_H(\tau^{(jM+k)}, \sigma^{(jM+k)}) \mid \tau^{(k)}, \sigma^{(k)} \right] \\ &\leq \left(1 - \frac{\alpha}{2}\right) \mathbb{E}_{\tau^{((j-1)M+k)}, \sigma^{((j-1)M+k)}} \left[ d_H(\tau^{((j-1)M+k)}, \sigma^{((j-1)M+k)}) \mid \tau^{(k)}, \sigma^{(k)} \right] \\ &\leq \dots \\ &\leq \left(1 - \frac{\alpha}{2}\right)^j \cdot d_H(\tau^{(k)}, \sigma^{(k)}),\end{aligned}$$

where  $\tau^{(0)} = \tau, \sigma^{(0)} = \sigma$  are arbitrary starting states, which need not be neighbors under  $S$ .

It follows that

$$\begin{aligned}
& \sum_{t=0}^{\infty} \mathbb{E}_{\tau^{(t)}, \sigma^{(t)}} \left[ d_H(\tau^{(t)}, \sigma^{(t)}) \mid \tau^{(0)} = \tau, \sigma^{(0)} = \sigma \right] \\
& \leq \sum_{k=0}^{M-1} \sum_{j=0}^{\infty} \mathbb{E}_{\tau^{(jM+k)}, \sigma^{(jM+k)}} \left[ d_H(\tau^{(jM+k)}, \sigma^{(jM+k)}) \mid \tau^{(k)}, \sigma^{(k)} \right] \\
& \leq \sum_{k=0}^{M-1} \mathbb{E} \left[ d_H(\tau^{(k)}, \sigma^{(k)}) \mid \tau^{(0)}, \sigma^{(0)} \right] \sum_{j=0}^{\infty} \left(1 - \frac{\alpha}{2}\right)^j \\
& = \frac{2}{\alpha} \sum_{k=0}^{M-1} \mathbb{E} \left[ d_H(\tau^{(k)}, \sigma^{(k)}) \mid \tau^{(0)}, \sigma^{(0)} \right] \\
& \leq \frac{2M}{\alpha} d_H(\tau^{(0)}, \sigma^{(0)}) \tag{*} \\
& \leq O(n) \cdot d_H(\tau^{(0)}, \sigma^{(0)}).
\end{aligned}$$

To justify (\*), note that  $T$  is the first time the Hamming distance changes, and that each time the Hamming distance changes, the expected Hamming distance contracts by a factor of  $1 - \alpha$ . ◀

## 6 Future Directions

Two concrete open problems are to bring down the required number of colors from  $(\frac{11}{6} - \epsilon) \Delta$  to  $\Delta + 2$ , and to remove the bounded-degree assumption, both in this work and in [16]. Another interesting question is if spectral independence implies any useful notion of correlation decay, such as strong spatial mixing, or the absence of zeros for partition function in a large region. This is relevant particularly for proper list-colorings, where we showed spectral independence when  $q \geq (\frac{11}{6} - \epsilon) \Delta$ , but correlation decay and absence of zeros are both open in general when number of colors is below  $2\Delta$ .

We also reiterate that one feature of our approach which we haven't exploited is that in order to obtain  $O(1)$ -spectral independence, it suffices for the Markov chain admitting the nice coupling to merely update  $O(1)$ -coordinates in a single move “on average”, as opposed to the worst-case starting state; see Remark 10. We leave it to future work to see if this can be exploited.

---

## References

- 1 Vedat Levi Alev and Lap Chi Lau. Improved analysis of higher order random walks and applications. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, page 1198–1211, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3357713.3384317.
- 2 Yeganeh Alimohammadi, Nima Anari, Kirankumar Shiragur, and Thuy-Duong Vuong. Fractionally log-concave and sector-stable polynomials: Counting planar matchings and more. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, page 433–446, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3406325.3451123.
- 3 Nima Anari and Michal Dereziński. Isotropy and log-concave polynomials: Accelerated sampling and high-precision counting of matroid bases. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1331–1344, Los Alamitos, CA, USA, November 2020. IEEE Computer Society.

- 4 Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials ii: High-dimensional walks and an fpras for counting bases of a matroid. In *Proceedings of the 51st Annual ACM SIGACT Symposium on the Theory Computing*, STOC 2019, pages 1–12, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3313276.3316385.
- 5 Nima Anari, Kuikui Liu, Shayan Oveis Gharan, Cynthia Vinzant, and Thuy-Duong Vuong. Log-concave polynomials iv: Approximate exchange, tight mixing times, and near-optimal sampling of forests. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on the Theory Computing*, STOC 2021, New York, NY, USA, 2021. Association for Computing Machinery.
- 6 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.
- 7 Antonio Blanca, Pietro Caputo, Zongchen Chen, Daniel Parisi, Daniel Štefankovič, and Eric Vigoda. On mixing of markov chains: Coupling, spectral independence, and entropy factorization. *arXiv preprint arXiv:2103.07459*, 2021.
- 8 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, page 287–296, New York, NY, USA, 2003. Association for Computing Machinery.
- 9 Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 1 edition, 2016.
- 10 Guy Bresler and Dheeraj Nagaraj. Stein’s method for stationary distributions of markov chains and application to ising models. *Annals of Applied Probability*, 29(5), 2019.
- 11 R. Bubley and M. Dyer. Path coupling: A technique for proving rapid mixing in markov chains. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, FOCS '97, pages 223–, Washington, DC, USA, 1997. IEEE Computer Society. URL: <http://dl.acm.org/citation.cfm?id=795663.796353>.
- 12 R. Bubley and M. E. Dyer. Path coupling, dobrushin uniqueness, and approximate counting. Technical report, University of Leeds, 1997.
- 13 Sitan Chen, Michelle Delcourt, Ankur Moitra, Guillem Perarnau, and Luke Postle. Improved bounds for randomly sampling colorings via linear programming. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, page 2216–2234, USA, 2019. Society for Industrial and Applied Mathematics.
- 14 Z. Chen, K. Liu, and E. 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, 2020.
- 15 Zongchen Chen, Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. *Rapid Mixing for Colorings via Spectral Independence*, page 1548–1557. Society for Industrial and Applied Mathematics, USA, 2021.
- 16 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, page 1537–1550, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3406325.3451035.
- 17 Zongchen Chen, Kuikui Liu, and Eric Vigoda. Spectral independence via stability and applications to holant-type problems. *arXiv preprint arXiv:2106.03366*, 2021.
- 18 M. Cryan, H. Guo, and G. Mousa. Modified log-sobolev inequalities for strongly log-concave distributions. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1358–1370, 2019.
- 19 Persi Diaconis and Laurent Saloff-Coste. Comparison theorems for reversible markov chains. *Annals of Applied Probability*, 3(3):696–730, 1993.
- 20 Persi Diaconis and Laurent Saloff-Coste. Logarithmic sobolev inequalities for finite markov chains. *Annals of Applied Probability*, 6(3), August 1996.

- 21 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.
- 22 R. L. Dobrushin and S. B. Shlosman. *Completely Analytical Gibbs Fields*, pages 371–403. Birkhäuser Boston, Boston, MA, 1985.
- 23 R. L. Dobrushin and S. B. Shlosman. *Constructive Criterion for the Uniqueness of Gibbs Field*, pages 347–370. Birkhäuser Boston, Boston, MA, 1985.
- 24 R. L. Dobrushin and S. B. Shlosman. Completely analytical interactions: Constructive description. *Journal of Statistical Physics*, 46(5):983–1014, 1987.
- 25 Roland Lvovich Dobrushin. Prescribing a system of random variables by conditional distributions. *Theory of Probability and its Applications*, 15(3):458–486, 1970.
- 26 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.
- 27 Charilaos Efthymiou, Andreas Galanis, Thomas P. Hayes, Daniel Stefankovic, and Eric Vigoda. Improved Strong Spatial Mixing for Colorings on Trees. 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 48:1–48:16, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.48.
- 28 Ronen Eldan, James R. Lee, and Joseph Lehec. Transport-entropy inequalities and curvature in discrete-space markov chains. *A Journey Through Discrete Mathematics*, pages 391–406, 2017.
- 29 Tomás Feder and Milena Mihail. Balanced matroids. In *STOC*, pages 26–38, 1992.
- 30 Weiming Feng, Heng Guo, Yitong Yin, and Chihao Zhang. *Rapid Mixing from Spectral Independence beyond the Boolean Domain*, page 1558–1577. Society for Industrial and Applied Mathematics, USA, 2021.
- 31 Alan Frieze and Eric Vigoda. A survey on the use of markov chains to randomly sample colourings. *Combinatorica*, January 2007. doi:10.1093/acprof:oso/9780198571278.003.0004.
- 32 David Gamarnik, Dmitriy Katz, and Sidhant Misra. Strong spatial mixing of list coloring of graphs. *Random Struct. Algorithms*, 46(4):599–613, 2015. doi:10.1002/rsa.20518.
- 33 Sharad Goel. Modified logarithmic sobolev inequalities for some models of random walk. *Stochastic Processes and their Applications*, 114:51–79, 2004.
- 34 Heng Guo and Giorgos Mousa. Local-to-global contraction in simplicial complexes. *arXiv preprint arXiv:2012.14317*, 2020.
- 35 Thomas P. Hayes. A simple condition implying rapid mixing of single-site dynamics on spin systems. *FOCS*, pages 39–46, 2006.
- 36 Thomas P. Hayes and Alistair Sinclair. A general lower bound for mixing of single-site dynamics on graphs. *Annals of Applied Probability*, 17(3):931–952, 2007.
- 37 Thomas P. Hayes and Eric Vigoda. Variable length path coupling. *Random Structures and Algorithms*, 31:251–272, 2007.
- 38 Mark Jerrum. A very simple algorithm for estimating the number of k-colorings of a low-degree graph. *Random Struct. Algorithms*, 7(2):157–165, 1995.
- 39 Tali Kaufman and David Mass. High dimensional random walks and colorful expansion. In *ITCS*, pages 4:1–4:27, 2017.
- 40 Tali Kaufman and Izhar Oppenheim. High order random walks: Beyond spectral gap. In *APPROX/RANDOM*, pages 47:1–47:17, 2018.
- 41 David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, 2 edition, 2017.
- 42 Katalin Marton. Logarithmic sobolev inequalities in discrete product spaces. *Combinatorics, Probability and Computing*, 28(6):919–935, 2019. doi:10.1017/S0963548319000099.

- 43 M. Mihail and U. Vazirani. On the expansion of 0/1 polytopes. *Journal of Combinatorial Theory*, 1989.
- 44 Yann Ollivier. Ricci curvature of markov chains on metric spaces. *Journal of Functional Analysis*, 256:810–864, February 2009.
- 45 Izhar Oppenheim. Local spectral expansion approach to high dimensional expanders part i: Descent of spectral gaps. *Discrete and Computational Geometry*, 59(2):293–330, 2018.
- 46 Gesine Reinert and Nathan Ross. Approximating stationary distributions of fast mixing glauber dynamics, with applications to exponential random graphs. *Annals of Applied Probability*, 29, 2019.
- 47 Marcus D. Sammer. *Aspects of Mass Transportation in Discrete Concentration Inequalities*. PhD thesis, Georgia Institute of Technology, April 2005.
- 48 Charles Stein. A bound for the error in the normal approximation to the distribution of a sum of dependent random variables. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability*, pages 583–602, 1972.
- 49 Eric Vigoda. Improved bounds for sampling colorings. *Journal of Mathematical Physics*, 41, 2000.
- 50 Jian-Sheng Wang, Robert H. Swendsen, and Roman Kotecký. Antiferromagnetic potts models. *Phys. Rev. Lett.*, 63:109–112, July 1989. doi:10.1103/PhysRevLett.63.109.

## A Variance and Entropy Decay

While we primarily use prior results on the spectral gap, and standard and modified log-Sobolev constants as blackboxes, to keep this paper self-contained, we define these constants here, and state their relevance to mixing and concentration. Fix a Markov kernel  $P$  on a finite state space  $\Omega$  which is reversible w.r.t. a distribution  $\pi$ . We may define an inner product using  $\pi$  by  $\langle f, g \rangle_\pi = \mathbb{E}_\pi[fg]$ . This inner product together with the kernel  $P$  induces a positive semidefinite quadratic form known as the Dirichlet form, defined as  $\mathcal{E}_P(f, g) = \langle f, (I - P)g \rangle_\pi$ . The variance of a function  $f : \Omega \rightarrow \mathbb{R}$  is given by  $\text{Var}_\pi(f) = \mathbb{E}_\pi(f^2) - \mathbb{E}_\pi(f)^2$ , while the entropy of a function  $f : \Omega \rightarrow \mathbb{R}_{\geq 0}$  is given by  $\text{Ent}_\pi(f) = \mathbb{E}_\pi(f \log f) - \mathbb{E}_\pi(f) \log \mathbb{E}_\pi(f)$ .

With these notions in hand, we may now define the following constants:

$$\begin{aligned}
 \text{(Spectral Gap)} \quad \lambda(P) &\stackrel{\text{def}}{=} \inf_{f \neq 0} \frac{\mathcal{E}(f, f)}{\text{Var}_\pi(f)} \\
 \text{(Modified Log-Sobolev Constant)} \quad \rho(P) &\stackrel{\text{def}}{=} \inf_{f \geq 0} \frac{\mathcal{E}(f, \log f)}{\text{Ent}_\pi(f)} \\
 \text{(Standard Log-Sobolev Constant)} \quad \kappa(P) &\stackrel{\text{def}}{=} \inf_{f \geq 0} \frac{\mathcal{E}(\sqrt{f}, \sqrt{f})}{\text{Ent}_\pi(f)}. \tag{4}
 \end{aligned}$$

It is known that  $4\kappa(P) \leq \rho(P) \leq 2\lambda(P)$  [8], with lower bounds on  $\kappa(P)$  being the most difficult to establish. For the reader's convenience, we collect some well-known relations between these constants, mixing, and concentration.

► **Proposition 30 (Mixing and Concentration).** *We have the following bounds on the mixing time of a Markov chain with transition probability matrix  $P$  and stationary distribution  $\pi$ .*

$$t_{\text{mix}}(\epsilon) \leq \frac{1}{\lambda(P)} \left( \frac{1}{2} \log \frac{1}{\pi_{\min}} + \log \frac{1}{2\epsilon} \right) \tag{41}$$

$$t_{\text{mix}}(\epsilon) \leq \frac{1}{\rho(P)} \left( \log \log \frac{1}{\pi_{\min}} + \log \frac{1}{2\epsilon^2} \right) \tag{8}$$

$$t_{\text{mix}}(\epsilon) \leq \frac{1}{4\kappa(P)} \left( \log \log \frac{1}{\pi_{\min}} + \log \frac{1}{2\epsilon^2} \right). \tag{20}$$

## 32:20 Coupling to Spectral Independence

Furthermore, for every function  $f : \Omega \rightarrow \mathbb{R}$  which is 1-Lipschitz w.r.t. graph distance under  $P$ , we have the following Chernoff-type concentration inequalities [33, 47, 9].

$$\Pr[f \geq \mathbb{E}_\pi(f) + \epsilon] \leq \exp\left(-\frac{\rho(P)\epsilon^2}{2v(f)}\right)$$

$$\Pr[f \leq \mathbb{E}_\pi(f) - \epsilon] \leq \exp\left(-\frac{\kappa(P)\epsilon^2}{2v(f)}\right),$$

where

$$v(f) \stackrel{\text{def}}{=} \max_{x \in \Omega} \left\{ \sum_{y \in \Omega} P(x \rightarrow y) \cdot (f(x) - f(y))^2 \right\}.$$

### B Missing Proofs

**Proof of Theorem 19.** We show that  $\eta_0 \leq \frac{4}{\alpha n} - 1$ . The bound  $\eta_k \leq \frac{4}{\alpha n} - 1$  follows by the same argument by instead considering the Glauber dynamics for the conditional distributions  $\mu \mid A$  of  $\mu$ . Because the Glauber dynamics only updates at most one coordinate in each step, it is 2-local w.r.t.  $d_H(\cdot, \cdot)$ . By Fact 18, we also have there is a  $C$ -amortized convergent coupling with  $C = \frac{1}{\alpha}$ . It follows from Theorem 7 that

$$\sum_{v \in V} \sum_{c' \in \Omega(v)} \left| \Pr_{\sigma \sim \mu} [\sigma(v) = c' \mid \sigma(u) = c] - \Pr_{\sigma \sim \mu} [\sigma(v) = c'] \right|$$

$$\leq \frac{2}{\alpha} \max_{\sigma \in \text{supp}(\mu \mid uc)} \sum_{\tau \neq \sigma} |P_\mu(\sigma \rightarrow \tau) - P_{\mu \mid uc}(\sigma \rightarrow \tau)|.$$

Now, by the definition of the Glauber dynamics, for each  $\sigma \in \text{supp}(\mu \mid uc)$ , we have

$$\sum_{\tau \neq \sigma} |P_\mu(\sigma \rightarrow \tau) - P_{\mu \mid uc}(\sigma \rightarrow \tau)|$$

$$= \sum_{v \in V} \sum_{c' \in L(v): c' \neq \sigma(v)} \left| \frac{1}{n} \mu^v(c' \mid \sigma_{-v}) - \frac{1}{n-1} \mu_{uc}^v(c' \mid \sigma_{-v}) \right|$$

$$= \sum_{v \in V: v \neq u} \sum_{c' \in L(v): c' \neq \sigma(v)} \left( \frac{1}{n-1} - \frac{1}{n} \right) \mu^v(c' \mid \sigma_{-v}) + \sum_{c' \in L(u): c' \neq c} \frac{1}{n} \mu^v(c' \mid \sigma_{-v})$$

$$\leq \frac{2}{n}.$$

The claim for the spectral gap in the case  $\alpha \geq \Omega(1/n)$  follows by combining with Theorem 14. The final claim for spin systems on bounded-degree graphs follows by combining with Theorem 15.  $\blacktriangleleft$

**Proof of Lemma 23.** The main detail one must be careful of is that the flip dynamics for sampling from  $\mu \mid uc$  always leaves the color for  $u$  fixed to  $c$ . Hence, flipping any Kempe component containing  $u$  leads to potentially different list-colorings under  $P_{\mu, \text{flip}}$  versus  $P_{\mu \mid uc, \text{flip}}$ . However, since we only flip components of  $O(1)$ -size, this isn't an issue for us.

Fix a  $\tau$  with  $\tau(u) = c$ , and let  $B(u, 6)$  denote the set of vertices of shortest path distance at most 6 away from  $u$  in  $G$ . Since we only flip Kempe components of size at most 6, we have that for any  $v \in V \setminus B(u, 6)$  and  $c \in L(u)$ , the flippable Kempe component  $S_\tau(v, c')$



does not contain  $u$ , and hence, flipping it leads to the same list-coloring under  $P_{\mu, \text{flip}}$  and  $P_{\mu|vc, \text{flip}}$ . Hence, we have



$$\begin{aligned}
& \sum_{\sigma \neq \tau} |P_{\mu, \text{flip}}(\tau \rightarrow \sigma) - P_{\mu|uc, \text{flip}}(\tau \rightarrow \sigma)| \\
&= \sum_{v \in V: v \notin B(u, 6)} \sum_{c' \in L(v)} \frac{1}{|L(v)|} \cdot \left( \frac{1}{n} - \frac{1}{n-1} \right) \cdot p_{|S_\tau(v, c')|} \\
&\quad + \sum_{v \in B(u, 6)} \sum_{c' \in [q]} |P_{\mu, \text{flip}}(\tau \rightarrow \sigma) - P_{\mu|uc, \text{flip}}(\tau \rightarrow \sigma)| \\
&\leq \frac{n - |B(u, 6)|}{n(n-1)} + \frac{|B(u, 6)|}{n} \\
&\leq \frac{|B(u, 6)| + 1}{n} \\
&\lesssim \frac{\Delta^6}{n} \\
&\leq O(1/n). \qquad \qquad \qquad \text{(Bounded-degree assumption)}
\end{aligned}$$

◀

► Remark 31. As one can see in the proof from the factor of  $\Delta^6$ , we have made no attempt to optimize constants.



# Singularity of Random Integer Matrices with Large Entries

Sankeerth Rao Karingula  

Department of Computer Science, University of California San Diego, CA, USA

Shachar Lovett  

Department of Computer Science, University of California San Diego, CA, USA

---

## Abstract

We study the singularity probability of random integer matrices. Concretely, the probability that a random  $n \times n$  matrix, with integer entries chosen uniformly from  $\{-m, \dots, m\}$ , is singular. This problem has been well studied in two regimes: large  $n$  and constant  $m$ ; or large  $m$  and constant  $n$ . In this paper, we extend previous techniques to handle the regime where both  $n, m$  are large. We show that the probability that such a matrix is singular is  $m^{-cn}$  for some absolute constant  $c > 0$ . We also provide some connections of our result to coding theory.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Randomness, geometry and discrete structures

**Keywords and phrases** Coding Theory, Random matrix theory, Singularity probability MDS codes, Error correction codes, Littlewood Offord, Fourier Analysis

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.33

**Category** RANDOM

**Funding** *Sankeerth Rao Karingula*: Research supported by NSF CCF award 1614023.

*Shachar Lovett*: Research supported by NSF CCF award 1614023.

**Acknowledgements** We would like to thank Roman Vershynin and Konstantin Tikhomirov for helpful discussions.

## 1 Introduction

In this paper we study the probability that a random  $n \times n$  matrix with uniform integer entries in  $\{-m, \dots, m\}$  is singular. Note that the probability that such a matrix is singular is at least  $(2m + 1)^{-n}$ , which is the probability that its first two rows are the same. We show that this bound is tight, up to polynomial factors.

► **Theorem 1** (Singularity of random matrices). *Let  $n, m \geq 1$ . Let  $M$  be an  $n \times n$  random integer matrix with entries chosen uniformly in  $\{-m, \dots, m\}$ . Then for some absolute constant  $c > 0$ ,*

$$\Pr[M \text{ is singular}] \leq m^{-cn}.$$

Note that if instead we chose  $M$  to be a random  $n \times n$  matrix over a finite field  $\mathbb{F}_q$ , then the probability that  $M$  is singular would be about  $1/q$ , independent of how large  $n$  is. This is the main point of difference between random matrices over integers and over finite fields - the singularity probability over integers decreases as the matrix becomes larger, whereas over finite fields it stabilizes.



© Sankeerth Rao Karingula and Shachar Lovett;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 33; pp. 33:1–33:16



Leibniz International Proceedings in Informatics

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

## 1.1 Connections to coding theory - alphabet size for MDS codes

Our motivation for proving Theorem 1 is a connection to coding theory. More specifically, the question of what alphabet size is needed for Maximum Distance Separable (MDS) codes over the integers.

Coding theory is the study of error correction codes. Codes are widely used in many applications, such as data compression, cryptography, error detection and correction, data transmission and data storage. Algorithms needed to implement codes perform arithmetic operations over an underlying alphabet, and hence their computational complexity is constrained by this alphabet size. Thus, understanding the alphabet size needed to support a given code structure is a central question in coding theory. By far, the most common approach to design codes is to use linear codes over finite fields. Our results in this paper help with investigating the possibility of designing codes over integers. In particular, we study the alphabet size needed to support basic code structures, and focus on the most basic and well-studied family of codes - Maximum Distance Separable (MDS) codes.

An MDS code is a code with the best possible tradeoff between the message length, codeword length and minimal distance. Concretely, an  $(n, k, d)$ -code is a code with message length  $k$ , codeword length  $n$  and minimal distance  $d$ . The Singleton bound [16] gives that  $d \leq n - k + 1$ . MDS codes are codes achieving this bound, namely  $(n, k, d)$ -codes with  $d = n - k + 1$ . If we consider linear codes, then it is well-known that MDS codes are generated by the row span of *MDS matrices*.

► **Definition 2** (MDS matrix). *Let  $n \geq k$ . A  $k \times n$  matrix is called an MDS matrix if any  $k$  columns in it are linearly independent. Equivalently, if any  $k \times k$  minor of it is nonsingular.*

Note that MDS matrices can be defined over finite fields or over the integers. If we define them over a finite field  $\mathbb{F}_q$ , then it is well-known that a linear field size is needed to support MDS matrices. Concretely, if we assume  $n \geq k + 2$ , then it is known that  $q \geq \max(k, n - k + 1)$  (see for example the introduction of [1] for a proof). In particular, this implies that  $q \geq n/2$ . Reed-Solomon codes can be constructed over fields of size  $q \geq n - 1$ , which is tight up to a factor of two. The MDS conjecture of Segre [15] speculates that this is indeed the best possible (except for a few special cases), and Ball [1] proved this over prime finite fields. In summary, over finite fields a linear field size  $q = \Theta(n)$  is both necessary and sufficient.

We show that over the integers, MDS matrices exist over much smaller alphabet sizes.

► **Theorem 3** (MDS matrices over integers). *Let  $n \geq k$ . There exist  $k \times n$  MDS matrices over integers whose entries are in  $\{-m, \dots, m\}$ , where  $m \leq (cn/k)^c$  for some absolute constant  $c > 0$ .*

The typical regime in coding theory is that of linear rate and linear distance; namely, where  $k = \alpha n$  for some constant  $\alpha \in (0, 1)$ . Note that in this regime Theorem 3 shows that MDS codes over the integers exist with a *constant* alphabet size, which is in stark contrast with the case over finite fields. It is easy to see that Theorem 3 is best possible, up to the unspecified constant  $c$ .

Theorem 3 follows directly from Theorem 1.

**Proof of Theorem 3.** Let  $M$  be a random  $k \times n$  matrix with entries chosen uniformly from  $\{-m, \dots, m\}$ . The number of  $k \times k$  minors for  $M$  is  $\binom{n}{k}$ , and the probability that each one is singular is at most  $m^{-ck}$  by Theorem 1. Thus

$$\Pr[M \text{ is not MDS}] \leq \binom{n}{k} m^{-ck} \leq \left(\frac{en}{k}\right)^k m^{-ck} = \left(\frac{en}{km^c}\right)^k.$$

In particular, this probability is at most  $2^{-k}$  (say) whenever  $m \geq (2en/k)^{1/c}$ . ◀

The following claim shows that the bound in Theorem 3 is best possible, up to the value of the unspecified constant  $c$ .

▷ **Claim 4.** Let  $n \geq k \geq 2$ . Let  $M$  be a  $k \times n$  MDS matrix whose entries are in an alphabet  $\Sigma$ . Then  $|\Sigma| \geq \sqrt{n/k}$ .

*Proof.* Let  $P_i = (M_{1,i}, M_{2,i}) \in \Sigma^2$  denote the first two elements in the  $i$ -th column of  $M$ . If  $n > |\Sigma|^2 k$ , then there must be  $k$  distinct columns  $i_1, \dots, i_k \in [n]$  such that  $P_{i_1} = \dots = P_{i_k}$ . But then  $M$  cannot be an MDS matrix, as the  $k \times k$  minor formed by taking these columns has the first two rows being a scalar multiple of each other, and hence cannot be nonsingular. ◁

Note that the proof of Theorem 3 is by choosing the matrix  $M$  randomly, and showing that with high probability it will be an MDS matrix. This is another aspect in which codes over integers seem to be different from codes over finite fields. Constructing MDS matrices over finite fields seems to require algebraic constructions (such as Reed-Solomon codes), unless the field size is exponential in  $n$ ; whereas over the integers, random matrices work well even for very small entries.

## 1.2 Related works on the singularity of random matrices

Most previous works in random matrix theory focused on random matrices whose entries are sampled independently from distributions with bounded tail behaviour. The most studied case is that of random sign matrices, namely with uniform  $\{-1, 1\}$  entries. Komlós [7] proved that the probability that a random  $n \times n$  sign matrix is singular is  $o(1)$  as  $n \rightarrow \infty$ , which already is a nontrivial result. It took nearly 30 years until Kahn, Komlós and Szemerédi [5] improved the bound to  $c^n$  for some constant  $c \in (0, 1)$ . A sequence of works [2, 17, 18] improved the value of the constant  $c$ , and recently Tikhomirov [19] proved that  $c = 1/2 + o(1)$ , which is best possible, as the probability that the first two rows of the matrix are equal is  $2^{-n}$ . For more general distributions, Rudelson and Vershynin [12, 13] proved that if the entries of an  $n \times n$  matrix are sampled from a sub-Gaussian distribution, then the probability it is singular is at most  $c^n$  for some  $c \in (0, 1)$ .

The other regime, of large  $m$  and constant  $n$ , was less explored. The only work we are aware of is by Katznelson [6] which gave a bound of the form  $c_n m^{-n}$  for some constant  $c_n$  depending on  $n$ . While having optimal dependence on  $m$  for constant  $n$ , it has a caveat - it only applies in the regime where  $m$  is much larger than  $n$  (more precisely, for every fixed  $n$ , in the limit of large  $m$ ).

A recent work that did address the regime of both large  $n$  and large  $m$ , but for a different entry distribution, is that of Vempala, Wang and Woodruff [20]. Fix  $\mu \in (0, 1)$ , and let  $\mathcal{D}_\mu$  be a distribution over  $\{-1, 0, 1\}$  with  $\mathcal{D}_\mu(0) = 1 - \mu$ ,  $\mathcal{D}_\mu(1) = \mathcal{D}_\mu(-1) = \mu/2$ . Let  $M$  be a random  $n \times n$  matrix, whose entries are the sum of  $m$  independent copies of  $\mathcal{D}_\mu$ . They show that the probability that  $M$  is singular is at most  $m^{-cn}$  for some constant  $c = c(\mu) > 0$ .

With respect to the connection to coding theory, we note that this result is sufficient to prove Theorem 3. However, we view the entry distribution in Theorem 1 (uniform in  $\{-m, \dots, m\}$ ) as more natural for coding applications. In fact, we conjecture that any entry distribution that doesn't give too much probability to any specific element would do, see Conjecture 5 for the details.

### 1.3 Proof techniques

We prove Theorem 1 following the approach of Rudelson and Vershynin [12, 13], in particular following the lecture notes of Rudelson [11] modified appropriately to handle the case of large  $m$ . On the other hand, Vempala et al. [20] follow the approach of Kahn et al. [5] and Tao and Vu [18]. We briefly discuss the difference between the two approaches below.

At a high level, both approaches aim to study “approximate periodicity” of random vectors. However, they take different routes. The first approach is more direct, using the notion of Lowest Common Denominator (LCD) to define and study approximate periodicity. The second approach is indirect, using Fourier analysis. Fourier analytic techniques seem useful when the underlying entry distribution has well-behaved Fourier tails; for example, this is the case for the distribution considered in [20], where the entries are a sum of  $m$  independent copies of a distribution over  $\{-1, 0, 1\}$ . However, the distribution we consider in this paper, uniform in  $\{-m, \dots, m\}$ , has less well-behaved Fourier tails, and Fourier analytic techniques seem less suited to analyze it.

### 1.4 Directions for further research and applications

#### Singularity of matrices over general distributions

As we discussed above, most works on the singularity of random matrices give a bound on the singularity of  $c^n$  for some absolute constant  $c \in (0, 1)$ . Theorem 1 shows that if the entries are uniformly sampled from  $\{-m, \dots, m\}$ , we can take  $c = 1/\text{poly}(m)$ . We speculate that this is an instance of a much more general phenomena - the singularity probability is determined by the anti-concentration of the underlying entries distribution. Given a distribution  $\mathcal{D}$  over  $\mathbb{R}$ , define its max-probability as  $\|\mathcal{D}\|_\infty = \max_x \mathcal{D}(x)$ . For example, if  $\mathcal{D}$  is the uniform distribution over  $\{-m, \dots, m\}$ , then  $\|\mathcal{D}\|_\infty = 1/(2m + 1)$ .

► **Conjecture 5.** *Let  $\mathcal{D}$  be a distribution over  $\mathbb{R}$  and set  $p = \|\mathcal{D}\|_\infty$ . Let  $M$  be a random  $n \times n$  matrix with independent entries from  $\mathcal{D}$ . Then for some absolute constant  $c > 0$ ,*

$$\Pr[M \text{ is singular}] \leq p^{cn}.$$

One can even speculate a more general conjecture, where each entry comes from a different underlying distribution, as long as they all have bounded max-probability.

#### Applications to coding theory

We view Theorem 3 as a first step towards the study of codes over integers. There are many intriguing questions that arise in coding theory, once we showed that random integer matrices are MDS with high probability.

- **Explicit constructions.** A natural question is to give an explicit construction of MDS matrices over integers with small integer values. Concretely, when  $k = \alpha n$  for some constant  $\alpha \in (0, 1)$ , to give an explicit construction of a  $k \times n$  MDS matrix with a constant alphabet size (namely, independent of  $n$ ).
- **Algorithms.** The next natural question, once there are explicit constructions, is to design efficient decoding algorithms for such codes. In particular, it would be intriguing to see if the smaller alphabet size can be utilized to obtain improved runtime (even by logarithmic factors).

## Paper Outline

We prove Theorem 1 in the remainder of the paper. We start with a high-level overview of our framework in Section 2. We compute some preliminary estimates in Section 3, define and study incompressible vectors in Section 4, define the LCD condition in Section 5, where we also prove some properties of it, and bound the LCD of random vectors in Section 6. We put all the ingredients together and complete the proof in Section 7.

## 2 General approach

We will follow the general approach of Rudelson [11] with several modifications needed to obtain effective bounds for large  $m$ .

### Notation

It will be convenient to scale the entries to be in  $[-1, 1]$ ; we denote by  $\mathcal{D}$  the uniform distribution over  $\{a/m : a \in \{-m, \dots, m\}\}$ . We denote by  $\mathcal{D}^n$  the distribution over  $n$ -dimensional vectors with independent entries from  $\mathcal{D}$ , and by  $\mathcal{D}^{n \times n}$  the distribution over  $n \times n$  matrices with independent entries from  $\mathcal{D}$ . We denote by  $S^{n-1}$  the unit sphere in  $\mathbb{R}^n$ , namely  $S^{n-1} = \{x \in \mathbb{R}^n : \|x\|_2 = 1\}$ . We will use the  $c, c', c_0$ , etc, to denote unspecified positive constants. Note that the same letter (e.g.  $c$ ) can mean different unspecified constants in different lemmas.

### We may assume that $n, m$ are large enough

We will assume throughout the proof that  $n, m$  are large enough; concretely, for any absolute constants  $n_0, m_0$ , we may assume that  $n \geq n_0, m \geq m_0$ , and this would only effect the value of the constant  $c$  in Theorem 1.

To see why, consider first the regime of constant  $m$  and large  $n$ . The distribution  $\mathcal{D}$  is symmetric and bounded in  $[-1, 1]$ . The results of [12] show that in such a case,

$$\Pr[M \text{ is singular}] \leq c^n$$

for some absolute constant  $c \in (0, 1)$ . This proves Theorem 1 for any constant  $m$ .

The other regime is that of constant  $n$  and large  $m$ . While we may appeal to the result of Katznelson [6] in this regime, which gives a sharp bound of  $c_n m^{-n}$ , there is a much simpler argument that gives a bound of the order of  $1/m$  which is good enough to establish Theorem 1 in this regime. As the determinant of an  $n \times n$  matrix is a polynomial of degree  $n$ , the Schwartz-Zippel lemma [14, 21] gives

$$\Pr[M \text{ is singular}] = \Pr[\det(M) = 0] \leq \frac{n}{m}.$$

In particular, for constant  $n$  and large  $m$ , this probability is of the order of  $1/m$ , which is consistent with the claimed bound of Theorem 1 (taking  $c < 1/n$ ).

### General approach

Let  $M \sim \mathcal{D}^{n \times n}$ , and let  $X_1, \dots, X_n$  denote its rows. If  $M$  is singular, then one of the rows belongs to the span of the other rows. By symmetry we have

$$\Pr[M \text{ is singular}] \leq n \cdot \Pr[X_n \in \text{Span}(X_1, \dots, X_{n-1})].$$

### 33:6 Singularity of Random Integer Matrices with Large Entries

Let  $X^*$  be any unit vector orthogonal to  $X_1, \dots, X_{n-1}$  (if there are multiple ones, choose one in some deterministic way). We call it a *random normal vector*. We will shorthand  $X = X_n$ . Observe that  $X, X^*$  are independent. Thus we can bound

$$\Pr[M \text{ is singular}] \leq n \cdot \Pr[\langle X^*, X \rangle = 0].$$

To do so, we will show that unless  $X^*$  belongs to a set of “bad” vectors, then the above probability is at most  $m^{-cn}$ , and that the probability that  $X^*$  is bad is also at most  $m^{-cn}$ .

### 3 Preliminary estimates

We establish some preliminary estimates in this section, which will be needed later in the proof.

#### Maximal eigenvalues of random matrices

The first ingredient is bounding the spectral norm of  $M$ . In fact, we would need this bound for rectangular matrices. Given an  $n \times k$  matrix  $R$  we denote its spectral norm as  $\|R\| = \max\{\|Rx\|_2 : x \in S^{k-1}\}$ . Note that  $\|R\| = \|R^T\|$  since  $\|R\| = \max_{x \in S^{k-1}, y \in S^{n-1}} y^T R x$ .

The following claim is a special case of [11, proposition 4.4], who showed that it holds for any symmetric distribution  $\mathcal{D}$  supported in  $[-1, 1]$ .

▷ **Claim 6.** Let  $R \sim \mathcal{D}^{n \times k}$  for  $n \geq k$ . Then for any  $\lambda > 0$ ,

$$\Pr[\|R\| \geq \lambda\sqrt{n}] \leq 2^{-c\lambda^2 k}.$$

#### Anti-concentration of projections

Next, we need anti-concentration results for projections of  $\mathcal{D}^n$ . To begin with we consider projections of the uniform distribution over the solid cube  $[-1, 1]^n$ .

▷ **Claim 7.** Let  $U \sim [-1, 1]^n$  be uniformly distributed. Then for every  $x \in S^{n-1}$  and  $\varepsilon > 0$ ,

$$\Pr_u [|\langle U, x \rangle| \leq \varepsilon] \leq c\varepsilon.$$

*Proof.* The uniform distribution  $U \sim [-1, 1]^n$  is a log-concave distribution. Let  $S = \langle U, x \rangle$  and note that  $S$  is a projection of  $U$  along the direction  $x$ . The Prékopa–Leindler inequality [8, 10] states that projections of log-concave distributions are log-concave, and so  $S$  is a log-concave distribution. Carbery and Wright [3, Theorem 8] show that the required anti-concentration bound holds for any log-concave distribution. ◁

We extend this anti-concentration to the discrete case using a coupling argument. Here and throughout, we denote by  $\log(\cdot)$  logarithm in base 2.

▷ **Claim 8.** Let  $X \sim \mathcal{D}^n$  and set  $\varepsilon_0 = \frac{\sqrt{\log m}}{m}$ . Then for every  $x \in S^{n-1}$  and  $\varepsilon \geq \varepsilon_0$ ,

$$\Pr[|\langle X, x \rangle| \leq \varepsilon] \leq c\varepsilon.$$

*Proof.* We apply a coupling argument between the uniform distribution in  $[-1, 1]^n$  and  $\mathcal{D}^n$ . Sample  $X \sim \mathcal{D}^n$ ,  $Y \sim [-1, 1]^n$  and set  $Z = X + Y/2m$ . Observe that  $Z$  is uniform in the solid cube  $[-1 - 1/2m, 1 + 1/2m]^n$ . Next, fix  $\varepsilon > 0$  and observe that  $\langle X, x \rangle = \langle Z, x \rangle - \langle Y, x \rangle/2m$ . Thus we can bound

$$\Pr[|\langle X, x \rangle| \leq \varepsilon] \leq \Pr[|\langle Z, x \rangle| \leq 2\varepsilon] + \Pr[|\langle Y, x \rangle| \geq 2\varepsilon m].$$



For the first term, Claim 7 bounds its probability by  $c_1\varepsilon$ . For the second term, the Chernoff bound bounds its probability for  $\varepsilon \geq \varepsilon_0$  by  $1/m$ . As we have  $1/m \leq \varepsilon$ , the claim follows.  $\triangleleft$

### Tensorization lemma

We also need the following “tensorization lemma” [11, Lemma 6.5].

$\triangleright$  **Claim 9.** Let  $Y_1, \dots, Y_n$  be independent real-valued random variables. Assume for some  $K, \varepsilon_0 > 0$  that

$$\Pr[|Y_i| \leq \varepsilon] \leq K\varepsilon \quad \text{for all } \varepsilon \geq \varepsilon_0.$$

Then

$$\Pr\left[\sum_{i=1}^n Y_i^2 \leq \varepsilon^2 n\right] \leq (cK\varepsilon)^n \quad \text{for all } \varepsilon \geq \varepsilon_0.$$

### Nets

A set of unit vectors  $\mathcal{N} \subset S^{n-1}$  is called an  $\varepsilon$ -net, for  $\varepsilon > 0$ , if it satisfies:

$$\forall x \in S^{n-1} \exists y \in \mathcal{N} \|x - y\|_2 \leq \varepsilon.$$

The following claim bounds the size of such a net. For a proof see for example [9, Lemma 2.6].

$\triangleright$  **Claim 10.** For any  $\varepsilon > 0$ , there exists a  $\varepsilon$ -net  $\mathcal{N} \subset S^{n-1}$  of size  $|\mathcal{N}| \leq (3/\varepsilon)^n$ .

### Integer points in ball

We need a bound on the number of integer vectors in a ball of a given radius. Let  $B_n(r) = \{x \in \mathbb{R}^n : \|x\|_2 \leq r\}$  denote the ball of radius  $r$  in  $\mathbb{R}^n$ . The following bound is well known.

$\triangleright$  **Claim 11.** The number of integer vectors in  $B_n(r)$  is at most  $\left(1 + \frac{cr}{\sqrt{n}}\right)^n$ .

## 4 Compressible vectors

The first set of “bad” vectors that we want to rule out are vectors which are close to sparse. A vector  $u \in \mathbb{R}^n$  is  $k$ -sparse if it has at most  $k$  nonzero coordinates.

$\blacktriangleright$  **Definition 12** (Compressible vectors). Let  $\alpha, \beta \in (0, 1)$ . A unit vector  $x \in S^{n-1}$  is called  $(\alpha, \beta)$ -compressible if it can be expressed as  $x = u + v$ , where  $u$  is  $(\alpha n)$ -sparse and  $\|v\|_2 \leq \beta$ . Otherwise, we say that  $x$  is  $(\alpha, \beta)$ -incompressible.

We will later choose  $\alpha, \beta$ , but we note here that  $\alpha$  will be a small enough absolute constant and  $\beta$  a small polynomial in  $1/m$ . Concrete values that work are  $\alpha = 1/50, \beta = 1/\sqrt{m}$ . We will implicitly assume that both  $n, m$  are large enough; concretely, at various places we assume that  $\alpha n \geq 2$ .

The main lemma we prove in this section is the following.

$\blacktriangleright$  **Lemma 13.** Let  $\alpha \in (0, 1/8), \beta \in (\varepsilon_0, 1/2)$  where  $\varepsilon_0 = \frac{\sqrt{\log m}}{m}$ . Then

$$\Pr[X^* \text{ is } (\alpha, \beta)\text{-compressible}] \leq (c\beta)^{n/8}.$$

### 33:8 Singularity of Random Integer Matrices with Large Entries

We need a bound on the smallest singular value of a rectangular matrix.

▷ **Claim 14.** Let  $R \sim \mathcal{D}^{n \times k}$  for  $n \geq k$ . Then for every  $x \in S^{k-1}$  and  $\varepsilon \geq \varepsilon_0$ ,

$$\Pr [\|Rx\|_2 \leq \varepsilon\sqrt{n}] \leq (c\varepsilon)^{n/2}.$$

*Proof.* Assume  $\|Rx\|_2 < \varepsilon\sqrt{n}$ . This implies that  $|(Rx)_i| \leq 2\varepsilon$  for at least  $n/2$  coordinates  $i \in [n]$ . Note that for each fixed  $i$ , the value  $(Rx)_i$  is distributed as  $\langle X, x \rangle$  for some  $X \sim \mathcal{D}^k$ . Applying Claim 8 and the union bound over the choice of the  $n/2$  coordinates gives

$$\Pr [\|Rx\|_2 \leq \varepsilon\sqrt{n}] \leq 2^n (c_1\varepsilon)^{n/2} = (c\varepsilon)^{n/2}. \quad \triangleleft$$

▷ **Claim 15.** Let  $R \sim \mathcal{D}^{n \times k}$  for  $n \geq 8k$ . Then for every  $\varepsilon \geq \varepsilon_0$ ,

$$\Pr \left[ \min_{x \in S^{k-1}} \|Rx\|_2 \leq \varepsilon\sqrt{n} \right] \leq (c\varepsilon)^{n/4}.$$

*Proof.* We may assume that  $\varepsilon \leq 1$  by taking  $c \geq 1$ . Let  $\mathcal{N}$  be an  $(\varepsilon^2)$ -net in  $S^{k-1}$  of size  $|\mathcal{N}| \leq (3/\varepsilon^2)^k$ , as given by Claim 10. Let  $E_1$  denote the event that there exists  $y \in \mathcal{N}$  for which  $\|Ry\|_2 \leq 2\varepsilon\sqrt{n}$ . Applying Claim 14 and a union bound gives

$$\Pr [E_1] \leq (3/\varepsilon^2)^k \cdot (c_1\varepsilon)^{n/2} \leq (c_2\varepsilon)^{n/4},$$

where we used the assumption  $n \geq 8k$ . Let  $E_2$  denote the event that  $\|R\| \geq \lambda\sqrt{n}$  for  $\lambda = \sqrt{\log(1/\varepsilon)}$ . Claim 6 shows that  $\Pr[E_2] \leq (c_3\varepsilon)^n$ . We next show that if  $E_1, E_2$  don't hold then the condition of the claim also doesn't hold, namely that  $\|Rx\|_2 > \varepsilon\sqrt{n}$  for all  $x \in S^{k-1}$ .

Let  $x \in S^{k-1}$  be arbitrary and let  $y \in \mathcal{N}$  be such that  $\|x - y\|_2 \leq \varepsilon^2$ . Then

$$\|Rx\|_2 \geq \|Ry\|_2 - \|R\| \cdot \|x - y\|_2 \geq (2\varepsilon - \varepsilon^2\lambda)\sqrt{n}.$$

It can be verified that for  $\varepsilon \leq 1$  we have  $\varepsilon\lambda \leq 1$ , which implies that  $\|Rx\|_2 \geq \varepsilon\sqrt{n}$ . ◁

We will now use these two claims to prove Lemma 13.

**Proof of Lemma 13.** Let  $M'$  be the  $(n-1) \times n$  matrix with rows  $X_1, \dots, X_{n-1}$ . Assume that there exists an  $(\alpha, \beta)$ -compressible vector  $x \in S^{n-1}$  in the kernel of  $M'$ . By definition,  $x = u + v$  where  $u$  is  $(\alpha n)$ -sparse and  $\|v\|_2 \leq \beta$ . In particular,  $M'(u + v) = 0$  and hence  $\|M'u\|_2 = \|M'v\|_2$ . In addition,  $\|u\|_2 \geq \|x\|_2 - \|v\|_2 \geq 1/2$  since  $x$  is a unit vector and  $\|v\|_2 \leq \beta \leq 1/2$ .

Let  $E$  denote the event that  $\|M'\| \geq \lambda\sqrt{n}$  for  $\lambda = c_1\sqrt{\log(1/\beta)}$ , where we choose  $c_1 \geq 1$  large enough so that by Claim 6,  $\Pr[E] \leq \beta^n$ . Note that as we assume  $\beta \leq 1/2$  we have  $\lambda \geq c_1 \geq 1$ . Assuming that  $E$  doesn't hold, we have

$$\|M'u\|_2 = \|M'v\|_2 \leq \|M'\| \cdot \|v\|_2 \leq \lambda\beta\sqrt{n}.$$

In particular,  $y = u/\|u\|_2$  is an  $(\alpha n)$ -sparse unit vector that satisfies  $\|M'y\|_2 \leq 2\lambda\beta\sqrt{n}$ . We next bound the probability that such a vector exists.

Let  $\varepsilon = 2\lambda\beta$ , and note that  $\varepsilon \geq \varepsilon_0$  since  $\beta \geq \varepsilon_0$  and  $\lambda \geq 1$ . There are  $\binom{n}{\alpha n}$  options for the support of  $y$ . Let  $I = \{i : y_i \neq 0\}$  denote a possible support, set  $k = |I|$  and let  $R$  be an  $(n-1) \times k$  matrix with columns  $(Y_i : i \in I)$ . As  $\alpha < 1/8$  we have  $n-1 \geq 8k$ . Thus we can apply Claim 15 and obtain that

$$\Pr [\neg E \quad \wedge \quad \exists y \in S^{k-1}, \|Ry\|_2 \leq \varepsilon\sqrt{n}] \leq (c_2\varepsilon)^{n/4} = \left( c_3\beta\sqrt{\log(1/\beta)} \right)^{n/4}.$$

Note that for  $\beta \leq 1$  we have  $\beta \log(1/\beta) \leq 1$  and hence the above bound is at most  $(c_4\beta)^{n/8}$ .

To conclude, we union bound over the choices for  $I$ , the number of which is  $\binom{n}{\alpha n} \leq 2^n$ . Thus we can bound the total probability by  $2^n (c_4\beta)^{n/8} = (c_5\beta)^{n/8}$ . ◀

## 5 The LCD condition

In this section we will introduce the notion of the Lowest Common Denominator (LCD) of a vector, which is a variant of the LCD definition in [11]. Informally, the LCD of a vector is a robust notion of “almost periodicity”; concretely, it is the least multiple where most of its entries are close to integers.

Given  $x \in \mathbb{R}^n$  let  $x = [x] + \{x\}$  be its decomposition into integer and fractional parts, where  $[x] \in \mathbb{Z}^n$  and  $\{x\} \in [-1/2, 1/2]^n$ .

► **Definition 16** (Least common denominator (LCD)). *Let  $\alpha, \beta \in (0, 1)$ . Given a unit vector  $x \in S^{n-1}$ , its least common denominator (LCD), denoted  $LCD_{\alpha, \beta}(x)$ , is the infimum of  $D > 0$  such that we can decompose  $\{Dx\} = u + v$ , where  $u$  is  $(\alpha n)$ -sparse and  $\|v\|_2 \leq \beta \min(D, \sqrt{n})$ .*

▷ **Claim 17.** Assume  $x \in S^{n-1}$  is  $(5\alpha, \beta)$ -incompressible. Then  $LCD_{\alpha, \beta}(x) > \sqrt{\alpha n}$ .

*Proof.* Let  $D = LCD_{\alpha, \beta}(x)$  and assume towards a contradiction that  $D \leq \sqrt{\alpha n}$ . Let  $y = Dx$ . As  $\|y\|_2^2 \leq \alpha n$  there are at most  $4\alpha n$  coordinates  $i \in [n]$  where  $|y_i| \geq 1/2$ . In all other coordinates  $\{y_i\} = y_i$ , and hence  $y - \{y\}$  is  $(4\alpha n)$ -sparse. By assumption we can decompose  $\{y\} = u + v$  where  $u$  is  $(\alpha n)$ -sparse and  $\|v\|_2 \leq \beta D$ . This implies that we can decompose  $y = u' + v$  where  $u'$  is  $(5\alpha n)$ -sparse. Thus, we can decompose  $x = y/D$  as  $x = u'' + v''$ , where  $u'' = u'/D$  is  $(5\alpha n)$ -sparse and  $v'' = v/D$  satisfies  $\|v''\|_2 \leq \beta$ . This violates the assumption that  $x$  is  $(5\alpha, \beta)$ -incompressible. ◁

Our main goal in this section is to prove the following lemma, which extends Claim 8 assuming  $x$  has large LCD. To get intuition, we note that the lemma below is useful as long as  $\beta \ll \gamma \ll 1$ . We will later set  $\gamma = \sqrt{\beta}$  to be such a choice. In particular, if we set  $\beta = m^{-1/2}$  then we have  $\gamma = m^{-1/4}$ .

► **Lemma 18.** *Let  $X \sim \mathcal{D}^n$ . Let  $\alpha, \beta, \gamma \in (0, 1/2)$ ,  $x \in S^{n-1}$  be  $(\alpha, \gamma)$ -incompressible and set  $D = LCD_{\alpha, \beta}(x)$ . Then for every  $\varepsilon \geq 1/2\pi m D$ , it holds that*

$$\Pr[|\langle X, x \rangle| \leq \varepsilon] \leq c \left( \frac{\varepsilon}{\gamma} + \frac{1}{(\alpha\beta m)^{\alpha n}} \right).$$

The proof of Lemma 18 relies on Esseen’s lemma [4].

► **Lemma 19** (Esseen’s Lemma). *Let  $Y$  be a real-valued random variable. Let  $\phi_Y(t) = \mathbb{E}[e^{itY}]$  denote the characteristic function of  $Y$ . Then for any  $\varepsilon > 0$ , it holds that*

$$\Pr[|Y| \leq \varepsilon] \leq c\varepsilon \int_{-1/\varepsilon}^{1/\varepsilon} |\phi_Y(t)| dt.$$

Before proving Lemma 18, we need some auxiliary claims. Fix some  $x \in S^{n-1}$ , let  $X \sim \mathcal{D}^n$  and let  $Y = \langle X, x \rangle$ . In order to apply Lemma 19, we need to compute the characteristic function of  $Y$ .

▷ **Claim 20.** Let  $X \sim \mathcal{D}^n$ ,  $x \in S^{n-1}$  and set  $Y = \langle X, x \rangle$ . For  $t \in \mathbb{R}$  it holds that

$$|\phi_Y(t)| = \prod_{k=1}^n F\left(\frac{x_k t}{2\pi m}\right)$$

where  $F : \mathbb{R} \rightarrow \mathbb{R}$  is defined as follows:

$$F(y) = \left| \frac{\sin((2m+1)\pi y)}{(2m+1)\sin(\pi y)} \right|.$$

### 33:10 Singularity of Random Integer Matrices with Large Entries

Proof. We have  $Y = \sum x_i \xi_i$  where  $\xi_1, \dots, \xi_n \sim \mathcal{D}$  are independent. Hence

$$\phi_Y(t) = \prod_{k=1}^n \mathbb{E}[e^{ix_k \xi_k t}].$$

Next we compute

$$\mathbb{E}[e^{ix_k \xi_k t}] = \frac{1}{2m+1} \sum_{\ell=-m}^m e^{ix_k(\ell/m)t} = \frac{1}{2m+1} \cdot \frac{\sin(\frac{2m+1}{2m} x_k t)}{\sin(\frac{1}{2m} x_k t)}.$$

Hence

$$|\mathbb{E}[e^{itx_k \xi_k}]| = F\left(\frac{x_k t}{2\pi m}\right). \quad \triangleleft$$

The next claim proves some basic properties of the function  $F$ .

▷ **Claim 21.** The function  $F$  satisfies the following properties:

1.  $F$  is symmetric:  $F(y) = F(-y)$  for all  $y \in \mathbb{R}$ .
2.  $F$  is invariant to shifts by integers:  $F(y) = F(\{y\})$  for  $y \in \mathbb{R}$ .
3.  $F$  is bounded:  $F(y) \in [0, 1]$  for all  $y \in \mathbb{R}$ .
4.  $F(y) \leq G(my)$  for  $y \in [0, 1/2]$ , where  $G : \mathbb{R}_+ \rightarrow [0, 1]$  is defined as follows:

$$G(y) = \begin{cases} e^{-\eta y^2} & \text{if } y \in [0, 1] \\ \frac{e^{-\eta}}{y} & \text{if } y \geq 1 \end{cases}$$

Here,  $\eta > 0$  is an absolute constant. Note that  $G$  is decreasing.

Proof. The first three claims follow immediately from the definition of  $F$  in Claim 20. In order to prove the last claim, we will prove that  $F(y) \leq \frac{c_1}{my}$  for  $y \in [1/m, 1/2]$  for some  $c_1 \in (0, 1)$ ; and that  $F(y) \leq \exp(-c_2(my)^2)$  for  $y \in [0, 1/m]$  for some  $c_2 > 0$ . The claim then follows by taking  $\eta = \min(\ln(1/c_1), c_2)$ .

First, note that  $F(y) \leq \frac{1}{(2m+1)|\sin(\pi y)|}$ . Using Taylor expansion at 0, we get for  $y \in [0, 1/2]$  that

$$\sin(\pi y) \geq \pi y - \frac{\pi^3 y^3}{6} \geq \frac{\pi y}{2}.$$

In particular,  $F(y) \leq \frac{1}{\pi my}$ , which gives the desired bound for  $c_1 = 1/\pi$ .

Next, note that  $F(y) = \frac{1}{2m+1} |\sin((2m+1)\pi y) \cdot \csc(\pi y)|$ . The Laurent series of  $\csc(x)$  at  $x \neq 0$  is  $\csc(x) = \frac{1}{x} + \frac{x}{6} + \frac{7x^3}{360} + \frac{31x^5}{15120} + \Theta(x^7)$  and the Taylor series for  $\sin(x)$  is  $\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \Theta(x^7)$ . So for  $y \in [0, 1/m]$  we have  $F(y) \leq 1 - c_2(my)^2 \leq \exp(-c_2(my)^2)$ . ◁

We also need the following claim, which shows that incompressible vectors retain a large fraction of their norm when restricted to small coordinates. We use the following notation: given  $x \in \mathbb{R}^n$  and a set of coordinates  $J \subset [n]$ , we denote by  $x|_J \in \mathbb{R}^J$  the restriction of  $x$  to coordinates in  $J$ .

▷ **Claim 22.** Let  $x \in S^{n-1}$  be  $(\alpha, \gamma)$ -incompressible. Let  $J = \left\{i : x_i \leq \frac{1}{\sqrt{\alpha n - 1}}\right\}$ . Then

$$\|x|_J\|_2^2 \geq \|x|_J\|_\infty^2 + \gamma^2.$$

Proof. Let  $J^c = [n] \setminus J$ . Since  $x$  is a unit vector, we have  $|J^c| \leq \alpha n - 1$ . Let  $j \in J$  be such that  $|x_j|$  is maximal and take  $K = J \setminus \{j\}$ . Then  $|K^c| \leq \alpha n$ , and since we assume that  $x$  is  $(\alpha, \gamma)$ -incompressible, we have  $\|x|_K\|_2 \geq \gamma$ . This completes the proof, since

$$\|x|_J\|_2^2 - \|x|_J\|_\infty^2 = \|x|_J\|_2^2 - x_j^2 = \|x|_K\|_2^2 \geq \gamma^2. \quad \triangleleft$$

We would need the following lemma in the computations later on.

► **Lemma 23.** *Let  $\gamma, \delta > 0$ . Let  $x \in \mathbb{R}^n$  be a vector such that  $\|x\|_\infty \leq \delta$  and  $\|x\|_2^2 \geq \|x\|_\infty^2 + \gamma^2$ . Let  $T = \pi m / \delta$ . Then*

$$I = \int_0^T \prod_{i=1}^n F\left(\frac{x_i t}{2\pi m}\right) dt \leq \frac{c}{\gamma}.$$

**Proof.** To simplify the proof, we may assume by Claim 21(1) that  $x_i \geq 0$  for all  $i$ . Reorder the coordinates of  $x$  so that  $x_1 \geq x_2 \geq \dots \geq x_n \geq 0$ . Observe that for  $x_i \in [0, T]$  we have  $\frac{x_i t}{2\pi m} \in [0, 1/2]$  and hence we can apply Claim 21(4) and bound each term by  $F\left(\frac{x_i t}{2\pi m}\right) \leq G\left(\frac{x_i t}{2\pi}\right)$ . Thus

$$I \leq \int_0^T \prod_{i=1}^n G\left(\frac{x_i t}{2\pi}\right) dt = 2\pi \int_0^{T/2\pi} \prod_{i=1}^n G(x_i t) dt \leq 2\pi \int_0^\infty \prod_{i=1}^n G(x_i t) dt.$$

We bound this last integral. Let  $t_i = 1/x_i$  so that  $t_1 \leq t_2 \leq \dots \leq t_n$ . For simplicity of notation set  $t_0 = 0, t_{n+1} = \infty$ . We break the computation of the integral to intervals  $[t_k, t_{k+1})$  for  $k = 0, \dots, n$ , and denote by  $I_k$  the integral in each interval:

$$I_k = \int_{t_k}^{t_{k+1}} \prod_{i=1}^n G(x_i t) dt = \int_{t_k}^{t_{k+1}} \prod_{i=1}^k \frac{e^{-\eta}}{x_i t} \cdot \prod_{i=k+1}^n e^{-\eta t^2 x_i^2} dt = e^{-\eta k} \int_{t_k}^{t_{k+1}} \frac{e^{-\eta t^2 \sum_{i=k+1}^n x_i^2}}{t^k \prod_{i=1}^k x_i} dt.$$

Fix  $k$  and consider first the case that  $\sum_{i=k+1}^n x_i^2 \geq \gamma^2/2$ . In this case, using the fact that  $x_i t \geq 1$  for  $i \in [k]$  and  $t \in [t_k, t_{k+1}]$ , we can bound  $I_k$  by

$$I_k \leq e^{-\eta k} \int_{t_k}^{t_{k+1}} e^{-\eta \gamma^2 t^2 / 2} dt \leq e^{-\eta k} \int_0^\infty e^{-\eta \gamma^2 t^2 / 2} dt \leq \frac{c_1 e^{-\eta k}}{\gamma}.$$

Next, consider the case that  $\sum_{i=k+1}^n x_i^2 < \gamma^2/2$ , which means that  $\sum_{i=1}^k x_i^2 > \|x\|_2^2 - \gamma^2/2 \geq \|x\|_\infty^2 + \gamma^2/2$ . Observe that this is impossible for  $k = 0$  or  $k = 1$ , and hence we may assume  $k \geq 2$ . In this case we bound

$$I_k \leq e^{-\eta k} \int_{t_k}^{t_{k+1}} \frac{1}{t^k \prod_{i=1}^k x_i} dt \leq e^{-\eta k} \int_{t_k}^\infty \frac{1}{t^k \prod_{i=1}^k x_i} dt = \frac{e^{-\eta k} x_k^{k-1}}{(k-1) \prod_{i=1}^k x_i} \leq \frac{e^{-\eta k}}{(k-1)x_1}.$$

By our assumption,  $\sum_{i=1}^k x_i^2 \geq \gamma^2/2$  and hence  $x_1^2 \geq \gamma^2/2k$ . Thus we can bound

$$I_k \leq \frac{e^{-\eta k}}{(k-1)\gamma/\sqrt{2k}} \leq \frac{c_2 e^{-\eta k}}{\gamma}.$$

Overall, we bounded the integral by

$$I \leq 2\pi \sum_{k=0}^n I_k \leq 2\pi \max(c_1, c_2) \sum_{k=0}^n \frac{e^{-\eta k}}{\gamma} \leq \frac{c}{\gamma},$$

where we used the fact that  $c_1, c_2, \eta > 0$  are all absolute constants. ◀

### 33:12 Singularity of Random Integer Matrices with Large Entries

Now we have all the ingredients to complete proof of Lemma 18.

**Proof of Lemma 18.** Let  $Y = \langle X, x \rangle$ . Lemma 19 and Claim 20 give the bound

$$\Pr[|Y| \leq \varepsilon] \leq c_1 \varepsilon I,$$

where  $I$  is the following integral:

$$I = \int_0^{1/\varepsilon} \prod_{i=1}^n F\left(\frac{x_i t}{2\pi m}\right) dt.$$

Let  $T = \pi m \sqrt{\alpha n - 1}$ . We will bound the integral in the regime  $[0, T]$  and  $[T, 1/\varepsilon]$ , and denote the corresponding integrals by  $I_1, I_2$ .

Consider first the integral  $I_1$  in  $[0, T]$ . Let  $\delta = 1/\sqrt{\alpha n - 1}$  and take  $J = \{i : x_i \leq \delta\}$ . Observe that by Claim 21(3), we can bound  $F\left(\frac{x_i t}{2\pi m}\right) \leq 1$  for  $i \notin J$ . Thus

$$I_1 = \int_0^T \prod_{i=1}^n F\left(\frac{x_i t}{2\pi m}\right) dt \leq \int_0^T \prod_{i \in J} F\left(\frac{x_i t}{2\pi m}\right) dt.$$

Next, as we assume that  $x$  is  $(\alpha, \gamma)$ -incompressible, Claim 22 gives that  $\|x|_J\|_2^2 \geq \|x|_J\|_\infty^2 + \gamma^2$ . Applying Lemma 23 to  $x|_J$ , we bound the first integral by

$$I_1 \leq \frac{c_2}{\gamma}.$$

Next, consider the second integral  $I_2$  in  $[T, 1/\varepsilon]$ . We will apply the LCD assumption to uniformly bound the integrand in this range. Given  $t \in [T, 1/\varepsilon]$ , let  $y(t) = \left\{\frac{x_i t}{2\pi m}\right\} \in [-1/2, 1/2]^n$ ,  $\beta(t) = \beta \min(t/\sqrt{n}, 1)$  and  $J(t) = \{i \in [n] : |y(t)_i| \geq \beta(t)\}$ . As  $t \leq 1/\varepsilon \leq 2\pi m D$ , we have that  $\frac{t}{2\pi m} \leq D = \text{LCD}_{\alpha, \beta}(x)$ , and hence  $|J(t)| \geq \alpha n$ . Applying Claim 21, we bound the integrand by

$$\prod_{i=1}^n F\left(\frac{x_i t}{2\pi m}\right) = \prod_{i=1}^n F(y_i) \leq \prod_{i \in J(t)} F(y_i) \leq \prod_{i \in J(t)} G(my_i) \leq \prod_{i \in J(t)} G(m\beta(t)) \leq G(m\beta(t))^{\alpha n}.$$

Following up on this, we have

$$\beta(t) \geq \beta(T) = \beta \frac{\sqrt{\alpha n - 1}}{\sqrt{n}} \geq \beta \sqrt{\alpha/2} \geq \alpha \beta,$$

where we used the assumptions that  $\alpha n \geq 2$  and  $\alpha \leq 1/2$ . We may assume that  $\alpha \beta m \geq 1$ , otherwise the conclusion of the lemma is trivial. In that case we have by Claim 21(4) that

$$G(m\beta(t)) \leq G(\alpha \beta m) \leq \frac{1}{\alpha \beta m}.$$

Thus we can bound the integral  $I_2$  by

$$I_2 = \int_T^{1/\varepsilon} \prod_{i=1}^n F\left(\frac{x_i t}{2\pi m}\right) dt \leq \frac{1/\varepsilon}{(\alpha \beta m)^{\alpha n}}.$$

Overall, we get

$$\Pr[|Y| \leq \varepsilon] \leq c_1 \varepsilon I = c_1 \varepsilon (I_1 + I_2) \leq \frac{c_1 c_2 \varepsilon}{\gamma} + \frac{c_1}{(\alpha \beta m)^{\alpha n}}. \quad \blacktriangleleft$$

## 6 Bounding the LCD

Our main goal in this section is to prove that a random normal vector  $X^*$  has large LCD with high probability. Let  $M'$  denote the  $(n-1) \times n$  matrix with rows  $X_1, \dots, X_{n-1}$ . Let  $D_0 = \sqrt{\alpha n}$  and  $D_1 = \beta(\alpha\beta m)^{\alpha n}$  in this section.

► **Lemma 24.** *Let  $\alpha \in (0, 1/40)$ ,  $\beta \in (0, 1/2)$  and  $D \in (1, D_1)$ . Then*

$$\Pr[\text{LCD}_{\alpha,\beta}(X^*) \leq D] \leq D^2 (1/\alpha c)^n \beta^{cn}$$

for some absolute constant  $c \in (0, 1)$ .

We set  $\gamma = \sqrt{\beta}$  throughout the section. We first condition on a number of bad events not holding. Define:

$$E_1 = [\|M\| \geq \sqrt{n \log(1/\beta)}]$$

$$E_2 = [X^* \text{ is } (5\alpha, \beta)\text{-compressible}]$$

$$E_3 = [X^* \text{ is } (\alpha, \gamma)\text{-compressible}]$$

Applying Claim 6 for  $E_1$ , and Lemma 13 for  $E_2, E_3$ , we get that

$$\Pr[E_1 \text{ or } E_2 \text{ or } E_3] \leq \beta^{cn}.$$

Thus, we will assume in this section that none of  $E_1, E_2, E_3$  hold. Assuming  $\neg E_2$ , Claim 17 yields that  $\text{LCD}_{\alpha,\beta}(X^*) \geq D_0$ . For  $D \geq D_0$  define

$$\mathcal{S}_D = \{x \in S^{n-1} : \text{LCD}_{\alpha,\beta}(x) \in [D, 2D] \text{ and } x \text{ is } (\alpha, \gamma)\text{-incompressible}\}.$$

The following is an analog of Lemma 7.2 in [11].

► **Claim 25.** Let  $D \geq D_0$  and set  $\nu = 6\beta\sqrt{n}/D$ . There exists a  $\nu$ -net  $\mathcal{N}_D \subset \mathcal{S}_D$  of size

$$|\mathcal{N}_D| \leq (D/\beta) \left( \frac{cD}{\sqrt{\alpha n}} \right)^n (1/\beta)^{\alpha n}.$$

Namely, for each  $x \in \mathcal{S}_D$  there exists  $y \in \mathcal{N}_D$  that satisfies  $\|x - y\|_2 \leq \nu$ .

*Proof.* Let  $x \in \mathcal{S}_D$  and shorthand  $D(x) = \text{LCD}_{\alpha,\beta}(x)$ . By definition, we can decompose  $\{D(x)x\} = u + v$  where  $u$  is  $(\alpha n)$ -sparse and  $\|v\|_2 \leq \beta \min(D, \sqrt{n}) \leq \beta\sqrt{n}$ .

Let  $W$  denote the set of  $(\alpha n)$ -sparse vectors  $w \in [-1/2, 1/2]^n$  such that each  $w_i$  is an integer multiple of  $\beta$ . Then  $|W| \leq \binom{n}{\alpha n} (1/\beta)^{\alpha n}$ , and there exists  $w \in W$  such that  $\|u - w\|_\infty \leq \beta$ , which implies  $\|u - w\|_2 \leq \beta\sqrt{n}$ . This implies that

$$\|\{D(x)x\} - w\|_2 \leq 2\beta\sqrt{n}.$$

Next, consider  $[D(x)x] \in \mathbb{Z}^n$ . As  $\lceil |a| \rceil \leq 2|a|$  for all  $a \in \mathbb{Z}$ , we have  $\|[D(x)x]\|_2 \leq 2D(x)\|x\|_2 \leq 4D$ . Let  $Z = \{z \in \mathbb{Z}^n : \|z\|_2 \leq 4D\}$ . Then  $[D(x)x] \in Z$ , and Claim 11 bounds  $|Z| \leq \left(1 + \frac{c_1 D}{\sqrt{n}}\right)^n$ . So there is  $z \in Z$  such that

$$\|D(x)x - z - w\|_2 \leq 2\beta\sqrt{n}.$$

Next, let  $R$  be set of integer multiples of  $\beta$  in the range  $[D, 2D]$ , so that  $|R| \leq D/\beta$  and there exists  $r \in R$  with  $|D(x) - r| \leq \beta$ . As  $\|x\|_2 = 1$  we have

$$\|rx - z - w\|_2 \leq 2\beta\sqrt{n} + \beta \leq 3\beta\sqrt{n}.$$

### 33:14 Singularity of Random Integer Matrices with Large Entries

Finally, define the set

$$Y = \{(z + w)/r : z \in Z, w \in W, r \in R\}.$$

Then there exists  $y \in Y$  such that

$$\|x - y\|_2 \leq 3\beta\sqrt{n}/D = \nu/2.$$

Take a maximal set  $\mathcal{N}_D \subset \mathcal{S}_D$  which is  $\nu$ -separated. That is, for any  $x', x'' \in \mathcal{N}_D$  we have  $\|x' - x''\|_2 > \nu$ . Note that by maximality,  $\mathcal{N}_D$  is a  $\nu$ -net in  $\mathcal{S}_D$ . Next, note that  $|\mathcal{N}_D| \leq |Y|$ , as any point  $x \in \mathcal{N}_D$  must be  $(\nu/2)$ -close to a distinct point in  $Y$ . To conclude, we need to bound  $|Y|$ . We have

$$|Y| \leq |W||Z||R| \leq \binom{n}{\alpha n} (1/\beta)^{\alpha n} \cdot \left(1 + \frac{cD}{\sqrt{n}}\right)^n \cdot (D/\beta).$$

As  $D \geq D_0 = \sqrt{\alpha n}$  we can simplify  $1 + \frac{cD}{\sqrt{n}} \leq \frac{(c+1)D}{\sqrt{\alpha n}}$ . We can trivially bound  $\binom{n}{\alpha n} \leq 2^n$ . Hence

$$|\mathcal{N}_D| \leq |Y| \leq (D/\beta) \left(\frac{2(c+1)D}{\sqrt{\alpha n}}\right)^n (1/\beta)^{\alpha n}. \quad \triangleleft$$

▷ **Claim 26.** For any  $D \in [D_0, D_1]$  we have

$$\Pr[X^* \in \mathcal{S}_D \text{ and } \neg E_1] \leq D^2 (c/\alpha)^n \beta^{n/8}.$$

*Proof.* First, note that we may assume  $\beta \leq \beta_0$  for any absolute constant  $\beta_0 \in (0, 1)$ , by choosing the constant  $c > 0$  large enough to compensate for that (namely, taking  $c \geq 1/\beta_0$ ). In particular, setting  $\beta_0 = 2^{-20}$  works.

If  $X^* \in \mathcal{S}_D$  then there exists  $y \in \mathcal{N}_D$  such that  $\|X^* - y\|_2 \leq \nu$  for  $\nu = 6\beta\sqrt{n}/D$ . By definition of  $X^*$  we have  $M'X^* = 0$ , and as we assume that  $\neg E_1$  hold, we have

$$\|M'y\|_2 \leq \|M'\| \|X^* - y\|_2 \leq \nu\sqrt{n \log(1/\beta)}.$$

Set  $\beta_1 = 6\beta\sqrt{\log(1/\beta)}$ . The assumption  $\beta \leq \beta_0$  implies that  $\beta_1 \leq \beta^{3/4}$ . Set  $\delta = \beta^{3/4}\sqrt{n}/D$ . We will bound the probability that there exists  $y \in \mathcal{N}_D$  such that  $\|M'y\|_2 \leq \delta\sqrt{n}$ .

Fix  $y \in \mathcal{N}_D$ , let  $X \sim \mathcal{D}^n$ , and define  $p(\varepsilon) = \Pr[|\langle X, y \rangle| \leq \varepsilon]$ . As  $y \in \mathcal{N}_D \subset \mathcal{S}_D$  we have that  $y$  is  $(\alpha, \gamma)$ -incompressible, and hence we can apply Lemma 18, which gives

$$p(\varepsilon) \leq c_1 \left( \frac{\varepsilon}{\gamma} + \frac{1}{(\alpha\beta m)^{\alpha n}} \right) \quad \text{for all } \varepsilon \geq 1/2\pi m D.$$

Next, we restrict attention to only  $\varepsilon \geq \delta$ , and note that in this regime the first term is dominant (since  $D \leq D_1$  we have  $\delta \geq \beta^{3/4}\sqrt{n}/D_1 \geq 1/(\alpha\beta m)^{\alpha n}$ ). We can then simplify the bound as

$$p(\varepsilon) \leq \frac{c_2 \varepsilon}{\gamma} \quad \text{for all } \varepsilon \geq \delta.$$

Applying Claim 9, and recalling that we set  $\gamma = \sqrt{\beta}$ , gives

$$\Pr[\|M'y\|_2 \leq \delta\sqrt{n}] \leq \left(\frac{c_3 \delta}{\gamma}\right)^{n-1} = \left(\frac{c_4 \beta^{1/4} \sqrt{n}}{D}\right)^{n-1}.$$



Union bounding over all  $y \in \mathcal{N}_D$ , using Claim 25 to bound its size, gives

$$\begin{aligned} \Pr[\exists y \in \mathcal{N}_D, \|M'y\|_2 \leq \delta\sqrt{n}] &\leq (D/\beta) \left(\frac{cD}{\sqrt{\alpha n}}\right)^n (1/\beta)^{\alpha n} \cdot \left(\frac{c_4\beta^{1/4}\sqrt{n}}{D}\right)^{n-1} \\ &\leq D^2 (c_5/\sqrt{\alpha})^n \beta^{n/4-\alpha n-2}. \end{aligned}$$

Our assumption  $\alpha < 1/40$  and the implicit assumption  $\alpha n \geq 2$  imply that  $\alpha n + 2 \leq n/8$ , which simplifies the above bound to the claimed bound.  $\triangleleft$

We are now in place to prove Lemma 24.

**Proof of Lemma 24.** We may assume that non of  $E_1, E_2, E_3$  hold, as the probability that any of them hold is at most  $\beta^{c_1 n}$  for some absolute constant  $c_1 \in (0, 1)$ . This in particular implies that  $\text{LCD}_{\alpha, \beta}(X^*) \geq D_0$ . Fix  $D \in [D_0, D_1]$ . As  $D \leq D_1$  we can applying Claim 26 to  $D_i = 2^i D_0$  as long as  $D_i \leq D/2$ . Summing the results we get

$$\Pr[\text{LCD}_{\alpha, \beta}(X^*) \leq D \text{ and } \neg E_1, \neg E_2, \neg E_3] \leq (2D)^2 (c_2/\alpha)^n \beta^{n/8}.$$

Thus overall we have

$$\Pr[\text{LCD}_{\alpha, \beta}(X^*) \leq D] \leq \beta^{c_1 n} + (2D)^2 (c_2/\alpha)^n \beta^{n/8}.$$

The lemma follows by taking  $c \in (0, 1)$  small enough.  $\blacktriangleleft$

## 7 Completing the proof

We now prove Theorem 1.

**Proof of Theorem 1.** Fix  $\alpha = 1/50, \beta = 1/\sqrt{m}$  and assume  $m \geq m_0$  for a large enough constant  $m_0$  to be determined soon. Let  $D$  to be determined soon. Lemma 24 gives

$$\Pr[\text{LCD}_{\alpha, \beta}(X^*) \leq D] \leq D^2 (1/\alpha c_1)^n \beta^{c_1 n}.$$

As  $\alpha$  is constant, and using the choice  $\beta = 1/\sqrt{m}$ , we can simplify the bound as follows. For a small enough constant  $c \in (0, 1)$ , setting  $D = m^{cn}$  and  $c_2 = 1/\alpha c_1$ , we have

$$\Pr[\text{LCD}_{\alpha, \beta}(X^*) \leq m^{cn}] \leq m^{2cn} c_2^n m^{-(c_1/2)n} \leq c_2^n m^{-(c_1/2-2c)n} \leq c_2^n m^{-cn}.$$

Assuming  $m \geq m_0$  for a large enough constant  $m_0$ , we can simplify this bound further as

$$\Pr[\text{LCD}_{\alpha, \beta}(X^*) \leq m^{cn}] \leq m^{-(c/2)n}.$$

Next, assume  $D = \text{LCD}_{\alpha, \beta}(X^*) \geq m^{cn}$ . In this case, Lemma 18 for  $\varepsilon = 1/2\pi m D$  gives that

$$\Pr[\langle X^*, X \rangle = 0] \leq \Pr[|\langle X^*, X \rangle| \leq \varepsilon] \leq c_3 \left( \frac{\varepsilon}{\gamma} + \frac{1}{(\alpha\beta m)^{\alpha n}} \right) \leq m^{-c'n}$$

for some  $c' \in (0, 1)$ . Overall we obtain the desired bound.  $\blacktriangleleft$

## References

- 1 Simeon Ball. On sets of vectors of a finite vector space in which every subset of basis size is a basis. *Journal of the European Mathematical Society*, 14(3):733–748, 2012.
- 2 Jean Bourgain, Van H Vu, and Philip Matchett Wood. On the singularity probability of discrete random matrices. *Journal of Functional Analysis*, 258(2):559–603, 2010.
- 3 Anthony Carbery and James Wright. Distributional and  $L^q$  norm inequalities for polynomials over convex bodies in  $\mathbb{R}^n$ . *Mathematical research letters*, 8(3):233–248, 2001.
- 4 CG Esseen. On the Kolmogorov-Rogozin inequality for the concentration function. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 5(3):210–216, 1966.
- 5 Jeff Kahn, János Komlós, and Endre Szemerédi. On the probability that a random  $\pm 1$ -matrix is singular. *Journal of the American Mathematical Society*, 8(1):223–240, 1995.
- 6 Yonathan Katznelson. Singular matrices and a uniform bound for congruence groups of  $SL_n(\mathbb{Z})$ . *Duke Mathematical Journal*, 69(1):121–136, 1993.
- 7 János Komlós. On determinant of  $(0, 1)$  matrices. *Studia Science Mathematica Hungarica*, 2:7–21, 1967.
- 8 L Leindler. On a certain converse of Hölder’s inequality. In *Proceedings of the 1971 Oberwolfach Conference*, BirkhHuser Verlag. Basel-Stuttgart, 1972.
- 9 Vitali D Milman and Gideon Schechtman. *Asymptotic theory of finite dimensional normed spaces: Isoperimetric inequalities in riemannian manifolds*, volume 1200. Springer, 2009.
- 10 András Prékopa. On logarithmic concave measures and functions. *Acta Scientiarum Mathematicarum*, 34:335–343, 1973.
- 11 Mark Rudelson. Lecture notes on non-asymptotic theory of random matrices, 2013.
- 12 Mark Rudelson and Roman Vershynin. The Littlewood–Offord problem and invertibility of random matrices. *Advances in Mathematics*, 218(2):600–633, 2008.
- 13 Mark Rudelson and Roman Vershynin. Non-asymptotic theory of random matrices: extreme singular values. In *Proceedings of the International Congress of Mathematicians 2010 (ICM 2010) (In 4 Volumes) Vol. I: Plenary Lectures and Ceremonies Vols. II–IV: Invited Lectures*, pages 1576–1602. World Scientific, 2010.
- 14 Jacob T Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM (JACM)*, 27(4):701–717, 1980.
- 15 Beniamino Segre. Curve razionali normali ek-archi negli spazi finiti. *Annali di Matematica Pura ed Applicata*, 39(1):357–379, 1955.
- 16 Richard Singleton. Maximum distance q-nary codes. *IEEE Transactions on Information Theory*, 10(2):116–118, 1964.
- 17 Terence Tao and Van Vu. On random  $\pm 1$  matrices: singularity and determinant. *Random Structures & Algorithms*, 28(1):1–23, 2006.
- 18 Terence Tao and Van Vu. On the singularity probability of random Bernoulli matrices. *Journal of the American Mathematical Society*, 20(3):603–628, 2007.
- 19 Konstantin Tikhomirov. Singularity of random Bernoulli matrices. *Annals of Mathematics*, 191(2):593–634, 2020.
- 20 Santosh S Vempala, Ruosong Wang, and David P Woodruff. The communication complexity of optimization. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1733–1752. SIAM, 2020.
- 21 Richard Zippel. Probabilistic algorithms for sparse polynomials. In *International symposium on symbolic and algebraic manipulation*, pages 216–226. Springer, 1979.

# Interplay Between Graph Isomorphism and Earth Mover's Distance in the Query and Communication Worlds

Sourav Chakraborty ✉🏠

Indian Statistical Institute, Kolkata, India

Arijit Ghosh ✉🏠

Indian Statistical Institute, Kolkata, India

Gopinath Mishra ✉🏠

Indian Statistical Institute, Kolkata, India

Sayantan Sen ✉🏠

Indian Statistical Institute, Kolkata, India

---

## Abstract

The graph isomorphism distance between two graphs  $G_u$  and  $G_k$  is the fraction of entries in the adjacency matrix that has to be changed to make  $G_u$  isomorphic to  $G_k$ . We study the problem of estimating, up to a constant additive factor, the graph isomorphism distance between two graphs in the query model. In other words, if  $G_k$  is a known graph and  $G_u$  is an unknown graph whose adjacency matrix has to be accessed by querying the entries, what is the query complexity for testing whether the graph isomorphism distance between  $G_u$  and  $G_k$  is less than  $\gamma_1$  or more than  $\gamma_2$ , where  $\gamma_1$  and  $\gamma_2$  are two constants with  $0 \leq \gamma_1 < \gamma_2 \leq 1$ . It is also called the tolerant property testing of graph isomorphism in the dense graph model. The non-tolerant version (where  $\gamma_1$  is 0) has been studied by Fischer and Matsliah (SICOMP'08).

In this paper, we prove a (interesting) connection between tolerant graph isomorphism testing and tolerant testing of the well studied Earth Mover's Distance (EMD). We prove that deciding tolerant graph isomorphism is equivalent to deciding tolerant EMD testing between multi-sets in the query setting. Moreover, the reductions between tolerant graph isomorphism and tolerant EMD testing (in query setting) can also be extended directly to work in the two party Alice-Bob communication model (where Alice and Bob have one graph each and they want to solve tolerant graph isomorphism problem by communicating bits), and possibly in other sublinear models as well.

Testing tolerant EMD between two probability distributions is equivalent to testing EMD between two multi-sets, where the multiplicity of each element is taken appropriately, and we sample elements from the unknown multi-set **with** replacement. In this paper, our (main) contribution is to introduce the problem of (*tolerant*) EMD testing between multi-sets (over Hamming cube) when we get samples from the unknown multi-set **without** replacement and to show that *this variant of tolerant testing of EMD is as hard as tolerant testing of graph isomorphism between two graphs*. Thus, while testing of equivalence between distributions is at the heart of the non-tolerant testing of graph isomorphism, we are showing that the estimation of the EMD over a Hamming cube (when we are allowed to sample **without** replacement) is at the heart of tolerant graph isomorphism. We believe that the introduction of the problem of testing EMD between multi-sets (when we get samples **without** replacement) opens an entirely new direction in the world of testing properties of distributions.

**2012 ACM Subject Classification** Theory of computation → Streaming, sublinear and near linear time algorithms

**Keywords and phrases** Graph Isomorphism, Earth Mover Distance, Query Complexity

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.34

**Category** RANDOM

**Related Version** *Full Version:* <https://eccc.weizmann.ac.il/report/2020/135/>

**Acknowledgements** The authors would like to thank an anonymous reviewer for pointing out a mistake in an earlier version of this paper, as well as the reviewers of RANDOM for various suggestions that improved the presentation of the paper.



© Sourav Chakraborty, Arijit Ghosh, Gopinath Mishra, and Sayantan Sen; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 34; pp. 34:1–34:23



Leibniz International Proceedings in Informatics

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

## 1 Introduction

Graph isomorphism (GI) has been one of the most celebrated problems in computer science. Roughly speaking, the graph isomorphism problem asks whether two graphs are structure-preserving. Namely, given two graphs  $G_u$  and  $G_k$ , graph isomorphism of  $G_u$  and  $G_k$  is a bijection  $\psi : V(G_u) \rightarrow V(G_k)$  such that for all pair of vertices  $u, v \in V(G_u)$ , the edges  $\{u, v\} \in E(G_u)$  if and only if  $\{\psi(u), \psi(v)\} \in E(G_k)$ <sup>1</sup>. One central open problem in complexity theory is whether the graph isomorphism problem can be solved in polynomial time. Recently in a breakthrough result, Babai [5] proved that the graph isomorphism problem could be decided in quasi-polynomial time.

For a central problem like the graph isomorphism, naturally, one would like to understand its (and related problems) computational complexity for various models of computation. While most of the focus has been on the standard time complexity in the RAM model for various classes of graphs (and hyper-graphs), other complexity measures like space complexity, parameterized complexity, and query complexity have also been studied over the past few decades (see the Dagstuhl Report [7] and PhD thesis of Sun [24]).

A natural extension of the GI problem is to estimate the “graph isomorphism distance” between two graphs. In other words, given two graphs  $G_u$  and  $G_k$ , what fraction of edges are necessary to add or delete to make the graphs isomorphic.

► **Definition 1.1.** Let  $G_u = (V_u, E_u)$  and  $G_k = (V_k, E_k)$  be two graphs with  $|V_u| = |V_k| = n$ . Given a bijection  $\phi : V_u \rightarrow V_k$ , the distance between the graphs  $G_u$  and  $G_k$  with respect to the bijection  $\phi$  is

$$d_\phi(G_u, G_k) := |\{(u, v) : \text{Exactly one among } (u, v) \in E_u \text{ or } (\phi(u), \phi(v)) \in E_k \text{ holds}\}|.$$

The GRAPH ISOMORPHISM DISTANCE (or GI-distance in short) between graphs  $G_u$  and  $G_k$  is defined as  $\min_{\phi: V_u \rightarrow V_k} d_\phi(G_u, G_k)/n^2$ , and is denoted by  $\delta_{GI}(G_u, G_k)$  (we will use  $d(G_u, G_k)$  to mean  $n^2 \delta_{GI}(G_u, G_k)$ ).

The problem of computing GI-distance between two graphs is known to be  $\#P$ -hard [18]. The next natural question is:

*What is the complexity for approximating (either by a constant additive or multiplicative factor) the graph isomorphism distance between two graphs?*

In [18], it was also proven that the problem of computing GI-distance between two graphs is APX-hard. So, approximating  $\delta_{GI}(G_u, G_k)$  up to a constant multiplicative factor is NP-hard. In this paper, we study this problem of approximating (up to a constant additive factor) the GI-distance between two graphs in the query model and two party communication complexity model.

### 1.1 Property Testing of Graph Isomorphism

Formally speaking, the main problem is: given two graphs  $G_u$  and  $G_k$  and an approximation parameter  $\zeta \in (0, 1)$ , the goal is to output an estimate  $\alpha$  such that

$$\delta_{GI}(G_u, G_k) - \zeta \leq \alpha \leq \delta_{GI}(G_u, G_k) + \zeta.$$

<sup>1</sup> In a graph  $G$ ,  $V(G)$  and  $E(G)$  denote the sets of vertices and edges in  $G$ , respectively.

In the query model, the problem is equivalent (up to a constant factor) to the tolerant property testing of graph isomorphism in the dense graph model (introduced in the work of Parnas, Ron and Rubinfeld [21]). For  $0 \leq \gamma < 1$ , two graphs  $G_u$  and  $G_k$ , with  $n$  vertices, are called  $\gamma$ -close or  $\gamma$ -far to isomorphic<sup>2</sup> if  $d(G_u, G_k) \leq \gamma n^2$  or  $d(G_u, G_k) \geq \gamma n^2$ , respectively. In  $(\gamma_1, \gamma_2)$ -tolerant GI testing, we are given two graphs  $G_u$  and  $G_k$ , and two parameters  $0 \leq \gamma_1 < \gamma_2 \leq 1$ , with the guarantee that either the graphs are  $\gamma_1$ -close or  $\gamma_2$ -far. One of the graphs (usually denoted as  $G_u$ ) is accessed by querying the entries of its adjacency matrix. In contrast, the other graph (usually denoted as  $G_k$ <sup>3</sup>) is known to the query algorithm, and no cost for accessing the entries of the adjacency matrix of  $G_k$  is incurred. The query complexity is the number of queries (to the adjacency matrix of  $G_u$ ) that are required for testing, (with correctness probability at least  $2/3$ <sup>4</sup>), whether  $G_u$  and  $G_k$  are  $\gamma_1$ -close or  $\gamma_2$ -far. The query algorithm is assumed to have unbounded computational power.

The non-tolerant property testing version of the graph isomorphism problem (that is, when  $\gamma_1 = 0$ ) was first studied by Fischer and Matsliah [13] and subsequently, Babai and Chakraborty [6] studied the non-tolerant property testing version of the hypergraph isomorphism problem. Recently, the non-tolerant testing of GI has been considered in various other models (like Goldreich [15] studied the problem for the *bounded degree graph model* of property testing and Levi and Medina [17] considered the problem in the *distributed* setting). However, the tolerant version of the problem remains elusive and it is surprising that the tolerant version of a fundamental problem like graph isomorphism (in query model) is not addressed in the literature, though the non-tolerant version of GI testing problem has been resolved more than a decade ago in [13] (when one graph is unknown). On a different note, there are also studies of non-tolerant version of graph isomorphism testing in the literature when both the graphs are unknown [13, 19]. We will not discuss much about that case as the main focus of this paper is different.

Before proceeding further, we want to note that there is a simple algorithm with query complexity  $\tilde{O}(n)$  for tolerant testing of graph isomorphism (when one of the graphs is known in advance). Basically, one goes over all possible  $n!$  bijections  $\phi: V_u \rightarrow V_k$  and estimates the distance between  $G_u$  and  $G_k$  with respect to the permutation. The samples may be reused<sup>5</sup>, and hence we have the following observation.

► **Observation 1.2.** Given a known graph  $G_k$  and an unknown graph  $G_u$  and any approximation parameter  $\zeta \in (0, 1)$ , there is a query algorithm that makes  $\tilde{O}(n)$  queries and outputs a number  $\alpha$  such that, with probability at least  $2/3$ , the following holds:

$$\delta_{GI}(G_u, G_k) - \zeta \leq \alpha \leq \delta_{GI}(G_u, G_k) + \zeta.$$

But obtaining a lower bound matching (at least up to a polylog factor) the upper bound of Observation 1.2 is not at all obvious. This paper's main contribution is to show an equivalence between tolerant testing of graph isomorphism and tolerant EMD testing between multi-sets (in the query setting).

<sup>2</sup> As a shorthand, rather than saying  $\gamma$ -close or  $\gamma$ -far to isomorphic, we will just say  $\gamma$ -close or  $\gamma$ -far respectively.

<sup>3</sup>  $G_u$  and  $G_k$  denote the unknown and known graphs, respectively.

<sup>4</sup> The correctness probability can be made any  $1 - \delta$  by incurring a multiplicative factor of  $O(\log \frac{1}{\delta})$  in the query complexity.

<sup>5</sup> If the samples are  $\Theta(\log(n!))$ , then the error probability can be bounded using the union bound.

Like many other property testing problems, the core difficulty in the testing of GI is understanding certain properties of distributions. In the case of the non-tolerant version of GI, it has been shown in [13] that the core problem is testing the variation distance between two distributions. Their upper bound result can be restated as: if there is a property testing algorithm, with query complexity  $q(n)$  for testing equivalence between two distributions, on support size  $n$ <sup>6</sup>, then GI can be tested using  $\tilde{\mathcal{O}}(q(n))$  queries, where the tilde hides a polylogarithmic factor of  $n$  (number of vertices). And since the query complexity for testing identity of distributions (from [8], [20], [1], [26]) is known to be  $\mathcal{O}(\frac{\sqrt{n}}{\epsilon^2})$ , the query complexity for non tolerant GI-testing is  $\tilde{\mathcal{O}}(\sqrt{n})$ .

In the lower bound proof of [13], there is no direct reduction of the graph isomorphism problem to the variation distance problem. But it is important to note that lower bound proofs for both of these problems use the tightness of the *birthday paradox*. So, in some sense, one can say that the heart of the non-tolerant testing of GI is in testing variation distance between two distributions.

## 1.2 Earth Mover's Distance (EMD)

Let  $H = \{0, 1\}^n$  be a Hamming cube of dimension  $n$ , and  $p, q$  be two probability distributions on  $H$ . The *Earth Mover's Distance* between  $p$  and  $q$  is denoted by  $EMD(p, q)$  and defined as the optimum solution to the following linear program:

$$\text{Minimize } \sum_{i,j \in H} f_{ij} d_H(i, j) \quad \text{Subject to } \sum_{j \in H} f_{ij} = p(i) \forall i \in H, \text{ and } \sum_{i \in H} f_{ij} = q(j) \forall j \in H.$$

A standard way to think of sampling from any probability distribution is to consider it as a multi-set of elements with appropriate multiplicities, and samples are drawn **with** replacement from that multi-set. While estimating EMD between two multi-sets, although the most natural way to access the unknown multi-set is sampling **with** replacement, we introduce the problem of tolerant EMD testing over multi-sets with the access of samples **without** replacement.

► **Definition 1.3 (EMD over multi-sets while sampling with and without replacement).** Let  $S_1$  and  $S_2$  denote two multi-sets, over  $n$ -dimensional Hamming cube  $H = \{0, 1\}^n$  such that  $|S_1| = |S_2| = n$ . Consider the two distributions  $p_1$  and  $p_2$  over the Hamming cube  $H$  that are naturally defined by the sets  $S_1$  and  $S_2$  where for all  $x \in H$  probability of  $x$  in  $p_1$  (and  $p_2$ ) is the number of occurrences of  $x$  in  $S_1$  (and  $S_2$ ) divided by  $n$ . We then define the EMD between the multi-sets  $S_1$  and  $S_2$  as

$$EMD(S_1, S_2) \triangleq n \cdot EMD(p_1, p_2).$$

The problem of estimating the EMD over multi-sets while sampling **with** (or **without**) replacement means designing an algorithm, that given any two constants  $\beta_1, \beta_2$  such that  $0 \leq \beta_1 < \beta_2 \leq 1$ , a known multi-set  $S_k$  and access to the unknown multi-set  $S_u$  by sampling **with** (or **without**) replacement, decides whether  $EMD(S_k, S_u) \leq \beta_1 n^2$  or  $EMD(S_k, S_u) \geq \beta_2 n^2$  with probability at least  $2/3$ . Note that estimating the EMD over multi-sets while sampling **with** replacement is exactly same as estimating EMD between the distributions  $p_u$  and  $p_k$  with samples drawn according to  $p_u$ .

---

<sup>6</sup> Testing identity between two distributions means to test if the unknown distribution (from where the samples are drawn) is identical to the known distribution or if the variation distance between them more than  $\epsilon$ .

We will denote by  $\text{QWR}_{\text{EMD}}(n, \beta_1, \beta_2)$  (and  $\text{QWOR}_{\text{EMD}}(n, \beta_1, \beta_2)$ ) the number of samples **with** (or **without**) replacement required to decide the above from the unknown multi-set  $S_u$ . For ease of presentation, we will write  $\text{QWOR}_{\text{EMD}}(n)$  ( $\text{QWR}_{\text{EMD}}(n)$ ) instead of  $\text{QWOR}_{\text{EMD}}(n, \beta_1, \beta_2)$  ( $\text{QWR}_{\text{EMD}}(n, \beta_1, \beta_2)$ ) when the proximity parameters are clear from the context.

Earth Mover's Distance (EMD) is a fundamental metric over the space of distributions supported on a fixed metric space. Estimating EMD between two distributions, up to a multiplicative factor, has been extensively studied in mathematics and computer science. It is closely related to the embedding of the EMD metric into a  $\ell_1$  metric. Even the problem of estimation of EMD between distributions up to an additive factor has been well studied, for reference see [12], [23]. The hardness of estimating EMD between distributions depends heavily on the structure of the domain on which the distributions are supported. In [12], the authors have proved a lower bound of  $\Omega((\Delta/\epsilon)^d)$  on the query complexity for estimating (up to an additive error of  $\epsilon$ ) EMD between two distributions supported on the real cube  $[0, \Delta]^d$ . At the same time, it is not hard to see that if the support has certain structures, estimating EMD may be easy. In this paper, we focus on the estimation of EMD between two distribution when the metric space is the Hamming cube.

As noted earlier, sample access to a probability distribution is precisely the same as uniform sampling from a multi-set **with** replacement. Thus, from the results of Valiant and Valiant [25], it can be shown that the sample complexity for estimating the EMD between two distribution over the Hamming cube of dimension  $n$  is  $\Omega(n/\log n)$ . In other words,  $\text{QWR}_{\text{EMD}}(n) = \Omega(n/\log n)$ , and this is tight ignoring polynomial factor in  $\log n$  (See Theorem B.10 of Appendix B). But what about  $\text{QWOR}_{\text{EMD}}(n)$ ? To the best of our knowledge, the sample complexity measure when the distributions are accessed by sampling a multi-set **without** replacement has never been studied before (for testing/estimating *distances* between distributions/multi-sets). However, it is interesting to note that, sampling **without** replacement model has been considered before in a different context by Raskhodnikova, Ron, Shpilka and Smith [22] for proving a lower bound of distinct elements problem. Also, recently Goldreich [15] considered a similar sampling **without** replacement model while studying the non-tolerant graph isomorphism in the bounded degree model.

Coming back to our context, it can be proven that: if  $\text{QWOR}_{\text{EMD}}(n) = o(\sqrt{n})$ , then  $\text{QWR}_{\text{EMD}}(n) = o(\sqrt{n})$  (See Proposition B.7 of Appendix B). As  $\text{QWR}_{\text{EMD}}(n) = \Omega(\frac{n}{\log n})$ , we have a lower bound of  $\Omega(\sqrt{n})$  on  $\text{QWOR}_{\text{EMD}}(n)$ . To the best of our knowledge, there is no known better lower bound than  $\Omega(\sqrt{n})$  for  $\text{QWOR}_{\text{EMD}}(n)$ , although a lower bound of  $\Omega(\frac{n}{\log n})$  exists for  $\text{QWR}_{\text{EMD}}(n)$  (using observation in [12]). We verified that the proof of [27] also goes through for  $\text{QWOR}_{\text{EMD}}(n)$  as well (See Theorem 1.5). We now present the following conjecture:

► **Conjecture 1.** *There exist two constants  $\beta_1$  and  $\beta_2$  with  $0 < \beta_1 < \beta_2 < 1$  such that in order to decide whether  $\text{EMD}(S_k, S_u) \leq \beta_1 n^2$  or  $\text{EMD}(S_k, S_u) \geq \beta_2 n^2$ , with probability at least  $2/3$ ,  $\Omega\left(\frac{n}{\text{poly}(\log n)}\right)$  samples **without** replacement from the unknown multi-set  $S_u$  are necessary.*

One of our main contributions in this paper is introducing this complexity measure of  $\text{QWOR}_{\text{EMD}}(n)$  as well as the above conjecture. In the rest of the paper, we focus on exploring the connection between  $\text{QWOR}_{\text{EMD}}(n)$  and the query complexity of tolerant GI-testing. For a formal discussion on EMD over Hamming cube, please refer to Appendix B.

### 1.3 Our Results

Our main result of this paper is that we prove estimating GI-distance is as hard as tolerant EMD testing over multi-sets with the access of samples **without** replacement over the unknown multi-set  $S_u$ , ignoring polynomial factors of  $\log n$ .

► **Theorem 1.4 (Main Result).** *Let  $G_k$  and  $G_u$  denote the known and the unknown graphs on  $n$  vertices, respectively, and  $Q_{GI}(G_u, G_k)$  denotes the number of adjacency queries to  $G_u$ , required by the best algorithm that takes two constants  $\gamma_1, \gamma_2$  with  $0 \leq \gamma_1 < \gamma_2 \leq 1$  and decides whether  $d(G_u, G_k) \leq \gamma_1 n^2$  or  $d(G_u, G_k) \geq \gamma_2 n^2$  with probability at least  $2/3$ . Then*

$$Q_{GI}(G_u, G_k) = \tilde{\Theta}(\text{QWoR}_{\text{EMD}}(n))$$

where  $\tilde{\Theta}(\cdot)$  hides polynomial factors in  $\frac{1}{\gamma_2 - \gamma_1}$  and  $\log n$ .

#### 1.3.1 Implication of Theorem 1.4 to Query Complexity of Tolerant GI

It is interesting to note that our lower bound proof is via a *pure reduction* from tolerant graph isomorphism to tolerant testing of *EMD* of multi-sets over the Hamming cube using samples **without** replacement. Thus our reductions also hold for other computational models such as the communication complexity model. Regarding the lower bound on the sample complexity of tolerant EMD testing of multi-sets (in the **with** replacement model), using observation in [12], we note that the tolerant EMD testing is as hard as tolerant testing of variation distance. In [27], they gave a lower bound of  $\Omega(n^{1-o(1)})$  on the sample complexity for tolerant  $\ell_1$  testing. Although the proof of [27] uses samples **with** replacement (when we think of a distribution as a multi-set), it can be verified that the proof also works for samples **without** replacement.

► **Theorem 1.5 (Follows from [27]).** *For any constants  $0 < \alpha < \beta < 1$ , distinguishing between distribution pairs with statistical distance less than  $\alpha$  from those with distance greater than  $\beta$  requires  $n^{1-o(1)}$  samples **without** replacement.*

From Theorem 1.5, a similar lower bound follows for tolerant EMD testing of multi-sets **without** replacement. Thus, from Theorem 1.4, we have the following corollary:

► **Corollary 1.6.** *Let  $G_k$  and  $G_u$  be the known and unknown graphs on  $n$  vertices, respectively. For any constants  $0 < \gamma_1 < \gamma_2 < 1$ , distinguishing between isomorphism distance of  $d(G_u, G_k) \leq \gamma_1 n^2$  with  $d(G_u, G_k) \geq \gamma_2 n^2$  requires  $n^{1-o(1)}$  queries to the adjacency matrix of  $G_u$ . On the other hand, for any constants  $0 < \gamma_1 < \gamma_2 < 1$ , distinguishing between isomorphism distance of  $d(G_u, G_k) \leq \gamma_1 n^2$  with  $d(G_u, G_k) \geq \gamma_2 n^2$  can be done in  $\tilde{O}(n)$  queries.*

The lower bound of [27] was later improved to  $\Omega(\frac{n}{\log n})$  in [25]. However, the arguments of [25] are much more delicate and it is not completely clear to us whether their result of  $\Omega(\frac{n}{\log n})$  can be carried over to the **without** replacement setting, even if we allow a loss of polylogarithmic factor. So, we propose the following conjecture:

► **Conjecture 2.** *Let  $G_k$  and  $G_u$  be the known and unknown graphs on  $n$  vertices, respectively. For any constants  $0 < \gamma_1 < \gamma_2 < 1$ , distinguishing between isomorphism distance of  $d(G_u, G_k) \leq \gamma_1 n^2$  with  $d(G_u, G_k) \geq \gamma_2 n^2$  requires  $\Omega(\frac{n}{\log n})$  queries to the adjacency matrix of  $G_u$ .*



Note that Conjecture 1 and Conjecture 2 are equivalent. Besides, the difference between sampling **with** and **without** replacement is much more subtle. Freedman [14] has shown the difference when we sample elements **with** replacement from a set and that **without** replacement from the same set. However, when the number of samples is  $o(\sqrt{n})$ , the distribution of answers to the queries when samples are drawn **with** replacement is very close (in  $\ell_1$  distance) to the distribution of answers to the queries when samples are drawn **without** replacement. Thus, following Proposition B.7 along with Theorem 1.4, we can get an alternative proof of the following lower bound proved by Fischer and Matsliah [13].

► **Corollary 1.7** (Fischer and Matsliah [13]). *There exists a constant  $\zeta \in (0, 1)$  such that any query algorithm that decides, with probability at least  $2/3$ , if a known graph  $G_k$  and an unknown graph  $G_u$  is isomorphic or  $\gamma$ -far from isomorphic, with  $\gamma \leq \zeta$ , must make  $\Omega(\sqrt{n})$  queries.*

### 1.3.2 Implication of Theorem 1.4 to Communication Complexity of Tolerant GI

One of the central models of computation (particularly in the context of theoretical computer science) is the 2-player communication game introduced by Yao [28] in 1979. Communication complexity is one of the most studied complexity measures and has wide-ranging applications in many different areas of computer science. But surprisingly, as far as we know, the communication complexity problem of GI (where Alice has graph  $G_a$  and Bob has graph  $G_b$ , and they want to decide if  $G_a$  and  $G_b$  are isomorphic) has never been studied. One of the main reasons may be that, in the communication setup, the standard GI problem reduces to the string equality checking problem, and hence GI in the (randomized) communication setup is not that interesting anymore, since the randomized communication complexity, trivially, becomes  $O(1)$  (see the full version for the proof).

But when it comes to tolerant GI testing, the communication version is not at all obvious. So, if Alice and Bob are given two graphs  $G_a$  and  $G_b$  respectively, what is the (randomized) communication complexity for checking if  $d(G_a, G_b) \leq \gamma_1 n^2$  or  $d(G_a, G_b) \geq \gamma_2 n^2$ ? While we don't have a complete answer to this question yet, the following theorem holds from Theorem 1.2:

► **Theorem 1.8** (Informally stated). *If Alice and Bob are given two graphs  $G_a$  and  $G_b$  with  $n$  vertices respectively and the (randomized) communication complexity for checking if the graphs are  $\gamma_1$ -close or  $\gamma_2$ -far is  $c(n, \gamma_1, \gamma_2)$  then the following holds: There exists an absolute constant  $C$  such that if Alice and Bob are given two  $n$ -grained distributions<sup>7</sup> over the  $Cn$ -dimension Hamming cube, then the (randomized) communication complexity of checking if the Earth Mover's Distance between the distributions is at most  $\beta_1 n$  or at least  $\beta_2 n$  is  $\Theta(c(n, \gamma'_1, \gamma'_2))$ , where  $\gamma'_1$  and  $\gamma'_2$  are constants that depend only on  $\beta_1$  and  $\beta_2$ , and  $\tilde{\Theta}(\cdot)$  hides multiplicative factor of  $\text{poly}(\log n)$ .*

Theorem 1.8 says that the communication complexity of solving tolerant graph isomorphism and tolerant EMD testing are essentially the same, ignoring the polylog factor. Note that in the case of the communication setting, the distinction between **with** replacement and **without** replacement is not present. Also, it is important to point out that the lower bounds on tolerant EMD in the sampling model ([27] and [25]) does not give a lower bound

<sup>7</sup> The probability of each element in the sample space is an integer multiple of  $\frac{1}{n}$ .

in the communication setting. Though the tolerant graph isomorphism problem has not been addressed at all in the literature of communication complexity, EMD (for different metric spaces) has been considered in communication, streaming, and sketching models [16, 3, 2, 4]. However, the EMD problem that we have considered in this paper is different from those considered in the literature, and we believe that it will be of independent interest.

We also observe that the deterministic communication complexity of graph isomorphism is  $\Omega(n^2)$  even for the non-tolerant setting.

► **Theorem 1.9.** *Deterministic communication complexity of non-tolerant version of Graph Isomorphism testing (hence the tolerant version) is  $\Theta(n^2)$ .*

The proof of the above theorem is present in the full version of the paper [10].

**Organization of the paper.** In Section 2, we discuss the proof techniques of our main results. We prove the lower bound part (tolerant graph isomorphism is as hard as tolerant EMD testing) and upper bound part (tolerant EMD testing is as hard as tolerant graph isomorphism) of Theorem 1.4 in Sections 3 and 4 respectively. We finally conclude in Section 5. For space constraint, we could not add all possible proofs. Please see [10] for the full version of the paper.

**Notations.** All graphs considered here are undirected, unweighted, and have no self-loops or parallel edges. For a graph  $G(V, E)$ ,  $V(G)$  and  $E(G)$  will denote the vertex set and the edge set of  $G$ , respectively. Since we are considering undirected graphs, we write an edge  $(u, v) \in E(G)$  as  $\{u, v\}$ . The *Hamming distance* between two points  $x$  and  $y$  in a Hamming cube  $\{0, 1\}^k$  will be denoted by  $d_H(x, y)$ .

## 2 Discussion on our proof of Theorem 1.4

### 2.1 Reduction from tolerant EMD testing to tolerant graph isomorphism testing (Lower bound part of Theorem 1.4)

In this reduction, we crucially use the fact that the multi-sets are composed of elements from the Hamming cube. The reduction is based upon an involved gadget construction. In fact, we prove the lower bound for a slightly more powerful query model rather than the standard adjacency matrix query model. The most interesting part of our lower bound proof is that thanks to our reduction, we get to observe the importance of the model of accessing the multi-set **without** replacement in the context of EMD testing.

Now, we discuss the overview of our reduction. Let  $S_k$  and  $S_u$  denote the known and the unknown multi-sets, over a Hamming cube  $\{0, 1\}^d$  (of dimension  $d$ ) with  $d = \Theta(n)$ , having  $n$  elements each. To start with, let us assume that we know both  $S_k$  and  $S_u$ . We will construct two graphs  $G_k$  and  $G_u$  on  $d + n$  vertices as follows:

- The vertex set of  $G_k$  (and  $G_u$ ) are partitioned into two sets  $A_k$  and  $B_k$  (and  $A_u$  and  $B_u$ ) with  $|A_k| = |A_u| = n$  and  $|B_k| = |B_u| = d$ .
- The graph induced by  $A_k$  is a clique, and similarly the graph induced by  $A_u$  is a clique.
- The graphs induced by  $B_k$  and  $B_u$  are copies of a special graph with certain nice properties which enable our reduction to work. The existence of such a graph is proved (in Lemma 3.3) using a probabilistic argument.
- Finally, for the cross edges between  $A_k$  and  $B_k$  (and  $A_u$  and  $B_u$ ), we have: there is an edge between the  $i$ -th vertex of  $A_k$  (or  $A_u$ ) and the  $j$ -th vertex of  $B_k$  (or  $B_u$ ) if and only if the  $j$ -th coordinate of the  $i$ -th element of  $S_k$  (or  $S_u$ ) is 1.
- Finally, a random permutation  $\pi$  is applied to the vertices of  $G_u$ .

The permutation  $\pi$  is not known to the GI-tester. Note that we can construct  $G_k$  explicitly as  $S_k$  is known. However, that is not the same with  $G_u$  as  $S_u$  is unknown. But since we know the permutation  $\pi$ , any query to the adjacency matrix of the graph  $G_u$  can be answered by a single query to one bit of  $S_u$ . But unfortunately we don't have query access to  $S_u$ , and only have sample access to  $S_u$ . To deal with this problem, it is easier to consider a slightly more powerful query. Say, the GI-tester wants to query the  $(i, j)$ -th bit of the graph  $G_u$ . Of course, if both  $i$  and  $j$  are in  $A_u$  or both are in  $B_u$ , we can answer without even sampling from  $S_u$ . But if  $i$  is in  $A_u$  and  $j$  is in  $B_u$ , then what we intend to do is to give the whole neighborhood of  $i$  in  $B_u$  as the answer to the query. This would be like neighbourhood query in a bipartite graph. But the question remains: how do we intend to answer the query by sampling. The key observation here is that since the GI-tester does not know the permutation  $\pi$  that was applied to the vertices in  $G_u$ , to its eye, all the vertices that have not been touched so far look same. So, every time it queries for  $(i, j)$ , where  $i \in A_u$  and  $j \in B_u$ , either of the two cases can happen:

- Either, previously a query of the form  $(i, j_1)$  was asked where  $j_1$  is also in  $B_u$ , but in that case, it must have already got the answer of  $(i, j)$  as we must have given all the neighbors of  $i$  in  $B_u$ . So in that case, we can give back the same answer without sampling.
- Or, previously  $i$  did not participate in any query of the form  $(i, j_1)$  where  $j_1$  is in  $B_u$ . In this case, to the GI-tester's eye,  $i$  is just a new vertex from  $A_u$ . We can then sample **without** replacement from  $S_u$  and whatever sample of the multi-set we have, we can assume that it is the element  $i$  and answer accordingly. Note that this is the exact place where sampling **without** replacement is crucial.

To complete our proof, we need to prove how the GI-distance between  $G_k$  and  $G_u$  is connected to the EMD between  $S_k$  and  $S_u$ . Consider the set  $\Phi$  of all SPECIAL bijections from  $V(G_k)$  to  $V(G_u)$  that maps  $A_k$  into  $A_u$  and  $B_k$  into  $B_u$  such that the  $i$ -th vertex of  $B_k$  is mapped to the  $i$ -th vertex of  $B_u$ . Observe that  $d_\Phi(G_k, G_u) = 2 \cdot \text{EMD}(S_k, S_u)$ , where  $d_\Phi(G_k, G_u) = \min_{\phi \in \Phi} d_\phi(G_k, G_u)$  (See [10], Lemma 3.5 for a formal proof). The factor 2 is because of the way we define  $d_\phi(G_k, G_u)$  (See Definition 1.1). This implies that tolerant isomorphism testing between  $G_k$  and  $G_u$  is at least as hard as tolerant EMD testing between  $S_k$  and  $S_u$  if we restrict the bijection from  $V(G_k)$  to  $V(G_u)$  to be a SPECIAL bijection. The reduction works for all possible bijections, because of the careful choice of the subgraph of  $G_k$  (and  $G_u$ ) induced by  $B_k$  (and  $B_u$ ), thus ensuring  $d(G_k, G_u)$  is close to  $d_\Phi(G_k, G_u)$  (See [10] Lemma 3.6 for a formal proof).

One might compare our proof technique to the lower bound proof of (non-tolerant) testing of GI from [13]. In [13],  $\Omega(\sqrt{n})$  lower bound was proved directly (using Yao's lemma) by constructing two distributions of YES instances and NO instances - the construction of the YES and NO instances were inspired from the tightness of the birthday paradox, which was also the core idea behind the lower bound proof of the equivalence testing of two probability distributions. But, there was no direct reduction from GI testing to equivalence testing of two probability distributions. But in our lower bound proof, we establish a direct reduction to estimating EMD of multi-sets on the Hamming cube with access to samples **without** replacement. This can be of much importance, mainly while considering other models of computation, like in the communication model. From our reduction, we can obtain an alternative proof of  $\Omega(\sqrt{n})$  lower bound for the (non-tolerant) GI testing via the  $\Omega(\sqrt{n})$  lower bound of the equivalence testing of distributions, as pointed out in Corollary 1.7.

## 2.2 Reduction from tolerant graph isomorphism to tolerant EMD testing (Upper bound part of Theorem 1.4)

Given a known graph  $G_k$  and query access to an unknown graph  $G_u$  (both on  $n$  vertices), we present an algorithm for tolerant testing of graph isomorphism between  $G_k$  and  $G_u$  by using a tolerant EMD tester (for distributions over  $H$ ) as a blackbox. Note that this will prove the upper bound part of Theorem 1.4.

**Algorithm for tolerant graph isomorphism using algorithm for tolerant EMD testing as a black box:**

Our testing algorithm is inspired by the algorithm of Fischer and Matsliah [13] for non-tolerant GI testing. But our algorithm significantly differs from that of Fischer-Matsliah in some crucial points. As we explain the high level picture of our algorithm, we will point out some of the crucial differences.

We split our algorithm into three phases. In Phase 1, we first choose a  $\mathcal{O}\left(\frac{1}{\gamma_2 - \gamma_1}\right)$  size collection of random subset of vertices, i.e. *coresets*  $C_u$  from the unknown graph  $G_u$  where each  $C_u \in \mathcal{C}_u$  is of size  $\mathcal{O}(\log n)$ . Thereafter we find all embeddings of  $C_u$  inside the known graph  $G_k$ . Let the embeddings be  $\eta_1, \eta_2, \dots, \eta_J$  where  $C_k^i = \eta_i(C_u)$ . Now each  $C_u$  (as well as each  $C_k^i$ ) defines a label distribution of the vertices of  $G_u$  (as well as  $G_k$ ). Let us denote the set of labels as  $X_{C_u}$  (and  $Y_{C_k^i}$ ). Now we test if the EMD between  $X_{C_u}$  and  $Y_{C_k^i}$  is close or far for each  $i \in [J]$  (See Claim 4.2). We keep only those  $(C_u, \eta_i)$  for Phase 2 such that  $EMD(X_{C_u}, Y_{C_k^i}) \leq (\gamma_1 + \frac{\gamma_2 - \gamma_1}{2000}) n |C_u|$ .

Although Phase 1 of our algorithm is similar to the algorithm of [13], there is a striking difference. Since the authors of [13] were testing the non-tolerant version of graph isomorphism, they were testing the identity of the label distributions of  $X_{C_u}$  and  $Y_{C_k^i}$ . However, since we are solving the tolerant version of the problem, we need to allow some error among the label distributions. We need to pass only those placements of  $C_u$  that under *good bijections* do not produce much error and testing of tolerant EMD fits exactly for this purpose. It is worth noting that Fischer-Matsliah uses an equivalence tester in their algorithm to identify the placements that do not produce “any” error. But, the proof of correctness of the algorithm would not go through even if we use the tolerant testing of the equivalence of distributions. The use of EMD in this phase is crucial for the proof of correctness of our algorithm to hold.

In Phase 2, we choose  $\mathcal{O}\left(\frac{\log^2 n}{(\gamma_2 - \gamma_1)^3}\right)$  many vertices from the unknown graph  $G_u$  randomly and call it  $W$ . We further find the labels of all the vertices of  $W$  under  $C_u$ -labelling by querying the corresponding entries of  $G_u$  for each  $C_u$  that has passed Phase 1. Then we try to match the vertices of  $W$  to the set of all possible labels  $\{l_1, l_2, \dots, l_t\}$  of the vertices of  $G_k$  under  $C_k^i$ -labelling where  $C_k^i = \eta_i(C_u)$ , for those  $\eta_i$  that have passed Phase 1. Ideally, we would like to find a mapping  $\psi : W \rightarrow \{l_1, l_2, \dots, l_t\}$  such that the total distance between the labels of the matched vertices is not too large. If no such  $\psi$  is possible, we reject the current embedding and try some other embedding that has passed Phase 1.

In Phase 3, we construct a random partial bijection  $\hat{\phi} : W \rightarrow V(G_k)$  that maps the vertices of  $W$  to the vertices of  $G_k$  while preserving the labels according to  $\psi$ . We achieve this by mapping each  $w \in W$  to one vertex of  $G_k$  randomly that has same label as determined by  $\psi$ . Finally, we randomly pair the vertices of  $W$  and find the fraction of edge mismatches between the paired up vertices of  $W$  and  $\hat{\phi}(W)$ . If this fraction is at most  $5\gamma_1 + \frac{3}{5}(\gamma_2 - \gamma_1)$ , we accept and say that  $G_u$  and  $G_k$  are  $\gamma_1$ -close. If there is no such embedding of any  $C_u \in \mathcal{C}_u$  that achieves this, we report that  $G_u$  and  $G_k$  are  $\gamma_2$ -far.

The proofs of completeness and soundness follow kind of similar route as Fischer-Matsliah’s proof but the arguments are way more complicated. Many things that were trivial or obvious

in the non-tolerant setting become major hurdles in the tolerant setting, and we overcome them with significantly difficult technical arguments. The proofs are present in the full version of the paper [10].

### 3 Tolerant graph isomorphism is as hard as tolerant EMD testing

In this section, we prove that it is necessary to perform  $\Omega(\text{QWOR}_{\text{EMD}}(n))$  many queries to the adjacency matrix of  $G_u$  to solve  $(\gamma_1, \gamma_2)$ -tolerant GI testing of  $G_k$  and  $G_u$ .

► **Theorem 3.1** (Restatement of the lower bound part of Theorem 1.4). *Let  $G_k$  be the known and  $G_u$  be the unknown graph on  $n$  vertices, where  $n \in \mathbb{N}$  is sufficiently large. There exists a constant  $\epsilon_{\text{ISO}} \in (0, 1)$  such that for any given constants  $\gamma_1, \gamma_2$  with  $0 < \gamma_1 < \gamma_2 < \epsilon_{\text{ISO}}$ , any algorithm that decides whether the graphs are  $\gamma_1$ -close or  $\gamma_2$ -far, requires  $\text{QWOR}_{\text{EMD}}(n)$  adjacency queries to the unknown graph  $G_u$  where  $\text{QWOR}_{\text{EMD}}$  is as defined in Definition 1.3.*

In Section 2.1, we have discussed an overview of our idea to prove the above theorem. To prove Theorem 3.1, we show a reduction from tolerant GI testing to tolerant EMD testing over multi-sets when we have samples **without** replacement from the unknown multi-set.

► **Lemma 3.2.** *Suppose there is a constant  $\epsilon_0 \in (0, \frac{1}{2})$  such that for all constants  $\gamma_1, \gamma_2$  with  $0 < \gamma_1 < \gamma_2 < \epsilon_0$  and any constant  $T \in \mathbb{N}$ , the following holds: There exists a  $(\gamma_1, \gamma_2)$ -tolerant tester for GI that, given a known graph  $G_k$  and an unknown graph  $G_u$  with  $|V(G_u)| = |V(G_k)| = (T+1)n$ , can distinguish whether  $d(G_u, G_k) \leq \gamma_1 T n^2$  or  $d(G_u, G_k) \geq \gamma_2 T n^2$  by performing  $Q$  adjacency queries to  $G_u$ .*

*Then, for any constants  $\beta_1$  and  $\beta_2$  with  $0 < \beta_1 < \beta_2 < \frac{\epsilon_0}{2}$ , the following holds where  $\kappa = \frac{\beta_2 - \beta_1}{8}$  and  $T_\kappa = \lceil \frac{30}{\kappa(2-\kappa)} \rceil$ . There is a tolerant tester for EMD such that, given a known and an unknown multi-set  $S_k$  and  $S_u$  respectively, of the Hamming cube  $\{0, 1\}^{T_\kappa n}$  with  $|S_k| = |S_u| = n$ , can distinguish whether  $\text{EMD}(S_k, S_u) \leq \beta_1 T_\kappa n^2$  or  $\text{EMD}(S_k, S_u) \geq \beta_2 T_\kappa n^2$  with  $Q$  many samples **without** replacement from  $S_u$ .*

► **Remark 1.** Observe that Lemma 3.2 talks about tolerant EMD testing between multi-sets with  $n$  elements over a Hamming cube of dimension  $T_\kappa n$ . But Theorem 3.1 states the lower bound of  $\text{QWOR}_{\text{EMD}}(n)$ , that is, of tolerant *EMD* testing of multi-sets with  $n$  elements over a Hamming cube of dimension  $n$ . However, the query complexity of *EMD* testing increases with the dimension of the Hamming cube (See Proposition B.9). So, we will be done with the proof of Theorem 3.1 by proving Lemma 3.2.

#### 3.1 Tolerant GI to Tolerant EMD testing: Proof of Lemma 3.2

To define the necessary reduction for the proof of Lemma 3.2, we need to show the existence of a graph  $G_p$  satisfying some unique properties.

► **Lemma 3.3.** *Let  $\kappa \in (0, 1)$  and  $s \geq 3$  be given constants. Then for  $C_{\kappa, s} = \lceil \frac{6s}{\kappa(2-\kappa)} \rceil$  and sufficiently large  $n \in \mathbb{N}$ <sup>8</sup>, there exists a graph  $G_p$  with  $C_{\kappa, s} n$  many vertices such that the following conditions hold.*

- (i) *The degree of each vertex in  $G_p$  is at least  $((1-\kappa)C_{\kappa, s} + 1)n - 1$ .*
- (ii) *The cardinality of symmetric difference between the sets of neighbors of any two (distinct) vertices in  $G_p$  is at least  $sn - 2$ .*

<sup>8</sup> The lower bound of  $n$  is a constant that depends on  $\kappa$  and  $s$ .

The proof of Lemma 3.3 uses probabilistic method (See [10] for the proof). Let  $ALG(\gamma_1, \gamma_2, T)$  be the algorithm that takes  $\gamma_1$  and  $\gamma_2$  with  $0 < \gamma_1 < \gamma_2 < \epsilon_0$  as input and decides whether  $d(G_k, G_u) \leq \gamma_1 T n^2$  or  $d(G_k, G_u) \geq \gamma_2 T n^2$ , where  $|V(G_k)| = |V(G_u)| = (T + 1)n$ . Now we show that for any two constants  $\beta_1$  and  $\beta_2$  with  $0 < \beta_1 < \beta_2 < \frac{\epsilon_0}{2}$ ,  $\kappa = \frac{\beta_2 - \beta_1}{8}$  and  $T_\kappa = \lceil \frac{6s}{\kappa(2-\kappa)} \rceil$ , there exists an algorithm  $\mathcal{A}(\beta_1, \beta_2, \kappa, T_\kappa)$  that can test whether two multi-sets  $S_k$  and  $S_u$  over the  $T_\kappa n$ -dimensional Hamming cube have EMD less than  $T_\kappa \beta_1 n^2$  or more than  $T_\kappa \beta_2 n^2$  with  $Q$  many queries to the multi-set  $S_u$ . To be specific, algorithm  $\mathcal{A}(\beta_1, \beta_2, \kappa, T_\kappa)$  for EMD testing will use algorithm  $ALG(\gamma_1, \gamma_2, T)$  for  $(\gamma_1, \gamma_2)$ -tolerant GI such that  $\gamma_1 = 2\beta_1$ ,  $\gamma_2 = 2\beta_2 - 2\kappa$  and  $T = T_\kappa$ . Note that, as  $0 < \beta_1 < \beta_2 < \frac{\epsilon_0}{2}$  and  $\kappa = \frac{\beta_2 - \beta_1}{8}$ ,  $0 < \gamma_1 < \gamma_2 < \epsilon_0$  holds. The details of the reduction, that is, algorithm  $\mathcal{A}$  is described below. Because of space constraint, we are not presenting the proof of correctness of the reduction in this extended abstract. Please refer to our full version [10].

### Description of the reduction

**Input:** A known multi-set  $S_k = \{k_1, \dots, k_n\}$  over  $H_{T_\kappa n} = \{0, 1\}^{T_\kappa n}$  and query access to an unknown multi-set  $S_u = \{u_1, \dots, u_n\}$  over  $H_{T_\kappa n}$ .

**Goal:** To decide whether  $EMD(S_k, S_u) \leq T_\kappa \beta_1 n^2$  or  $EMD(S_k, S_u) \geq T_\kappa \beta_2 n^2$ .

**Construction of  $G_k$  and  $G_u$  from  $S_k$  and  $S_u$ :** Let us first construct the graph  $G_k$  from  $S_k$ .  $G_k$  has  $(T_\kappa + 1)n$  vertices partitioned into two parts  $A_k = \{a_1, \dots, a_n\}$  and  $B_k = \{b_1, \dots, b_{T_\kappa n}\}$ . Now the edges of  $G_k$  are described as follows:

- $G_k[A_k]$  is a clique with  $n$  vertices.
- $G_k[B_k]$  is a copy of the graph  $G_p(V_p, E_p)$  on  $T_\kappa n$  vertices as stated in Lemma 3.3 with parameters  $s = 5$ ,  $\kappa = \frac{\beta_2 - \beta_1}{8}$  and  $T_\kappa = C_{\kappa, 5}$ .
- For the cross edges between the vertices in  $A_k$  and  $B_k$ , we add the edge  $(a_i, b_j)$  to  $E(G_k)$  if and only if the  $j$ -th coordinate of  $k_i$  is 1 for all  $i \in [n]$  and  $j \in [T_\kappa n]$ .

Note that the graph  $G_k$  constructed above is unique for a given multi-set  $S_k$ . The graph  $G_u$  with the vertex sets  $A_u = \{a'_1, \dots, a'_n\}$  and  $B_u = \{b'_1, \dots, b'_{T_\kappa n}\}$  is constructed from the multi-set  $S_u$  in a similar fashion, but at the end, the vertices of  $A_u$  are permuted using a random permutation. So,

- $G_u[A_u]$  is a clique with  $n$  vertices.
- $G_u[B_u]$  is a copy of the graph  $G_p(V_p, E_p)$  on  $T_\kappa n$  vertices as stated in Lemma 3.3, with parameters  $s = 5$ ,  $\kappa = \frac{\beta_2 - \beta_1}{8}$  and  $T_\kappa = C_{\kappa, 5}$ .
- Let us first pick a random permutation  $\pi$  on  $[n]$ . For the cross edges between the vertices in  $A_u$  and  $B_u$ , we add the edge  $(a'_{\pi(i)}, b_j)$  to  $E(G_u)$  if and only if the  $j$ -th coordinate of  $u_i$  is 1 for all  $i \in [n]$  and  $j \in [T_\kappa n]$ .

Note that our final objective is to prove a lower bound on the query complexity for tolerant testing of GI, that is, when we have an adjacency query access to  $G_u$ . We will instead show that the lower bound holds even if we have the following query access, named as  *$A_u$ -neighborhood-query*: the tester can choose a vertex  $a'_i \in A_u$  and in one go obtain the information about the entire neighborhood of  $a'_i$  in  $B_u$ .

Observe that the only part of  $G_u$  that is not known to the tester is the cross edges between  $A_u$  and  $B_u$ . So, in this case, the  $A_u$ -neighborhood query is way more stronger than the standard queries to  $G_u$ , and a lower bound for the  $A_u$ -neighborhood query would imply a lower bound on adjacency query.

### Simulating Queries to $G_u$ by samples drawn from $S_u$ *without* replacement

Following the above discussion, we will only have to show how to simulate  $A_u$ -neighborhood queries using samples drawn from  $S_u$  **without** replacement. So, we can assume that the queries are of the form: *what are the neighbors of  $a'_i$  in  $B_u$ ?* And since in each query the entire neighborhood of  $a'_i$  is obtained, the tester would pick different  $a'_i$  for every query. Note that in  $G_u$ , by construction, the vertices of  $A_u$  were permuted using a random permutation. So, from the point of view of the tester, the  $a'_i$  are just randomly drawn from  $A_u$  minus the set of  $a'_i$  already queried. In other word, the  $a'_i$  are just randomly drawn from  $A_u$  **without** replacement. Now because of the way the edges between  $A_u$  and  $B_u$  are constructed, the neighborhood of a random  $a'_i$  drawn from  $A_u$  **without** replacement is same as obtaining random samples from  $S_u$  **without** replacement. It is also important to note that because of the randomness, the queries made by the tester are actually non-adaptive.

### Description of algorithm $\mathcal{A}$ for testing $EMD(S_k, S_u)$

Run ALG on  $G_k$  and  $G_u$  with parameters  $\gamma_1 = 2\beta_1$  and  $\gamma_2 = 2\beta_2 - 2\kappa$ . If ALG reports  $d(G_k, G_u) \leq T_\kappa \gamma_1 n^2$ , output that  $EMD(S_k, S_u) \leq T_\kappa \beta_1 n^2$ . Similarly, if ALG reports that  $d(G_k, G_u) \geq T_\kappa \gamma_2 n^2$ , then output  $EMD(S_k, S_u) \geq T_\kappa \beta_2 n^2$ .

## 4 Tolerant EMD testing is as hard as tolerant graph isomorphism testing

In this section, we prove the following theorem, that discusses about algorithm for tolerant graph isomorphism testing with a blackbox access to tolerant EMD testing over multi-sets.

► **Theorem 4.1** (Restatement of the upper bound part of Theorem 1.4). *Let  $G_k$  and  $G_u$  be the known and unknown graphs, respectively. There exists an algorithm that takes parameters  $\gamma_1$  and  $\gamma_2$  as input such that  $0 \leq \gamma_1 < \gamma_2 \leq 1$ , performs  $\tilde{\mathcal{O}}(\text{QWoREMD}(n))$  many queries to the adjacency matrix of  $G_u$  for appropriate  $\beta_1$  and  $\beta_2$  depending on  $\gamma_1$  and  $\gamma_2$ , and decides whether  $d(G_u, G_k) \leq \gamma_1 n^2$  or  $d(G_u, G_k) \geq \gamma_2 n^2$ , with probability at least  $2/3$ . Here  $\tilde{\mathcal{O}}(\cdot)$  hides a polynomial factor in  $\frac{1}{\beta_2 - \beta_1}$  and  $\log n$ .*

► **Remark 2.** The theorem stated above works for any  $\gamma_1, \gamma_2$  such that  $0 \leq \gamma_1 < \gamma_2 \leq 1$ . However, for simplicity of representation, we have assumed  $\gamma_2 \geq 11\gamma_1$ .

► **Remark 3.** Note that Theorem 4.1 can also be stated in terms of  $\text{QWR}_{\text{EMD}}(n)$  as  $\text{QWoREMD}(n) \leq \text{QWR}_{\text{EMD}}(n)$  as we can simulate samples **with** replacement when we have query access to samples **without** replacement (See Proposition B.5).

Our algorithm for tolerant GI testing, as stated in Theorem 4.1, uses a special kind of tolerant  $EMD$  tester over multi-sets: we know  $t$  many multi-sets, one multi-set is unknown and two parameters  $\epsilon_1$  and  $\epsilon_2$  are given; the objective is to test tolerant  $EMD$  of each known multi-set with the unknown one. The following theorem gives us the special  $EMD$  tester.

► **Theorem 4.2.** *Let  $H = \{0, 1\}^n$  be a  $n$ -dimensional Hamming cube. Let  $\{S_k^i : i \in [t]\} \cup \{S_u\}$  denote the multi-sets with  $n$  elements from  $H$  where  $\{S_k^i : i \in [t]\}$  denote the set of  $t$  many known multi-sets and  $S_u$  denotes the unknown multi-set. There exists an algorithm ALG-EMD that takes two proximity parameters  $\epsilon_1, \epsilon_2$  with  $0 \leq \epsilon_1 < \epsilon_2 \leq 1$  and a  $\delta \in (0, 1)$  as input and decides whether  $EMD(S_u, S_k^i) \leq \epsilon_1 n^2$  or  $EMD(S_u, S_k^i) \geq \epsilon_2 n^2$ , with probability at least  $1 - \delta$ , for each  $i \in [t]$ . Moreover, ALG-EMD uses  $\text{QWoREMD}(n) \cdot \mathcal{O}(\log \frac{t}{\delta})$  many samples **without** replacement from  $S_u$ .*

The above theorem follows from the definition of  $\text{QWor}_{\text{EMD}}(n)$  (See Definition 1.3) along with union bound and standard argument for amplifying the success probability.

► **Remark 4.** The algorithm of Theorem 4.1, to be discussed in Section 4.1, formulates a tolerant *EMD* instance of multi-sets having  $n$  elements in  $H = \{0, 1\}^d$ , where  $d = \mathcal{O}(\log n / (\gamma_2 - \gamma_1))$ . But ALG-EMD is an algorithm for tolerant *EMD* testing between two multi-sets having  $n$  elements in  $\{0, 1\}^n$ . This is not a problem as the query complexity of *EMD* is an increasing function in dimension (See Proposition B.9 in Appendix B). Moreover, the algorithm in Section 4.1 calls ALG-EMD with parameters  $\epsilon_1 = (\gamma_1 + \frac{\gamma_2 - \gamma_1}{2000})$ ,  $\epsilon_2 = \gamma_2/5$ ,  $t = 2^{\mathcal{O}(\log^2 n / (\gamma_2 - \gamma_1))}$  and  $\delta$  is a suitable constant depending upon  $\gamma_1$  and  $\gamma_2$ , where  $\gamma_1$  and  $\gamma_2$  are parameters as stated in Theorem 4.1. So, each call to ALG-EMD, in our context, makes  $\tilde{\mathcal{O}}(\text{QWor}_{\text{EMD}}(n))$  many queries.

#### 4.1 Algorithm for tolerant graph isomorphism testing

For our algorithm, we need the following definitions of *label* and *embedding*.

► **Definition 4.3.** (*Label of a vertex*) Given a graph  $G$  and  $C \subset V(G) = \{c_1, \dots, c_{|C|}\}$ , the  $C$ -labelling of  $V(G)$  is a function  $\mathcal{L}_C : V(G) \rightarrow \{0, 1\}^{|C|}$  such that the  $i$ -th entry of  $\mathcal{L}_C(v)$  is 1 if and only if  $v$  is a neighbor of  $c_i \in C$ . Also,  $\mathcal{L}_C(v)$  is referred as the label of  $v$  under  $C$ -labelling of  $V(G)$ .

► **Definition 4.4.** (*Embedding of a Vertex Set into another Vertex Set*) Let  $G_u$  and  $G_k$  be two graphs. Consider  $A \subseteq V(G_u)$  and  $B \subseteq V(G_k)$  such that  $|A| \leq |B|$ . An injective mapping  $\eta$  from  $A$  to  $B$  is referred as an *embedding* of  $A$  into  $B$ .

Now we present our query algorithm **TolerantGI**( $G_u, G_k, \gamma_1, \gamma_2$ ) that comprises three phases. The technical overview of the algorithm is already presented in Section 2.2

#### Formal Description of TolerantGI( $G_u, G_k, \gamma_1, \gamma_2$ ):

The three phases of our algorithm are as follows:

##### 4.1.1 Phase 1

The first phase of our algorithm consists of the following three steps.

**Step 1** First we sample a collection  $\mathcal{C}_u$  of  $\mathcal{O}(\log n)$  sized random subsets of  $V(G_u)$  with  $|\mathcal{C}_u| = \mathcal{O}(\frac{1}{\gamma_2 - \gamma_1})$ . We perform **Step 2** and **Step 3** for each  $C_u \in \mathcal{C}_u$ .

**Step 2** We determine all possible embeddings, that is,  $\eta_1, \dots, \eta_J$ , of  $C_u$  into  $V(G_k)$ , where  $J = \binom{n}{\mathcal{O}(\log n)} \leq 2^{\mathcal{O}(\log^2 n)}$ . For each  $i \in [J]$ , let  $C_k^i$  be the set of images of  $C_u$  under the  $i$ -th embedding of  $C_u$  into  $V(G_k)$ , that is,  $C_k^i = \eta_i(C_u)$ . For all  $i \in [J]$ , we construct the multi-set  $Y_{C_k^i}$  that contains  $C_k^i$ -labellings of all the vertices of  $G_k$ .

**Step 3** Now for each vertex  $v \in V(G_u)$ , there is a  $C_u$ -labelling of  $v$ . Let  $X_{C_u}$  be the multi-set of  $C_u$ -labellings of all the vertices in  $V(G_u)$ . However,  $X_{C_u}$  is unknown to the algorithm. We call ALG-EMD (as stated in Theorem 4.2) by setting parameters as described in Remark 4 to decide whether  $\text{EMD}(X_{C_u}, Y_{C_k^i}) \leq (\gamma_1 + \frac{\gamma_2 - \gamma_1}{2000})n|C_u|$  or  $\text{EMD}(X_{C_u}, Y_{C_k^i}) \geq \gamma_2 n|C_u|/5$ , for each  $i \in [J]$ . Let us pair up  $C_u$ 's and their accepted embeddings into  $G_k$  and call the set  $\Gamma$ , that is,

$$\Gamma = \left\{ (C_u, \eta_i) \mid \text{ALG-EMD decides } \text{EMD}(X_{C_u}, Y_{C_k^i}) \leq (\gamma_1 + \frac{\gamma_2 - \gamma_1}{2000})n|C_u| \right\}.$$



### 4.1.2 Phase 2

In the second phase, the algorithm performs the following two steps.

**Step 1** We sample a subset  $W$  of  $\mathcal{O}(\log^2 n / (\gamma_2 - \gamma_1)^3)$  vertices randomly from  $G_u$ .

**Step 2** For each  $(C_u, \eta_i) \in \Gamma$  that has passed **Phase 1**, we perform the following steps:

- (i) We find the  $C_k^i = \eta_i(C_u)$ -labelling of the vertices of  $G_k$ . Let  $l_1, \dots, l_t$  be the labels of the vertices where  $t = 2^{\lfloor C_k^i \rfloor}$  and  $V_j \subseteq V(G_k)$  be the set of vertices with label  $l_j$ .
- (ii) We define a matrix  $M$  of size  $|W| \times 2^{\lfloor C_k^i \rfloor}$  where each row represents the label of a vertex  $w \in W$  and each column represents one of the possible  $C_k^i$ -labelling of  $V(G_k)$ <sup>9</sup>. The  $(i, j)$ -th entry of  $M$  is defined as:  $M_{ij} = d_H(\mathcal{L}_{C_u}(w_i), l_j)$ .
- (iii) We choose a function  $\psi : W \rightarrow \{l_1, \dots, l_t\}$  randomly satisfying

$$\sum_{w \in W} d_H(\mathcal{L}_{C_u}(w), \psi(w)) \leq \frac{2\gamma_2}{5} |C_u| |W| \text{ and } |\{w : \psi(w) = l_j\}| \leq |V_j| \forall j \in [t]. \quad (1)$$

Let  $\Gamma_W$  be the set of tuples such that

$$\Gamma_W = \{(C_u, \eta_i, \psi) : (C_u, \eta_i) \in \Gamma \text{ and } \psi \text{ satisfies Equation (1)}\}.$$

### 4.1.3 Phase 3

The third phase of our algorithm comprises the following four steps.

**Step 1** We randomly pair up the vertices of  $W$ . Let  $\{(a_1, b_1), \dots, (a_p, b_p)\}$  be the pairs of the vertices, where  $p = \mathcal{O}(\log^2 n / (\gamma_2 - \gamma_1)^3)$ . We now determine which  $(a_i, b_i)$  pairs form edges in  $G_u$  by querying the corresponding entries of the adjacency matrix of  $G_u$ .

**Step 2** For each  $(C_u, \eta_i, \psi) \in \Gamma_W$  that has passed **Phase 2**, we perform **Step 3** and **Step 4** as follows.

**Step 3** We choose an embedding  $\hat{\phi} : W \rightarrow V(G_k)$  randomly, satisfying  $\hat{\phi}(w) \in V_j$  if and only if  $\psi(w) = l_j$  and modulo permutation of the vertices in  $V_j$  for all  $j \in [t]$ . In other words, we map each  $w \in W$  to a vertex in  $G_k$  randomly having  $\psi(w) = l_j$  as its  $C_k^i$ -labelling in  $G_k$ .

**Step 4** We find the fraction  $\zeta(C_u, \eta_i, \psi, \hat{\phi}) = |\{(a_i, b_i) : \mathbb{1}_{(a_i, b_i)} = 1\}| / p$ , where  $\mathbb{1}_{(a_i, b_i)} = 1$  if exactly one among  $(a_i, b_i) \in E(G_u)$  and  $(\hat{\phi}(a_i), \hat{\phi}(b_i)) \in E(G_k)$  holds.

If  $\zeta(C_u, \eta_i, \psi, \hat{\phi}) \leq 5\gamma_1 + \frac{3}{5}(\gamma_2 - \gamma_1)$ , then **HALT and REPORT** that  $G_u$  and  $G_k$  are  $\gamma_1$ -close.

While executing **Step 3** and **Step 4** for each tuple in  $\Gamma_W$ , if we did not **HALT**, then we **HALT** now and **REPORT** that  $G_u$  and  $G_k$  are  $\gamma_2$ -far.

## 5 Conclusion

In this paper, we proved that the query complexity of tolerant GI testing between a known graph  $G_k$  and an unknown graph  $G_u$  is the same as (up to polylogarithmic factor) tolerant testing of  $EMD$  between a known multi-set  $S_k$  and an unknown multi-set  $S_u$  when we have

<sup>9</sup> Let  $C_u = \{x_1, \dots, x_{\mathcal{O}(\log n / (\gamma_2 - \gamma_1))}\}$ . Note that for each  $w_i \in W$ ,  $\mathcal{L}_{C_u}(w_i) \in \{0, 1\}^{\mathcal{O}(\log n / (\gamma_2 - \gamma_1))}$  such that the  $j$ -th coordinate is 1 if and only if  $w_i$  is a neighbour of  $x_j$ , where  $i \in [\mathcal{O}(\log^2 n / (\gamma_2 - \gamma_1)^3)]$  and  $j \in [\mathcal{O}(\log n / (\gamma_2 - \gamma_1))]$ . Similarly,  $l_j \in \{0, 1\}^{\mathcal{O}(\log n / (\gamma_2 - \gamma_1))}$  such that the  $i$ -th coordinate of  $l_j$  is 1 if and only if  $\eta(x_i)$  is a neighbour of  $v \in V_j$ , where  $j \in [2^{\lfloor C_k^i \rfloor}]$ .

samples **without** replacement from  $S_u$ . In Lemma B.10, we have shown that the sample complexity of testing of  $EMD$  between a known multi-set  $S_k$  and an unknown multi-set  $S_u$  when we have samples **with** replacement from  $S_u$  is  $\Omega(n/\log n)$ . Thus the natural open question is

*What is the query complexity of tolerant  $EMD$  testing when we have samples **without** replacement from the unknown multi-set?*

As mentioned before, it is interesting to note that our lower bound proof is via a *pure reduction* from tolerant graph isomorphism to tolerant testing of  $EMD$  of multi-sets over the Hamming cube using samples **without** replacement. Using our lower bound technique (and Proposition B.7), we can get an alternative proof of Fischer and Matsliah’s lower bound result for testing non-tolerant graph isomorphism [13]. Our upper bound proof is also a pure reduction from tolerant testing of  $EMD$  of multi-sets over the Hamming cube to tolerant graph isomorphism problem. Thus our reductions also hold for other computational models such as the communication complexity model. So, in the communication model (that is, when Alice and Bob have graphs  $G_a$  and  $G_b$  respectively and they want to estimate the GI-distance between them), the amount of bits of communication is same (up to a polylogarithmic factors) to the problem of estimating the  $EMD$  between two distributions over Hamming cube, where Alice and Bob have access to one distribution each. The question we would like to pose is:

*What is the randomized communication complexity of testing tolerant graph isomorphism problem?*

Fischer and Matsliah [13] studied the non-tolerant version of the graph isomorphism problem in two scenarios: (i) one graph is known and the other graph is unknown, (ii) both the graphs are unknown. They resolved the query complexity of (i), whereas Onak and Sun [19] resolved (ii). With this paper, we initiate the study of tolerant graph isomorphism problem in the query and communication world. So, another natural open question to look for is:

*What is the query complexity of tolerant graph isomorphism when both the graphs are unknown?*

---

## References

- 1 Jayadev Acharya, Constantinos Daskalakis, and Gautam Kamath. Optimal testing for properties of distributions. *arXiv preprint arXiv:1507.05952*, 2015.
- 2 Alexandr Andoni, Khanh Do Ba, Piotr Indyk, and David Woodruff. Efficient sketches for earth-mover distance, with applications. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 324–330. IEEE, 2009.
- 3 Alexandr Andoni, Piotr Indyk, and Robert Krauthgamer. Earth mover distance over high-dimensional spaces. In *SODA*, volume 8, pages 343–352, 2008.
- 4 Alexandr Andoni, Robert Krauthgamer, and Ilya Razenshteyn. Sketching and embedding are equivalent for norms. *SIAM Journal on Computing*, 47(3):890–916, 2018.
- 5 László Babai. Graph Isomorphism in Quasipolynomial Time. In *Proceedings of the 48th Annual ACM symposium on Theory of Computing, STOC*, pages 684–697, 2016.
- 6 Laszlo Babai and Sourav Chakraborty. Property Testing of Equivalence under a Permutation Group Action. *ACM Transactions on Computation Theory (ToCT)*, 2010.

- 7 László Babai, Anuj Dawar, Pascal Schweitzer, and Jacobo Torán. The Graph Isomorphism Problem (Dagstuhl Seminar 15511). *Dagstuhl Reports*, 5(12):1–17, 2015. doi:10.4230/DagRep.5.12.1.
- 8 Tugkan Batu, Eldar Fischer, Lance Fortnow, Ravi Kumar, Ronitt Rubinfeld, and Patrick White. Testing Random Variables for Independence and Identity. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science, FOCS*, pages 442–451, 2001.
- 9 Clément L Canonne. A survey on distribution testing: Your data is big. but is it blue? *Theory of Computing*, pages 1–100, 2020.
- 10 Sourav Chakraborty, Arijit Ghosh, Gopinath Mishra, and Sayantan Sen. Interplay between graph isomorphism and earth mover’s distance in the query and communication worlds. In *Electron. Colloquium Comput. Complex.*, volume 27, page 135, 2020.
- 11 Luc Devroye and Gábor Lugosi. *Combinatorial methods in density estimation*. Springer Science & Business Media, 2012.
- 12 Khanh Do Ba, Huy L Nguyen, Huy N Nguyen, and Ronitt Rubinfeld. Sublinear time algorithms for earth mover’s distance. *Theory of Computing Systems*, 48(2):428–442, 2011.
- 13 Eldar Fischer and Arie Matsliah. Testing Graph Isomorphism. *SIAM Journal on Computing*, 38(1):207–225, 2008.
- 14 David Freedman. A remark on the difference between sampling with and without replacement. *Journal of the American Statistical Association*, 72(359):681–681, 1977.
- 15 Oded Goldreich. Testing isomorphism in the bounded-degree graph model. *Electron. Colloquium Comput. Complex.*, 26:102, 2019. URL: <https://ecc.weizmann.ac.il/report/2019/102>.
- 16 Subhash Khot and Assaf Naor. Nonembeddability theorems via fourier analysis. *Mathematische Annalen*, 334(4):821–852, 2006.
- 17 Reut Levi and Moti Medina. Distributed testing of graph isomorphism in the congest model. *arXiv preprint arXiv:2003.00468*, 2020.
- 18 Chih-Long Lin. Hardness of Approximating Graph Transformation Problem. In *Proceedings of the 5th International Symposium on Algorithms and Computation, ISAAC.*, pages 74–82, 1994.
- 19 Krzysztof Onak and Xiaorui Sun. The Query Complexity of Graph Isomorphism: Bypassing Distribution Testing Lower Bounds. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 165–171, 2018.
- 20 Liam Paninski. A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE Transactions on Information Theory*, 54(10):4750–4755, 2008.
- 21 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *Journal of Computer and System Sciences*, 72(6):1012–1042, 2006.
- 22 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.
- 23 Shashank Singh and Barnabás Póczos. Minimax distribution estimation in wasserstein distance. *arXiv preprint arXiv:1802.08855*, 2018.
- 24 Xiaorui Sun. *On the Isomorphism Testing of Graphs*. PhD thesis, Columbia University, 2016.
- 25 Gregory Valiant and Paul Valiant. The Power of Linear Estimators. In *Proceedings of the 52nd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 403–412, 2011.
- 26 Gregory Valiant and Paul Valiant. An automatic inequality prover and instance optimal identity testing. *SIAM Journal on Computing*, 46(1):429–455, 2017.
- 27 Paul Valiant. Testing Symmetric Properties of Distributions. *SIAM Journal on Computing*, 40(6):1927–1968, 2011.
- 28 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In Michael J. Fischer, Richard A. DeMillo, Nancy A. Lynch, Walter A. Burkhard, and Alfred V. Aho, editors, *Proceedings of the 11th Annual ACM Symposium on*

*Theory of Computing*, April 30 - May 2, 1979, Atlanta, Georgia, USA, pages 209–213. ACM, 1979. doi:10.1145/800135.804414.

## A Preliminaries

All graphs considered here are undirected, unweighted and have no self-loops or parallel edges. For a graph  $G(V, E)$ ,  $V(G)$  and  $E(G)$  will denote the vertex set and the edge set of  $G$ , respectively. Since we are considering undirected graphs, we write an edge  $(u, v) \in E(G)$  as  $\{u, v\}$ . The *Hamming distance* between two points  $x$  and  $y$  in a Hamming cube  $\{0, 1\}^k$  will be denoted by  $d_H(x, y)$ .

### A.1 Notion of distance between two graphs

First let us define the notion of DECIDER of a vertex and then the notion of distance between two graphs, using decider of vertices, that is conceptually same as that of GRAPH ISOMORPHISM DISTANCE defined in Definition 1.1.

► **Definition A.1.** (DECIDER of a vertex) Given two graphs  $G_k$  and  $G_u$  and a bijection  $\phi : V(G_u) \rightarrow V(G_k)$ , DECIDER of a vertex  $x \in V(G_u)$  with respect to  $\phi$  is defined as the set of vertices of  $G_u$  that create the edge difference in  $x$  and  $\phi(x)$ 's neighbourhood in  $G_u$  and  $G_k$ , respectively. Formally,

$$\text{DECIDER}_\phi(x) := \{y \in V(G_u) : \text{one of the edges } \{x, y\} \text{ and } \{\phi(x), \phi(y)\} \text{ is not present}\}$$

► **Definition A.2.** (DISTANCE between two graphs) Let  $G_u$  and  $G_k$  be two graphs and  $\phi : V(G_u) \rightarrow V(G_k)$  be a bijection from the vertex set of  $G_u$  to that of  $G_k$ . The *distance* between  $G_u$  and  $G_k$  under  $\phi$  is defined as the sum of the sizes of the deciders of all the vertices in  $G_u$ , that is,

$$d_\phi(G_u, G_k) := \sum_{x \in V(G_u)} |\text{DECIDER}_\phi(x)|.$$

The *distance* between two graphs  $G_u$  and  $G_k$  is the minimum distance under all possible bijections  $\phi$  from  $V(G_u)$  to  $V(G_k)$ , that is,  $d(G_u, G_k) := \min_{\phi} d_\phi(G_u, G_k)$ .

► **Remark 5.** Recall the definition of  $\delta_{GI}(G_u, G_k)$ , GRAPH ISOMORPHISM DISTANCE between  $G_u$  and  $G_k$ , that is given in Definition 1.1. Observe that  $d(G_u, G_k) = 2 \binom{n}{2} \delta_{GI}(G_u, G_k)$ . Though,  $d(G_u, G_k)$  and  $\delta_{GI}(G_u, G_k)$  represent the same thing, conceptually, we will do our calculations by using  $d(G_u, G_k)$  for simplicity of presentation.

Next we define the concept of closeness between two graphs.

► **Definition A.3.** (CLOSE and FAR) For  $\gamma \in [0, 1)$ , two graphs  $G_u$  and  $G_k$  with  $n$  vertices are  $\gamma$ -close to isomorphic if  $d(G_u, G_k) \leq \gamma n^2$ . Otherwise, we say  $G_u$  and  $G_k$  are  $\gamma$ -far from being isomorphic.<sup>10</sup>

<sup>10</sup>By abuse of notation, we will say  $G_u$  and  $G_k$  are  $\gamma$ -far when  $d(G_u, G_k) \geq \gamma n^2$ .

## A.2 Property Testing of Distribution Properties

Understanding different properties of probability distributions have been an active area of research in property testing (For reference, see [9]). The authors studied these problems assuming random sample access from the unknown distributions. Considering the relation between the distributions and their corresponding representative multi-sets, we can say that all these results hold for multi-sets along with access over sampling **with** replacement.

Although it seems that the change of query model from sample **with** replacement to sample **without** replacement does not make much difference, following the work of Freedman [14], we know that the variation distance between probability distributions when accessed via samples **with** and **without** replacement, becomes arbitrary close to  $1/2$  when the number of samples is  $\Omega(\sqrt{n})$ . Because of this reason, many techniques developed for sampling **with** replacement for various problems no longer work anymore. Most importantly, proving any lower bound better than  $\Omega(\sqrt{n})$  is often nontrivial.

## B Earth Mover's Distance (EMD) over Hamming Cube

In this section, we study some properties of *Earth Mover's distance (EMD)* over probability distributions and multi-sets, which are crucial in the context of both our lower and upper bound. Before proceeding to the discussion on EMD, let us first recall the definition of  $\ell_1$  distance between two distributions.

► **Definition B.1** ( $\ell_1$  distance between two distributions). Let  $p$  and  $q$  be two probability distributions over  $[n]$ . The  $\ell_1$  distance between  $p$  and  $q$  is defined as

$$d_{\ell_1}(p, q) = \sum_{i=1}^n |p(i) - q(i)|$$

► **Definition B.2** (*EMD* between two probability distributions). Let  $H = \{0, 1\}^d$  be a Hamming cube of dimension  $d$ , and  $p, q$  be two probability distributions on  $H$ . The *EMD* between  $p$  and  $q$  is denoted by  $EMD(p, q)$  and defined as the optimum solution to the following linear program:

$$\begin{aligned} & \text{Minimize} && \sum_{x, y \in H} f_{xy} d_H(x, y) \\ & \text{Subject to} && \sum_{y \in H} f_{xy} = p(x) \quad \forall x \in H, \text{ and } \sum_{x \in H} f_{xy} = q(y) \quad \forall y \in H. \end{aligned}$$

Now we define *EMD* between two multi-sets.

► **Definition B.3** (*EMD* between two multi-sets). Let  $S_1, S_2$  be two multi-sets on a Hamming cube  $H = \{0, 1\}^d$  of dimension  $d$  with  $|S_1| = |S_2|$ . The *EMD* between  $S_1$  and  $S_2$  is denoted by  $EMD(S_1, S_2)$  and defined as  $EMD(S_1, S_2) = \min_{\phi: S_1 \rightarrow S_2} \sum_{x \in S_1} d_H(x, \phi(x))$  where  $\phi$  is a bijection from  $S_1$  to  $S_2$ .

Note that an unknown distribution  $p$  is accessed by taking samples from  $p$ . However, a multi-set is accessed as follows:

► **Definition B.4** (Query accesses to multi-sets). A multi-set  $S$  of  $n$  elements is accessed in one of the following ways:

**Sample Access with replacement:** Each element of  $S$  is reported uniformly at random independent of all previous queries.

**Sample Access without replacement:** Let us assume we make  $Q$  queries to  $S$ , where  $Q \leq n$ .

The answer to the first query, say  $s_1$ , is an element from  $S$  chosen uniformly at random.

For any  $2 \leq i \leq Q$ , the answer of the  $i$ -th query is an element chosen uniformly at random from  $S \setminus \{s_1, \dots, s_{i-1}\}$ . Here  $s_j, 1 \leq j \leq Q$ , denotes the answer to the  $j$ -th query.

Although sampling **with** replacement is more natural query model, we need sampling **without** replacement for our lower bound proof. We now note that we can simulate samples **with** replacement when we have samples **without** replacement.

► **Proposition B.5** (Simulating samples **with** replacement from samples **without** replacement). *Given  $Q$  many samples **without** replacement from an unknown multi-set  $S_u$  with  $n$  elements, we can simulate  $Q$  many samples **with** replacement from  $S_u$  where  $Q \leq n$ .*

For a formal proof of the above proposition, see [10]. The following observation connects the *EMD* between two probability distributions with that of between two multi-sets.

► **Observation B.6.** Let  $p, q$  be two  $K$ -grained probability distributions<sup>11</sup> on a  $n$  dimensional Hamming cube  $H = \{0, 1\}^n$ . Then  $p$  and  $q$  induces two multi-sets  $S_1$  and  $S_2$  on  $H$ , respectively, as follows.  $S_1$  ( $S_2$ ) is the multi-set containing  $x \in H$  with multiplicity  $p(x)K$  ( $q(x)K$ ) for each  $x \in H$ . Moreover,  $EMD(p, q) = \frac{EMD(S_1, S_2)}{K}$ .

See [10] for a formal proof.

► **Remark 6.** Note that sample access from a probability distribution is exactly same as uniform sampling from a multi-set **with** replacement.

► **Proposition B.7.** *Let  $\mathcal{D}$  be the set of all multi-sets of size  $n$  over a universe  $[m]$ ; let  $S_k$  and  $S_u$  in  $\mathcal{D}$  denote the known and unknown multi-sets over  $[n]$ ; and  $\text{PROP} : \mathcal{D} \times \mathcal{D} \rightarrow \{0, 1\}$  be a boolean function. Then the following holds:*

*If there exists an algorithm that determines  $\text{PROP}$  by  $Q$  many samples **without** replacement from  $S_u$  with probability at least  $2/3$ , then there exists an algorithm that determines  $\text{PROP}$  by  $\min\{Q, \sqrt{\min\{n, m\}}\}$  many samples **with** replacement from  $S_u$  with probability at least  $2/3 - o(1)$ .*

This follows from the fact that when  $Q = o(\sqrt{n})$  and  $D_{WR}$  ( $D_{WR}$ ) be the probability distribution over all the subsets having  $Q$  elements from  $[n]$  **with** (**without**) replacement, the  $\ell_1$  distance between  $D_{WR}$  and  $D_{WR}$  is  $o(1)$ .

► **Definition B.8** (EMD over multi-sets while sampling **with** and **without** replacement). Let  $S_k$  and  $S_u$  denote the known and the unknown multi-sets, respectively, over  $n$ -dimensional Hamming cube  $H = \{0, 1\}^n$  such that  $|S_u| = |S_k| = n$ . Consider the two distributions  $p_u$  and  $p_k$  over the Hamming cube  $H$  that are naturally defined by the sets  $S_u$  and  $S_k$  where for all  $x \in H$  probability of  $x$  in  $p_u$  (and  $p_k$ ) is the number of occurrences of  $x$  in  $S_u$  (and  $S_k$ ) divided by  $n$ . We then define the EMD between the multi-sets  $S_u$  and  $S_k$  as

$$EMD(S_u, S_k) \triangleq n \cdot EMD(p_u, p_k).$$

The problem of estimating the EMD over multi-sets while sampling **with** (or **without**) replacement means designing an algorithm, that given any two constants  $\beta_1, \beta_2$  such that  $0 \leq \beta_1 < \beta_2 \leq 1$ , and access to the unknown set  $S_u$  by sampling **with** (or **without**)

<sup>11</sup>The probability of each element in the sample space is an integer multiple of  $\frac{1}{K}$ .

replacement decides whether  $EMD(S_k, S_u) \leq \beta_1 n^2$  or  $EMD(S_k, S_u) \geq \beta_2 n^2$  with probability at least  $2/3$ .

Note that estimating the EMD over multi-sets while sampling **with** replacement is exactly same as estimating EMD between the distributions  $p_u$  and  $p_k$  with samples drawn according to  $p_u$ .

Let  $QWR_{EMD}(n, d, \beta_1, \beta_2)$  (and  $QWoR_{EMD}(n, d, \beta_1, \beta_2)$ ) denote the number of samples **with** ( and **without**) replacement required to decide the above from the unknown multi-set  $S_u$ . For ease of presentation, we write  $QWoR_{EMD}(n, d)$  ( $QWR_{EMD}(n, d)$ ) instead of  $QWoR_{EMD}(n, d)$  ( $QWR_{EMD}(n, \beta_1, \beta_2)$ ) when the proximity parameters are clear from the context.

► **Proposition B.9** (Query complexity of EMD increases with number of points as well as dimension). *Let  $n, n_1, n_2, d, d_1, d_2 \in \mathbb{N}$  be such that  $d_1 \leq d_2$  and  $n_1 \leq n_2$ . Then*

- (i)  $QWR_{EMD}(n_1, d) \leq QWR_{EMD}(n_2, d)$ ;
- (ii)  $QWoR_{EMD}(n_1, d) \leq QWoR_{EMD}(n_2, d)$ ;
- (iii)  $QWR_{EMD}(n, d_1) \leq QWR_{EMD}(n, d_2)$ ; and
- (iv)  $QWoR_{EMD}(n, d_1) \leq QWoR_{EMD}(n, d_2)$ .

► **Remark 7.** For  $d = n$  (as considered in Definition 1.3),  $QWoR_{EMD}(n, d)$  (and  $QWR_{EMD}(n, d)$ ) are denoted as  $QWoR_{EMD}(n)$  (and  $QWR_{EMD}(n)$ ).

Now let us state the lower bound of  $QWR_{EMD}(n)$ .

► **Theorem B.10.**  $QWR_{EMD}(n) = \Omega\left(\frac{n}{\log n}\right)$ .

Thus following Proposition B.7, we have

► **Theorem B.11.**  $QWoR_{EMD}(n) = \Omega(\sqrt{n})$ .

Note that an upper bound of  $QWoR_{EMD}(n) = \tilde{O}(n)$  is trivial. In the rest of the section, we focus on proving Theorem B.10 that states the lower bound on  $QWR_{EMD}(n)$ . We also provide an upper bound for  $QWR_{EMD}(n)$  at Lemma B.16 that shows that  $\tilde{O}(n)$  many samples **with** replacement from  $S_u$  to estimate  $QWR_{EMD}(n)$ . Note that by Remark 6, it is enough to show the following lemma that states the lower bound for tolerant EMD testing between two distributions.

► **Lemma B.12.** *Let  $S$  be a subset of a Hamming cube  $H = \{0, 1\}^n$  such that the minimum distance between any pair of points in  $S$  is at least  $\frac{n}{2}$ . Also, let  $p$  and  $q$  be two known and unknown distributions, respectively, supported over a subset of  $S$ . Then there exists a constant  $\epsilon_{EMD}$  such that the following holds. Given two constants  $\beta_1, \beta_2$  with  $0 < \beta_1 < \beta_2 < \epsilon_{EMD}(c)$ ,  $\Omega\left(\frac{n}{\log n}\right)$  samples from the distribution  $q$  are necessary in order to decide whether  $EMD(p, q) \leq \beta_1 n$  or  $EMD(p, q) \geq \beta_2 n$ . More over,  $\epsilon_{EMD} = \frac{1 - \epsilon_{\ell_1}}{4}$ , where  $\epsilon_{\ell_1}$  is the constant that is mentioned in Theorem B.14.*

To prove the above lower bound, let us first consider the following lower bound for tolerant  $\ell_1$  testing between two probability distributions.

► **Theorem B.13** (Valiant and Valiant [25]). *Let  $p$  and  $q$  be two known and unknown probability distributions respectively over  $[n]$ . There is an absolute constant  $\epsilon$  such that in order to decide whether  $\|p - q\|_1 \leq \epsilon$  or  $\|p - q\|_1 \geq 1 - \epsilon$ ,  $\Omega\left(\frac{n}{\log n}\right)$  samples, from the distribution  $q$ , are necessary.*<sup>12</sup>

<sup>12</sup>Note that this is rephrasing of the result proved in [25]. For reference, see Chapter 5 of the survey by Canonne [9].

## 34:22 Graph Isomorphism and EMD

Now, we restate the above result for our purpose.

► **Theorem B.14.** *Let  $p$  and  $q$  be two known and unknown probability distributions, having support size  $n$ , over a Hamming cube  $H = \{0, 1\}^n$ . There is an absolute constant  $\epsilon_{\ell_1}$  such that in order to decide whether  $\|p - q\|_1 \leq \alpha_1$  or  $\|p - q\|_1 \geq \alpha_2$  with  $0 < \alpha_1 < \alpha_2 \leq 1 - \epsilon_{\ell_1}$ ,  $\Omega(\frac{n}{\log n})$  samples, from the distribution  $q$ , are necessary.*

As noted earlier, we will prove Theorem B.10 by using Lemma B.14. However, Theorem B.10 is regarding  $EMD$  between two distributions whereas Lemma B.14 is regarding  $\ell_1$  distance between two distributions. The following observation (from [12]) gives a connection between  $EMD$  between two distributions with the  $\ell_1$  distance between them, which will be required in lower bound proof.

► **Proposition B.15** ([12]). *Let  $(M, D)$  be a finite metric space and  $p$  and  $q$  be two probability distributions on  $M$ . Minimum distance between any two points of  $M$  is  $\Delta_{\min}$  and diameter of  $M$  is  $\Delta_{\max}$ . Then the following condition holds:*

$$\frac{\|p - q\|_1 \Delta_{\min}}{2} \leq EMD(p, q) \leq \frac{\|p - q\|_1 \Delta_{\max}}{2}.$$

Note that the above proposition gives interesting result when  $\frac{\Delta_{\max}}{\Delta_{\min}}$  is bounded by a constant. Note that  $S \subset \{0, 1\}^n$  satisfies  $\frac{\Delta_{\max}}{\Delta_{\min}} \leq 2$ .

**Proof of Lemma B.12.** In  $S \subset H = \{0, 1\}^n$ , the pairwise Hamming distance between any two elements in  $S$  is at least  $\frac{n}{2}$ , to have  $\frac{\Delta_{\max}}{\Delta_{\min}} \leq 2$  in our context. It is well known that  $|S| = \Omega(n)$ . We will show that if there exists an algorithm  $\mathcal{A}$  that decides  $EMD(p, q) \leq \beta_1 n$  or  $EMD(p, q) \geq \beta_2 n$  by using  $t$  samples from  $q$ , then there exists an algorithm  $\mathcal{P}$  that decides whether  $\|p - q\|_1 \leq \alpha_1$  or  $\|p - q\|_1 \geq \alpha_2$  by using  $t$  samples from  $q$ , where  $\alpha_1 = 2\beta_1$  and  $\alpha_2 = 4\beta_2$ . Note that we have  $0 < \beta_1 < \beta_2 < \frac{1 - \epsilon_{\ell_1}}{4}$ . So,  $0 < \alpha_1 < \alpha_2 < 1 - \epsilon_{\ell_1}$ , which satisfies the requirement of Theorem B.14.

### Algorithm $\mathcal{P}$ :

- (1) First run algorithm  $\mathcal{A}$ .
- (2) If the output of algorithm  $\mathcal{A}$  is  $EMD(p, q) \leq \beta_1 n$ , algorithm  $\mathcal{P}$  returns  $\|p - q\|_1 \leq \alpha_1$ .
- (3) If the output of algorithm  $\mathcal{A}$  is  $EMD(p, q) \geq \beta_2 n$ , algorithm  $\mathcal{P}$  returns  $\|p - q\|_1 \geq \alpha_2$ .

To complete the proof, we only need to show that  $\mathcal{P}$  gives desired output with probability at least  $2/3$ . The result then follows from Theorem B.14.

Let us first consider the case  $\|p - q\|_1 \leq \alpha_1$ . Then by Observation B.15, we can say that  $EMD(p, q) \leq \frac{\alpha_1 n}{2} = \beta_1 n$ . Therefore algorithm  $\mathcal{A}$  will output that  $EMD(p, q) \leq \beta_1 n$ . This implies that the algorithm  $\mathcal{P}$  will output  $\|p - q\|_1 \leq \alpha_1$ .

Now, let us consider the case  $\|p - q\|_1 \geq \alpha_2$ . Using the fact that any pair elements in  $S \subset H$  is at least  $\frac{n}{2}$  along with Observation B.15, we get  $EMD(p, q) \geq \frac{\alpha_2 n}{4} = \beta_2 n$ . This implies  $\mathcal{P}$  will output  $\|p - q\|_1 \geq \alpha_2$ . ◀

Till now, we were discussing the proof of Lemma B.12 that states  $QWR_{EMD}(n) = \Omega(\frac{n}{\log n})$ . The lower bound is almost tight, up to a polynomial factor of  $\log n$ . The upper bound is stated in the following observation.

► **Observation B.16.**  $QWR_{EMD}(n) = \tilde{O}(n)$ , where  $\tilde{O}(\cdot)$  hides a polynomial factor in  $\frac{1}{\beta_2 - \beta_1}$  and  $\log n$ .



Instead of proving the above observation, we prove the following lemma that states the upper bound of tolerant EMD testing between two distributions when we know one distribution and have sample access to the unknown distribution. By Remark 6, we will be done with the proof of Observation B.16.

► **Lemma B.17.** *Let  $H = \{0, 1\}^n$  be a  $n$ -dimensional Hamming cube, and let  $p$  and  $q$  denote two known and unknown  $n$ -grained distribution over  $H$ . There exists an algorithm that takes two parameters  $\beta_1, \beta_2$  with  $0 \leq \beta_1 < \beta_2 \leq 1$  and a  $\delta \in (0, 1)$  as input and decides whether  $EMD(p, q) \leq \beta_1 n$  or  $EMD(p, q) \geq \beta_2 n$  with probability at least  $1 - \delta$ . Moreover, the algorithm ALG-EMD queries for  $\tilde{O}(n)$  many samples from  $q$ , where  $\tilde{O}(\cdot)$  hides a polynomial factor in  $\frac{1}{\beta_2 - \beta_1}$  and  $\log n$ .*

**Proof.** Let  $\epsilon$  be a constant less than  $(\beta_2 - \beta_1)$ . We construct a probability distribution  $q'$  such that the  $\ell_1$  distance between  $q$  and  $q'$  will be at most  $\epsilon$ , that is,  $\sum_{i \in [L]} |q(i) - q'(i)| \leq \epsilon$ .



Note that such a  $q'$  can be constructed with probability at least  $1 - \delta$  by querying for  $\tilde{O}(n)$  many samples of  $q$  which follows from [11]. Then, we find  $EMD(p, q')$ . Observe that  $|EMD(p, q) - EMD(p, q')| \leq \frac{\epsilon n}{2}$ . This is because

$$\begin{aligned} |EMD(p, q) - EMD(p, q')| &\leq |EMD(p, q') + EMD(q', q) - EMD(p, q')| \\ &\leq EMD(q, q') \\ &\leq \frac{\epsilon d}{2} \text{ (By Proposition B.15)} \end{aligned}$$

As  $EMD(p, q) \leq \beta_1 n$  or  $EMD(p, q) \geq \beta_2 n$ , by the above observation, we will get either  $EMD(p, q') \leq (\beta_1 + \frac{\epsilon}{2}) n$  or  $EMD(p, q') \geq (\beta_1 + \frac{\epsilon}{2}) n$ , respectively. By our choice of  $\epsilon < \beta_2 - \beta_1$ , we can decide  $EMD(p, q) \leq \beta_1 n$  or  $EMD(p, q) \geq \beta_2 n$  from the value of  $EMD(p, q')$ . ◀



# The Product of Gaussian Matrices Is Close to Gaussian

Yi Li  

Division of Mathematical Sciences, Nanyang Technological University, Singapore, Singapore

David P. Woodruff 

Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

---

## Abstract

We study the distribution of the *matrix product*  $G_1 G_2 \cdots G_r$  of  $r$  independent Gaussian matrices of various sizes, where  $G_i$  is  $d_{i-1} \times d_i$ , and we denote  $p = d_0$ ,  $q = d_r$ , and require  $d_i = d_{r-1}$ . Here the entries in each  $G_i$  are standard normal random variables with mean 0 and variance 1. Such products arise in the study of wireless communication, dynamical systems, and quantum transport, among other places. We show that, provided each  $d_i$ ,  $i = 1, \dots, r$ , satisfies  $d_i \geq Cp \cdot q$ , where  $C \geq C_0$  for a constant  $C_0 > 0$  depending on  $r$ , then the matrix product  $G_1 G_2 \cdots G_r$  has variation distance at most  $\delta$  to a  $p \times q$  matrix  $G$  of i.i.d. standard normal random variables with mean 0 and variance  $\prod_{i=1}^{r-1} d_i$ . Here  $\delta \rightarrow 0$  as  $C \rightarrow \infty$ . Moreover, we show a converse for constant  $r$  that if  $d_i < C' \max\{p, q\}^{1/2} \min\{p, q\}^{3/2}$  for some  $i$ , then this total variation distance is at least  $\delta'$ , for an absolute constant  $\delta' > 0$  depending on  $C'$  and  $r$ . This converse is best possible when  $p = \Theta(q)$ .

**2012 ACM Subject Classification** Mathematics of computing  $\rightarrow$  Probability and statistics

**Keywords and phrases** random matrix theory, total variation distance, matrix product

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.35

**Category** RANDOM

**Funding** *Yi Li*: Supported in part by Singapore Ministry of Education (AcRF) Tier 2 grant MOE2018-T2-1-013.

*David P. Woodruff*: Supported in part by Office of Naval Research (ONR) grant N00014-18-1-256 and a Simons Investigator Award.

**Acknowledgements** D. Woodruff would like to thank Sébastien Bubeck, Sitan Chen, and Jerry Li for many helpful discussions.

## 1 Introduction

Random matrices play a central role in many areas of theoretical, applied, and computational mathematics. One particular application is dimensionality reduction, whereby one often chooses a rectangular random matrix  $G \in \mathbb{R}^{m \times n}$ ,  $m \ll n$ , and computes  $G \cdot x$  for a fixed vector  $x \in \mathbb{R}^n$ . Indeed, this is the setting in compressed sensing and sparse recovery [12], randomized numerical linear algebra [18, 20, 36], and sketching algorithms for data streams [25]. Often  $G$  is chosen to be a Gaussian matrix, and in particular, an  $m \times n$  matrix with entries that are i.i.d. normal random variables with mean 0 and variance 1, denoted by  $N(0, 1)$ . Indeed, in compressed sensing, such matrices can be shown to satisfy the Restricted Isometry Property (RIP) [10], while in randomized numerical linear algebra, in certain applications such as support vector machines [29] and non-negative matrix factorization [19], their performance is shown to often outperform that of other sketching matrices.

Our focus in this paper will be on understanding the *product* of two or more Gaussian matrices. Such products arise naturally in different applications. For example, in the over-constrained ridge regression problem  $\min_x \|Ax - b\|_2^2 + \lambda \|x\|_2^2$ , the design matrix  $A \in \mathbb{R}^{n \times d}$ ,



© Yi Li and David P. Woodruff;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 35; pp. 35:1–35:22



Leibniz International Proceedings in Informatics

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

$n \gg d$ , is itself often assumed to be Gaussian (see, e.g., [26]). In this case, the “sketch-and-solve” algorithmic framework for regression [32] would compute  $G \cdot A$  and  $G \cdot b$  for an  $m \times n$  Gaussian matrix  $G$  with  $m \approx sd_\lambda$ , where  $sd_\lambda$  is the so-called statistical dimension [2], and solve for the  $x$  which minimizes  $\|G \cdot Ax - G \cdot b\|_2^2 + \lambda \|x\|_2^2$ . While computing  $G \cdot A$  is slower than computing the corresponding matrix product for other kinds of sketching matrices  $G$ , it often has application-specific [29, 19] as well as statistical benefits [31]. Notice that  $G \cdot A$  is the product of two independent Gaussian matrices, and in particular,  $G$  has a small number of rows while  $A$  has a small number of columns – this is precisely the rectangular case we will study below. Other applications in randomized numerical linear algebra where the product of two Gaussian matrices arises is when one computes the product of a Gaussian sketching matrix and Gaussian noise in a spiked identity covariance model [37].

The product of two or more Gaussian matrices also arises in diverse fields such as multiple-input multiple-output (MIMO) wireless communication channels [24]. Indeed, similar to the above regression problem in which one wants to reconstruct an underlying vector  $x$ , in such settings one observes the vector  $y = G_1 \cdots G_r \cdot x + \eta$ , where  $x$  is the transmitted signal and  $\eta$  is background noise. This setting corresponds to the situation in which there are  $r$  scattering environments separated by major obstacles, and the dimensions of the  $G_i$  correspond to the number of “keyholes” [24]. To determine the mutual information of this channel, one needs to understand the singular values of the matrix  $G_1 \cdots G_r$ . If one can show the distribution of this product is close to that of a Gaussian distribution in total variation distance, then one can use the wide range of results known for the spectrum of a single Gaussian matrix (see, e.g., [35]). Other applications of products of Gaussian matrices include disordered spin chains [11, 3, 15], stability of large complex dynamical systems [22, 21], symplectic maps and Hamiltonian mechanics [11, 4, 28], quantum transport in disordered wires [23, 13], and quantum chromodynamics [27]; we refer the reader to [14, 1] for an overview.

The main question we ask in this work is:

*What is the distribution of the product  $G_1 G_2 \cdots G_r$  of  $r$  independent Gaussian matrices of various sizes, where  $G_i$  is  $d_{i-1} \times d_i$ ?*

Our main interest in the question above will be when  $G_1$  has a small number  $p = d_0$  of rows, and  $G_r$  has a small number  $q = d_r$  of columns. Despite the large body of work on random matrix theory (see, e.g., [34] for a survey), we are not aware of any work which attempts to bound the total variation distance of the entire distribution of  $G_1 G_2 \cdots G_r$  to a Gaussian distribution itself.

## 1.1 Our Results

Formally, we consider the problem of distinguishing the product of normalized Gaussian matrices

$$A_r = \left( \frac{1}{\sqrt{d_1}} G_1 \right) \left( \frac{1}{\sqrt{d_2}} G_2 \right) \cdots \left( \frac{1}{\sqrt{d_{r-1}}} G_{r-1} \right) \left( \frac{1}{\sqrt{d_1}} G_r \right)$$

from a single normalized Gaussian matrix

$$A_1 = \frac{1}{\sqrt{d_1}} G_1.$$

We show that, when  $r$  is a constant, with constant probability we cannot distinguish the distributions of these two random matrices when  $d_i \gg p, q$  for all  $i$ ; and, conversely, with constant probability, we can distinguish these two distributions when the  $d_i$  are not large enough.

► **Theorem 1** (Main theorem). *Suppose that  $d_i \geq \max\{p, q\}$  for all  $i$  and  $d_{r-1} = d_1$ .*

(a) *It holds that*

$$d_{TV}(A_r, A_1) \leq C_1 \sum_{i=1}^{r-1} \sqrt{\frac{pq}{d_i}},$$

where  $d_{TV}(A_r, A_1)$  denotes the total variation distance between  $A_r$  and  $A_1$ , and  $C_1 > 0$  is an absolute constant.

(b) *If  $p, q, d_1, \dots, d_r$  further satisfy that*

$$\sum_{j=1}^{r-1} \frac{1}{d_j} \geq \frac{C_2^r}{\max\{p, q\}^{\frac{1}{2}} \min\{p, q\}^{\frac{3}{2}}},$$

where  $C_2 > 0$  is an absolute constant, then  $d_{TV}(A_r, A_1) \geq 2/3$ .

Part (a) states that  $d_{TV}(A_r, A_1) < 2/3$  when  $d_i \geq C'_1 pq$  for all  $i$  for a constant  $C'_1$  depending on  $r$ . The converse in (b) implies that  $d_{TV}(A_r, A_1) \geq 2/3$  when  $d_i \leq C'_2 \max\{p, q\}^{1/2} \min\{p, q\}^{3/2}$  for some  $i$  for a constant  $C'_2$  depending on  $r$ . When  $p = \Theta(q)$  and  $r$  is a constant, we obtain a dichotomy (up to a constant factor) for the conditions on  $p, q$  and  $d_i$ .

## 1.2 Our Techniques

**Upper Bound.** We start by explaining our main insight as to why the distribution of a product  $G_1 \cdot G_2$  of a  $p \times d$  matrix  $G_1$  of i.i.d.  $N(0, 1)$  random variables and a  $d \times q$  matrix  $G_2$  of i.i.d.  $N(0, 1)$  random variables has low variation distance to the distribution of a  $p \times q$  matrix  $A$  of i.i.d.  $N(0, d)$  random variables. One could try to directly understand the probability density function as was done in the case of Wishart matrices in [7, 30], which corresponds to the setting when  $G_1 = G_2$ . However, there are certain algebraic simplifications in the case of the Wishart distribution that seem much less tractable when manipulating the density function of the product of independent Gaussians [9]. Another approach would be to try to use entropic methods as in [8, 6]. Such arguments try to reveal entries of the product  $G_1 \cdot G_2$  one-by-one, arguing that for most conditionings of previous entries, the new entry still looks like an independent Gaussian. However, the entries are clearly not independent – if  $(G_1 \cdot G_2)_{i,j}$  has large absolute value, then  $(G_1 \cdot G_2)_{i,j'}$  is more likely to be large in absolute value, as it could indicate that the  $i$ -th row of  $G_1$  has large norm. One could try to first condition on the norms of all rows of  $G_1$  and columns of  $G_2$ , but additional issues arise when one looks at submatrices: if  $(G_1 \cdot G_2)_{i,j}, (G_1 \cdot G_2)_{i,j'}$ , and  $(G_1 \cdot G_2)_{i',j}$  are all large, then it could mean the  $i$ -th row of  $G_1$  and the  $i'$ -th row of  $G_1$  are correlated with each other, since they both are correlated with the  $j$ -th column of  $G_2$ . Consequently, since  $(G_1 \cdot G_2)_{i,j'}$  is large, it could make it more likely that  $(G_1 \cdot G_2)_{i',j'}$  has large absolute value. This makes the entropic method difficult to apply in this context.

Our upper bound instead leverages beautiful work of Jiang [16] and Jiang and Ma [17] which bounds the total variation distance between the distribution of an  $r \times \ell$  submatrix of a random  $d \times d$  orthogonal matrix (orthonormal rows and columns) and an  $r \times \ell$  matrix with i.i.d.  $N(0, 1/d)$  entries. Their work shows that if  $r \cdot \ell/d \rightarrow 0$  as  $d \rightarrow \infty$ , then the total variation distance between these two matrix ensembles goes to 0. It is not immediately clear how to apply such results in our context. First of all, which submatrix should we be looking at? Note though, that if  $V^T$  is a  $p \times d$  uniformly random (Haar measure) matrix with orthonormal rows, and  $E$  is a  $d \times q$  uniformly random matrix with orthonormal columns,

## 35:4 The Product of Gaussian Matrices Is Close to Gaussian

then by rotational invariance,  $V^T E$  is identically distributed to a  $p \times q$  submatrix of a  $d \times d$  random orthonormal matrix. Thus, setting  $r = p$  and  $\ell = q$  in the above results, they imply that  $V^T E$  is close in variation distance to a  $p \times q$  matrix  $H$  with i.i.d.  $N(0, 1/d)$  entries. Given  $G_1$  and  $G_2$ , one could then write them in their *singular value decomposition*, obtaining  $G_1 = U\Sigma V^T$  and  $G_2 = E^T F^T$ . Then  $V^T$  and  $E$  are independent and well-known to be uniformly random  $p \times d$  and  $d \times q$  orthonormal matrices, respectively. Thus  $G_1 \cdot G_2$  is close in total variation distance to  $U\Sigma H^T F^T$ . However, this does not immediately help either, as it is not clear what the distribution of this matrix is. Instead, the “right” way to utilize the results above is to (1) observe that  $G_1 \cdot G_2 = U\Sigma V^T G_2$  is identically distributed as  $U\Sigma X$ , where  $X$  is a matrix of i.i.d. normal random variables, given the rotational invariance of the Gaussian distribution. Then (2)  $X$  is itself close to a product  $W^T Z$  where  $W^T$  is a random  $p \times d$  matrix with orthonormal rows, and  $Z$  is a random  $d \times q$  matrix with orthonormal columns, by the above results. Thus,  $G_1 \cdot G_2$  is close to  $U\Sigma W^T Z$ . Then (3)  $U\Sigma W^T$  has the same distribution as  $G_1$ , so  $U\Sigma W^T Z$  is close to  $G'_1 Z$ , where  $G'_1$  and  $G_1$  are identically distributed, and  $G'_1$  is independent of  $Z$ . Finally, (4)  $G'_1 Z$  is identically distributed as a matrix  $A_1$  of standard normal random variables because  $G'_1$  is Gaussian and  $Z$  has orthonormal columns, by rotational invariance of the Gaussian distribution.

We hope that this provides a general method for arguments involving Gaussian matrices - in step (2) we had the quantity  $U\Sigma X$ , where  $X$  was a Gaussian matrix, and then viewed  $X$  as a product of a short-fat random orthonormal matrix  $W^T$  and a tall-thin random orthonormal matrix  $Z$ . Our proof for the product of more than 2 matrices recursively uses similar ideas, and bounds the growth in variation distance as a function of the number  $r$  of matrices involved in the product.

**Lower Bound.** For our lower bound for constant  $r$ , we show that the fourth power of the Schatten 4-norm of a matrix, namely,  $\|X\|_{S_4}^4 = \text{tr}((X^T X)^2)$ , can be used to distinguish a product  $A_r$  of  $r$  Gaussian matrices and a single Gaussian matrix  $A_1$ . We use Chebyshev’s inequality, for which we need to find the expectation and variance of  $\text{tr}((X^T X)^2)$  for  $X = A_r$  and  $X = A_1$ .

Let us consider the expectation first. An idea is to calculate the expectation recursively, that is, for a fixed matrix  $M$  and a Gaussian random matrix  $G$  we express  $\mathbb{E} \text{tr}(((MG)^T(MG))^2)$  in terms of  $\mathbb{E} \text{tr}((M^T M)^2)$ . The real situation turns out to be slightly more complicated. Instead of expressing  $\mathbb{E} \text{tr}(((MG)^T(MG))^2)$  in terms of  $\mathbb{E} \text{tr}((M^T M)^2)$  directly, we decompose  $\mathbb{E} \text{tr}(((MG)^T(MG))^2)$  into the sum of expectations of a few functions in terms of  $M$ , say,

$$\mathbb{E} \text{tr}(((MG)^T(MG))^2) = \mathbb{E} f_1(M) + \mathbb{E} f_2(M) + \dots + \mathbb{E} f_s(M)$$

and build up the recurrence relations for  $\mathbb{E} f_1(MG), \dots, \mathbb{E} f_s(MG)$  in terms of  $\mathbb{E} f_1(M), \mathbb{E} f_2(M), \dots, \mathbb{E} f_s(M)$ . It turns out that the recurrence relations are all linear, i.e.,

$$\mathbb{E} f_i(MG) = \sum_{j=1}^s a_{ij} \mathbb{E} f_j(M), \quad i = 1, \dots, s, \tag{1}$$

whence we can solve for  $\mathbb{E} f_i(A_r)$  and obtaining the desired expectation  $\mathbb{E} \text{tr}((A_r^T A_r)^2)$ .

Now we turn to variance. One could try to apply the same idea of finding recurrence relations for  $\text{Var}(Q) = \mathbb{E}(Q^2) - (\mathbb{E} Q)^2$  (where  $Q = \text{tr}(((MG)^T(MG))^2)$ ), but it quickly becomes intractable for the  $\mathbb{E}(Q^2)$  term as it involves products of eight entries of  $M$ , which all need to be handled carefully as to avoid any loose bounds; note, the subtraction of  $(\mathbb{E} Q)^2$

is critically needed to obtain a small upper bound on  $\text{Var}(Q)$  and thus loose bounds on  $\mathbb{E}(Q^2)$  would not suffice. For a tractable calculation, we keep the product of entries of  $M$  to 4th order throughout, without involving any terms of 8th order. To do so, we invoke the law of total variance,

$$\text{Var}_{M,G}(\text{tr}((MG)^T(MG))^2) = \mathbb{E}_M \left( \text{Var}_G(\text{tr}((G^T M^T M G)^2)) \middle| M \right) + \text{Var}_M \left( \mathbb{E}_G \text{tr}((G^T M^T M G)^2) \middle| M \right). \quad (2)$$

For the first term on the right-hand side, we use Poincaré’s inequality to upper bound it. Poincaré’s inequality for the Gaussian measure states that

$$\text{Var}_{g \sim N(0, I_m)}(f(g)) \leq C \mathbb{E}_{g \sim N(0, I_m)} \|\nabla f(g)\|_2^2$$

for a differentiable function  $f$  on  $\mathbb{R}^m$ . Here we can simply let  $f(X) = \text{tr}((MX)^T(MX))^2$  and calculate  $\mathbb{E} \|\nabla f(G)\|_2^2$ . This is tractable since  $\mathbb{E} \|\nabla f(G)\|_2^2$  involves the products of at most 4 entries of  $M$ , and we can use the recursive idea for the expectation above to express

$$\mathbb{E} \|\nabla f(G)\|_2^2 = \sum_i a_{ij} \mathbb{E} g_i(M)$$

for a few functions  $g_i$ ’s and establish a recurrence relation for each  $g_i$ .

The second term on the right-hand side of (2) can be dealt with by plugging in (1), and turns out to depend on a new quantity  $\text{Var}(\text{tr}^2(M^T M))$ . We again apply the recursive idea and the law of total variance to

$$\text{Var}_{M,G}(\text{tr}^2(G^T M^T M G)) = \mathbb{E}_M \left( \text{Var}_G(\text{tr}^2((G^T M^T M G))) \middle| M \right) + \text{Var}_M \left( \mathbb{E}_G \text{tr}^2(G^T M^T M G) \middle| M \right).$$

Again, the first term on the right-hand side can be handled by Poincaré’s inequality and the second-term turns out to depend on  $\text{Var}(\text{tr}((M^T M)^2))$ , which is crucial. We have now obtained a double recurrence involving inequalities on  $\text{Var}(\text{tr}((M^T M)^2))$  and  $\text{Var}(\text{tr}^2((M^T M)^2))$ , from which we can solve for an upper bound on  $\text{Var}(\text{tr}(A_r^T A_r)^2)$ . This upper bound, however, grows exponentially in  $r$ , which is impossible to improve due to our use of Poincaré’s inequality.

## 2 Preliminaries

**Notation.** For a random variable  $X$  and a probability distribution  $\mathcal{D}$ , we use  $X \sim \mathcal{D}$  to denote that  $X$  is subject to  $\mathcal{D}$ . For two random variables  $X$  and  $Y$  defined on the same sample space, we write  $X \stackrel{d}{=} Y$  if  $X$  and  $Y$  are identically distributed.

We use  $\mathcal{G}_{m,n}$  to denote the distribution of  $m \times n$  Gaussian random matrices of i.i.d. entries  $N(0, 1)$  and  $\mathcal{O}_{m,n}$  to denote the uniform distribution (Haar) of an  $m \times n$  random matrix with orthonormal rows. For a distribution  $\mathcal{D}$  on a linear space and a scaling factor  $\alpha \in \mathbb{R}$ , we use  $\alpha\mathcal{D}$  to denote the distribution of  $\alpha X$ , where  $X \sim \mathcal{D}$ .

For two probability measures  $\mu$  and  $\nu$  on the Borel algebra  $\mathcal{F}$  of  $\mathbb{R}^m$ , the total variation distance between  $\mu$  and  $\nu$  is defined as

$$d_{TV}(\mu, \nu) = \sup_{A \in \mathcal{F}} |\mu(A) - \nu(A)| = \frac{1}{2} \int_{\mathbb{R}^m} \left| \frac{d\mu}{d\nu} - 1 \right| d\nu.$$

If  $\nu$  is absolutely continuous with respect to  $\mu$ , one can define the Kullback-Leibler Divergence between  $\mu$  and  $\nu$  as

$$D_{KL}(\mu \parallel \nu) = \int_{\mathbb{R}^m} \frac{d\mu}{d\nu} \log_2 \frac{d\mu}{d\nu} d\nu.$$

If  $\nu$  is not absolutely continuous with respect to  $\mu$ , we define  $D_{KL}(\mu \parallel \nu) = \infty$ .

## 35:6 The Product of Gaussian Matrices Is Close to Gaussian

When  $\mu$  and  $\nu$  correspond to two random variables  $X$  and  $Y$ , respectively, we also write  $d_{TV}(\mu, \nu)$  and  $D_{\text{KL}}(\mu\|\nu)$  as  $d_{TV}(X, Y)$  and  $D_{\text{KL}}(X\|Y)$ , respectively.

The following is the well-known relation between the Kullback-Leibler divergence and the total variation distance between two probability measures.

► **Lemma 2** (Pinsker's Inequality [5, Theorem 4.19]).  $d_{TV}(\mu, \nu) \leq \sqrt{\frac{1}{2}D_{\text{KL}}(\mu\|\nu)}$ .

The following result, concerning the distance between the submatrix of a properly scaled Gaussian random matrix and a submatrix of a random orthogonal matrix, is due to Jiang and Ma [17].

► **Lemma 3** ([17]). Let  $G \sim \mathcal{G}_{d,d}$  and  $Z \sim \mathcal{O}_{d,d}$ . Suppose that  $p, q \leq d$  and  $\hat{G}$  is the top-left  $p \times q$  block of  $G$  and  $\hat{Z}$  the top-left  $p \times q$  block of  $Z$ . Then

$$d_{\text{KL}}\left(\frac{1}{\sqrt{d}}\hat{G}\left\|\hat{Z}\right.\right) \leq C\frac{pq}{d}, \quad (3)$$

where  $C > 0$  is an absolute constant.

The original paper [17] does not state explicitly the bound in (3) and only states that the Kullback-Leibler divergence tends to 0 as  $d \rightarrow \infty$ . A careful examination of the proof of [17, Theorem 1(i)], by keeping track of the order of the various  $o(1)$  terms, reveals the quantitative bound (3).

**Useful Inequalities.** We list two useful inequalities below.

► **Lemma 4** (Poincaré's inequality for Gaussian measure [5, Theorem 3.20]). Let  $X \sim N(0, I_n)$  be the standard  $n$ -dimensional Gaussian distribution and  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be any continuously differentiable function. Then

$$\text{Var}(f(X)) \leq \mathbb{E}\left(\|\nabla f(X)\|_2^2\right).$$

► **Lemma 5** (Trace inequality, [33]). Let  $A$  and  $B$  be symmetric, positive semidefinite matrices and  $k$  be a positive integer. Then

$$\text{tr}((AB)^k) \leq \min\left\{\|A\|_{\text{op}}^k \text{tr}(B^k), \|B\|_{\text{op}}^k \text{tr}(A^k)\right\}.$$

### 3 Upper Bound

Let  $r \geq 2$  be an integer. Suppose that  $G_1, \dots, G_r$  are independent Gaussian random matrices, where  $G_i \sim \mathcal{G}_{d_{i-1}, d_i}$  and  $d_0 = p$ ,  $d_r = q$  and  $d_{r-1} = d_1$ . Consider the product of normalized Gaussian matrices

$$A_r = \left(\frac{1}{\sqrt{d_1}}G_1\right) \left(\frac{1}{\sqrt{d_2}}G_2\right) \cdots \left(\frac{1}{\sqrt{d_{r-1}}}G_{r-1}\right) \left(\frac{1}{\sqrt{d_1}}G_r\right)$$

and a single normalized Gaussian random matrix

$$A_1 = \frac{1}{\sqrt{d_1}}G'_1$$

where  $G'_1 \sim \mathcal{G}_{p,q}$ . In this section, we shall show that when  $p, q \ll d_i$  for all  $i$ , we cannot distinguish  $A_r$  from  $A_1$  with constant probability.

For notational convenience, let  $W_i = \frac{1}{\sqrt{d_i}}G_i$  for  $i \leq r$  and  $W_r = \frac{1}{\sqrt{d_1}}G_r$ . Assume that  $pq \leq \beta d_i$  for some constant  $\beta$  for all  $i$ . Our question is to find the total variation distance between the matrix product  $W_1W_2 \cdots W_r$  and the product  $W_1W_r$  of two matrices.



► **Lemma 6.** Let  $p, q, d, d'$  be positive integers satisfying that  $pq \leq \beta d$  and  $pq \leq \beta d'$  for some constant  $\beta < 1$ . Suppose that  $A \in \mathbb{R}^{p \times d}$ ,  $G \sim \frac{1}{\sqrt{d}} \mathcal{G}_{d, d'}$ , and  $L \sim \mathcal{O}_{d', d}$ . Further suppose that  $G$  and  $L$  are independent. Let  $Z \sim \mathcal{O}_{q, d}$  be independent of  $A, G$  and  $L$ . Then

$$d_{TV}(AGL, AZ^T) \leq C \sqrt{\frac{pq}{d}},$$

where  $C > 0$  is an absolute constant.

**Proof.** Let  $A = U\Sigma V^T$  be its singular value decomposition, where  $V$  has dimension  $d \times p$ . Then

$$AGL = U\Sigma(V^T GL) \stackrel{d}{=} U\Sigma X,$$

where  $X$  is a  $p \times q$  random matrix of i.i.d.  $N(0, 1/d)$  entries. Suppose that  $\tilde{Z}$  consists of the top  $p$  rows of  $Z^T$ . Then

$$AZ^T = U\Sigma(V^T Z^T) \stackrel{d}{=} U\Sigma \tilde{Z}.$$

Note that  $X$  and  $Z$  are independent of  $U$  and  $\Sigma$ . It follows from Lemma 3 that

$$d_{\text{KL}}(AGL \| AZ^T) = d_{\text{KL}}(U\Sigma X \| U\Sigma \tilde{Z}) = d_{\text{KL}}(X \| \tilde{Z}) \leq C \frac{pq}{d},$$

where  $C > 0$  is an absolute constant. The result follows from Pinsker's inequality (Lemma 2). ◀

The next theorem follows from the lemma above.

► **Theorem 7.** It holds that

$$d_{TV}(W_1 \cdots W_r, W_1 W_r) \leq C \sum_{i=1}^r \sqrt{\frac{pq}{d_i}},$$

where  $C > 0$  is an absolute constant.

**Proof.** Let  $W_r = U\Sigma V^T$  and  $X_i \sim \mathcal{O}_{q, d_i}$ , independent from each other and from the  $W_i$ 's. Applying the preceding lemma with  $A = W_1 \cdots W_{r-2}$ ,  $G = W_{r-1}$  and  $L = U$ , we have

$$d_{TV}(W_1 \cdots W_{r-2} W_{r-1} W_r, W_1 \cdots W_{r-2} X_{r-1}^T \Sigma V^T) \leq C \sqrt{\frac{pq}{d_{r-1}}},$$

Next, applying the preceding lemma with  $A = W_1 \cdots W_{r-3}$ ,  $G = W_{r-1}$  and  $L = X_r$ , we have

$$d_{TV}(W_1 \cdots W_{r-2} X_r \Sigma V^T, W_1 \cdots W_{r-3} X_{r-2}^T \Sigma V^T) \leq C \sqrt{\frac{pq}{d_{r-2}}},$$

Iterating this procedure, we have in the end that

$$d_{TV}(W_1 W_2 X_3 \Sigma V^T, W_1 X_2 \Sigma V^T) \leq C \sqrt{\frac{pq}{d_2}}.$$

Since  $U, \Sigma$  and  $V$  are independent and  $X_2 \stackrel{d}{=} U$ , it holds that  $X_2 \Sigma V^T \stackrel{d}{=} W_r$ . Therefore,

$$d_{TV}(W_1 \cdots W_r, W_1 W_r) \leq C \sum_{i=2}^{r-1} \sqrt{\frac{pq}{d_i}}. \quad \blacktriangleleft$$

Repeating the same argument for  $W_1 W_r$ , we obtain the following corollary immediately.

► **Corollary 8.** It holds that

$$d_{TV}(A_r, A_1) \leq C \sum_{i=1}^{r-1} \sqrt{\frac{pq}{d_i}},$$

where  $C > 0$  is an absolute constant.

#### 4 Lower Bound

Suppose that  $r$  is a constant. We shall show that one can distinguish the product of  $r$  Gaussian random matrices

$$A_r = \left( \frac{1}{\sqrt{d_1}} G_1 \right) \left( \frac{1}{\sqrt{d_2}} G_2 \right) \cdots \left( \frac{1}{\sqrt{d_{r-1}}} G_{r-1} \right) \left( \frac{1}{\sqrt{d_1}} G_r \right),$$

from one Gaussian random matrix

$$A_1 = \frac{1}{\sqrt{d_1}} G_1'$$

when the intermediate dimensions  $d_1, \dots, d_{r-1}$  are not large enough. Considering  $h(X) = \text{tr}((X^T X)^2)$ , it suffices to show that one can distinguish  $h(A_r)$  and  $h(A_1)$  with a constant probability for constant  $r$ . By Chebyshev's inequality, it suffices to show that

$$\max \left\{ \sqrt{\text{Var}(h(A_1))}, \sqrt{\text{Var}(h(A_r))} \right\} \leq c(\mathbb{E} h(A_r) - \mathbb{E} h(A_1))$$

for a small constant  $c$ . We calculate that:

► **Lemma 9.** *Suppose that  $r$  is a constant,  $d_i \geq \max\{p, q\}$  for all  $i = 1, \dots, r$ . When  $p, q, d_1, \dots, d_r \rightarrow \infty$ ,*

$$\mathbb{E} h(A_r) = \frac{pq(p+q+1)}{d_r^2} + (1+o(1)) \frac{pq(p-1)(q-1)}{d_r^2} \sum_{j=1}^{r-1} \frac{1}{d_j}.$$

► **Lemma 10.** *Suppose that  $r$  is a constant,  $d_i \geq \max\{p, q\}$  for all  $i = 1, \dots, r$ . There exists an absolute constant  $C$  such that, when  $p, q, d_1, \dots, d_r$  are sufficiently large,*

$$\text{Var}(h(A_r)) \leq \frac{C^r(p^3q + pq^3)}{d_1^4}.$$

We conclude with the following theorem, which can be seen as a tight converse to Corollary 8 up to a constant factor on the conditions for  $p, q, d_1, \dots, d_r$ .

► **Theorem 11.** *Suppose that  $r$  is a constant and  $d_i \geq \max\{p, q\}$  for all  $i = 1, \dots, r$ . Further suppose that  $d_1 = d_r$ . When  $p, q, d_1, \dots, d_r$  are sufficiently large and satisfy that*

$$\sum_{j=1}^{r-1} \frac{1}{d_j} \geq \frac{C^r}{\max\{p, q\}^{\frac{1}{2}} \min\{p, q\}^{\frac{3}{2}}},$$

where  $C > 0$  is some absolute constant, with probability at least  $2/3$ , one can distinguish  $A_r$  from  $A_1$ .

#### 4.1 Calculation of the Mean

Suppose that  $A$  is a  $p \times q$  random matrix, and is rotationally invariant under left- and right-multiplication by orthogonal matrices. We define

$$\begin{aligned} S_1(p, q) &= \mathbb{E} A_{11}^4 \quad (\text{diagonal}) \\ S_2(p, q) &= \mathbb{E} A_{21}^4 \quad (\text{off-diagonal}) \\ S_3(p, q) &= \mathbb{E} A_{i1}^2 A_{j1}^2 \quad (i \neq j) \quad (\text{same column}) \end{aligned}$$

$$\begin{aligned} S_4(p, q) &= \mathbb{E} A_{1i}^2 A_{1j}^2 \quad (i \neq j) \quad (\text{same row}) \\ S_5(p, q) &= \mathbb{E} A_{1i}^2 A_{2j}^2 \quad (i \neq j) \\ S_6(p, q) &= \mathbb{E} A_{ik} A_{il} A_{jk} A_{jl} \quad (i \neq j, k \neq l) \quad (\text{rectangle}) \end{aligned}$$

Since  $A$  is left- and right-invariant under rotations, these quantities are well-defined. Then

$$\begin{aligned} \mathbb{E} \operatorname{tr}((A^T A)^2) &= \mathbb{E} \sum_{1 \leq i, j \leq q} (A^T A)_{ij}^2 = \sum_{i=1}^q \mathbb{E} (A^T A)_{ii}^2 + \sum_{1 \leq i, j \leq q, i \neq j} \mathbb{E} (A^T A)_{ij}^2 \\ &= q \mathbb{E} (A^T A)_{11}^2 + q(q-1) \mathbb{E} (A^T A)_{12}^2 \end{aligned}$$

and

$$\begin{aligned} \mathbb{E} (A^T A)_{11}^2 &= \mathbb{E} \left( \sum_{i=1}^p A_{i1}^2 \right)^2 = \sum_{i=1}^p \mathbb{E} A_{i1}^4 + \sum_{1 \leq i, j \leq p, i \neq j} \mathbb{E} A_{i1}^2 A_{j1}^2 \\ &= \mathbb{E} A_{11}^4 + (p-1) \mathbb{E} A_{21}^4 + p(p-1) \mathbb{E} A_{11}^2 A_{21}^2 \\ &=: S_1(p, q) + (p-1) S_2(p, q) + p(p-1) S_3(p, q) \\ \mathbb{E} (A^T A)_{12}^2 &= \mathbb{E} \left( \sum_{i=1}^p A_{i1} A_{i2} \right)^2 = \sum_{i=1}^p \mathbb{E} A_{i1}^2 A_{i2}^2 + \sum_{1 \leq i, j \leq p, i \neq j} \mathbb{E} A_{i1} A_{i2} A_{j1} A_{j2} \\ &= p S_4(p, q) + p(p-1) S_6(p, q). \end{aligned}$$

When  $S_1(p, q) = S_2(p, q)$ , we have

$$\begin{aligned} \mathbb{E} \operatorname{tr}((A^T A)^2) &= q(p S_1(p, q) + p(p-1) S_3(p, q)) + q(q-1)(p S_4(p, q) + p(p-1) S_6(p, q)) \\ &= pq S_1(p, q) + pq(p-1) S_3(p, q) + pq(q-1) S_4(p, q) + p(p-1) q(q-1) S_6(p, q). \end{aligned}$$

When  $A = G$ , we have

$$S_1(p, q) = S_2(p, q) = 3, \quad S_3(p, q) = S_4(p, q) = S_5(p, q) = 1, \quad S_6(p, q) = 0$$

and so

$$\mathbb{E} \operatorname{tr}((A^T A)^2) = 3pq + pq(p-1) + pq(q-1) = pq(p+q+1).$$

Next, consider  $A = BG$ , where  $B$  is a  $p \times d$  random matrix and  $G$  a  $d \times q$  random matrix of i.i.d.  $N(0, 1)$  entries. The following proposition is easy to verify, and its proof is postponed to Appendix A.

► **Proposition 12.** *It holds that  $\mathbb{E} A_{21}^4 = \mathbb{E} A_{11}^4$ .*

Suppose that the associated functions of  $B$  are named  $T_1, T_2, T_3, T_4, T_6, T_5$ . Then we can calculate that (detailed calculations can be found in Appendix B)

$$\begin{aligned} S_1(p, q) &= 3dT_1(p, d) + 3d(d-1)T_4(p, d) \\ S_3(p, q) &= 3dT_3(p, d) + d(d-1)T_5(p, d) + 2d(d-1)T_6(p, d) \\ S_4(p, q) &= dT_1(p, d) + d(d-1)T_4(p, d) \\ S_5(p, q) &= dT_3(p, d) + d(d-1)T_5(p, d) \\ S_6(p, q) &= dT_3(p, d) + d(d-1)T_6(p, d) \end{aligned}$$

It is clear that  $S_1, S_3, S_4, S_5, S_6$  depend only on  $d$  (not on  $p$  and  $q$ ) if  $T_1, T_3, T_4, T_5, T_6$  do so. Furthermore, if  $T_1 = 3T_4$  then we have  $S_1 = 3S_4$  and thus  $S_4 = d(d+2)T_4$ . If  $T_3 = 2T_6 + T_5$  then  $S_3 = d(d+2)T_3$  and  $S_3 = 2S_6 + S_5$ . Hence, if  $T_3 = T_4$  then  $S_3 = S_4$ . We can verify

## 35:10 The Product of Gaussian Matrices Is Close to Gaussian

that all these conditions are satisfied with one Gaussian matrix and we can iterate it to obtain these quantities for the product of  $r$  Gaussian matrices with intermediate dimensions  $d_1, d_2, \dots, d_{r-1}$ . We have that

$$S_3 = S_4 = \prod_{i=1}^{r-1} d_i(d_i + 2), \quad S_1 = 3S_4, \quad S_6 = \sum_{j=1}^{r-1} \left( \prod_{i=1}^{j-1} d_i(d_i + 2) \right) d_j \left( \prod_{i=j+1}^{r-1} d_i(d_i - 1) \right).$$

Therefore, normalizing the  $i$ -th matrix by  $1/\sqrt{d_i}$ , that is,

$$A = \left( \frac{1}{\sqrt{d_1}} G_1 \right) \left( \frac{1}{\sqrt{d_2}} G_2 \right) \cdots \left( \frac{1}{\sqrt{d_{r-1}}} G_{r-1} \right) \left( \frac{1}{\sqrt{d_r}} G_r \right),$$

we have for constant  $r$  that

$$\begin{aligned} \mathbb{E} \operatorname{tr}((A^T A)^2) &= \frac{1}{d_1^2 d_2^2 \cdots d_{r-1}^2 d_r^2} (pq(p+q+1)S_3 + pq(p-1)(q-1)S_6) \\ &\approx \frac{pq(p+q+1)}{d_r^2} + \frac{pq(p-1)(q-1)}{d_r^2} \sum_{j=1}^{r-1} \frac{1}{d_j}. \end{aligned} \quad (4)$$

### 4.2 Calculation of the Variance

Let  $M \in \mathbb{R}^{p \times p}$  be a random symmetric matrix, and let  $G \in \mathbb{R}^{p \times q}$  be a random matrix of i.i.d.  $N(0, 1)$  entries. We want to find the variance of  $\operatorname{tr}((G^T M G)^2)$ . The detailed calculations of some steps can be found in Appendix C.

Our starting point is the law of total variance, which states that

$$\operatorname{Var}(\operatorname{tr}((G^T M G)^2)) = \mathbb{E}_M \left( \operatorname{Var}_G(\operatorname{tr}((G^T M G)^2)) \mid M \right) + \operatorname{Var}_M \left( \mathbb{E}_G \operatorname{tr}((G^T M G)^2) \mid M \right) \quad (5)$$

**Step 1a.** We shall handle each term separately. Consider the first term, which we shall bound using the Poincaré inequality for Gaussian measures. Define  $f(X) = \operatorname{tr}((X^T M X)^2)$ , where  $X \in \mathbb{R}^{p \times q}$ . We shall calculate  $\nabla f$ .

$$f(X) = \|X^T M X\|_F^2 = \sum_{1 \leq i, j \leq q} (X^T M X)_{ij}^2 = \sum_{1 \leq i, j \leq q} \left( \sum_{1 \leq k, l \leq p} M_{kl} X_{ki} X_{lj} \right)^2.$$

Then

$$\frac{\partial f}{\partial X_{rs}} = \sum_{1 \leq i, j \leq q} 2 \left( \sum_{1 \leq u, v \leq p} M_{uv} X_{ui} X_{vj} \right) \left( \sum_{1 \leq k, l \leq p} \frac{\partial}{\partial X_{rs}} (M_{kl} X_{ki} X_{lj}) \right).$$

Note that

$$\frac{\partial}{\partial X_{rs}} (M_{kl} X_{ki} X_{lj}) = \begin{cases} M_{kl} X_{lj}, & (k, i) = (r, s) \text{ and } (l, j) \neq (r, s) \\ M_{kl} X_{ki}, & (k, i) \neq (r, s) \text{ and } (l, j) = (r, s) \\ 2M_{rr} X_{rs}, & (k, i) = (r, s) \text{ and } (l, j) = (r, s) \\ 0, & \text{otherwise.} \end{cases}$$

we have that

$$\frac{\partial f}{\partial X_{rs}} = 4 \left( \sum_{1 \leq u, v \leq p} M_{uv} X_{us} X_{vs} \right) M_{rr} X_{rs} + 2 \sum_{(l, j) \neq (r, s)} \left( \sum_{1 \leq u, v \leq p} M_{uv} X_{us} X_{vj} \right) M_{rl} X_{lj}$$

$$\begin{aligned}
& + 2 \sum_{(k,i) \neq (r,s)} \left( \sum_{1 \leq u,v \leq p} M_{uv} X_{ui} X_{vs} \right) M_{kr} X_{ki} \\
& = 4 \left[ \left( \sum_{1 \leq u,v \leq p} M_{uv} X_{us} X_{vs} \right) M_{rr} X_{rs} + \sum_{(l,j) \neq (r,s)} \left( \sum_{u,v} M_{uv} X_{us} X_{vj} \right) M_{rl} X_{lj} \right] \\
& = 4 \sum_{l,j} \left( \sum_{u,v} M_{uv} X_{us} X_{vj} \right) M_{rl} X_{lj}.
\end{aligned}$$

Next we calculate  $\mathbb{E}(\partial f / \partial X_{rs})^2$  when  $X$  is i.i.d. Gaussian.

$$\left( \frac{1}{4} \frac{\partial f}{\partial X_{rs}} \right)^2 = \sum_{\substack{l,j \\ l',j'}} \sum_{\substack{u,v \\ u',v'}} M_{uv} M_{u'v'} M_{rl} M_{r'l'} \mathbb{E} X_{us} X_{u's} X_{vj} X_{l_j} X_{v'j'} X_{l'j'}$$

We discuss different cases of  $j, j', s$ .

When  $j \neq j' \neq s$ , it must hold that  $u = u', v = l$  and  $v' = l'$  for a possible nonzero contribution, and the total contribution in this case is at most  $q(q-1)B_{r,s}^{(1)}$ , where

$$B_{r,s}^{(1)} = \sum_{1 \leq l, l' \leq p} \sum_u M_{ul} M_{ul'} M_{rl} M_{r'l'} = \sum_u \langle M_{u,\cdot}, M_{r,\cdot} \rangle^2.$$

When  $j = j' \neq s$ , it must hold that  $u = u'$  for a possible nonzero contribution, and the total contribution in this case is at most  $(q-1)B_{r,s}^{(2)}$ , where

$$\begin{aligned}
B_{r,s}^{(2)} & = \sum_{l,l'} \sum_{u,v,v'} M_{uv} M_{u'v'} M_{rl} M_{r'l'} \mathbb{E} X_{us}^2 X_{vj} X_{l_j} X_{v'j} X_{l'j} \\
& = \|M\|_F^2 \|M_{r,\cdot}\|_2^2 + 2 \sum_u \langle M_{u,\cdot}, M_{r,\cdot} \rangle^2.
\end{aligned}$$

When  $j = s \neq j'$ , it must hold that  $v' = l'$  for possible nonzero contribution, and the total contribution in this case is at most  $(q-1)B_{r,s}^{(3)}$ , where

$$\begin{aligned}
B_{r,s}^{(3)} & = \sum_{j' \neq s} \left[ \sum_{l,l'} \sum_{u,v} M_{uv} M_{u'l'} M_{rl} M_{r'l'} \mathbb{E} X_{us} X_{u's} X_{vs} X_{l_s} X_{l'j'}^2 \right] \\
& = \sum_{l,l'} (2 \langle M_{l,\cdot}, M_{l',\cdot} \rangle + \text{tr}(M) M_{ll'}) M_{rl} M_{r'l'}.
\end{aligned}$$

When  $j = j' = s$ , the nonzero contribution is

$$B_{r,s}^{(4)} = \sum_{l,l'} \sum_{\substack{u,v \\ u',v'}} M_{uv} M_{u'v'} M_{rl} M_{r'l'} \mathbb{E} X_{us} X_{u's} X_{vs} X_{l_s} X_{v's} X_{l's}.$$

Since  $u, u', v, v', l, l'$  needs to be paired, the only case which is not covered by  $B_{r,s}^{(1)}$ ,  $B_{r,s}^{(3)}$  and  $B_{r,s}^{(4)}$  is when  $u = v, u' = v'$  and  $l = l'$ , in which case the contribution is at most

$$\sum_l \sum_{u,u'} M_{uu} M_{u'u'} M_{rl}^2 \mathbb{E} X_{us}^2 X_{u's}^2 X_{l_s}^2 \lesssim \text{tr}^2(M) \|M_{r,\cdot}\|_2^2.$$

Hence

$$B_{r,s}^{(4)} \lesssim B_{r,s}^{(1)} + B_{r,s}^{(2)} + B_{r,s}^{(3)} + \text{tr}^2(M) \|M_{r,\cdot}\|_2^2.$$

## 35:12 The Product of Gaussian Matrices Is Close to Gaussian

It follows that

$$\begin{aligned}
\sum_{r,s} B_{r,s}^{(1)} &= q \sum_{u,r} \langle M_{u,\cdot}, M_{r,\cdot} \rangle^2 = q \operatorname{tr}(M^4) \\
\sum_{r,s} B_{r,s}^{(2)} &= q \sum_r \|M\|_F^2 \|M_{r,\cdot}\|_2^2 + 2q \sum_{u,r} \langle M_{u,\cdot}, M_{r,\cdot} \rangle^2 = q \|M\|_F^4 + 2q \operatorname{tr}(M^4) \\
\sum_{r,s} B_{r,s}^{(3)} &= \sum_{r,s} \sum_{l,l'} (2 \langle M_{l,\cdot}, M_{l',\cdot} \rangle + \operatorname{tr}(M) M_{l'l'}) M_{rl} M_{r'l'} \\
&\leq 2q \operatorname{tr}(M^4) + q \operatorname{tr}(M) \|M\|_F \sqrt{\operatorname{tr}(M^4)}
\end{aligned}$$

Note that  $\operatorname{tr}(M^4) \leq \operatorname{tr}^2(M^2) = \|M\|_F^4$ . Hence

$$\begin{aligned}
\frac{1}{16} \mathbb{E} \|\nabla f\|_2^2 &\leq \sum_{r,s} ((q-1)(q-2)B_{rs}^{(1)} + (q-1)B_{rs}^{(2)} + (q-1)B_{rs}^{(3)} + B_{rs}^{(4)}) \\
&\lesssim \sum_{r,s} (q^2 B_{rs}^{(1)} + qB_{rs}^{(2)} + qB_{rs}^{(3)} + \operatorname{tr}^2(M) \|M_{r,\cdot}\|_2^2) \\
&\lesssim q^3 \operatorname{tr}(M^4) + q^2 \|M\|_F^4 + q^2 \operatorname{tr}(M) \|M\|_F \sqrt{\operatorname{tr}(M^4)} + q \operatorname{tr}^2(M) \|M\|_F^2.
\end{aligned}$$

By the Gaussian Poincaré inequality,

$$\begin{aligned}
&\operatorname{Var}_G(\operatorname{tr}((G^T M G)^2) | M) \\
&\lesssim \mathbb{E} \|\nabla f\|_2^2 \\
&\lesssim q^3 \operatorname{tr}(M^4) + q^2 \|M\|_F^4 + q^2 \operatorname{tr}(M) \|M\|_F \sqrt{\operatorname{tr}(M^4)} + q \operatorname{tr}^2(M) \|M\|_F^2.
\end{aligned} \tag{6}$$

For the terms on the right-hand side, we calculate that (using the trace inequality (Lemma 5))

$$\begin{aligned}
\mathbb{E} \operatorname{tr}((G^T M G)^4) &= \mathbb{E} \operatorname{tr}((M G G^T)^4) \leq \mathbb{E} \|G G^T\|_{op}^4 \operatorname{tr}(M^4) = \mathbb{E} \|G\|_{op}^8 \operatorname{tr}(M^4) \\
&\lesssim \max\{p, q\}^4 \operatorname{tr}(M^4), \\
\mathbb{E} \|G^T M G\|_F^4 &\leq \mathbb{E} \|G\|_{op}^8 \|M\|_F^4 \lesssim \max\{p, q\}^4 \|M\|_F^4, \\
\mathbb{E} \operatorname{tr}^2(G^T M G) \|G^T M G\|_F^2 &\leq \mathbb{E} \|G\|_{op}^8 \operatorname{tr}^2(M) \|M\|_F^2 \lesssim \max\{p, q\}^4 \operatorname{tr}^2(M) \|M\|_F^2
\end{aligned}$$

and

$$\begin{aligned}
&\mathbb{E} \operatorname{tr}(G^T M G) \|G^T M G\|_F \sqrt{\operatorname{tr}((G^T M G)^4)} \\
&\leq \mathbb{E} \|G\|_{op}^2 \operatorname{tr}(G) \cdot \|G\|_{op}^2 \|M\|_F^2 \cdot \sqrt{\|G\|_{op}^8 \operatorname{tr}(M^4)} \\
&= \mathbb{E} \|G\|_{op}^8 \operatorname{tr}(M) \|M\|_F \sqrt{\operatorname{tr}(M^4)} \\
&\lesssim \max\{p, q\}^4 \operatorname{tr}(M) \|M\|_F \sqrt{\operatorname{tr}(M^4)}.
\end{aligned}$$

This implies that each term on the right-hand of (6) grows geometrically.

**Step 1b.** Next we deal with the second term in (5). We have

$$\begin{aligned}
\mathbb{E}_G \operatorname{tr}((G^T M G)^2) &= \sum_{i,j} \mathbb{E}_G (G^T M G)_{ij}^2 = \sum_{i,j} \mathbb{E}_G \left( \sum_{k,l} M_{kl} G_{ki} G_{lj} \right)^2 \\
&= \sum_{i,j} \sum_{k,l,k',l'} M_{kl} M_{k'l'} \mathbb{E}_G G_{ki} G_{lj} G_{k'i} G_{l'j}.
\end{aligned}$$

When  $i \neq j$ , for non-zero contribution, it must hold that  $k = l$  and  $k' = l'$  and thus the nonzero contribution is

$$\sum_{i \neq j} \sum_{k,l} M_{kl}^2 = q(q-1) \|M\|_F^2.$$

When  $i = j$ , the contribution is

$$\sum_i \sum_{k,l,k',l'} M_{kl} M_{k'l'} \mathbb{E} G_{ki} G_{li} G_{k'i} G_{l'i} = 2q \|M\|_F^2 + q \operatorname{tr}^2(M). \quad (7)$$

Hence

$$\mathbb{E}_G \operatorname{tr}((G^T M G)^2) = q(q+1) \|M\|_F^2 + q \operatorname{tr}^2(M)$$

and when  $M$  is random,

$$\begin{aligned} & \operatorname{Var}(\mathbb{E} \operatorname{tr}((G^T M G)^2) | M) \\ &= \operatorname{Var}\left(q(q+1) \|M\|_F^2 + q \operatorname{tr}^2(M)\right) \\ &\leq q^2(q+1)^2 \operatorname{Var}(\|M\|_F^2) + q^2 \operatorname{Var}(\operatorname{tr}^2(M)) + 2q^2(q+1) \sqrt{\operatorname{Var}(\|M\|_F^2) \operatorname{Var}(\operatorname{tr}^2(M))}. \end{aligned} \quad (8)$$

**Step 2a.** Note that the  $\operatorname{Var}(\operatorname{tr}^2(M))$  term on the right-hand side of (8). To bound this term, we examine the variance of  $g(G)$ , where  $g(X) = \operatorname{tr}^2(X^T M X)$ . We shall again calculate  $\nabla g$ . Note that

$$\frac{\partial g}{\partial X_{rs}} = 2 \operatorname{tr}(X^T M X) \sum_i \sum_{k,l} M_{kl} \frac{\partial}{\partial X_{rs}} X_{ki} X_{li}$$

and

$$\frac{\partial}{\partial X_{rs}} (X_{ki} X_{li}) = \begin{cases} X_{li}, & (k, i) = (r, s) \text{ and } (l, i) \neq (r, s) \\ X_{ki}, & (k, i) \neq (r, s) \text{ and } (l, i) = (r, s) \\ 2X_{rs}, & (k, i) = (r, s) \text{ and } (l, i) = (r, s) \\ 0, & \text{otherwise.} \end{cases}$$

We have

$$\frac{\partial g}{\partial X_{rs}} = 4 \operatorname{tr}(X^T M X) \sum_l M_{rl} X_{ls} = 4 \sum_{\substack{1 \leq j \leq q \\ 1 \leq l, u, v \leq p}} M_{uv} M_{rl} X_{ls} X_{uj} X_{vj}$$

Next we calculate  $\mathbb{E}(\partial g / \partial X_{rs})^2$  when  $X$  is i.i.d. Gaussian.

$$\left(\frac{1}{4} \frac{\partial g}{\partial X_{rs}}\right)^2 = \sum_{\substack{j,l,u,v \\ j',l',u',v'}} M_{uv} M_{u'v'} M_{rl} M_{r'l'} \mathbb{E} X_{ls} X_{l's} X_{uj} X_{vj} X_{u'j'} X_{v'j'}$$

In order for the expectation in the summand to be non-zero, we must have one of the following cases: (1)  $s \neq j \neq j'$ , (2)  $s = j \neq j'$ , (3)  $s = j' \neq j$ , (4)  $s \neq j = j'$ , (5)  $s = j = j'$ . We calculate the contribution in each case below.

### 35:14 The Product of Gaussian Matrices Is Close to Gaussian

Case 1: it must hold that  $l = l'$ ,  $u = v$  and  $u' = v'$ . The contribution is  $(q-1)(q-2)B_{rs}^{(1)}$ , where

$$B_{rs}^{(1)} = \sum_{l,u,u'} M_{uu} M_{u'u'} M_{rl}^2 = \text{tr}^2(M) \|M_{r,\cdot}\|_2^2.$$

Case 2: it must hold that  $u' = v'$ . The contribution is  $(q-1)B_{rs}^{(2)}$ , where

$$\begin{aligned} B_{rs}^{(2)} &= \sum_{l,l',u,u',v} M_{uv} M_{u'u'} M_{rl} M_{r'l'} \mathbb{E} X_{ls} X_{l's} X_{us} X_{vs} X_{u'j'}^2 \\ &= \text{tr}(M) \left( \text{tr}(M) \|M_{r,\cdot}\|_2^2 + 2 \sum_{l,l'} M_{ll'} M_{rl} M_{r'l'} \right) \end{aligned}$$

Case 3: this gives the same bound as Case 2.

Case 4: it must hold that  $l = l'$ . The contribution is  $(q-1)B_{rs}^{(4)}$ , where

$$B_{rs}^{(4)} = \sum_{l,u,u',v,v'} M_{uv} M_{u'v'} M_{rl}^2 \mathbb{E} X_{uj} X_{vj} X_{u'j} X_{v'j} = 3 \|M_{r,\cdot}\|_2^2 \|M\|_F^2$$

Case 5: the contribution is  $B_{rs}^{(5)}$ , where

$$B_{rs}^{(5)} = \sum_{\substack{l,u,v \\ l',u',v'}} M_{uv} M_{u'v'} M_{rl} M_{r'l'} \mathbb{E} X_{ls} X_{us} X_{vs} X_{l's} X_{u's} X_{v's}.$$

The only uncovered case is  $l = u'$ ,  $l' = v$ ,  $u = v'$  and its symmetries. In such a case the contribution is at most

$$C \sum_{l,u,v} M_{uv} M_{lu} M_{rl} M_{rv} = C \sum_u \langle M_{r,\cdot}, M_{u,\cdot} \rangle^2.$$

Note that

$$\begin{aligned} \sum_{r,s} B_{rs}^{(1)} &= q \text{tr}^2(M) \|M\|_F^2, \\ \sum_{r,s} B_{rs}^{(2)} &= q \text{tr}^2(M) \|M\|_F^2 + 2q \text{tr}(M) \sum_{l,l'} M_{ll'} \langle M_{l,\cdot}, M_{l',\cdot} \rangle \\ &\leq q \text{tr}^2(M) \|M\|_F^2 + 2q \text{tr}(M) \|M\|_F \sqrt{\text{tr}(M^4)}, \\ \sum_{r,s} B_{rs}^{(4)} &= q \|M\|_F^4, \\ \sum_{r,s} B_{rs}^{(5)} &\lesssim \sum_{r,s} B_{rs}^{(1)} + \sum_{r,s} B_{rs}^{(2)} + \text{tr}(M^4). \end{aligned}$$

Therefore,

$$\begin{aligned} \frac{1}{16} \mathbb{E} \|\nabla g\|_2^2 &\leq \sum_{r,s} ((q-1)(q-2)B_{rs}^{(1)} + (q-1)B_{rs}^{(2)} + (q-1)B_{rs}^{(4)} + B_{rs}^{(5)}) \\ &\lesssim q^3 \text{tr}^2(M) \|M\|_F^2 + q^2 \text{tr}(M) \|M\|_F \sqrt{\text{tr}(M^4)} + q^2 \|M\|_F^4 + q \text{tr}(M^4). \end{aligned}$$

By Poincaré's inequality,

$$\begin{aligned} &\text{Var}_G(\text{tr}^2(G^T M G)) \\ &\lesssim \mathbb{E} \|\nabla g\|_2^2 \\ &\lesssim q^3 \text{tr}^2(M) \|M\|_F^2 + q^2 \text{tr}(M) \|M\|_F \sqrt{\text{tr}(M^4)} + q^2 \|M\|_F^4 + q \text{tr}(M^4). \end{aligned} \tag{9}$$

Similar to before, each term on the right-hand side grows geometrically.



**Step 2b.** Next we deal with  $\text{Var}_M(\mathbb{E}_G \text{tr}^2(G^T M G) | M)$ .

$$\mathbb{E} \text{tr}^2(G^T M G) = \mathbb{E} \left( \sum_{i,k,l} M_{kl} G_{ki} G_{li} \right)^2 = \sum_{i,j} \sum_{k,l,k',l'} M_{kl} M_{k'l'} \mathbb{E} G_{ki} G_{li} G_{k'j} G_{l'j}.$$

When  $i \neq j$ , for non-zero contribution, it must hold that  $k = l$  and  $k' = l'$  and thus the nonzero contribution is

$$\sum_{i \neq j} \sum_{k,k'} M_{kk} M_{k'k'} = q(q-1) \text{tr}^2(M).$$

When  $i = j$ , the contribution is (this is exactly the same as (7) in Step 1b.)

$$\sum_i \sum_{k,k',l,l'} M_{kl} M_{k'l'} \mathbb{E} G_{ki} G_{li} G_{k'i} G_{l'i} = 2q \|M\|_F^2 + q \text{tr}^2(M).$$

Hence

$$\mathbb{E} \text{tr}^2(G^T M G) = 2q \|M\|_F^2 + q^2 \text{tr}^2(M)$$

and when  $M$  is random,

$$\begin{aligned} & \text{Var}(\mathbb{E} \text{tr}^2(G^T M G) | M) \\ &= \text{Var}(2q \|M\|_F^2 + q^2 \text{tr}^2(M)) \\ &\leq 4q^2 \text{Var}(\|M\|_F^2) + q^4 \text{Var}(\text{tr}^2(M)) + 2q^3 \sqrt{\text{Var}(\|M\|_F^2) \text{Var}(\text{tr}^2(M))}. \end{aligned} \tag{10}$$

**Step 3.** Let  $U_r$  denote the variance of  $\text{tr}((A_r^T A_r)^2)$  and  $V_r$  the variance of  $\text{tr}^2(A_r^T A_r)$ . Combining (5), (6), (8), (9), (10), we have the following recurrence relations, where  $C_1, C_2, C_3, C_4 > 0$  are absolute constants.

$$\begin{aligned} U_{r+1} &\leq C_1 P_r + 2U_r + \frac{1}{d_r^2} V_r + \frac{3}{d_r} \sqrt{U_r V_r} \\ V_{r+1} &\leq C_2 Q_r + \frac{1}{d_r^2} U_r + V_r + \frac{2}{d_r} \sqrt{U_r V_r} \\ P_{r+1} &\leq C_3 P_r \\ Q_{r+1} &\leq C_4 Q_r \\ U_0 &= V_0 = 0 \end{aligned}$$

In the base case, set  $M = I_p$  (the  $p \times p$  identity matrix in (6)) and note that the second term in (5) vanishes. We see that  $P_1 \lesssim (p^3 q + pq^3)/d_1^4$  after proper normalization. (Alternatively we can calculate this precisely, see Appendix D.) Similarly we have  $Q_1 \lesssim p^3 q^3/d_1^4$ . Note that  $Q_1/d_1^2 \lesssim (p^3 q + pq^3)/d_1^4$ . Now, we can solve that

$$U_{r+1} \leq C^r \frac{p^3 q + pq^3}{d_1^4}$$

for some absolute constant  $C > 0$ .

## References

- 1 Gernot Akemann and Jesper R Ipsen. Recent exact and asymptotic results for products of independent random matrices. *Acta Physica Polonica B*, pages 1747–1784, 2015.
- 2 Ahmed El Alaoui and Michael W. Mahoney. Fast randomized kernel ridge regression with statistical guarantees. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS’15, page 775–783, Cambridge, MA, USA, 2015. MIT Press.
- 3 Richard Bellman. Limit theorems for non-commutative operations. I. *Duke Mathematical Journal*, 21(3):491–500, 1954.
- 4 Giancarlo Benettin. Power-law behavior of lyapunov exponents in some conservative dynamical systems. *Physica D: Nonlinear Phenomena*, 13(1-2):211–220, 1984.
- 5 Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.
- 6 Matthew Brennan, Guy Bresler, and Dheeraj Nagaraj. Phase transitions for detecting latent geometry in random graphs. *Probability Theory and Related Fields*, pages 1215–1289, 2020. doi:10.1007/s00440-020-00998-3.
- 7 Sébastien Bubeck, Jian Ding, Ronen Eldan, and Miklós Z. Rácz. Testing for high-dimensional geometry in random graphs. *Random Structures & Algorithms*, 49(3):503–532, 2016. doi:10.1002/rsa.20633.
- 8 Sébastien Bubeck and Shirshendu Ganguly. Entropic CLT and Phase Transition in High-dimensional Wishart Matrices. *International Mathematics Research Notices*, 2018(2):588–606, December 2016. doi:10.1093/imrn/rnw243.
- 9 Z. Burda, R. A. Janik, and B. Waclaw. Spectrum of the product of independent random gaussian matrices. *Physical Review E*, 81(4), April 2010. doi:10.1103/physreve.81.041132.
- 10 Emmanuel J Candes. The restricted isometry property and its implications for compressed sensing. *Comptes rendus mathématique*, 346(9-10):589–592, 2008.
- 11 Andrea Crisanti, Giovanni Paladin, and Angelo Vulpiani. *Products of random matrices: in Statistical Physics*, volume 104. Springer Science & Business Media, 2012.
- 12 David L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.
- 13 S. Iida, H.A. Weidenmüller, and J.A. Zuk. Statistical scattering theory, the supersymmetry method and universal conductance fluctuations. *Annals of Physics*, 200(2):219–270, 1990.
- 14 Jesper R. Ipsen. *Products of independent Gaussian random matrices*. PhD thesis, Bielefeld University, 2015.
- 15 Hiroshi Ishitani. A central limit theorem for the subadditive process and its application to products of random matrices. *Publications of the Research Institute for Mathematical Sciences*, 12(3):565–575, 1977.
- 16 Tiefeng Jiang. How many entries of a typical orthogonal matrix can be approximated by independent normals? *The Annals of Probability*, 34(4):1497–1529, July 2006. doi:10.1214/009117906000000205.
- 17 Tiefeng Jiang and Yutao Ma. Distances between random orthogonal matrices and independent normals. *Transactions of the American Mathematical Society*, 372(3):1509–1553, 2019. doi:10.1090/tran/7470.
- 18 Ravindran Kannan and Santosh Vempala. *Spectral algorithms*. Now Publishers Inc, 2009.
- 19 Michael Kapralov, Vamsi Potluru, and David Woodruff. How to fake multiply by a gaussian matrix. In *International Conference on Machine Learning*, pages 2101–2110. PMLR, 2016.
- 20 Michael W. Mahoney. Randomized algorithms for matrices and data. *Found. Trends Mach. Learn.*, 3(2):123–224, 2011. doi:10.1561/22000000035.
- 21 Satya N Majumdar and Grégory Schehr. Top eigenvalue of a random matrix: large deviations and third order phase transition. *Journal of Statistical Mechanics: Theory and Experiment*, 2014(1):P01012, 2014.
- 22 Robert M May. Will a large complex system be stable? *Nature*, 238(5364):413–414, 1972.

- 23 P.A. Mello, P. Pereyra, and N. Kumar. Macroscopic approach to multichannel disordered conductors. *Annals of Physics*, 181(2):290–317, 1988.
- 24 Ralf R Muller. On the asymptotic eigenvalue distribution of concatenated vector-valued fading channels. *IEEE Transactions on Information Theory*, 48(7):2086–2091, 2002.
- 25 Shanmugavelayutham Muthukrishnan. *Data streams: Algorithms and applications*. Now Publishers Inc, 2005.
- 26 Guillaume Obozinski, Martin J Wainwright, Michael I Jordan, et al. Support union recovery in high-dimensional multivariate regression. *The Annals of Statistics*, 39(1):1–47, 2011.
- 27 James C Osborn. Universal results from an alternate random-matrix model for QCD with a baryon chemical potential. *Physical review letters*, 93(22):222001, 2004.
- 28 G Paladin and A Vulpiani. Scaling law and asymptotic distribution of lyapunov exponents in conservative dynamical systems with many degrees of freedom. *Journal of Physics A: Mathematical and General*, 19(10):1881, 1986.
- 29 Saurabh Paul, Christos Boutsidis, Malik Magdon-Ismail, and Petros Drineas. Random projections for linear support vector machines. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(4):1–25, 2014.
- 30 Miklós Z. Rácz and Jacob Richey. A smooth transition from wishart to goe. *Journal of Theoretical Probability*, pages 898–906, 2019. doi:10.1007/s10959-018-0808-2.
- 31 Garvesh Raskutti and Michael W Mahoney. A statistical perspective on randomized sketching for ordinary least-squares. *The Journal of Machine Learning Research*, 17(1):7508–7538, 2016.
- 32 Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 143–152. IEEE, 2006.
- 33 Khalid Shebrawi and Hussein Albadawi. Trace inequalities for matrices. *Bulletin of the Australian Mathematical Society*, 87(1):139–148, 2013. doi:10.1017/S0004972712000627.
- 34 Terence Tao. *Topics in random matrix theory*, volume 132. American Mathematical Soc., 2012.
- 35 Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. In Yonina C. Eldar and Gitta Kutyniok, editors, *Compressed Sensing: Theory and Applications*, pages 210–268. Cambridge University Press, 2012. doi:10.1017/CB09780511794308.006.
- 36 David P. Woodruff. Sketching as a tool for numerical linear algebra. *Found. Trends Theor. Comput. Sci.*, 10(1–2):1–157, 2014. doi:10.1561/04000000060.
- 37 Fan Yang, Sifan Liu, Edgar Dobriban, and David P Woodruff. How to reduce dimension with PCA and random projections? arXiv:2005.00511 [math.ST], 2020.

## A Proof of Proposition 12

**Proof.** We have

$$\begin{aligned} \mathbb{E} A_{11}^4 &= \mathbb{E} \left( \sum_i B_{1i} G_{i1} \right)^4 = \sum_{i,j,k,l} \mathbb{E} B_{1i} B_{1j} B_{1k} B_{1l} \mathbb{E} G_{i1} G_{j1} G_{k1} G_{l1} \\ &= 3 \sum_i \mathbb{E} B_{1i}^4 + 3 \sum_{i \neq j} \mathbb{E} B_{1i}^2 B_{1j}^2 \end{aligned}$$

and

$$\begin{aligned} \mathbb{E} A_{21}^4 &= \mathbb{E} \left( \sum_i B_{2i} G_{i1} \right)^4 = \sum_{i,j,k,l} \mathbb{E} B_{2i} B_{2j} B_{2k} B_{2l} \mathbb{E} G_{i1} G_{j1} G_{k1} G_{l1} \\ &= 3 \sum_i \mathbb{E} B_{2i}^4 + 3 \sum_{i \neq j} \mathbb{E} B_{2i}^2 B_{2j}^2 = \mathbb{E} A_{11}^4. \end{aligned} \quad \blacktriangleleft$$

**B** Omitted Calculations in Section 4.1

$$S_1(p, q) = 3 \sum_i \mathbb{E} B_{1i}^4 + 3 \sum_{i \neq j} \mathbb{E} B_{1i}^2 B_{1j}^2 = 3dT_1(p, d) + 3d(d-1)T_4(p, d)$$

$$\begin{aligned} S_3(p, q) &= \mathbb{E} A_{11}^2 A_{21}^2 = \mathbb{E} \left( \sum_i B_{1i} G_{i1} \right)^2 \left( \sum_k B_{2k} G_{k1} \right)^2 \\ &= \sum_{i,j,k,l} \mathbb{E} B_{1i} B_{1j} B_{2k} B_{2l} \mathbb{E} G_{i1} G_{j1} G_{k1} G_{l1} \\ &= 3 \sum_i \mathbb{E} B_{1i}^2 B_{2i}^2 + \sum_{i \neq j} \mathbb{E} B_{1i}^2 B_{2j}^2 + 2 \sum_{i \neq j} \mathbb{E} B_{1i} B_{2i} B_{1j} B_{2j} \\ &= 3dT_3(p, d) + d(d-1)T_5(p, d) + 2d(d-1)T_6(p, d) \end{aligned}$$

$$\begin{aligned} S_4(p, q) &= \mathbb{E} A_{11}^2 A_{12}^2 = \mathbb{E} \left( \sum_i B_{1i} G_{i1} \right)^2 \left( \sum_k B_{1k} G_{k2} \right)^2 \\ &= \sum_{i,j,k,l} \mathbb{E} B_{1i} B_{1j} B_{1k} B_{1l} \mathbb{E} G_{i1} G_{j1} G_{k2} G_{l2} \\ &= \sum_i \mathbb{E} B_{1i}^4 + \sum_{i \neq j} \mathbb{E} B_{1i}^2 B_{1j}^2 = dT_1(p, d) + d(d-1)T_4(p, d). \end{aligned}$$

$$\begin{aligned} S_5(p, q) &= \mathbb{E} A_{11}^2 A_{22}^2 = \mathbb{E} \left( \sum_i B_{1i} G_{i1} \right)^2 \left( \sum_k B_{2k} G_{k2} \right)^2 \\ &= \sum_{i,j,k,l} \mathbb{E} B_{1i} B_{1j} B_{2k} B_{2l} \mathbb{E} G_{i1} G_{j1} G_{k2} G_{l2} \\ &= \sum_{i,j} \mathbb{E} B_{1i}^2 B_{2j}^2 = dT_3(p, d) + d(d-1)T_5(p, d) \end{aligned}$$

$$\begin{aligned} S_6(p, q) &= \mathbb{E} A_{11} A_{12} A_{21} A_{22} = \sum_{i,j,k,l} \mathbb{E} B_{1i} B_{1j} B_{2k} B_{2l} \mathbb{E} G_{i1} G_{j2} G_{k1} G_{l2} \\ &= \sum_i \mathbb{E} B_{1i}^2 B_{2i}^2 + \sum_{i \neq j} \mathbb{E} B_{1i} B_{1j} B_{2i} B_{2j} \\ &= dT_3(p, d) + d(d-1)T_6(p, d) \end{aligned}$$

## C

 Omitted Calculations in Section 4.2

In Step 1a.

$$\begin{aligned}
B_{r,s}^{(2)} &= \sum_{l,l'} \sum_{u,v,v'} M_{uv} M_{uv'} M_{rl} M_{r'l'} \mathbb{E} X_{us}^2 X_{vj} X_{lj} X_{v'j} X_{l'j} \\
&= \underbrace{\sum_{l \neq l'} \sum_u M_{ul} M_{ul'} M_{rl} M_{r'l'}}_{v=l \neq v'=l'} + \underbrace{\sum_l \sum_{\substack{u \\ v \neq l}} M_{uv}^2 M_{rl}^2}_{v=v' \neq l=l'} + \underbrace{\sum_{l \neq l'} \sum_u M_{ul'} M_{ul} M_{rl} M_{r'l'}}_{v=l' \neq l=v'} \\
&\quad + 3 \underbrace{\sum_{l,u} M_{ul}^2 M_{rl}^2}_{v=v'=l=l'} \\
&= \left( \sum_{u,v} M_{uv}^2 \right) \left( \sum_l M_{rl}^2 \right) + 2 \sum_{l,l',u} M_{ul'} M_{ul} M_{rl} M_{r'l'} \\
&= \|M\|_F^2 \|M_{r,\cdot}\|_2^2 + 2 \sum_u \langle M_{u,\cdot}, M_{r,\cdot} \rangle^2. \\
\\
B_{r,s}^{(3)} &= \sum_{j' \neq s} \left[ \sum_{l,l'} \sum_{u,v} M_{uv} M_{u'l'} M_{rl} M_{r'l'} \mathbb{E} X_{us} X_{u's} X_{vs} X_{l's} X_{l'j'}^2 \right] \\
&= \underbrace{\sum_{l,l'} \sum_{u \neq l} M_{ul} M_{ul'} M_{rl} M_{r'l'}}_{u=u' \neq v=l} + \underbrace{\sum_{l,l'} \sum_{u \neq l} M_{uu} M_{ll'} M_{rl} M_{r'l'}}_{u=v \neq u'=l} + \underbrace{\sum_{l,l'} \sum_v M_{lv} M_{v'l'} M_{rl} M_{r'l'}}_{u=l \neq u'=v} \\
&\quad + 3 \underbrace{\sum_{l,l'} M_{ll} M_{ll'} M_{rl} M_{r'l'}}_{u=u'=v=l} \\
&= \sum_{l,l'} \sum_u (2M_{ul} M_{ul'} + M_{uu} M_{ll'}) M_{rl} M_{r'l'} \\
&= \sum_{l,l'} (2\langle M_{l,\cdot}, M_{l',\cdot} \rangle + \text{tr}(M) M_{ll'}) M_{rl} M_{r'l'}. \\
\\
\sum_{r,s} B_{r,s}^{(3)} &= \sum_{r,s} \sum_{l,l'} (2\langle M_{l,\cdot}, M_{l',\cdot} \rangle + \text{tr}(M) M_{ll'}) M_{rl} M_{r'l'} \\
&= q \sum_{l,l'} (2\langle M_{l,\cdot}, M_{l',\cdot} \rangle + \text{tr}(M) M_{ll'}) \sum_r M_{rl} M_{r'l'} \\
&= q \sum_{l,l'} (2\langle M_{l,\cdot}, M_{l',\cdot} \rangle + \text{tr}(M) M_{ll'}) \langle M_{l,\cdot}, M_{l',\cdot} \rangle \\
&= q \sum_{l,l'} 2\langle M_{l,\cdot}, M_{l',\cdot} \rangle^2 + q \text{tr}(M) \sum_{l,l'} M_{ll'} \langle M_{l,\cdot}, M_{l',\cdot} \rangle \\
&\leq 2q \text{tr}(M^4) + q \text{tr}(M) \left( \sum_{l,l'} M_{ll'}^2 \right)^{\frac{1}{2}} \left( \sum_{l,l'} \langle M_{l,\cdot}, M_{l',\cdot} \rangle^2 \right)^{\frac{1}{2}} \\
&\leq 2q \text{tr}(M^4) + q \text{tr}(M) \|M\|_F \sqrt{\text{tr}(M^4)}
\end{aligned}$$

In Step 1b.

$$\begin{aligned}
 & \sum_i \sum_{\substack{k,l \\ k',l'}} M_{kl} M_{k'l'} \mathbb{E} G_{ki} G_{li} G_{k'i} G_{l'i} \\
 &= \sum_i \left( \underbrace{\sum_{k \neq l} M_{kl}^2}_{k=k' \neq l=l'} + \underbrace{\sum_{k \neq l} M_{kl}^2}_{k=l' \neq k'=l} + \underbrace{\sum_{k \neq l} M_{kk} M_{k'l'}}_{k=l \neq k'=l'} + 3 \underbrace{\sum_k M_{kk}^2}_{k=k'=l=l'} \right) \\
 &= \sum_i \left( \sum_{k,l} M_{kl}^2 + \sum_{k,l} M_{kl}^2 + \sum_{k,l} M_{kk} M_{k'l'} \right) \\
 &= \sum_i (2 \|M\|_F^2 + \text{tr}^2(M)) \\
 &= 2q \|M\|_F^2 + q \text{tr}^2(M).
 \end{aligned}$$

In Step 2a.

$$\begin{aligned}
 B_{rs}^{(2)} &= \sum_{l,l',u,u',v} M_{uv} M_{u'u'} M_{rl} M_{r'l'} \mathbb{E} X_{ls} X_{l's} X_{us} X_{v's} X_{u'j'}^2 \\
 &= \underbrace{\sum_{\substack{l \neq u \\ u'}} M_{uu} M_{u'u'} M_{rl}^2}_{l=l' \neq u=v} + \underbrace{\sum_{\substack{l \neq l' \\ u'}} M_{ll'} M_{u'u'} M_{rl} M_{r'l'}}_{l=u \neq l'=v} + \underbrace{\sum_{\substack{l \neq l' \\ u'}} M_{ll'} M_{u'u'} M_{rl} M_{r'l'}}_{l=v \neq l'=u} \\
 &\quad + 3 \underbrace{\sum_{l,u'} M_{ll} M_{u'u'} M_{rl}^2}_{l=u=l'=v} \\
 &= \text{tr}(M) \left( \sum_{l,u} M_{uu} M_{rl}^2 + 2 \sum_{l,l'} M_{ll'} M_{rl} M_{r'l'} \right) \\
 &= \text{tr}(M) \left( \text{tr}(M) \|M_{r,\cdot}\|_2^2 + 2 \sum_{l,l'} M_{ll'} M_{rl} M_{r'l'} \right)
 \end{aligned}$$

$$\begin{aligned}
 B_{rs}^{(4)} &= \sum_{l,u,u',v,v'} M_{uv} M_{u'v'} M_{rl}^2 \mathbb{E} X_{uj} X_{vj} X_{u'j} X_{v'j} \\
 &= \left( \sum_l M_{rl}^2 \right) \left( \underbrace{\sum_{u,v} M_{uv}^2}_{u=u' \neq v=v'} + \underbrace{\sum_{u,v} M_{uv}^2}_{u=v' \neq u=v} + \underbrace{\sum_{u,v} M_{uv}^2}_{u=v \neq u'=v'} + 3 \underbrace{\sum_u M_{uu}^2}_{u=v=u'=v'} \right) \\
 &= 3 \left( \sum_l M_{rl}^2 \right) \sum_{u,v} M_{u,v}^2 \\
 &= 3 \|M_{r,\cdot}\|_2^2 \|M\|_F^2
 \end{aligned}$$

### D Exact Variance when $r = 2$

Suppose that  $A$  is rotationally invariant under both left- and right-multiplication of an orthogonal matrix. Define

$$\begin{aligned}
U_1(p, q) &= \text{Var}((A^T A)_{ii}^2) \\
U_2(p, q) &= \text{Var}((A^T A)_{ij}^2) \quad i \neq j \\
U_3(p, q) &= \text{cov}((A^T A)_{ii}^2, (A^T A)_{ik}^2) \quad i \neq k \quad (\text{same row, one entry on diagonal}) \\
U_4(p, q) &= \text{cov}((A^T A)_{ij}^2, (A^T A)_{ik}^2) \quad j \neq k \quad (\text{same row, both entries off-diagonal}) \\
U_5(p, q) &= \text{cov}((A^T A)_{ii}^2, (A^T A)_{jj}^2) \quad i \neq j \quad (\text{diff. rows and cols, both entries on diagonal}) \\
U_6(p, q) &= \text{cov}((A^T A)_{ii}^2, (A^T A)_{jk}^2) \quad i \neq j \neq k \quad (\text{diff. rows and cols, one entry on diagonal}) \\
U_7(p, q) &= \text{cov}((A^T A)_{ij}^2, (A^T A)_{kl}^2) \quad i \neq j \neq k \neq l \quad (\text{diff. rows and cols, nonsymmetric around diag.})
\end{aligned}$$

It is clear that they are well-defined.

$$\begin{aligned}
& \text{Var}(\text{tr}((A^T A)^2)) \\
&= \text{Var}\left(\sum_{i,j} (A^T A)_{ij}^2\right) \\
&= \sum_{i,j,k,l} \text{cov}((A^T A)_{ij}^2, (A^T A)_{kl}^2) \\
&= \sum_{i,j} \text{Var}((A^T A)_{ij}^2) + 2 \sum_i \sum_{j \neq l} \text{cov}((A^T A)_{ij}^2, (A^T A)_{il}^2) + \sum_{\substack{i \neq k \\ j \neq l}} \text{cov}(\mathbb{E}(A^T A)_{ij}^2, (A^T A)_{kl}^2) \\
&= q \text{Var}((A^T A)_{11}^2) + q(q-1) \text{Var}(\mathbb{E}(A^T A)_{12}^2) \\
&\quad + 2 [2q(q-1) \text{cov}((A^T A)_{11}^2, (A^T A)_{12}^2) + q(q-1)(q-2) \text{cov}((A^T A)_{12}^2, (A^T A)_{13}^2)] \\
&\quad + q(q-1) \text{cov}(\mathbb{E}(A^T A)_{11}^2, (A^T A)_{22}^2) + q(q-1) \text{cov}(\mathbb{E}(A^T A)_{12}^2, (A^T A)_{21}^2) \\
&\quad + 2q(q-1)(q-2) \text{cov}((A^T A)_{11}^2, (A^T A)_{23}^2) \\
&\quad + 2q(q-1)(q-2) \text{cov}((A^T A)_{12}^2, (A^T A)_{31}^2) \\
&\quad + q(q-1)(q-2)(q-3) \mathbb{E}(A^T A)_{12}^2 (A^T A)_{34}^2 \\
&= qU_1(p, q) + q(q-1)U_2(p, q) + 2q(q-1)(2U_3(p, q) + (q-2)U_4(p, q)) \\
&\quad + q(q-1)(U_5(p, q) + U_2(p, q)) + 2q(q-1)(q-2)(U_6(p, q) + U_4(p, q)) \\
&\quad + q(q-1)(q-2)(q-3)U_7(p, q) \\
&= qU_1(p, q) + q(q-1)(2U_2(p, q) + 4U_3(p, q) + U_5(p, q)) \\
&\quad + 2q(q-1)(q-2)(2U_4(p, q) + U_6(p, q)) + q(q-1)(q-2)(q-3)U_7(p, q).
\end{aligned}$$

Let us calculate  $U_1, \dots, U_7$  for a  $p \times q$  Gaussian random matrix  $G$ .

$$\begin{aligned}
U_1(p, q) &= \mathbb{E}(G^T G)_{11}^4 - (\mathbb{E}(G^T G)_{11}^2)^2 = \mathbb{E} \|G_1\|_2^8 - (\mathbb{E} \|G_1\|_2^4)^2 \\
&= p(p+2)(p+4)(p+6) - (p(p+2))^2 \\
&= 8p(p+2)(p+3) \\
U_2(p, q) &= \mathbb{E}(G^T G)_{12}^4 - (\mathbb{E}(G^T G)_{12}^2)^2 = \mathbb{E} \left( \sum_r G_{r1} G_{r2} \right)^4 - (\mathbb{E} \langle G_1, G_2 \rangle)^2 \\
&= \sum_{r,s,t,u} \mathbb{E} G_{r1} G_{s1} G_{t1} G_{u1} G_{r2} G_{s2} G_{t2} G_{u2} - p^2 \\
&= 3 \sum_{r \neq t} \mathbb{E} G_{r1}^2 G_{t1}^2 G_{r2}^2 G_{t2}^2 + \sum_r G_{r1}^4 G_{r2}^4 - p^2 \\
&= 3p(p-1) + 9p - p^2 = 2p(p+3).
\end{aligned}$$

**35:22 The Product of Gaussian Matrices Is Close to Gaussian**

$$\begin{aligned}
U_3(p, q) &= \mathbb{E}(G^T G)_{11}^2 (G^T G)_{12}^2 - \mathbb{E}(G^T G)_{11}^2 \mathbb{E}(G^T G)_{12}^2 \\
&= \mathbb{E}(G_1^T G_1)^2 G_1^T G_2 G_2^T G_1 - \mathbb{E} \|G_1\|_2^4 \mathbb{E} \langle G_1, G_2 \rangle^2 \\
&= \mathbb{E}(G_1^T G_1)^2 G_1^T (\mathbb{E} G_2 G_2^T) G_1 - p(p+2) \cdot p \\
&= \mathbb{E}(G_1^T G_1)^3 - p^2(p+2) \\
&= \mathbb{E} \|G_1\|_2^6 - p^2(p+2) = p(p+2)(p+4) - p^2(p+2) = 4p(p+2) \\
U_4(p, q) &= \mathbb{E}(G^T G)_{12}^2 (G^T G)_{13}^2 - \mathbb{E}(G^T G)_{12}^2 \mathbb{E}(G^T G)_{13}^2 \\
&= \mathbb{E} G_1^T G_2 G_2^T G_1 G_1^T G_3 G_3^T G_1 - p^2 \\
&= \mathbb{E} G_1^T \mathbb{E}(G_2 G_2^T) G_1 G_1^T \mathbb{E}(G_3 G_3^T) G_1 - p^2 \\
&= \mathbb{E}(G_1^T G_1)^2 - p^2 = \mathbb{E} \|G_1\|_2^4 - p^2 = p(p+2) - p^2 = 2p \\
U_5(p, q) &= \mathbb{E}(G^T G)_{11}^2 (G^T G)_{22}^2 - \mathbb{E}(G^T G)_{11}^2 \mathbb{E}(G^T G)_{22}^2 \\
&= \mathbb{E} \|G_1\|_2^4 \|G_2\|_2^4 - \mathbb{E} \|G_1\|_2^4 \mathbb{E} \|G_2\|_2^4 = 0 \\
U_6(p, q) &= \mathbb{E}(G^T G)_{11}^2 (G^T G)_{23}^2 - \mathbb{E}(G^T G)_{11}^2 \mathbb{E}(G^T G)_{23}^2 \\
&= \mathbb{E} \|G_1\|_2^4 \langle G_2, G_3 \rangle^2 - \mathbb{E} \|G_1\|_2^4 \mathbb{E} \langle G_2, G_3 \rangle^2 = 0 \\
U_7(p, q) &= \mathbb{E}(G^T G)_{12}^2 (G^T G)_{34}^2 - \mathbb{E}(G^T G)_{12}^2 \mathbb{E}(G^T G)_{34}^2 \\
&= \mathbb{E} \langle G_1, G_2 \rangle^2 \langle G_3, G_4 \rangle^2 - \mathbb{E} \langle G_1, G_2 \rangle^2 \mathbb{E} \langle G_3, G_4 \rangle^2 = 0
\end{aligned}$$

Therefore

$$\begin{aligned}
\text{Var}(\text{tr}((G^T G)^2)) &= qU_1 + q(q-1)(2U_2 + 4U_3 + U_5) + 2q(q-1)(q-2)(2U_4 + U_6) \\
&\quad + q(q-1)(q-2)(q-3)U_7 \\
&= qU_1 + q(q-1)(2U_2 + 4U_3) + 4q(q-1)(q-2)U_4 \\
&= 4pq(5 + 5p + 5q + 2p^2 + 5pq + 2q^2).
\end{aligned}$$

When  $r = 2$ , recalling that  $\mathbb{E}(A_2 - A_1) = (1 + o(1))p^2q^2/d^3$  (see (4)), we have that

$$\frac{\sqrt{\text{Var}(\text{tr}((\frac{1}{\sqrt{d}}G^T \cdot \frac{1}{\sqrt{d}}G)^2))}}{p^2q^2/d^3} \leq \frac{6d}{\max\{p, q\}^{\frac{1}{2}} \min\{p, q\}^{\frac{3}{2}}}.$$

If the right-hand side above is at most a small constant  $c$ , we can distinguish  $A_2$  from  $A_1$  with probability at least a constant.



# Fast Mixing via Polymers for Random Graphs with Unbounded Degree

Andreas Galanis ✉

Department of Computer Science, University of Oxford, UK

Leslie Ann Goldberg ✉

Department of Computer Science, University of Oxford, UK

James Stewart ✉

Department of Computer Science, University of Oxford, UK

---

## Abstract

The polymer model framework is a classical tool from statistical mechanics that has recently been used to obtain approximation algorithms for spin systems on classes of bounded-degree graphs; examples include the ferromagnetic Potts model on expanders and on the grid. One of the key ingredients in the analysis of polymer models is controlling the growth rate of the number of polymers, which has been typically achieved so far by invoking the bounded-degree assumption. Nevertheless, this assumption is often restrictive and obstructs the applicability of the method to more general graphs. For example, sparse random graphs typically have bounded average degree and good expansion properties, but they include vertices with unbounded degree, and therefore are excluded from the current polymer-model framework.

We develop a less restrictive framework for polymer models that relaxes the standard bounded-degree assumption, by reworking the relevant polymer models from the edge perspective. The edge perspective allows us to bound the growth rate of the number of polymers in terms of the total degree of polymers, which in turn can be related more easily to the expansion properties of the underlying graph. To apply our methods, we consider random graphs with unbounded degrees from a fixed degree sequence (with minimum degree at least 3) and obtain approximation algorithms for the ferromagnetic Potts model, which is a standard benchmark for polymer models. Our techniques also extend to more general spin systems.

**2012 ACM Subject Classification** Theory of computation → Generating random combinatorial structures; Theory of computation → Design and analysis of algorithms

**Keywords and phrases** Markov chains, approximate counting, Potts model, expander graphs, random graphs

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.36

**Category** RANDOM

**Related Version** *A full version with all proofs can be found at:* <https://arxiv.org/abs/2105.00524>

## 1 Introduction

The polymer model framework [21, 14] is a classical tool from statistical mechanics which has recently been used to obtain efficient approximation algorithms for analysing spin systems (such as the Potts model) in parameter regimes where standard algorithmic approaches are provably inefficient/inaccurate on general graphs. These algorithms apply to certain classes of graphs that typically have sufficiently strong expansion properties relative to their local growth rates. Typically, the local growth rate is restricted by a bounded-degree assumption. Examples of such classes include bounded-degree expanders [20, 22, 7, 5, 6, 2, 10, 15] and the  $d$ -dimensional grid [16, 4, 20, 17]. The purpose of this work is to expand the current framework for applying polymer models by relaxing the bounded-degree assumption and using alternative methods to capture the growth of the graph.



© Andreas Galanis, Leslie Ann Goldberg, and James Stewart;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 36; pp. 36:1–36:13



Leibniz International Proceedings in Informatics

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

To briefly review the current framework, we use as a running example the  $q$ -state ferromagnetic Potts model with parameter  $\beta > 0$ . For a graph  $G = (V_G, E_G)$ , the set  $\Omega_{G,q}$  of configurations of the model is the set of all (not necessarily proper)  $q$ -colourings  $\sigma$  of  $V_G$  using the set of colours  $[q] = \{1, \dots, q\}$  where  $q \geq 3$ . The weight of a colouring  $\sigma$  is  $w(\sigma) = e^{\beta m_G(\sigma)}$  where  $m_G(\sigma)$  is the number of monochromatic edges under  $\sigma$ . The so-called partition function  $Z = Z_{G,q,\beta}$  is the aggregate weight of all  $\sigma$  and the Gibbs distribution  $\mu = \mu_{G,q,\beta}$  is the probability distribution on the set of all  $\sigma$ , in which each  $\sigma$  has mass proportional to its weight, i.e.,  $\mu(\sigma) = w(\sigma)/Z$ . We will study the computational problems of approximating the partition function and approximately sampling from the Gibbs distribution. In general, these problems are computationally hard ( $\#$ BIS-hard) when the parameter  $\beta$  is sufficiently large [13, 12].

The recent works [16, 20] introduced a framework based on polymer models that bypasses the worst-case hardness, on classes of bounded-degree graphs with expansion properties. The rough intuition for the Potts model is that, for large  $\beta$ , due to the expansion properties, the colourings with non-negligible weight are close to the so-called ground-states of the model, i.e., the  $q$  configurations in which all vertices get the same colour. Polymer models capture the deviation of configurations from these ground states. Given a ground state with colour  $r$ , a polymer is a connected set of vertices, none of which is coloured with  $r$ , and a polymer configuration (with respect to the ground state  $r$ ) corresponds to the set of all polymers (see Example 4 for more details). The Potts model can then be decomposed into  $q$  polymer models, each of which can be studied using relatively streamlined algorithmic methods (based on interpolation [1] and Markov chains). This framework has already found multiple algorithmic applications in far more general settings [16, 4, 17, 20, 19, 22, 7, 9, 5, 6, 10, 15].

Despite these advances, the current applications of polymer models rely crucially on the fact that the maximum degree of the underlying graph is bounded. This fact is used to control the number of polymers of a given size (which is crucially needed for the algorithmic analysis). As a result of this limitation, applications to several other interesting classes of graphs are ruled out, excluding for example sparse random graphs, which have bounded average degree and good expansion properties, but include vertices with unbounded degree.

## 1.1 Main Results

In this paper, we propose a framework for polymer models that overcomes the bounded-degree limitations of previous algorithms, by revisiting the Markov chain approach of [7]. We introduce a new condition which requires that the weight of each polymer decays exponentially in its total degree (the sum of the degrees of the vertices in the polymer) instead of decaying exponentially in the polymer's size. This new condition allows us to prove rapid mixing for a Markov chain which is an adapted edge-version of the so-called polymer dynamics of [7]. Crucially, the fact that the new condition is formulated in terms of the total degree of a polymer allows us to relax the assumption that the instance has bounded degree.

As an application of our method, we consider the  $q$ -state ferromagnetic Potts model on sparse random graphs of unbounded degree with a given degree sequence, as detailed below.

► **Definition 1.** *Let  $d$  be a positive real number and  $n$  be a positive integer. We define  $\mathcal{D}_{n,d}$  to be the set of all degree sequences  $\{x_1, x_2, \dots, x_n\}$  that satisfy*

1. *For all  $i \in [n]$ ,  $3 \leq x_i \leq n^\rho$  where  $\rho = \frac{1}{50}$ , and*
2.  *$\sum_{i \in [n]} x_i^2 \leq dn$ .*

*We write  $G \sim \mathcal{G}(n, \vec{x})$  to indicate that  $G$  is a graph chosen uniformly at random from the set of all simple  $n$ -vertex graphs with degree sequence  $\vec{x}$ .  $G$  satisfies a property with high probability (whp) if the probability that  $G$  satisfies the property is  $1 - o(1)$ , as a function of  $n$  (uniformly over  $\vec{x}$ ).*

Note that  $\mathcal{D}_{n,d}$  is empty unless  $d \geq 9$ . The assumption that all degrees are greater than or equal to 3 (rather than 2) guarantees that the random graph  $G$  is connected and has good expansion properties. The degree lower bound also means that our results do not apply to Erdős-Rényi random graphs. The upper bound on the degrees is mild and can in fact be relaxed somewhat further (but in general cannot be made to be linear in  $n$  due to the sparsity assumption in Item 2).

We give an efficient algorithm for approximately sampling<sup>1</sup> from and approximating the partition function<sup>2</sup> of the ferromagnetic Potts model on random graphs with a given degree sequence for all sufficiently large  $\beta$ .

► **Theorem 2.** *Let  $d$  be a real number and  $q \geq 3$  be an integer. For the ferromagnetic Potts model, there is  $\beta_0$  such that for all  $\beta \geq \beta_0$  there is a poly-time approximate sampling algorithm for  $\mu_{G,q,\beta}$  and an FPRAS for  $Z_{G,q,\beta}$  that work with high probability on random graphs  $G \sim \mathcal{G}(n, \vec{x})$  for any degree sequence  $\vec{x} \in \mathcal{D}_{n,d}$ .*

► **Remark 3.** Note that  $\beta_0$  depends on  $d$  and  $q$ , and our arguments later (see Remark 17) show that  $\beta_0 = Cd \log d \log q$  for some  $C > 0$  (independent of  $d$  or  $q$ ). If the desired accuracy  $\varepsilon$  is at least  $e^{-n}$  then the running time of the sampling algorithm is  $O(n \log \frac{n}{\varepsilon} \log \frac{1}{\varepsilon})$  and the running time of the FPRAS is  $O(n^2 (\log \frac{n}{\varepsilon})^3)$ .

We further remark that the bounded-degree assumption has also been relaxed in [17] for the ferromagnetic Potts model on lattice graphs; the approach therein however is tailored to a certain flow representation of the Potts model, which is used as a basis for the corresponding polymer models and therefore does not extend to general spin systems. Our approach applies to general polymer models as detailed in the next section and our focus on the ferromagnetic Potts model is mainly to illustrate the method without further technical overhead; the approach for example can be adapted to general spin systems on bipartite random graphs with a given degree sequence (analogously to [10]).

## 2 Polymers

The main technique that we use to prove Theorem 2 is to refine the polymer framework by introducing a new polymer sampling condition which requires that the weight of each polymer decays exponentially in its total degree. In order to state the new condition we first give some relevant definitions.

Let  $G = (V_G, E_G)$  be a graph – we refer to  $G$  as the “host graph” of the polymer model. Let  $[q] = \{1, \dots, q\}$  be a set of spins and  $g = \{g_v\}_{v \in V_G}$  specify a set of ground-state spins for the vertices, i.e.,  $g_v \subseteq [q]$  for each  $v \in V_G$ . A polymer is a pair  $\gamma = (V_\gamma, \sigma_\gamma)$  consisting of a connected set of vertices  $V_\gamma$  and an assignment  $\sigma_\gamma: V_\gamma \rightarrow [q]$  such that  $\sigma_\gamma(v) \in [q] \setminus g_v$ . Let  $\mathcal{P}_G$  be the set of all polymers. A polymer model for the host graph  $G$  is specified by a subset of allowed polymers  $\mathcal{C}_G \subseteq \mathcal{P}_G$ , and a weight function  $w_G: \mathcal{C}_G \rightarrow \mathbb{R}_{\geq 0}$ . For polymers  $\gamma, \gamma' \in \mathcal{P}_G$ , we write  $\gamma \sim \gamma'$  to denote that  $\gamma, \gamma'$  are compatible, i.e., that

<sup>1</sup> A polynomial-time approximate sampling algorithm for  $\mu_{G,q,\beta}$  is an algorithm that, given an accuracy parameter  $\varepsilon > 0$  and a graph  $G = (V_G, E_G)$  as input, outputs a sample from a probability distribution that is within total variation distance  $\varepsilon$  of  $\mu_{G,q,\beta}$ , in time  $\text{poly}(|V_G|, 1/\varepsilon)$ .

<sup>2</sup> Given an accuracy parameter  $\varepsilon > 0$ , we say that  $\hat{Z}$  is an  $\varepsilon$ -approximation to the quantity  $Z$  if  $e^{-\varepsilon} Z \leq \hat{Z} \leq e^\varepsilon Z$ . A fully polynomial randomised approximation scheme (FPRAS) for  $Z_{G,q,\beta}$  is a randomised algorithm that, given an accuracy parameter  $\varepsilon > 0$  and a graph  $G = (V_G, E_G)$  as input, outputs a random variable that is an  $\varepsilon$ -approximation to  $Z_{G,q,\beta}$  with probability at least  $3/4$ , in time  $\text{poly}(|V_G|, 1/\varepsilon)$ .

for every  $u \in \gamma$  and  $u' \in \gamma$  the graph distance in  $G$  between  $u$  and  $u'$  is at least 2. We define  $\Omega_G = \{\Gamma \subseteq \mathcal{C}_G \mid \forall \gamma, \gamma' \in \Gamma, \gamma \sim \gamma'\}$  to be the set of all sets of mutually compatible polymers of  $\mathcal{C}_G$ ; elements of  $\Omega_G$  are called polymer configurations. We define the partition function as  $Z_G = \sum_{\Gamma \in \Omega_G} \prod_{\gamma \in \Gamma} w_G(\gamma)$ , and the Gibbs distribution on  $\Gamma \in \Omega_G$  as  $\mu_G(\Gamma) = \prod_{\gamma \in \Gamma} w_G(\gamma) / Z_G$ . We use  $(\mathcal{C}_G, w_G)$  to denote the polymer model.

► **Example 4** (The polymer model  $(\mathcal{C}_{G,q}^r, w_{G,\beta})$ , [20]). Consider the  $q$ -state ferromagnetic Potts model with parameter  $\beta$ , and let  $r \in [q]$  be a colour. Let  $G$  be a graph and set  $g_v = \{r\}$  for every  $v \in V_G$ . The weight of a polymer  $\gamma = (V_\gamma, \sigma_\gamma)$  is defined as  $w_{G,\beta}(\gamma) = e^{-\beta B_\gamma}$ , where  $B_\gamma$  denotes the number of edges from  $V_\gamma$  to  $V_G \setminus V_\gamma$  plus the number of edges of  $G$  with both endpoints in  $V_\gamma$  that are bichromatic under  $\sigma_\gamma$ . We let  $\mathcal{P}_{G,q}^r$  denote the set of all polymers and the set of allowed polymers  $\mathcal{C}_{G,q}^r$  to be the set of polymers  $\gamma \in \mathcal{P}_{G,q}^r$  such that  $|V_\gamma| < |V_G|/2$ . Note that a polymer configuration  $\Gamma$  consisting of the polymers  $\gamma_1, \dots, \gamma_k$  corresponds to a colouring  $\sigma$ , where a vertex  $v$  takes the colour  $\sigma_{\gamma_i}(v)$  if  $v \in V_{\gamma_i}$  for some  $i \in [k]$ , and the colour  $r$  otherwise; moreover,  $e^{\beta|E_G|} \prod_{i \in [k]} w_G(\gamma_i) = w_G(\sigma)$ .

Polymer models have been used to approximate the partition function of spin systems on bounded-degree host graphs. There are several existing algorithmic frameworks which can be used to sample from these resulting polymer models. One such deterministic algorithm uses the polynomial interpolation method of Barvinok [1] combined with the cluster expansion to approximate the partition function of the polymer model (see [16] for more details). Typical running times for these deterministic algorithms are of the form  $n^{O(\log(\Delta))}$ , where  $\Delta$  is the maximum degree of the host graph, though for polymer models these have been improved to give a running time of  $n^{1+o_\Delta(1)}$ , see [20]. Another approach, described in Section 4 of the full version [11], uses a Markov chain called the polymer dynamics to sample from  $\mu_G$  (see also [7] for more details). The running times of algorithms obtained using the Markov chain approach are usually faster and of the form  $O(n \log n)$ . Both of these approaches work for roughly the same range of parameters, and the essential condition required is that the weight of each polymer decays exponentially in the number of vertices it contains. To obtain our results, we give a simple generic way to modify this condition, as detailed below.

For a vertex  $v \in V_G$  we write  $\deg_G(v)$  to denote the degree of  $v$  in  $G$ , and for a vertex subset  $S \subseteq V_G$  we write  $\deg_G(S)$  to denote  $\sum_{v \in S} \deg_G(v)$ .

► **Definition 5.** Let  $q \geq 2$  be an integer,  $\mathcal{G}$  be a class of graphs, and  $\mathcal{F}_\mathcal{G} = \{(\mathcal{C}_G, w_G) \mid G \in \mathcal{G}\}$  be a family of  $q$ -spin polymer models. We say that  $\mathcal{F}_\mathcal{G}$  satisfies the polymer sampling condition with constant  $\tau \geq 3 \log(8e^3(q-1))$  if  $w_G(\gamma) \leq e^{-\tau \deg_G(V_\gamma)}$  for all  $G \in \mathcal{G}$  and all  $\gamma \in \mathcal{C}_G$ .<sup>3</sup>

Using Definition 5, we will show (Lemma 8, below) that if a “computationally feasible” family of polymer models on a class of graphs  $\mathcal{G}$  satisfies this new condition, then there is an efficient algorithm which, given a graph  $G \in \mathcal{G}$ , approximately samples from the Gibbs distribution of the polymer model corresponding to  $G$ .

The new polymer sampling condition in Definition 5 is analogous to the original one in [7] except that the original condition requires the weight of a polymer to decay exponentially in its size, and in particular that the constant  $\tau$  is sufficiently big relative to the maximum degree of  $G$ . The new condition relaxes this, allowing us to choose the constant  $\tau$  in Definition 5 so that it does not depend on the maximum degree of the host graph, which is how we overcome the limitations of previous work. Technically, the improvement comes from the fact that previous work relies on bounding the number of connected vertex subsets of a given size

<sup>3</sup> Unless we specify otherwise, the base of all logarithms in this paper is assumed to be  $e$ .

(with bounds that depend on the maximum degree of the graph), but here we are able to instead rely on the following lemma which bounds the number of connected vertex subsets with a given total degree and this enables us to avoid restricting the maximum degree of the graph. The new condition, which replaces the notion of “size” with total degree, fits well with the original abstract polymer model framework of [21], where the notion of the “size” of a polymer is an abstract function.

► **Lemma 6.** *Let  $G = (V_G, E_G)$  be a graph,  $v \in V_G$ , and  $\ell \geq 1$  be an integer. The number of connected vertex subsets  $S \subseteq V_G$  such that  $v \in S$  and  $\deg_G(S) = \ell$  is at most  $(2e)^{2\ell-1}$ .*

In addition to the bound on the number of connected vertex subsets in Lemma 6, we will use the fact that these connected vertex subsets can be enumerated in time exponential in the total degree  $\ell$  (see Lemma 21 of the full version). Although the bound in Lemma 6 is exponential in  $\ell$ , this will be mitigated by the fact that the new polymer sampling condition ensures that the weight of each polymer is exponentially small in its total degree. The new polymer sampling condition therefore allows us to prove that the following condition holds – this condition is analogous to the polymer mixing condition of [7], except that we consider edges instead of vertices. For a polymer  $\gamma \in \mathcal{P}_G$ , let  $E_\gamma$  denote the set of edges of  $G$  with at least one endpoint in  $V_\gamma$ .

► **Definition 7.** *Let  $q \geq 2$  be an integer,  $\mathcal{G}$  be a class of graphs, and  $\mathcal{F}_\mathcal{G} = \{(\mathcal{C}_G, w_G) \mid G \in \mathcal{G}\}$  be a family of  $q$ -spin polymer models. We say that  $\mathcal{F}_\mathcal{G}$  satisfies the polymer mixing condition with constant  $\theta \in (0, 1)$  if  $\sum_{\gamma' \sim \gamma} |E_{\gamma'}| \cdot w_G(\gamma') \leq \theta |E_\gamma|$  for all  $G \in \mathcal{G}$  and all  $\gamma \in \mathcal{C}_G$ .*

In contrast to the conditions in [7], the two new conditions consider edges since we modify the polymer dynamics algorithm to sample edges instead of vertices. Subject to these new conditions, the techniques of [7] can be adapted to show that the modified polymer dynamics mixes rapidly, therefore giving the efficient algorithm for sampling from the Gibbs distribution of a polymer model. We give the details of the modified dynamics in Section 4 of the full version [11].

Finally, in order to use the modified polymer dynamics as an efficient algorithm for computing an approximate sample from  $\mu_G$ , we will need a mild computational condition for polymers. More precisely, we say that a family of polymer models  $\{(\mathcal{C}_G, w_G) \mid G \in \mathcal{G}\}$  is *computationally feasible* if for all  $G \in \mathcal{G}$  and all  $\gamma \in \mathcal{P}_G$ , it is possible to decide whether  $\gamma \in \mathcal{C}_G$  and to compute  $w_G(\gamma)$ , if it is, in  $O(e^{\deg_G(V_\gamma)})$  time. Computational feasibility serves exactly the same purpose as it did in Definition 3 of [7], which requires that the same operations are able to be carried out in time depending on  $|V_\gamma|$  (instead of  $\deg_G(V_\gamma)$  that we use here).

In Section 4 of the full version [11], we prove the following lemma which gives an efficient algorithm for sampling<sup>4</sup> from the Gibbs distribution of a polymer model and for approximating its partition function. In order to prove the lemma, we extend the polymer dynamics algorithm of [7] to the unbounded degree setting. The proof of the lemma uses the fact (see Lemma 18 of the full version) that the polymer sampling condition implies the polymer mixing condition.

<sup>4</sup> Given an accuracy parameter  $\varepsilon > 0$ , we say that a random variable  $X$  is an  $\varepsilon$ -sample from the probability distribution  $\mu$  if the total variation distance between the distribution of  $X$  and  $\mu$  is at most  $\varepsilon$ .

► **Lemma 8.** *Let  $q \geq 2$  be an integer,  $\mathcal{G}$  be a class of graphs, and  $\mathcal{F}_{\mathcal{G}}$  be a family of computationally feasible  $q$ -spin polymer models satisfying the polymer sampling condition.*

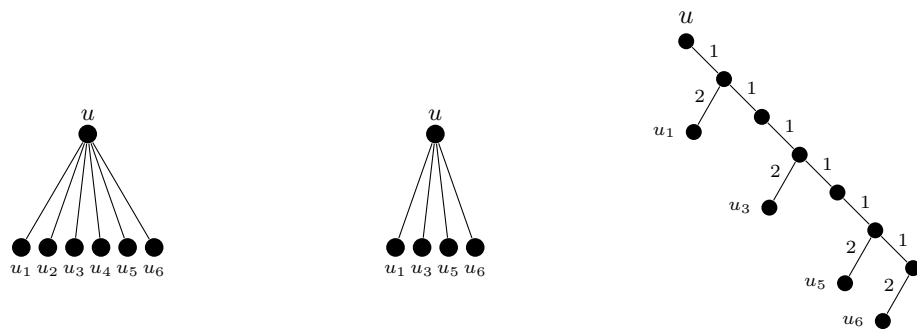
*There are randomised algorithms which, given as input a graph  $G \in \mathcal{G}$  with  $m$  edges and an accuracy parameter  $\varepsilon > 0$ , output an  $\varepsilon$ -sample from  $\mu_G$  in  $O(m \log \frac{m}{\varepsilon} \log \frac{1}{\varepsilon})$  time, and an  $\varepsilon$ -approximation to  $Z_G$ , with probability at least  $3/4$ , in  $O(m^2 \log(\frac{m}{\varepsilon})^3)$  time.*

We next give the proof of Lemma 6 which is one of the key technical ingredients allowing us to relax the bounded-degree restriction and to remove the dependence of the constant  $\tau$  in Definition 5 on the maximum degree of the host graph, as noted earlier.

**Proof of Lemma 6.** Let  $N(G, v, \ell)$  be the set of subtrees  $T = (V_T, E_T)$  of  $G$  such that  $v \in V_T$ ,  $\deg_G(V_T) = \ell$ . We will show that  $|N(G, v, \ell)| \leq (2e)^{2\ell-1}$ , which gives us the desired result for the following reason. Let  $S \subseteq V_G$  be a connected vertex subset such that  $v \in S$  and  $\deg_G(S) = \ell$ . Since  $S$  is connected, it has at least one spanning tree  $T = (V_T = S, E_T)$  such that  $v \in V_T$  and  $\deg_G(V_T) = \ell$ . Since  $S$  is the unique connected vertex subset that  $T$  spans, this gives us an injective map from the set of all connected vertex subsets containing  $v$  with total degree  $\ell$ , to  $N(G, v, \ell)$ .

We now give an injective map from  $N(G, v, \ell)$  to  $T^*(2\ell, 3)$  – the set of subtrees of size  $2\ell$  that contain the root, of the infinite rooted 3-regular tree. By a result of Bollobás [3, p. 129], we know that  $|T^*(2\ell, 3)|$  is at most  $(2e)^{2\ell-1}$ . Let  $T = (V_T, E_T)$  be a subtree from  $N(G, v, \ell)$ . We will map  $T$  to a rooted subtree  $T' = (V_{T'}, E_{T'})$  from  $T^*(2\ell, 3)$ . For each vertex of  $V_G$ , fix an ordering of its neighbours. In the infinite rooted 3-regular tree, label the edges incident to the root with  $\{1, 2, 3\}$ , and for each other vertex label the edges connecting it to its two children with  $\{1, 2\}$ . As we construct  $T'$ , we will label its edges so that it is clear which subtree from  $T^*(2\ell, 3)$  we are constructing, we will also label some of its vertices. We construct  $T'$  as follows (see Figure 1 for an example of the following construction).

1. Add the root to  $V_{T'}$  and label it  $v$ .
2. While there is a labelled vertex of  $T'$  (call its label  $u$ ) such that  $u$  has a child  $w$  in  $T$  but no vertex of  $T'$  is labelled  $w$ , then we do the following. First, we create a path  $P$  of length  $\deg_G(u)$  where each edge is labelled 1. We then connect the vertex of  $T'$  labelled  $u$  to  $P$  via an edge labelled 1. Finally, for  $1 \leq i \leq \deg_G(u)$ , we connect a vertex labelled  $w$  to the  $i^{\text{th}}$  vertex of  $P$  via an edge labelled 2, if  $w$  is the  $i^{\text{th}}$  neighbour of  $u$  in  $G$  and  $w$  is a child of  $u$  in  $T$ .



(a) Neighbourhood of  $u$  in  $G$ .      (b) Neighbourhood of  $u$  in  $T$ .      (c) Neighbourhood of  $u$  in  $T'$ .

■ **Figure 1** Constructing  $T'$ .

Each  $T \in N(G, v, \ell)$  maps to a different  $T' \in T^*(2\ell, 3)$ . When constructing  $T'$ , we used edge labels from  $\{1, 2, 3\}$ , therefore the maximum degree of  $T'$  is 3. For each  $v \in V_T$ , we added at most  $2 \deg_G(v)$  vertices to  $T'$ , therefore the size of  $T'$  is at most  $2 \deg_G(V_T) = 2\ell$ . ◀

### 3 Application to unbounded-degree graphs

Let  $\alpha > 0$  be a real number. We say that a graph  $G$  is an  $\alpha$ -total-degree expander if, for all connected vertex subsets  $S \subseteq V_G$  with  $|S| \leq |V_G|/2$ , we have  $e_G(S, S^c) \geq \alpha \deg_G(S)$ , where  $e_G(S, S^c)$  denotes the number of edges with one endpoint in  $S$  and the other in  $S^c := V_G \setminus S$ . Let  $\mathcal{G}_\alpha$  denote the set of all  $\alpha$ -total-degree expanders. Note, every connected  $G \in \mathcal{G}_\alpha$  is also an  $\alpha$ -expander (i.e.,  $e_G(S, S^c) \geq \alpha|S|$ ).

When  $\beta$  is sufficiently large, the polymer model from Example 4 satisfies the polymer sampling condition (Definition 5) with constant  $\tau = \alpha\beta$ . To see this, consider  $\gamma \in \mathcal{C}_{G,q}^r$  and observe that since  $B_\gamma \geq e_G(V_\gamma, V_\gamma^c)$  and  $|V_\gamma| < |V_G|/2$ , it follows that

$$w_{G,\beta}(\gamma) \leq \exp\{-\alpha\beta \deg_G(V_\gamma)\} = e^{-\tau \deg_G(V_\gamma)}, \tag{1}$$

where  $\tau \geq 3 \log(8e^3(q-1))$  if  $\beta \geq \frac{3}{\alpha} \log(8e^3(q-1))$ .

We may therefore apply Lemma 8 in order to efficiently sample from the ferromagnetic Potts model and to estimate  $Z_G$  for  $G \in \mathcal{G}_\alpha$ , provided that  $\beta$  is sufficiently large. The proof of the following theorem is in Section 5 of the full version [11].

► **Theorem 9.** *Let  $\alpha > 0$  be a real number. Let  $q \geq 3$  be an integer and  $\beta \geq \frac{3}{\alpha} \log(8e^3(q-1))$  be a real. For the Potts model on  $G \in \mathcal{G}_\alpha$ , there is a poly-time approximate sampling algorithm for  $\mu_{G,q,\beta}$  and an FPRAS for  $Z_{G,q,\beta}$ .*

*In fact, for  $n = |V_G|$  and  $m = |E_G|$ , if the desired accuracy  $\varepsilon$  satisfies  $\varepsilon \geq e^{-n}$  then the running time of the sampler is  $O(m \log \frac{m}{\varepsilon} \log \frac{1}{\varepsilon})$  and the running time of the FPRAS is  $O(m^2 (\log \frac{m}{\varepsilon})^3)$ .*

#### 3.1 Expansion of random graphs with specified degree sequences

Let  $d$  be a real number. In this section, we will show that a graph  $G \sim \mathcal{G}(n, \vec{x})$  for a degree sequence  $\vec{x} \in \mathcal{D}_{n,d}$  is whp an  $\alpha$ -total-degree-expander for some constant  $\alpha > 0$ , i.e., that  $G \in \mathcal{G}_\alpha$ .

To work with  $G \sim \mathcal{G}(n, \vec{x})$ , we consider the standard configuration model, where a random multigraph  $H = (V_H, E_H)$  with the given degree sequence  $\vec{x}$  is sampled by the following process. For each  $i \in [n]$ , we attach  $x_i$  half-edges to the vertex  $i$ . We then sample a uniformly random perfect matching on the half-edges to give  $E_H$ . This uniformly random perfect matching can be sampled by performing the following until no half-edges remain: choose any remaining half-edge, choose another remaining half-edge uniformly at random, then pair these two half-edges and remove them from the set of remaining half-edges. We write  $H \sim \text{CM}(n, \vec{x})$ . Note, for two vertices  $i, j \in V_H$  such that  $i \neq j$ , the probability that a half edge attached to  $i$  and a half edge attached to  $j$  are paired is

$$p_{\{i,j\}} = \frac{x_i x_j}{2m - 1}, \text{ where } m = \frac{1}{2} \sum_{k=1}^n x_k, \tag{2}$$

and similarly the probability that two half-edges of  $i$  are connected is  $p_{\{i,i\}} = \frac{x_i(x_i-1)}{2m(2m-1)}$ .

We first prove results about  $\text{CM}(n, \vec{x})$ , since asymptotic properties of  $\text{CM}(n, \vec{x})$  can easily be transferred back to  $\mathcal{G}(n, \vec{x})$  using the following straightforward consequence of [18, Theorem 1.1]. A proof is included in the full version [11] for completeness.

► **Lemma 10.** *Let  $d$  be a positive real number. For every positive integer  $n$ , let  $\mathcal{E}_n$  be a set of  $n$ -vertex multigraphs. If, for some  $\vec{x} \in \mathcal{D}_{n,d}$ ,  $G \sim \mathcal{G}(n, \vec{x})$  and  $H \sim \text{CM}(n, \vec{x})$  then the following is true. If  $H \in \mathcal{E}_n$  with high probability, then  $G \in \mathcal{E}_n$  with high probability.*

For a (multi)graph  $H = (V_H, E_H)$  we define the tree-excess to be  $t_H = |E_H| - (|V_H| - 1)$ ; that is, the number of edges more than a tree that  $H$  has. First, we show that multigraphs drawn from the configuration model have locally bounded tree excess.

► **Lemma 11.** *Let  $d$  be a positive real number. The following is true with high probability when  $H = (V_H, E_H)$  is drawn from  $\text{CM}(n, \vec{x})$  uniformly over all degree sequences  $\vec{x} \in \mathcal{D}_{n,d}$ . For all connected vertex sets  $S \subseteq V_H$  with  $|S| \leq (\log n)^2$  and  $\deg_H(S) \geq 36$ , we have that  $t_{H[S]} \leq \frac{1}{6} \deg_H(S)$ .*

**Proof.** For positive integers  $k$  and  $\ell$ , and a non-negative integer  $t$ , let the random variable  $X_{k,\ell,t}$  denote the number of connected vertex subsets  $S \subseteq V_H$  such that  $|S| = k$ ,  $\deg_H(S) = \ell$ , and  $t_{H[S]} = t$ . To prove the lemma, we will show that whp

$$\sum_{k \leq \lfloor (\log n)^2 \rfloor} \sum_{\ell \geq 36} \sum_{t \geq \lfloor \ell/6 \rfloor + 1} X_{k,\ell,t} = 0.$$

In fact, we can further restrict the range of summation. From the lower bound in Item 1 of Definition 1, we have that  $x_i \geq 3$  for all  $i$ , and therefore  $\ell \geq 3k$ . Item 2 shows that  $\sum_i x_i \leq dn$ , and therefore  $\ell \leq dn$  and  $t \leq \ell/2 \leq dn/2$ . So, consider any integer  $\ell$  in the range  $36 \leq \ell \leq dn$ , any integer  $k$  in the range  $1 \leq k \leq \min\{(\log n)^2, \ell/3\}$ , and any integer  $t > \ell/6$ . There are at most  $\binom{n}{k}$  vertex subsets  $S \subseteq V_H$  with  $|S| = k$  and  $\deg_G(S) = \ell$ . Let  $j = k - 1 + t$  be the number of edges with both endpoints in  $S$ . Given such a set  $S$ , there are at most  $\binom{\ell}{2j}$  possibilities for the set of half-edges in these  $j$  edges. On a given set of  $2j$  half-edges, there are  $(2j - 1)!! = \frac{(2j)!}{2^j j!}$  perfect matchings. Using the upper bound on the degrees from Item 1 of Definition 1, the probability that a set of  $j$  edges is present in  $H$  is at most

$$\frac{n^{2\rho}}{2m-1} \frac{n^{2\rho}}{2m-3} \cdots \frac{n^{2\rho}}{2m-2j+1} \leq \left( \frac{n^{2\rho}}{2m-2j} \right)^j \leq \left( \frac{n^{2\rho}}{n} \right)^j,$$

where the final inequality follows from the fact that  $k \leq (\log n)^2$  and therefore that  $2m - 2j \geq \deg_G(S^c) \geq 3|S^c| = 3(n - k) > n$  (as long as  $n$  is sufficiently big). We also have that

$$\binom{\ell}{2j} \cdot \frac{(2j)!}{2^j j!} < \frac{\ell!}{(\ell - 2j)! j!} < \frac{\ell^{2j} e^j}{j^j} \leq \left( \frac{e\ell^2}{t} \right)^j.$$

Putting everything together, it follows that

$$\mathbb{E}[X_{k,\ell,t}] \leq \binom{n}{k} \left( \frac{e\ell^2}{t} \right)^{k-1+t} \left( \frac{n^{2\rho}}{n} \right)^{k-1+t} < \left( \frac{e^2 \ell^2}{t} \right)^{k-1+t} \frac{n^{2\rho(k-1+t)}}{n^{t-1}}.$$

Furthermore, since  $t > \ell/6$ ,  $k < 2t$ , and (by the upper bound in Item 1 of Definition 1)  $\ell \leq kn^\rho \leq n^{2\rho}$ , we have that

$$\mathbb{E}[X_{k,\ell,t}] < \frac{(6e^2 n^{4\rho})^{3t-1}}{n^{t-1}}. \quad (3)$$

Let

$$X = \sum_{\ell=36}^{dn} \sum_{k=1}^{\lfloor \min\{(\log n)^2, \ell/3\} \rfloor} \sum_{t=\lfloor \ell/6 \rfloor + 1}^{\lfloor dn/2 \rfloor} X_{k,\ell,t}.$$



Since  $t > \ell/6 \geq 6$ , it follows that  $t \geq 7$ . For big enough  $n$ , (3) shows that  $\mathbb{E}[X_{k,\ell,t}] \leq n^{13\rho t}/n^{t-1}$ . Since  $\rho \leq 2/91$  and  $t \geq 7$ ,  $1 - 13\rho \geq 5/7 \geq 5/t$  so  $13\rho t \leq t - 5$  and  $\mathbb{E}[X_{k,\ell,t}]$  is at most  $n^{-4}$ . Taking a union bound over all permissible values for  $\ell$ ,  $k$ , and  $t$ , we find that  $\mathbb{E}[X] = o(1)$ . Applying Markov's inequality, we have that  $\Pr(X > 0) = \Pr(X \geq 1) \leq \mathbb{E}[X] = o(1)$ , and the result follows.  $\blacktriangleleft$

To obtain the expansion bounds in Lemmas 13 and 14, we require the following result from [8, Proposition 4.5]. Although this result is stated in [8] in terms of the random graph model, it is first proved for the configuration model, so this is how we state it. Also, Fountoulakis and Reed require that the vector  $\vec{x}$  be in  $\mathcal{D}_{n,d}$  but this is only important for lifting their result to the random graph model, so it is not relevant for us.

► **Lemma 12.** (Fountoulakis, Reed) *Let  $H = (V_H, E_H)$  be drawn from  $\text{CM}(n, \vec{x})$  for some length- $n$  degree sequence  $\vec{x}$ . For any set  $S \subseteq V_H$  we have  $\Pr(e_H(S, S^c) = 0) \leq \binom{m}{\deg_H(S)/2}^{-1}$ , where  $m = \frac{1}{2} \sum_i x_i$ .*

Note that Fountoulakis and Reed's lemma was stated for  $S$  such that  $\deg_H(S)$  is even, but if  $\deg_H(S)$  is odd, it is not possible to have  $e_H(S, S^c) = 0$ . Next, we show that in a multigraph  $H$  drawn from the configuration model, small vertex subsets satisfy certain expansion properties.

► **Lemma 13.** *Let  $d$  be a positive real number. The following is true with high probability when  $H = (V_H, E_H)$  is drawn from  $\text{CM}(n, \vec{x})$  uniformly over all degree sequences  $\vec{x} \in \mathcal{D}_{n,d}$ . For all connected vertex sets  $S \subseteq V_H$  with  $|S| \leq (\log n)^2$ , we have that  $e_H(S, S^c) \geq |S|/4$ .*

**Proof.** For positive integers  $k$  and  $\ell$ , and a non-negative integer  $j$ , let the random variable  $X_{k,j,\ell}$  denote the number of connected vertex subsets  $S \subseteq V_H$  with  $|S| = k$ ,  $e_H(S, S^c) = j$ , and  $\deg_H(S) = \ell$ . By Item 1 of Definition 1, we need only consider  $\ell$  satisfying  $3k \leq \ell \leq kn^\rho$ . Let

$$X = \sum_{k=1}^{\lfloor (\log n)^2 \rfloor} \sum_{j=0}^{\lfloor k/4 \rfloor} \sum_{\ell=3k}^{\lfloor kn^\rho \rfloor} X_{k,j,\ell}.$$

To prove the lemma we will show that  $X = 0$ , whp. Consider any integer  $k$  in the range  $1 \leq k \leq (\log n)^2$ , any integer  $j$  in the range  $0 \leq j < k/4$ , and any integer  $\ell$  in the range  $3k \leq \ell \leq kn^\rho$ . There are at most  $\binom{n}{k}$  candidates for vertex sets  $S$  with  $|S| = k$  and  $\deg_H(S) = \ell$ . There are then at most  $\binom{\ell}{j}$  choices for the  $j$  half-edges emanating from vertices of  $S$  that will be matched with half-edges emanating from vertices of  $S^c$ , once  $H$  is drawn. Applying Lemma 12 to the degree sequence derived from  $\vec{x}$  by removing the  $j$  half-edges (and their partners), the probability that the remaining  $\ell - j$  half-edges are matched amongst themselves is at most

$$\left( \frac{m'}{(\ell - j)/2} \right)^{-1} \leq \left( \frac{(\ell - j)}{2m'} \right)^{\frac{(\ell - j)}{2}} \leq \left( \frac{kn^\rho}{n} \right)^{\frac{11k}{8}},$$

where  $2m' = (\sum_{i=1}^n x_i) - 2j$  and the last inequality follows (for big enough  $n$ ) since  $11k/4 \leq$

## 36:10 Fast Mixing via Polymers for Random Graphs with Unbounded Degree

$\ell - j \leq kn^\rho$  and  $2m' \geq 3n - 2j > n$ . We therefore have that

$$\begin{aligned} \mathbb{E}[X] &\leq \sum_{k=1}^{\lfloor (\log n)^2 \rfloor} \sum_{j=0}^{\lfloor k/4 \rfloor} \sum_{\ell=3k}^{\lfloor kn^\rho \rfloor} \binom{n}{k} \binom{\ell}{j} \left( \frac{kn^\rho}{n} \right)^{\frac{11k}{8}} \\ &\leq \sum_{k=1}^{\lfloor (\log n)^2 \rfloor} \sum_{j=0}^{\lfloor k/4 \rfloor} \sum_{\ell=3k}^{\lfloor kn^\rho \rfloor} \left( \frac{ne}{k} \right)^k \left( \frac{e\ell}{j} \right)^j \left( \frac{kn^\rho}{n} \right)^{\frac{11k}{8}} \\ &\leq \sum_{k=1}^{\lfloor (\log n)^2 \rfloor} \left( \frac{(\log n)^{O(1)} n^{\rho(2+11/8)}}{n^{3/8}} \right)^k. \end{aligned}$$

This is  $o(1)$  since  $\rho < 1/9$ . Applying Markov's inequality, we have that  $\Pr(X > 0) = \Pr(X \geq 1) = o(1)$ , and the result follows.  $\blacktriangleleft$

The next lemma handles the expansion of sets  $S$  with relative big size.

**► Lemma 14.** *Let  $d$  be a positive real number. There is a positive real number  $\alpha$  (depending on  $d$ ) such that the following is true with high probability when  $H = (V_H, E_H)$  is drawn from  $\text{CM}(n, \vec{x})$  uniformly over all degree sequences  $\vec{x} \in \mathcal{D}_{n,d}$ . For all connected vertex sets  $S \subseteq V_H$  with  $(\log n)^2 \leq |S| \leq n/2$ , we have that  $e_H(S, S^c) \geq \alpha \deg_H(S)$ .*

**Proof.** We will give the proof for vertex sets  $S \subseteq V_H$  with  $(\log n)^2 \leq |S| \leq n/2$  and  $\deg_H(S) > \max\{100d|S|, n/2\}$ , the cases where  $\deg_H(S) \leq \max\{100d|S|, n/2\}$  follow by arguments that are close to those in [8], and are given in the full version [11].

Let  $C = 10^4 d$ . By the Cauchy-Schwarz inequality, we have that  $|S| \sum_{i \in S} x_i^2 \geq (\deg_H(S))^2 \geq 10^4 d^2 |S|^2$ , so using Item 2 of Definition 1 which ensures that  $\sum_{i \in S} x_i^2 \leq dn$ , we find that  $|S| \leq n/C$ .

Let  $f = (eC)^{1/C}$ . The number of sets  $S$  satisfying  $|S| \leq n/C$  is at most  $n \binom{n}{n/C} \leq n f^n$  since there are at most  $n$  possibilities for  $|S|$  to consider, and for each of them  $\binom{n}{|S|} \leq \binom{n}{n/C}$ .

Fix any set  $S \subseteq V_H$  with  $|S| \leq n/C$  and consider the random construction of  $H$ , starting from half-edges in  $S^c$  (and choosing their mates in the pairing). Let

$$j = \left\lfloor \frac{\deg_H(S^c)}{2} \right\rfloor \geq \left\lfloor \frac{3|S^c|}{2} \right\rfloor \geq \left\lfloor \frac{3n(1 - \frac{1}{C})}{2} \right\rfloor \geq \frac{3n(1 - \frac{2}{C})}{2},$$

where the first inequality uses the fact that each  $x_i$  is at least 3 (from Item 1 of Definition 1) and the final inequality uses the fact that  $n$  is sufficiently large.

Note that the process initiates a pairing from at least  $j$  half-edges in  $S^c$ . For each  $i \in [j]$ , let  $Y_i$  be the indicator random variable for the event that the  $i$ 'th half-edge from which pairing is initiated connects to an endpoint in  $S$  (conditioned on the pairings of the first  $i - 1$  half-edges initiated from  $S^c$ ).

Let  $\varepsilon = 3(1 - 2/C)/(8d) \leq 1/2$ . Recall from Item 2 in Definition 1 that  $\sum_{i=1}^n x_i \leq dn$ . For any  $t \in [j]$  satisfying  $\sum_{i=1}^{t-1} Y_i < \varepsilon n/2$  we have

$$\Pr(Y_t = 1) \geq \frac{\deg_H(S) - \varepsilon n/2}{dn} > \frac{1 - \varepsilon}{2d} \geq \frac{1}{4d}.$$

Now let  $X_1, \dots, X_j$  be i.i.d. Bernoulli random variables which are 1 with probability  $1/(4d)$ . We can couple the evolution of these variables so that, for any  $t \in [j]$  satisfying  $\sum_{i=1}^{t-1} Y_i < \varepsilon n/2$ , we have  $\sum_{i=1}^t Y_i \geq \sum_{i=1}^t X_i$ . We conclude that  $\Pr(\sum_{i=1}^j Y_i < \varepsilon n/2) \leq \Pr(\sum_{i=1}^j X_i < \varepsilon n/2)$ .

To conclude we will show that  $nf^n \Pr(\sum_{i=1}^j X_i < \varepsilon n/2) = o(1)$ , implying that we can take  $\alpha = \varepsilon/(2d)$  since  $\varepsilon n/2 = \alpha dn \geq \alpha \deg_H(S)$ .

Let  $X = \sum_{i=1}^j X_i$  and  $\delta = 1/2$ . Note that  $\mathbb{E}[X] = j/(4d)$  and that

$$\frac{(1-\delta)j}{4d} \geq \frac{(1-\delta)3n(1-\frac{2}{C})}{8d} = \frac{\varepsilon n}{2}.$$

By a Chernoff bound,  $\Pr(X \leq \varepsilon n/2) \leq \Pr(X \leq (1-\delta)j/(4d)) \leq \exp(-j\delta^2/(8d))$ .

To conclude that  $nf^n \exp(-j\delta^2/(8d)) = o(1)$  we observe that  $f < \exp(3(1-2/C)\delta^2/(16d))$ . Taking  $\alpha = \varepsilon/(2d)$ , we conclude the proof. ◀

We can now prove the following result, which establishes the desired expansion properties of the multigraphs generated by the configuration model.

► **Lemma 15.** *Let  $d$  be a positive real number. There is a positive real number  $\alpha$  (depending on  $d$ ) such that the following is true with high probability when  $H = (V_H, E_H)$  is drawn from  $\text{CM}(n, \vec{x})$  uniformly over all degree sequences  $\vec{x} \in \mathcal{D}_{n,d}$ . For all connected vertex sets  $S \subseteq V_H$  with  $|S| \leq n/2$ , we have that  $e_H(S, S^c) \geq \alpha \deg_H(S)$ .*

**Proof.** We consider three cases.

**Case 1.** Consider all connected subsets  $S \subseteq V_H$  with  $(\log n)^2 \leq |S| \leq n/2$ . By Lemma 14 there is a positive real number  $\alpha'$  such that, whp, every such subset  $S$  has  $e_H(S, S^c) \geq \alpha' \deg_H(S)$ .

**Case 2.** Consider all connected subsets  $S \subseteq V_H$  with  $|S| \leq (\log n)^2$  and  $\deg_H(S) \geq 36$ .

- Consider first those subsets  $S$  with  $|S| \leq \frac{1}{6} \deg_H(S)$ . We have that

$$e_H(S, S^c) = \deg_H(S) - 2(t_{H[S]} + |S| - 1) \geq \frac{2}{3} \deg_H(S) - 2|S| \geq \frac{1}{3} \deg_H(S),$$

by Lemma 11 and our assumption on the size of  $S$ .

- Now consider those subsets  $S$  with  $|S| > \frac{1}{6} \deg_H(S)$ , then by Lemma 13, we have that  $e_H(S, S^c) \geq |S|/4 \geq \deg_H(S)/24$ .

**Case 3.** Finally, consider connected subsets  $S \subseteq V_H$  with  $|S| \leq (\log n)^2$  and  $\deg_H(S) < 36$ .

By Lemma 13, we have that  $e_H(S, S^c) \geq |S|/4 \geq 1/4 = 36/144 > \deg_H(S)/144$ .

The result follows from the three cases by taking  $\alpha = \min\{1/144, \alpha'\} = \alpha'$ . ◀

Using the definition of  $\mathcal{G}_\alpha$  and Lemma 10, we have the following corollary of Lemma 15.

► **Corollary 16.** *Let  $d$  be a real number. There is a positive real number  $\alpha$  (depending on  $d$ ) such that the following holds. With high probability, when  $G \sim \mathcal{G}(n, \vec{x})$  for some  $\vec{x} \in \mathcal{D}_{n,d}$ , it holds that  $G \in \mathcal{G}_\alpha$ .*

Combining Corollary 16 with Theorem 9 implies our main theorem.

► **Theorem 2.** *Let  $d$  be a real number and  $q \geq 3$  be an integer. For the ferromagnetic Potts model, there is  $\beta_0$  such that for all  $\beta \geq \beta_0$  there is a poly-time approximate sampling algorithm for  $\mu_{G,q,\beta}$  and an FPRAS for  $Z_{G,q,\beta}$  that work with high probability on random graphs  $G \sim \mathcal{G}(n, \vec{x})$  for any degree sequence  $\vec{x} \in \mathcal{D}_{n,d}$ .*

**Proof.** Let  $d$  be a real number and let  $q \geq 2$  be an integer. Let  $\alpha$  be the positive real number from Corollary 16. Let  $\beta_0 = \frac{3}{\alpha} \log(8e^3(q-1))$ .

Consider  $\vec{x} \in \mathcal{D}_{n,d}$  and let  $G$  be drawn from  $\mathcal{G}(n, \vec{x})$ . By Corollary 16,  $G \in \mathcal{G}_\alpha$  whp. The result then follows by using the algorithms from Theorem 9. ◀

► Remark 17. The bounds on  $\beta$  in Remark 3 follow from the choice of  $\beta_0$  in the proof of Theorem 2 and from the fact that  $\alpha = O(\frac{1}{d \log d})$  which follows from the proofs of Lemmas 14 and 15. The running time bounds in Remark 3 come from those in Theorem 9 using the fact that  $|E_G| = O(n)$  which follows from Item 2 of Definition 1.

---

## References

- 1 Alexander Barvinok. *Combinatorics and complexity of partition functions*, volume 9. Springer, 2016.
- 2 Alexander Barvinok and Guus Regts. Weighted counting of solutions to sparse systems of equations. *Combinatorics, Probability and Computing*, 28(5):696–719, 2019.
- 3 Béla Bollobás. *The art of mathematics: Coffee time in Memphis*. Cambridge University Press, 2006.
- 4 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*, pages 738–751, 2020.
- 5 Sarah Cannon and Will Perkins. Counting independent sets in unbalanced bipartite graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1456–1466, 2020.
- 6 Charles Carlson, Ewan Davies, and Alexandra Kolla. Efficient algorithms for the Potts model on small-set expanders. *arXiv preprint arXiv:2003.01154*, 2020.
- 7 Zongchen Chen, Andreas Galanis, Leslie Ann Goldberg, Will Perkins, James Stewart, and Eric Vigoda. Fast algorithms at low temperatures via Markov chains. *Random Struct. Algorithms*, 58(2):294–321, 2021. Theorems 5 and 6 from [arxiv.org/abs/1901.0665](https://arxiv.org/abs/1901.0665).
- 8 Nikolaos Fountoulakis and Bruce A Reed. The evolution of the mixing rate of a simple random walk on the giant component of a random graph. *Random Structures & Algorithms*, 33(1):68–86, 2008.
- 9 Tobias Friedrich, Andreas Göbel, Martin S. Krejca, and Marcus Pappik. A spectral independence view on hard-spheres via block dynamics, 2021. [arXiv:2102.07443](https://arxiv.org/abs/2102.07443).
- 10 Andreas Galanis, Leslie Ann Goldberg, and James Stewart. Fast Algorithms for General Spin Systems on Bipartite Expanders. In *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*, pages 37:1–37:14, 2020.
- 11 Andreas Galanis, Leslie Ann Goldberg, and James Stewart. Fast mixing via polymers for random graphs with unbounded degree. *CoRR*, abs/2105.00524, 2021. [arXiv:2105.00524](https://arxiv.org/abs/2105.00524).
- 12 Andreas Galanis, Daniel Stefankovic, Eric Vigoda, and Linji Yang. Ferromagnetic Potts model: Refined #BIS-hardness and related results. *SIAM Journal on Computing*, 45(6):2004–2065, 2016.
- 13 Leslie Ann Goldberg and Mark Jerrum. Approximating the partition function of the ferromagnetic Potts model. *Journal of the ACM*, 59(5):1–31, 2012.
- 14 Christian Gruber and Hervé Kunz. General properties of polymer systems. *Communications in Mathematical Physics*, 22(2):133–161, 1971.
- 15 Tyler Helmuth, Matthew Jenssen, and Will Perkins. Finite-size scaling, phase coexistence, and algorithms for the random cluster model on random graphs. *arXiv preprint arXiv:2006.11580*, 2020.
- 16 Tyler Helmuth, Will Perkins, and Guus Regts. Algorithmic Pirogov–Sinai theory. *Probability Theory and Related Fields*, 176(3):851–895, 2020.
- 17 Jeroen Huijben, Viresh Patel, and Guus Regts. Sampling from the low temperature Potts model through a Markov chain on flows. *CoRR*, abs/2103.07360, 2021. [arXiv:2103.07360](https://arxiv.org/abs/2103.07360).
- 18 Svante Janson. The probability that a random multigraph is simple. II. *Journal of Applied Probability*, 51(A):123–137, 2014.
- 19 Matthew Jenssen and Peter Keevash. Homomorphisms from the torus, 2020. [arXiv:2009.08315](https://arxiv.org/abs/2009.08315).

- 20 Matthew Jenssen, Peter Keevash, and Will Perkins. Algorithms for #BIS-hard problems on expander graphs. *SIAM Journal on Computing*, 49(4):681–710, 2020.
- 21 Roman Kotecký and David Preiss. Cluster expansion for abstract polymer models. *Communications in Mathematical Physics*, 103(3):491–498, 1986.
- 22 Chao Liao, Jiabao Lin, Pinyan Lu, and Zhenyu Mao. Counting independent sets and colorings on random regular bipartite graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*, pages 34:1–34:12, 2019.



# Deterministic Approximate Counting of Polynomial Threshold Functions via a Derandomized Regularity Lemma

Rocco A. Servedio ✉

Columbia University, New York, NY, USA

Li-Yang Tan ✉

Stanford University, CA, USA

---

## Abstract

We study the problem of deterministically approximating the number of satisfying assignments of a polynomial threshold function (PTF) over Boolean space. We present and analyze a scheme for transforming such algorithms for PTFs over *Gaussian space* into algorithms for the more challenging and more standard setting of Boolean space. Applying this transformation to existing algorithms for Gaussian space leads to new algorithms for Boolean space that improve on prior state-of-the-art results due to Meka and Zuckerman [19] and Kane [13]. Our approach is based on a bias-preserving derandomization of Meka and Zuckerman’s regularity lemma for polynomials [19] using the [23] pseudorandom generator for PTFs.

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

**Keywords and phrases** Derandomization, Polynomial threshold functions, deterministic approximate counting

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.37

**Category** RANDOM

**Funding** Rocco A. Servedio: NSF grants CCF-1814873, IIS-1838154, CCF-1563155, and Simons Collaboration on Algorithms and Geometry.

Li-Yang Tan: NSF CAREER award CCF-1942123.

**Acknowledgements** This material is based upon work supported by the National Science Foundation under grant numbers listed above. 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 (NSF).

## 1 Introduction

Unconditional derandomization has emerged as a major topic of inquiry in complexity theory over the past several decades. One important strand in this study is the development of deterministic algorithms that can perform *approximate counting* for various function classes: given the description of a function  $f \in \mathcal{C}$  and an accuracy parameter  $\epsilon > 0$ , deterministically output an estimate of the acceptance probability of  $f$  (i.e.  $\Pr_{\mathbf{x} \leftarrow \{-1,1\}^n} [f(\mathbf{x}) = 1]$ ) that is additively accurate to within  $\pm\epsilon$ . This problem is trivially easy to solve with a randomized algorithm, but is much more challenging if a deterministic algorithm is required. Indeed, recall that the P vs. BPP problem is essentially equivalent to solving the deterministic approximate counting problem for  $\mathcal{C}$  being the class of all polynomial-size circuits (and  $\epsilon = 0.1$ ).

In this work we focus on the class of *low-degree polynomial threshold functions*, an important class of functions that has been the subject of intensive study in unconditional derandomization in recent years [5, 12, 11, 13, 16, 19, 14, 3, 4, 15, 9, 10, 17, 23, 22].



© Rocco A. Servedio and Li-Yang Tan;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 37; pp. 37:1–37:18



Leibniz International Proceedings in Informatics

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

## 1.1 Background: Deterministic approximate counting algorithms for PTFs

A *degree- $d$  polynomial threshold function* (PTF) is a function  $f(x) = \text{sign}(p(x))$  where  $p(x_1, \dots, x_n)$  is a polynomial of degree at most  $d$  and  $\text{sign} : \mathbb{R} \rightarrow \{-1, 1\}$  outputs 1 iff its argument is nonnegative. Deterministic approximate counting algorithms for PTFs have been well studied in a number of different works, and the following table summarizes the runtimes of the fastest known algorithms prior to this work:

■ **Table 1** Prior work on deterministic approximate counting algorithms for degree- $d$  PTFs. In the runtime of [13],  $O_d(\cdot)$  hides an Ackermann-type dependence on  $d$ , and likewise, in the runtime of [4],  $O_{d,\epsilon}(\cdot)$  hides an Ackermann-type dependence on  $d$  and  $1/\epsilon$ . With the exception of [4], all these algorithms are based on the construction of pseudorandom generators for PTFs.

Reference	Runtime
[19]	$n^{(d/\epsilon)^{O(d)}}$
[13]	$n^{O_d(\text{poly}(1/\epsilon))}$
[4]	$O_{d,\epsilon}(1) \cdot n^{O(d)}$
[17]	$\text{poly}(n) \cdot \exp(2^{\tilde{O}(\sqrt{\log(1/\epsilon)})})$ for $d = 2$
[23]	$\exp(2^{\sqrt{d \log n}}) \cdot \text{quasipoly}(1/\epsilon)$

### 1.1.1 Algorithms for PTFs over Gaussian space

A fruitful theme that has emerged in the study of PTFs concerns the relationship between PTFs in the standard setting of *Boolean space* – PTFs over  $\{-1, 1\}^n$  endowed with the uniform distribution – and PTFs in the setting of *Gaussian space*: PTFs over  $\mathbb{R}^n$  endowed with the Gaussian measure  $N(0, 1)^n$ .

The Gaussian setting enjoys numerous useful features that are not afforded by the Boolean setting, most of them owing to the continuous nature of  $\mathbb{R}^n$  and the rotational invariance of the Gaussian measure. In fact, the problem of approximate counting of degree- $d$  PTFs over Gaussian space (i.e. approximating  $\Pr_{\mathbf{g} \leftarrow N(0,1)^n}[f(\mathbf{g}) = 1]$  where  $f$  is a degree- $d$  PTF) can be seen to be a *special case* of the same problem over Boolean space, in the sense that an algorithm for the latter setting can be used to obtain an algorithm with a comparable runtime for the former setting. (This is a consequence of the invariance principle [20].) For this reason, many works have focused on the special case of designing deterministic approximate counting algorithms for PTFs over Gaussian space. There are by now a number of results in this setting for which no counterparts are yet known for the more challenging Boolean setting:

**Contrasting the state of the art over Boolean and Gaussian space.** Comparing Tables 1 and 2, we note that the runtime of [12] is strictly better than those of [19]’s and [13]’s algorithms for the Boolean setting; the runtime of [14] is strictly better than that of [13]; the runtime of [3] is strictly better than that of [17]; and the runtime of [22] remains subexponential for  $d = 2^{\tilde{\Omega}(\sqrt{\log n})}$ , whereas all algorithms for the Boolean setting trivialize once  $d = \Omega(\log n)$ .



■ **Table 2** Current best deterministic approximate counting algorithms for degree- $d$  PTFs over Gaussian space. With the exception of [3], all these algorithms are based on the construction of pseudorandom generators for PTFs over Gaussian space.

Reference	Running time
[12]	$n^{2^{O(d)} \cdot \text{poly}(1/\epsilon)}$
[14]	$n^{O_{d,\kappa}(1/\epsilon)^\kappa}$ for all $\kappa > 0$
[3]	$\text{poly}(n, 1/\epsilon)$ for $d = 2$
[22]	$n^{(d/\epsilon)^{O(\log d)}}$

## 1.2 This work: Upgrading algorithms for Gaussian space into ones for Boolean space

In this work we establish a new connection between the derandomization of PTFs over Boolean and Gaussian space. We leverage this connection to transform existing deterministic approximate counting algorithms for the Gaussian setting (i.e. those summarized in Table 2) into new state-of-the-art deterministic algorithms for approximate counting of PTFs for the more challenging Boolean setting, improving upon those summarized in Table 1.

The runtimes of our new algorithms improve upon the prior state of the art for a broad range of parameters. For  $d = \Theta(1)$ , we obtain a strict improvement for all  $\epsilon$  satisfying  $2^{-\Theta(\sqrt{\log n})} \leq \epsilon \leq o_n(1)$ . For  $d = \omega_n(1)$ , we obtain a strict improvement for all  $\epsilon$  satisfying  $d \log(d/\epsilon) \leq \Theta(\log n)$ . We now give precise statements of the runtimes of our new algorithms, and provide example parameter settings that highlight the main qualitative advantages of these new runtimes.

## 1.3 Our results: New deterministic approximate counting algorithms for PTFs over Boolean space

First, by instantiating our framework with [12]’s algorithm for the Gaussian setting, we obtain the following algorithm for the Boolean setting:

► **Theorem 1.** *There is a deterministic algorithm for  $\epsilon$ -approximate counting  $n$ -variable degree- $d$  PTFs over Boolean space that runs in time*

$$\exp\left(2^{O(d\sqrt{\log(d/\epsilon)})}\right) \cdot n^{2^{O(d)} \cdot \text{poly}(1/\epsilon)}.$$

This runtime is a strict improvement of [19]’s runtime and very nearly matches the  $n^{2^{O(d)} \cdot \text{poly}(1/\epsilon)}$  running time of the [12] algorithm for Gaussian space. For any  $\epsilon = \Theta(1/\text{polylog}(n))$ , our runtime remains  $n^{\text{polylog}(n)}$  for  $d$  as large as  $\tilde{\Omega}(\log \log n)$ , whereas the runtimes of all previous algorithms for the Boolean setting exceed  $n^{\text{polylog}(n)}$  once  $d$  is even a slightly superconstant function of  $n$ .

Next, by instantiating our framework with [14]’s algorithm for the Gaussian setting, we obtain the following algorithm for the Boolean setting:

► **Theorem 2.** *For all  $\kappa > 0$ , there is a deterministic algorithm for  $\epsilon$ -approximate counting  $n$ -variable degree- $d$  PTFs over Boolean space that runs in time*

$$n^{O_{d,\kappa}(1/\epsilon)^\kappa}.$$

This runtime is a strict improvement of [13]’s runtime and matches that of [14]’s algorithm for the Gaussian setting. For arbitrarily large constants  $c, d \in \mathbb{N}$  and  $\epsilon = 1/(\log n)^c$ , our runtime is barely superpolynomial,  $n^{O((\log n)^\kappa)}$  for any arbitrarily small constant  $\kappa > 0$ , whereas all previous algorithms for the Boolean setting run in time at least  $n^{(\log n)^{\Omega(c)}}$  or  $n^{(\log n)^{\Omega(d)}}$ .

■ **Table 3** Our new algorithms for deterministic approximate counting of degree- $d$  PTFs over Boolean space. The runtime of Theorem 1 is a strict improvement of [19]’s; the runtime of Theorem 2 is a strict improvement of [13]’s, and matches that of [14]’s algorithm for Gaussian space.

	Runtime	Follows by instantiating our framework with:
Theorem 1	$\exp\left(2^{O(d\sqrt{\log(d/\epsilon)})}\right) \cdot n^{2^{O(d)} \cdot \text{poly}(1/\epsilon)}$	[12]’s Gaussian PRG
Theorem 2	$n^{O_{d,\kappa}(1/\epsilon)^\kappa}$	[14]’s Gaussian PRG

Finally, we remark that:

- Our framework can also be instantiated with [3]’s algorithm for degree-2 PTFs in the Gaussian setting to recover [17]’s PRG-based algorithm for degree-2 PTFs in the Boolean setting (and in fact, we are able to improve it slightly by eliminating the  $\text{polylog}(1/\epsilon)$  factor suppressed by the  $\tilde{O}(\cdot)$  in its runtime);
- Our framework can also be instantiated with [22]’s algorithm for degree- $d$  PTFs in the Gaussian setting, yielding a deterministic algorithm for degree- $d$  PTFs over Boolean space that runs in time  $\exp\left(2^{O(d\sqrt{\log(d/\epsilon)})}\right) \cdot n^{(d/\epsilon)^{O(\log d)}}$ . Like Theorem 1, this runtime is a strict improvement of [19]’s runtime.

### 1.4 Our approach: Derandomizing Meka and Zuckerman’s regularity lemma

Our main new tool is a *derandomization of the [19] regularity lemma*. To explain what this means, we begin by recalling the basics of the original [19] regularity lemma.

A multivariate polynomial  $p$  is said to be *regular* if, intuitively, no variable has high influence (we give precise definitions in Section 2). Let us recall the original [19] regularity lemma: given any degree- $d$  polynomial over  $\{-1, 1\}^n$ , it gives an efficient (deterministic) algorithm which builds a not-too-large decision tree such that at almost every leaf  $\rho$ , the resulting polynomial  $p_\rho$  ( $p$  restricted according the partial assignment corresponding to that leaf) is such that either (i)  $p_\rho$  is regular, or (ii)  $\text{sign}(p_\rho)$  is close to either the constant  $+1$  function or the constant  $-1$  function.<sup>1</sup>

<sup>1</sup> We note that a similar regularity lemma was given in simultaneous work of [6] (see also [18]); that work employed a slightly different technical definition of what it means for a polynomial to be “regular”, and it gave a similar algorithm to build a decision tree with similar properties for that related notion. However, in the current work for technical reasons it is essential that we use the [19] notion of regularity; we explain this in more detail in Remark 22 in Appendix B.2.

The [19] regularity lemma is useful for derandomization because it lets one reduce the general Boolean case to the “regular” Boolean case, which can be easier because sophisticated mathematical tools like central limit theorems and invariance principles can be brought to bear on regular polynomials over  $\{-1, 1\}^n$  to relate them to the corresponding polynomials over the Gaussian domain. Indeed, the [19] regularity lemma and related results play an important role in a number of PRG and approximate counting results for polynomial threshold functions, including the works of [19, 13, 4] mentioned above. However there is often a substantial algorithmic cost associated with the use of a regularity lemma, because building the decision tree (or equivalently, exploring all of its leaves) can be relatively expensive.

**Our derandomization of the regularity lemma.** The above-described standard strategy of building and visiting all of the leaves of a decision tree corresponds to using true uniform randomness to choose a path through the decision tree. The intuition behind our derandomized version of the regularity lemma is as follows: by choosing a path through the decision tree according to a suitable *pseudorandom* distribution, it is possible, from an algorithmic perspective, to “build and visit only a tiny fraction of the leaves of the decision tree.” This can be much more efficient than visiting all leaves.

Intuitively, the leaves that our derandomized regularity lemma constructs are determined by the output of a PRG for degree- $d$  PTFs over  $m$  variables where  $m$  is the depth of the decision tree.<sup>2</sup> As our analysis shows, for the purpose of deterministic approximate counting for the original PTF  $\text{sign}(p)$ , it suffices to do deterministic approximate counting on just the PTFs  $\text{sign}(p_\rho)$  for these (relatively few) leaves  $\rho$ .

More precisely, we prove a general result, Theorem 15, which encapsulates the above approach. It outputs a collection of restrictions (which can be thought of as a very small subset of the leaves of the decision tree that the original regularity lemma constructs) with the following property: given an accurate estimate of the fraction of assignments satisfying  $\text{sign}(p_\rho)$  for each restriction  $\rho$  in the collection, combining these estimates in the obvious way gives an accurate estimate of the overall fraction of inputs in  $\{-1, 1\}^n$  that satisfy the original PTF  $\text{sign}(p)$ . Moreover (and crucially), each restriction  $\rho$  in the collection is such that either the restricted polynomial  $p_\rho$  is highly regular, or else  $\text{sign}(p_\rho)$  is a close-to-constant function.

For the purpose of deterministic approximate counting, restrictions where  $\text{sign}(p_\rho)$  is a close-to-constant function are easy to handle, and thanks to the invariance principle, at restrictions where  $p_\rho$  is regular we can do Gaussian deterministic approximate counting and the resulting estimate of  $\Pr_{\mathbf{g} \leftarrow N(0,1)^n}[\text{sign}(p_\rho(\mathbf{g})) = 1]$  will be an accurate estimate of  $\Pr_{\mathbf{x} \leftarrow \{-1,1\}^n}[\text{sign}(p_\rho(\mathbf{x}))]$  for the Boolean problem. Thus, the overall running time of the deterministic approximate counting algorithm we obtain from the regularity lemma is essentially the running time of (i) “fooling Boolean PTFs over few variables” (to build the tree) times the running time of (ii) “Gaussian deterministic approximate counting” (to handle the regular leaves).

### 1.4.1 Applying the derandomized regularity lemma

To obtain an efficient deterministic approximate counting algorithm from this approach, in part (i) above it is okay to use a PRG with a relatively poor dependence on the number of variables, since the number of variables is quite small. Such a generator is provided for us by

<sup>2</sup> This is actually an oversimplification: the regularity lemma works in a sequence of “atomic stages” to build a tree, and our approach actually works by derandomizing each atomic stage separately. The cost of a single atomic stage provides the dominant contribution to the overall cost, though, so the intuition is correct.

the [23] PRG (or rather a slight variant of it which we need for technical reasons); we can afford this generator's poor dependence on the number of variables, and using it lets us take advantage of its better dependence on the other parameters (it is clear from Table 1 that [23] has a better dependence on both  $d$  and  $\epsilon$  than any of the other algorithms in that table).

By applying our derandomized regularity lemma with (essentially) the [23] PRG for part (i) and the Gaussian result of [12] for part (ii), we obtain Theorem 1. By using instead [14]'s algorithm for the Gaussian setting for part (ii), we obtain Theorem 2.

## 2 Preliminaries

We start by establishing some basic notation. We write  $[n]$  to denote  $\{1, 2, \dots, n\}$  and  $[k, \ell]$  to denote  $\{k, k+1, \dots, \ell\}$ . We use bold font to denote random variables. We write  $\mathbf{E}[\mathbf{X}]$  and  $\mathbf{Var}[\mathbf{X}]$  to denote expectation and variance of a random variable  $\mathbf{X}$  and write  $\mathbf{E}_{\mathbf{X} \leftarrow \mathcal{D}}[\mathbf{X}]$ ,  $\mathbf{Var}_{\mathbf{X} \leftarrow \mathcal{D}}[\mathbf{X}]$ , and the like to indicate that the random variable  $\mathbf{X}$  has distribution  $\mathcal{D}$ . If  $S$  is a finite set then " $\mathbf{X} \leftarrow S$ " means that  $\mathbf{X}$  is distributed uniformly over  $S$ ; if no distribution is specified for a random variable taking values in  $\{-1, 1\}^n$  then the implied distribution is uniform over  $\{-1, 1\}^n$ . For  $x \in \{-1, 1\}^n$  and  $A \subseteq [n]$  we write  $x_A$  to denote  $(x_i)_{i \in A}$ .

For a function  $f : \{-1, 1\}^n \rightarrow \mathbb{R}$  and  $q \geq 1$ , we denote by  $\|f\|_q$  its  $\ell_q$  norm with respect to the uniform distribution, i.e.,  $\|f\|_q \stackrel{\text{def}}{=} \mathbf{E}[|p(\mathbf{x})|^q]^{1/q} = \mathbf{E}_{\mathbf{x} \leftarrow \{-1, 1\}^n}[|p(\mathbf{x})|^q]^{1/q}$ . We write  $\|f\|_{q, \mathcal{D}}$  to denote its  $\ell_q$  norm with respect to the distribution  $\mathcal{D}$ , i.e.  $\|f\|_{q, \mathcal{D}} \stackrel{\text{def}}{=} \mathbf{E}_{\mathbf{x} \leftarrow \mathcal{D}}[|p(\mathbf{x})|^q]^{1/q}$ .

For Boolean-valued functions  $f, g : \{-1, 1\}^n \rightarrow \{-1, 1\}$  the distance between  $f$  and  $g$ , denoted  $\text{dist}(f, g)$ , is  $\Pr[f(\mathbf{x}) \neq g(\mathbf{x})]$ .

### 2.1 Fourier analysis of Boolean functions

**Fourier Analysis over  $\{-1, 1\}^n$  and Influences.** We consider functions  $f : \{-1, 1\}^n \rightarrow \mathbb{R}$ , and we think of the inputs  $x$  to  $f$  as being distributed according to the uniform probability distribution. The set of such functions forms a  $2^n$ -dimensional real inner product space with inner product given by  $\langle f, g \rangle = \mathbf{E}[f(\mathbf{x})g(\mathbf{x})]$ . The set of functions  $(\chi_S)_{S \subseteq [n]}$  defined by  $\chi_S(x) = \prod_{i \in S} x_i$  forms a complete orthonormal basis for this space. Given a function  $f : \{-1, 1\}^n \rightarrow \mathbb{R}$  we define its *Fourier coefficients* by  $\hat{f}(S) \stackrel{\text{def}}{=} \mathbf{E}[f(\mathbf{x})\chi_S(\mathbf{x})]$ , and we have that  $f(x) = \sum_S \hat{f}(S)\chi_S(x)$ . We refer to the maximum  $|S|$  over all nonzero  $\hat{f}(S)$  as the *Fourier degree* of  $f$ .

As a consequence of orthonormality we have *Plancherel's identity*  $\langle f, g \rangle = \sum_S \hat{f}(S)\hat{g}(S)$ , which has as a special case *Parseval's identity*,  $\mathbf{E}[f(\mathbf{x})^2] = \sum_S \hat{f}(S)^2$ . From this it follows that for every  $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$  we have  $\sum_S \hat{f}(S)^2 = 1$ . We recall the well-known fact that the total influence  $\text{Inf}(f)$  of any Boolean function equals  $\sum_S \hat{f}(S)^2 |S|$ . Note that, in this setting, the expectation and the variance can be expressed in terms of the Fourier coefficients of  $f$  by  $\mathbf{E}[f] = \hat{f}(\emptyset)$  and  $\mathbf{Var}[f] = \sum_{\emptyset \neq S \subseteq [n]} \hat{f}(S)^2$ .

Let  $f : \{-1, 1\}^n \rightarrow \mathbb{R}$  and  $f(x) = \sum_S \hat{f}(S)\chi_S(x)$  be its Fourier expansion. The *influence* of variable  $i$  on  $f$  is  $\text{Inf}_i(f) \stackrel{\text{def}}{=} \sum_{S \ni i} \hat{f}(S)^2$ , and the *total influence* of  $f$  is  $\text{Inf}(f) = \sum_{i=1}^n \text{Inf}_i(f)$ .

**Bounded independence and bounded uniformity distributions.** A distribution  $\mathcal{D}$  on  $\{-1, 1\}^n$  is said to be *k-wise independent* if any collection of  $k$  distinct coordinates  $i_1, \dots, i_k$  are such that  $\mathbf{x}_{i_1}, \dots, \mathbf{x}_{i_k}$  are independent when  $\mathbf{x} \leftarrow \mathcal{D}$ . If each  $\mathbf{x}_i$  additionally is uniform

over  $\{-1, 1\}$ , then the distribution  $\mathcal{D}$  is said to be *k-wise uniform*. We note that if  $\mathcal{D}$  is a *k-wise uniform* distribution over  $\{-1, 1\}^n$ , then for any degree-*k* polynomial  $p$ , it holds by linearity of expectation that

$$\mathbf{E}_{z \leftarrow \mathcal{D}} [p(z)] = \mathbf{E}_{x \leftarrow \{-1, 1\}^n} [p(x)], \tag{1}$$

and hence if  $\mathcal{D}$  is  $(qk)$ -wise uniform for even  $q$ , it holds that

$$\|p\|_q = \|p\|_{q, \mathcal{D}}. \tag{2}$$

**Useful probability bounds.** We first recall the (2,4)-Hypercontractivity theorem of [2, 7]:

► **Theorem 3** ((2,4)-Hypercontractivity, special case of Theorem 9.21 of [21] / Lemma 4.3 of [8]). *Let  $p : \{-1, 1\}^n \rightarrow \mathbb{R}$  be a degree- $d$  multilinear polynomial. Then*

$$\|p\|_4 \leq 3^{d/2} \cdot \|p\|_2.$$

For our purposes we will need a *derandomized* version of Theorem 3, where the expectations are with respect to a suitable pseudorandom distribution. As an immediate consequence of Equation (2), we obtain the following corollary of Theorem 3:

► **Corollary 4** ((2,4)-Hypercontractivity for bounded-uniformity distributions). *Let  $p : \{-1, 1\}^n \rightarrow \mathbb{R}$  be a degree- $d$  multilinear polynomial. If  $\mathcal{D}$  is a  $4d$ -wise uniform distribution over  $\{-1, 1\}^n$ , then*

$$\|p\|_{4, \mathcal{D}} \leq 3^{d/2} \cdot \|p\|_{2, \mathcal{D}}.$$

We will need the following fact, which is a consequence of (2, 4)-hypercontractivity and states that a low-degree polynomial must exceed its expectation with nonnegligible probability:

► **Fact 5** (Lemma 5.4 of [8]). *Let  $p : \{-1, 1\}^n \rightarrow \mathbb{R}$  be a degree- $d$  multilinear polynomial normalized so that  $\mathbf{Var}[p] = 1$ . Then there is an absolute constant  $C > 0$  such that*

$$\Pr_{x \leftarrow \{-1, 1\}^n} [p(x) \geq \mathbf{E}[p] + 2^{-Cd}] \geq 2^{-O(d)}.$$

We will also need a derandomized version of Fact 5:

► **Fact 6** (Fact 5 for bounded-uniformity distributions). *Let  $p : \{-1, 1\}^n \rightarrow \mathbb{R}$  be a degree- $d$  multilinear polynomial normalized so that  $\mathbf{Var}[p] = 1$ . If  $\mathcal{D}$  is a  $4d$ -wise uniform distribution over  $\{-1, 1\}^n$ , then there is an absolute constant  $C > 0$  such that*

$$\Pr_{x \leftarrow \{-1, 1\}^n} [p(x) \geq \mathbf{E}[p] + 2^{-Cd}] \geq 2^{-O(d)}.$$

For the sake of completeness, we include the short proof of Fact 6 in Appendix A.

**Invariance.** We recall the invariance principle of Mossel, O’Donnell and Oleszkiewicz, specifically Theorem 3.19 under hypothesis **H4** in [20]:

► **Theorem 7** ([20]). *Let  $p(x) = \sum_{S \subseteq [n], |S| \leq d} \widehat{p}(S) \chi_S(x)$  be a degree- $d$  multilinear polynomial with  $\mathbf{Var}[p] = 1$ . Suppose each coordinate  $i \in [n]$  has  $\text{Inf}_i(p) \leq \tau$ . Then,*

$$\sup_{t \in \mathbb{R}} \left| \Pr_{x \leftarrow \{-1, 1\}^n} [p(x) \leq t] - \Pr_{g \leftarrow N(0, 1)^n} [p(g) \leq t] \right| \leq O(d\tau^{1/(8d)}).$$

## 2.2 PTFs, regularity, and the critical index

► **Definition 8** (Regularity). *We say that a degree- $d$  polynomial  $p$  is  $\tau$ -regular if*

$$\sqrt{\sum_{j=i}^n \text{Inf}_j(p)^2} \leq \tau \sum_{j=1}^n \text{Inf}_j(p).$$

► **Definition 9** ( $\tau$ -critical index). *Let  $p$  be a degree- $d$  polynomial, and assume (without loss of generality) that the variables of  $p$  are ordered so that  $\text{Inf}_1(p) \geq \text{Inf}_2(p) \geq \dots \geq \text{Inf}_n(p)$ . The  $\tau$ -critical index of  $p$  is the least  $i$  such that*

$$\text{Inf}_{i+1}(p) \leq \tau^2 \sum_{j=i+1}^n \text{Inf}_j(p).$$

## 3 Derandomizing Meka and Zuckerman's regularity lemma

### 3.1 Overview of [19]'s regularity lemma and its proof

In this subsection we state Meka and Zuckerman's regularity lemma for low-degree polynomials [19], and recall its key elements at a level of detail which will enable us to build on those elements.

► **Lemma 10** (Implicit in the proof of Lemma 5.17 of [19]). *There is a deterministic algorithm which, on input a degree- $d$  polynomial  $p$  and parameters  $\tau, \epsilon, \delta$ , outputs a decision tree of depth*

$$\text{depth}(d, \tau, \epsilon, \delta) := \frac{2^{O(d)}}{\tau^2} \cdot \log\left(\frac{1}{\delta}\right) \log\left(\frac{1}{\epsilon}\right)$$

*with the following property: with probability  $1 - \epsilon$ , a random path down the tree reaches a leaf  $\rho$  such that  $p_\rho$  is either*

1.  $\tau$ -regular, or
2. the PTF  $\text{sign}(p_\rho)$  is  $\delta$ -close to the constant function  $\text{sign}(\mathbf{E}[p_\rho])$ .

*The running time of this tree construction algorithm is  $\text{poly}(n^d, 2^{\text{depth}(d, \tau, \epsilon, \delta)})$ .*

The algorithm of [19] recursively constructs the tree in a sequence of simple “atomic steps”. We now describe how a single atomic step works. Consider a leaf  $\rho$  of the decision tree; initially the leaf  $\rho$  is simply the root of the tree corresponding to the empty restriction. The algorithm behaves differently depending on how large the  $\tau$ -critical index of  $p_\rho$  is:

**Large critical index.** If the polynomial  $p_\rho$  has “large”  $\tau$ -critical index (larger than a parameter  $K$  which is  $2^{O(d)} \log(1/\delta)/\tau^2$ ) then an “atomic step” consists of fixing the  $K$  variables which have the highest influence in  $p_\rho$ , i.e. replacing the current leaf with a complete depth- $K$  decision tree that exhaustively queries those variables. The key to the analysis of this case is the following structural result, which is Lemma 5.2 of [8]:

► **Lemma 11** (restatement of [8]'s Lemma 5.2: Large critical index). *There is a universal constant  $C_1 > 0$  such that the following holds. Let  $p : \{-1, 1\}^n \rightarrow \mathbb{R}$  be a degree- $d$  multilinear polynomial with  $\tau$ -critical index at least  $K := 2^{C_1 d} \log(1/\delta)/\tau^2$ . Then*

$$\Pr_{\rho \leftarrow \{-1, 1\}^{[K]}} [\mathbf{Var}[p_\rho] \leq \delta \mathbf{E}[p_\rho]^2] \geq 2^{-O(d)},$$

*and consequently by Chebyshev's inequality,*

$$\Pr_{\rho \leftarrow \{-1, 1\}^{[K]}} [\text{sign}(p_\rho) \text{ is } \delta\text{-close to } \text{sign}(\mathbf{E}[p_\rho])] \geq 2^{-O(d)}.$$

**Small critical index.** If the polynomial has small  $\tau$ -critical index (smaller than  $K$ ), then an “atomic step” consists of fixing the “head” variables  $[k]$  up to the critical index (again building a complete decision tree over those  $k \leq K$  variables). The key to the analysis of this case is the following structural result, which is Lemma 5.1 of [8]:

► **Lemma 12** (restatement of [8]’s Lemma 5.1: Small critical index). *Let  $p : \{-1, 1\}^n \rightarrow \mathbb{R}$  be a degree- $d$  multilinear polynomial with  $\tau$ -critical index  $k \in [n]$ . Then*

$$\Pr_{\rho \leftarrow \{-1, 1\}^{[k]}} [p_\rho \text{ is } \tau'\text{-regular}] \geq 2^{-O(d)}, \quad \text{where } \tau' \leq 2^{O(d)} \cdot \tau.$$

Given these two results, a relatively straightforward analysis (which is in fact a special case of the analysis we give in the subsequent subsections) shows that after at most  $2^{O(d)} \log(1/\epsilon)$  levels of these “atomic steps”, at most an  $\epsilon$  fraction of paths will not have terminated either in a close-to-constant leaf or a regular leaf.

### 3.2 The high level idea of our approach: Derandomizing each “atomic step” in a bias-preserving manner

The first important technical ingredient of our approach is in the following two lemmas, Lemma 13 and 14, which give *derandomized* versions of Lemma 11 and 12 respectively. Intuitively, these results say that in each of Lemma 11 and 12, rather than considering the uniform distribution over all restrictions fixing  $[K]$  and  $[k]$  respectively, it suffices to consider instead a suitable *pseudorandom* distribution over restrictions.

► **Lemma 13** (Bounded uniformity suffices for Lemma 11: Large critical index). *Let  $p : \{-1, 1\}^n \rightarrow \mathbb{R}$  be a degree- $d$  multilinear polynomial with  $\tau$ -critical index at least  $K := 2^{C_1 d} \log(1/\delta)/\tau^2$ , where  $C_1$  is the universal constant from Lemma 11. Let  $\mathcal{D}$  be a  $4d$ -wise uniform distribution over  $\{-1, 1\}^K$ . Then*

$$\Pr_{\rho \leftarrow \mathcal{D}} [\mathbf{Var}[p_\rho] \leq \delta \mathbf{E}[p_\rho]^2] \geq 2^{-O(d)},$$

and consequently by Chebyshev’s inequality,

$$\Pr_{\rho \leftarrow \mathcal{D}} [\text{sign}(p_\rho) \text{ is } \delta\text{-close to } \text{sign}(\mathbf{E}[p_\rho])] \geq 2^{-O(d)}.$$

► **Lemma 14** (Bounded uniformity suffices for Lemma 12: Small critical index). *Let  $p : \{-1, 1\}^n \rightarrow \mathbb{R}$  be a degree- $d$  multilinear polynomial with  $\tau$ -critical index  $k \in [n]$ . Let  $\mathcal{D}$  be a  $4d$ -wise uniform distribution over  $\{-1, 1\}^k$ . Then*

$$\Pr_{\rho \leftarrow \mathcal{D}} [p_\rho \text{ is } \tau'\text{-regular}] \geq 2^{-O(d)}, \quad \text{where } \tau' \leq 2^{O(d)} \cdot \tau.$$

We prove Lemma 13 in Appendix B.1 and prove Lemma 14 in Appendix B.2. Combining these results with a PRG that fools degree- $d$  PTFs over  $m$  variables (where “ $m$ ” should be thought of as  $\ll n$ ), we establish our bias-preserving derandomized regularity lemma:

► **Theorem 15** (Bias-preserving derandomization of [19]’s regularity lemma, Lemma 10). *Let  $\mathcal{G}_{\text{PTF}}$  be a PRG with seed length  $s(m, d, \eta)$  that*

- (i) *is  $4d$ -wise uniform and*
- (ii)  *$\eta$ -fools degree- $d$  PTFs over  $m$  variables.*

### 37:10 Deterministic Approximate Counting of Polynomial Threshold Functions

There is a deterministic algorithm *BUILD-RESTRICTIONS* which, on input a degree- $d$  polynomial  $p : \{-1, 1\}^n \rightarrow \mathbb{R}$  and parameters  $\epsilon, \delta$ , and  $\tau$ , outputs a collection  $\mathcal{R}$  of restrictions,

$$|\mathcal{R}| \leq \exp(s(m, d, \eta) \cdot 2^{O(d)} \log(\frac{1}{\epsilon}))$$

where

$$m \leq \frac{2^{O(d)}}{\tau^2} \log(\frac{1}{\delta}) \quad \text{and} \quad \eta = \frac{\epsilon}{2^{O(d)} \log(\frac{1}{\epsilon})}$$

with the following property: with probability  $1 - \epsilon$  over a draw  $\rho \leftarrow \mathcal{R}$ , the restricted polynomial  $p_\rho$  is either

1.  $\tau$ -regular, or
2. the PTF  $\text{sign}(p_\rho)$  is  $\delta$ -close to the constant function  $\text{sign}(\mathbf{E}[p_\rho])$ .

Furthermore the collection of restrictions  $\mathcal{R}$  is bias-preserving, in the sense that

$$\left| \mathbf{E}_{\rho \leftarrow \mathcal{R}} \left[ \mathbf{E}_{\mathbf{x} \leftarrow \{-1, 1\}^n} [\text{sign}(p_\rho(\mathbf{x}))] \right] - \mathbf{E}_{\mathbf{x} \leftarrow \{-1, 1\}^n} [\text{sign}(p(\mathbf{x}))] \right| \leq \epsilon. \quad (3)$$

The running time of *BUILD-RESTRICTIONS* is  $\text{poly}(n^d, |\mathcal{R}|)$ .

We prove Theorem 15 in Section 3.3. At a high level, the argument is an extension of the analysis that establishes the original regularity lemma of [19] from Lemma 11 and 12.

In Section 4 we apply Theorem 15 to obtain new deterministic approximate counting results for degree- $d$  Boolean PTFs. We do this by instantiating the pseudorandom generator  $\mathcal{G}_{\text{PTF}}$  using (a slight variant of) the [23] pseudorandom generator, and by using invariance principles and pseudorandom generators for Gaussian PTFs to obtain the required estimates of  $\mathbf{E}_{\mathbf{x} \leftarrow \{-1, 1\}^n} [\text{sign}(p_\rho)]$  for the regular polynomials  $p_\rho$ .

### 3.3 Proof of Theorem 15: Bias-preserving derandomization of [19]'s regularity lemma

We start with a simple fact about bias preservation:

► **Fact 16** (Bias preservation). *Let  $p : \{-1, 1\}^n \rightarrow \mathbb{R}$  be a degree- $d$  polynomial and let  $H \sqcup T$  be a partition of  $[n]$  into two disjoint sets. Let  $\mathcal{D}$  be a pseudorandom distribution over  $\{-1, 1\}^H$  that  $\eta$ -fools degree- $d$  PTFs over the variables in  $H$ . Then*

$$\left| \mathbf{E}_{\substack{\mathbf{x} \leftarrow \{-1, 1\}^H \\ \mathbf{y} \leftarrow \{-1, 1\}^T}} [\text{sign}(p(\mathbf{x}, \mathbf{y}))] - \mathbf{E}_{\substack{\mathbf{z} \leftarrow \mathcal{D} \\ \mathbf{y} \leftarrow \{-1, 1\}^T}} [\text{sign}(p(\mathbf{z}, \mathbf{y}))] \right| \leq \eta. \quad (4)$$

**Proof.** This follows directly from the fact that for any fixed outcome  $y \in \{-1, 1\}^T$ , the function  $\text{sign}(p_y(x)) = \text{sign}(p(x, y))$  is a degree- $d$  PTF over the variables in  $H$  (i.e. the class of degree- $d$  PTFs is closed under restrictions). ◀

The following will be the key subroutine for our algorithm:

► **Lemma 17** (Single atomic step). *Let  $\mathcal{G}_{\text{PTF}}$  be a PRG with seed length  $s(m, d, \eta)$  that*

- (i) *is  $4d$ -wise uniform and*
- (ii)  *$\eta$ -fools degree- $d$  PTFs over  $m$  variables.*



There is a universal constant  $C_2 > 0$  such that the following holds. There is a deterministic algorithm *BUILD-RESTRICTIONS-ATOMIC* which, on input a degree- $d$  polynomial  $p : \{-1, 1\}^n \rightarrow \mathbb{R}$  and parameters  $\delta, \eta$ , and  $\tau$ , outputs a collection  $\mathcal{R}_{\text{atomic}}(p)$  of restrictions,

$$|\mathcal{R}_{\text{atomic}}(p)| \leq \exp(s(m, d, \eta)), \quad m \leq \frac{2^{O(d)}}{\tau^2} \log\left(\frac{1}{\delta}\right)$$

with the following property: with probability  $\geq 2^{-C_2 d}$  over a draw  $\rho \leftarrow \mathcal{R}_{\text{atomic}}(p)$ , the restricted polynomial  $p_\rho$  is either

1.  $\tau$ -regular, or
2. satisfies  $\mathbf{Var}[p_\rho] \leq \delta \mathbf{E}[p_\rho]^2$ , and consequently by Chebyshev's inequality, the PTF  $\text{sign}(p_\rho)$  is  $\delta$ -close to the constant function  $\text{sign}(\mathbf{E}[p_\rho])$ .

Furthermore, this collection of restrictions  $\mathcal{R}_{\text{atomic}}(p)$  is bias-preserving, in the sense that:

$$\left| \mathbf{E}_{\rho \leftarrow \mathcal{R}_{\text{atomic}}(p)} \left[ \mathbf{E}_{\mathbf{x} \leftarrow \{-1, 1\}^n} [\text{sign}(p_\rho(\mathbf{x}))] \right] - \mathbf{E}_{\mathbf{x} \leftarrow \{-1, 1\}^n} [\text{sign}(p(\mathbf{x}))] \right| \leq \eta. \quad (5)$$

The running time of *BUILD-RESTRICTIONS-ATOMIC* is  $\text{poly}(n^d, |\mathcal{R}_{\text{atomic}}(p)|)$ .

**Proof.** Define  $\bar{\tau} := \tau \cdot 2^{-C_3 d}$  where  $C_3 > 0$  is a universal constant that we will set later. The algorithm *BUILD-RESTRICTIONS-ATOMIC* begins by computing  $\text{Inf}_i(p)$  for all  $i \in [n]$ , which can be done deterministically in time  $\text{poly}(n^d)$  via the Fourier formula  $\text{Inf}_i(p) = \sum_{S \ni i} \hat{p}(S)^2$ . With these values, the algorithm then determines whether the  $\bar{\tau}$ -critical index of  $p$  is large (i.e. at least  $K := 2^{C_1 d} \log(1/\delta)/\bar{\tau}^2$  where  $C_1$  is the universal constant from Lemma 11) or small (i.e. at most  $k < K$ ). Let  $H \subseteq [n]$  be the  $K$  most influential variables of  $p$  in the case where  $p$  has large  $\bar{\tau}$ -critical index and the  $k$  most influential ones otherwise, and let  $T := [n] \setminus H$ .

We define  $\mathcal{R}_{\text{atomic}}(p)$  to be the set of all restrictions  $\rho \in \{-1, 1\}^H \times \{*\}^T$  such that  $\rho_H \in \text{range}(\mathcal{G}_{\text{PTF}})$ , where  $\mathcal{G}_{\text{PTF}}$  is a PRG with seed length  $s(|H|, d, \eta)$  that is  $4d$ -wise uniform and  $\eta$ -fools degree- $d$  PTFs over  $\{-1, 1\}^H$ . Note that the size of  $\mathcal{R}_{\text{atomic}}(p)$  is indeed as claimed in the statement of the lemma:

$$|\mathcal{R}_{\text{atomic}}(p)| \leq \exp(s(|H|, d, \eta)),$$

$$\text{where } |H| \leq |\text{range}(\mathcal{G}_{\text{PTF}})| \leq \frac{2^{O(d)}}{\bar{\tau}^2} \log\left(\frac{1}{\delta}\right) = \frac{2^{O(d)}}{\tau^2} \log\left(\frac{1}{\delta}\right).$$

By the  $4d$ -uniformity of  $\mathcal{G}_{\text{PTF}}$  and our definition of  $H$ , it follows from Lemma 13 and 14 that with probability  $\geq 2^{-O(d)}$  over a draw  $\rho \leftarrow \mathcal{R}_{\text{atomic}}$ , the restricted polynomial  $p_\rho$  is either

1.  $(2^{O(d)} \cdot \bar{\tau})$ -regular, or
2. satisfies  $\mathbf{Var}[p_\rho] \leq \delta \mathbf{E}[p_\rho]^2$ , and consequently by Chebyshev's inequality, the PTF  $\text{sign}(p_\rho)$  is  $\delta$ -close to the constant function  $\text{sign}(\mathbf{E}[p_\rho])$ .

We choose the universal constant  $C_3$  to ensure that  $2^{O(d)} \cdot \bar{\tau} = 2^{O(d)} \cdot \tau \cdot 2^{-C_3 d} \leq \tau$ . Finally, using the fact that  $\mathcal{G}_{\text{PTF}}$   $\eta$ -fools degree- $d$  PTFs over  $\{-1, 1\}^H$ , Equation (5) follows from Fact 16 and this completes the proof.  $\blacktriangleleft$

### 3.3.1 Composing single atomic steps: Proof of Theorem 15 given Lemma 17

At a very high level, Theorem 15 follows by recursive applications of Lemma 17. Given a degree- $d$  polynomial  $p$ , *BUILD-RESTRICTIONS* begins by calling the subroutine *BUILD-RESTRICTIONS-ATOMIC* of Lemma 17, which returns a set  $\mathcal{R}_{\text{atomic}}(p) =: \mathcal{R}^{(1)}$  of  $\exp(s(m, d, \eta))$  many restrictions satisfying the conclusion of Lemma 17. We call a restriction

### 37:12 Deterministic Approximate Counting of Polynomial Threshold Functions

$\rho \in \mathcal{R}_{\text{atomic}}(p)$  *good* if  $p_\rho$  is either  $\tau$ -regular or satisfies  $\mathbf{Var}[p_\rho] \leq \delta \mathbf{E}[p_\rho]^2$  (i.e. if  $p_\rho$  satisfies the conclusion of Lemma 17), and we call  $\rho$  *bad* otherwise. By Lemma 17, we have that

$$\Pr_{\rho \leftarrow \mathcal{R}^{(1)}}[\rho \text{ is good}] \geq 2^{-C_2 d}$$

and

$$\left| \mathbf{E}_{\rho \leftarrow \mathcal{R}^{(1)}} \left[ \mathbf{E}_{\mathbf{x} \leftarrow \{-1,1\}^n} [\text{sign}(p_\rho(\mathbf{x}))] \right] - \mathbf{E}_{\mathbf{x} \leftarrow \{-1,1\}^n} [\text{sign}(p(\mathbf{x}))] \right| \leq \eta.$$

BUILD-RESTRICTIONS cycles through all  $\rho \in \mathcal{R}_{\text{atomic}}(p)$  and determines if each is good or bad. Note that this can be done deterministically in overall time  $|\mathcal{R}_{\text{atomic}}(p)| \cdot \text{poly}(n^d)$  via the Fourier formulas  $\text{Inf}_i(q) = \sum_{S \ni i} \hat{q}(S)^2$  and  $\mathbf{Var}(q) = \sum_{S \neq \emptyset} \hat{q}(S)^2$ . For each bad restriction  $\rho$ , BUILD-RESTRICTIONS recursively calls the subroutine BUILD-RESTRICTIONS-ATOMIC on the restricted polynomial  $p_\rho$ , obtaining another set  $\mathcal{R}_{\text{atomic}}(p_\rho)$  of  $\exp(s(m, d, \eta))$  many restrictions satisfying the conclusion of Lemma 17. Consider the overall set of restrictions comprising of the good restrictions in  $\mathcal{R}_{\text{atomic}}(p)$ , along with the the bad  $\rho \in \mathcal{R}_{\text{atomic}}(p)$  extended by those in  $\mathcal{R}_{\text{atomic}}(p_\rho)$ , i.e.

$$\mathcal{R}^{(2)} := \{\rho: \rho \in \mathcal{R}_{\text{atomic}}(p) \text{ is good}\} \cup \{\rho \circ \rho': \rho \in \mathcal{R}_{\text{atomic}}(p) \text{ is bad, } \rho' \in \mathcal{R}_{\text{atomic}}(p_\rho)\}.$$

We have that

$$\begin{aligned} \Pr_{\rho \leftarrow \mathcal{R}^{(2)}}[\rho \text{ is good}] &= 1 - \Pr_{\rho \leftarrow \mathcal{R}^{(2)}}[\rho \text{ is bad}] \\ &= 1 - \Pr_{\rho \leftarrow \mathcal{R}^{(1)}}[\rho \text{ is bad}] \cdot \Pr_{\substack{\rho' \leftarrow \mathcal{R}^{(1)} \\ \rho' \leftarrow \mathcal{R}_{\text{atomic}}(p_\rho)}}[\rho' \text{ is bad} \mid \rho \text{ is bad}] \\ &\geq 1 - (1 - 2^{-C_2 d})^2, \end{aligned}$$

where  $C_2$  is the universal constant from Lemma 17, and

$$\left| \mathbf{E}_{\rho \leftarrow \mathcal{R}^{(2)}} \left[ \mathbf{E}_{\mathbf{x} \leftarrow \{-1,1\}^n} [\text{sign}(p_\rho(\mathbf{x}))] \right] - \mathbf{E}_{\mathbf{x} \leftarrow \{-1,1\}^n} [\text{sign}(p(\mathbf{x}))] \right| \leq 2\eta.$$

Iterating this argument and defining  $\mathcal{R}^{(j)}$  analogously for  $j > 2$ , we have that

$$\Pr_{\rho \leftarrow \mathcal{R}^{(j)}}[\rho \text{ is good}] \geq 1 - (1 - 2^{-C_2 d})^j \tag{6}$$

and

$$\left| \mathbf{E}_{\rho \leftarrow \mathcal{R}^{(j)}} \left[ \mathbf{E}_{\mathbf{x} \leftarrow \{-1,1\}^n} [\text{sign}(p_\rho(\mathbf{x}))] \right] - \mathbf{E}_{\mathbf{x} \leftarrow \{-1,1\}^n} [\text{sign}(p(\mathbf{x}))] \right| \leq j\eta. \tag{7}$$

By choosing  $j = 2^{O(d)} \log(1/\epsilon)$  we can make the RHS of Equation (6) at least  $1 - \epsilon$ , and by our choice of  $\eta = \epsilon/2^{O(d)} \log(1/\epsilon)$  we have that the RHS of Equation (7) is at most  $\epsilon$ . Finally, we note that

$$|\mathcal{R}^{(j)}| \leq \exp(s(m, d, \eta) \cdot j) = \exp(s(m, d, \eta) \cdot 2^{O(d)} \log(\frac{1}{\epsilon}))$$

and this completes the proof of Theorem 15 given Lemma 17. ◀

## 4 Instantiating our derandomized regularity lemma: Proofs of Theorems 1 and 2

### 4.1 The $\mathcal{G}_{\text{PTF}}$ PRG

To apply Theorem 15 we need a PRG  $\mathcal{G}_{\text{PTF}}$  that (i) is  $4d$ -wise uniform, and (ii)  $\eta$ -fools degree- $d$  PTFs over  $\{-1, 1\}^m$ . Since  $m$  (the number of variables) is quite small in Theorem 15, the idea is to use a PRG which has a poor dependence on this parameter but a good dependence on the error parameter  $\eta$  and the degree parameter  $d$ , since we will be able to take advantages of these good dependences on  $\eta$  and  $d$  while not having to “pay too much” for the poor dependence on  $m$ .

As mentioned earlier, the PRG for degree- $d$  PTFs from [23] is well suited to the purpose of achieving item (ii) above with good parameters for us. We recall the performance guarantee of [23]:

► **Theorem 18** (Special case<sup>3</sup> of Theorem 2 of [23]). *There is an efficient explicit PRG with seed length  $2^{O(\sqrt{d \log m})} + \text{polylog}(1/\eta)$  that  $\eta$ -fools the class of degree- $d$  PTFs over  $\{-1, 1\}^m$ .*

Regarding item (i), it is not clear that the [23] PRG is  $4d$ -wise uniform but we can easily augment it to be  $4d$ -wise uniform by simply performing a bitwise XOR with a  $4d$ -wise uniform distribution. The correctness of this is ensured by the following simple lemma:

► **Lemma 19.** *Let  $G_1 : \{-1, 1\}^{s_1} \rightarrow \{-1, 1\}^m$  be a PRG that  $\epsilon$ -fools the class of degree- $d$  PTFs over  $\{-1, 1\}^m$ . Let  $G_2 : \{-1, 1\}^{s_2} \rightarrow \{-1, 1\}^m$  be such that the distribution of  $G_2(\mathbf{r})$  is  $k$ -wise uniform for  $\mathbf{r}$  uniform random over  $\{-1, 1\}^{s_2}$ . Define  $G : \{-1, 1\}^{s_1+s_2} \rightarrow \{-1, 1\}^m$  by*

$$(G(r_1, r_2))_j = (G(r_1))_j \cdot (G(r_2))_j \quad \text{for } j \in [m].$$

*Then (i)  $G$   $\epsilon$ -fools the class of degree- $d$  PTFs over  $\{-1, 1\}^m$ , and (ii)  $G(\mathbf{r}_1, \mathbf{r}_2)$  is  $k$ -wise uniform for  $(\mathbf{r}_1, \mathbf{r}_2)$  uniform random over  $\{-1, 1\}^{s_1+s_2}$ .*

**Proof.** For (i), observe that if  $p(x_1, \dots, x_m)$  is a degree- $d$  polynomial, then for any fixed  $a \in \{-1, 1\}^m$  the function  $q(x_1, \dots, x_m) = p(a_1 x_1, \dots, a_m x_m)$  is also a degree- $d$  polynomial. It follows that for any fixed setting of  $r_1 \in \{-1, 1\}^{s_1}$ , the distribution

$$(G(\mathbf{r}_1, r_2))_{\mathbf{r}_1 \leftarrow \{-1, 1\}^{s_1}} = ((G(\mathbf{r}_1))_1 \cdot (G(r_2))_1, \dots, (G(\mathbf{r}_1))_m \cdot (G(r_2))_m)_{\mathbf{r}_1 \leftarrow \{-1, 1\}^{s_1}}$$

$\epsilon$ -fools the class of degree- $d$  PTFs over  $\{-1, 1\}^m$ , and (i) follows directly from this.

For (ii), similarly observe that if  $\mathbf{X} = (\mathbf{X}_1, \dots, \mathbf{X}_m)$  is a  $k$ -wise uniform random variable over  $\{-1, 1\}^m$  then for any fixed  $a \in \{-1, 1\}^m$ , the random variable  $(a_1 \mathbf{X}_1, \dots, a_m \mathbf{X}_m)$  is also  $k$ -wise uniform. It follows that for any fixed setting of  $r_1 \in \{-1, 1\}^{s_1}$ , the distribution

$$(G(r_1, \mathbf{r}_2))_{\mathbf{r}_2 \leftarrow \{-1, 1\}^{s_2}} = ((G(r_1))_1 \cdot (G(\mathbf{r}_2))_1, \dots, (G(r_1))_m \cdot (G(\mathbf{r}_2))_m)_{\mathbf{r}_2 \leftarrow \{-1, 1\}^{s_2}}$$

is  $k$ -wise uniform, and (ii) follows directly from this. ◀

<sup>3</sup> Theorem 2 of [23] gives a PRG with seed length  $2^{O(\sqrt{\log S})} + \text{polylog}(1/\eta)$  that  $\eta$ -fools the class of size- $S$  Threshold-of- $AC^0$  circuits. The current statement follows as a special case of this by observing that any degree- $d$   $m$ -variable PTF can be viewed as a Threshold-of-AND circuit of size at most  $S = O(m^d)$ , since there are at most  $\binom{m}{0} + \dots + \binom{m}{d}$  many ANDs over at most  $d$  out of  $m$  input variables.

## 37:14 Deterministic Approximate Counting of Polynomial Threshold Functions

Recalling the well-known fact (see e.g. [24]) that there are simple explicit pseudorandom generators with seed length  $O(k \log m)$  that output  $k$ -wise independent distributions over  $\{-1, 1\}^m$ , we get the following corollary of Theorem 18:

► **Corollary 20.** *There is an efficient explicit PRG  $\mathcal{G}_{\text{PTF}}$  with seed length  $s(m, d, \eta) = 2^{O(\sqrt{d \log m})} + \text{polylog}(1/\eta)$  that is  $4d$ -wise uniform and  $\eta$ -fools the class of degree- $d$  PTFs over  $\{-1, 1\}^m$ .*

### 4.2 Proof of Theorem 1

Recall that to prove Theorem 1, we must give a deterministic algorithm for  $\epsilon$ -approximate counting  $n$ -variable degree- $d$  PTFs over Boolean space that runs in time

$$\exp\left(2^{O(d\sqrt{\log(d/\epsilon)})}\right) \cdot n^{2^{O(d)} \cdot \text{poly}(1/\epsilon)}.$$

The algorithm operates in two stages. In the first stage, it runs the BUILD-RESTRICTIONS procedure given in Theorem 15 with its “ $\epsilon$ ” and “ $\delta$ ” parameters both set to  $\epsilon$ , its “ $\tau$ ” parameter set to  $(\epsilon/d)^{O(d)}$ , and  $\mathcal{G}_{\text{PTF}}$  being the PRG given in Corollary 20. With these parameter settings the “ $m$ ” of Theorem 15 is  $m = (d/\epsilon)^{O(d)}$  and the “ $\eta$ ” is  $\eta = \frac{\epsilon}{2^{O(d)} \log(\frac{1}{\epsilon})}$ . Recall that the running time of BUILD-RESTRICTIONS is

$$\begin{aligned} & \text{poly}\left(n^d, \exp\left(s(m, d, \eta) \cdot 2^{O(d)} \log\left(\frac{1}{\epsilon}\right)\right)\right) \\ &= \text{poly}\left(n^d, \exp\left(\left(2^{O(d\sqrt{\log(d/\epsilon)})} + \text{poly}(d, \log(1/\epsilon))\right) \cdot 2^{O(d)} \log\left(\frac{1}{\epsilon}\right)\right)\right) \\ &= \text{poly}\left(n^d, \exp\left(2^{O(d\sqrt{\log(d/\epsilon)})}\right)\right), \end{aligned} \tag{8}$$

and that BUILD-RESTRICTIONS outputs a set  $\mathcal{R}$  of at most

$$\exp\left(s(m, d, \eta) \cdot 2^{O(d)} \log\left(\frac{1}{\epsilon}\right)\right) = \exp\left(2^{O(d\sqrt{\log(d/\epsilon)})}\right)$$

many restrictions.

In the second stage, the algorithm exhaustively iterates over each restriction  $\rho \in \mathcal{R}$ . For each  $\rho \in \mathcal{R}$  it computes a value  $v_\rho$  as follows:

1. First, it computes  $\text{Inf}_i(p_\rho)$  for all variables  $i$  that were not fixed by the restriction  $\rho$  (recall that this can be done deterministically in time  $\text{poly}(n^d)$  via the Fourier formula  $\text{Inf}_i(p_\rho) = \sum_{S \ni i} \hat{p}_\rho(S)^2$ ). It uses these computed influences to determine whether or not  $p_\rho$  is  $\tau$ -regular according to Definition 8 (recall that  $\tau = (\epsilon/d)^{O(d)}$ ).
2. If  $p_\rho$  is  $\tau$ -regular, then it runs the [12] deterministic PRG-based algorithm for Gaussian space (recall Table 2) to obtain a  $\pm\epsilon$ -accurate estimate of  $\mathbf{E}_{\mathbf{g} \sim N(0,1)^n}[\text{sign}(p_\rho(\mathbf{g}))]$ . (Recall that the [12] algorithm takes time  $n^{2^{O(d)} \cdot \text{poly}(1/\epsilon)}$ .) It sets  $v_\rho$  to be the output of this algorithm.
3. Otherwise, if  $p_\rho$  is not  $\tau$ -regular, it simply sets  $v_\rho$  to be the value  $\text{sign}(\mathbf{E}[p_\rho]) = \text{sign}(\hat{p}_\rho(\emptyset)) \in \{-1, 1\}$ .

The final value  $v$  returned by the algorithm is the average over all  $\rho \in \mathcal{R}$  of the values  $v_\rho$ .

From Equation (8) and item (2) above we have that the running time of the algorithm is

$$\exp\left(2^{O(d\sqrt{\log(d/\epsilon)})}\right) \cdot n^{2^{O(d)} \cdot \text{poly}(1/\epsilon)},$$

as claimed in Theorem 1. To conclude the proof it remains only to argue that  $v$  is indeed within an additive  $\pm O(\epsilon)$  of the true value of  $\mathbf{E}_{\mathbf{x} \leftarrow \{-1, 1\}^n}[\text{sign}(p(\mathbf{x}))]$ . Recalling Equation (3)

and the fact that at most an  $\epsilon$  fraction of restrictions  $\rho \in \mathcal{R}$  are such that neither is  $p_\rho$   $\tau$ -regular nor is  $\text{sign}(p_\rho)$   $\delta$ -close (i.e.  $\epsilon$ -close) to the constant function  $\text{sign}(\mathbf{E}[p_\rho])$ , using item (2) above it suffices to show that for every  $\rho$  such that  $p_\rho$  is  $\tau$ -regular, it holds that

$$\left| \mathbf{E}_{\mathbf{g} \sim N(0,1)} [\text{sign}(p_\rho(\mathbf{g}))] - \mathbf{E}_{\mathbf{x} \sim \{-1,1\}^n} [\text{sign}(p_\rho(\mathbf{x}))] \right| \leq O(\epsilon).$$

Since  $p_\rho$  is  $\tau$ -regular, we have that

$$\max_{i \in [n]} \text{Inf}_i(p_\rho) \leq \sqrt{\sum_{j=i}^n \text{Inf}_j(p_\rho)^2} \leq \tau \sum_{j=1}^n \text{Inf}_j(p_\rho) \leq \tau d \cdot \mathbf{Var}[p_\rho],$$

where the last inequality uses that the total influence of a degree- $d$  polynomial is at most  $d$  times its variance. Hence by the invariance principle (Theorem 7), we have that

$$\left| \mathbf{E}_{\mathbf{g} \sim N(0,1)} [\text{sign}(p_\rho(\mathbf{g}))] - \mathbf{E}_{\mathbf{x} \sim \{-1,1\}^n} [\text{sign}(p_\rho(\mathbf{x}))] \right| \leq O(d(\tau d)^{1/8d}) = O(\epsilon)$$

as desired, where the last inequality is by our choice of  $\tau = (\epsilon/d)^{O(d)}$ . This concludes the proof of Theorem 1.  $\blacktriangleleft$

### 4.3 Proof of Theorem 2

To prove Theorem 2, we must give a deterministic algorithm for  $\epsilon$ -approximate counting  $n$ -variable degree- $d$  PTFs over Boolean space that runs in time  $n^{O_{d,\kappa}(1/\epsilon)^\kappa}$ .

The first stage of the algorithm here is identical to the first stage of the algorithm in the previous subsection, with the same parameter settings and running time. The second stage differs only in item (2), where now in the  $\tau$ -regular case we run the [14] deterministic PRG-based algorithm for Gaussian space, which runs in time  $n^{O_{d,\kappa}(1/\epsilon)^\kappa}$ .

The analysis of correctness (showing that the output of this algorithm is  $\pm O(\epsilon)$ -close to the right value) is identical to the previous subsection. The running time of the algorithm is  $\exp(2^{O(d\sqrt{\log(d/\epsilon)})}) \cdot n^{O_{d,\kappa}(1/\epsilon)^\kappa}$ , where we recall that the function of  $d$  and  $1/\kappa$  hidden by the big-Oh notation is very fast-growing, in fact of Ackermann type. We now observe that the first component of the running time,  $\exp(2^{O(d\sqrt{\log(d/\epsilon)})})$ , is asymptotically dominated by the second  $n^{O_{d,\kappa}(1/\epsilon)^\kappa}$  component, which gives us the final claimed  $n^{O_{d,\kappa}(1/\epsilon)^\kappa}$  runtime. We establish this by comparing  $d$  and  $\epsilon$  as follows:

- If  $d$  is less than  $(\log(1/\epsilon))^{1/3}$ , then the first component  $\exp(2^{O(d\sqrt{\log(d/\epsilon)})})$  is less than  $\exp(2^{(\log(1/\epsilon))^{0.9}})$ , whereas the second expression is  $n^{O_{d,\kappa}(1) \cdot (1/\epsilon)^\kappa} \geq \exp(O_{d,\kappa}(1) \cdot (1/\epsilon)^\kappa)$ . For  $(1/\epsilon)^\kappa$  to be as small as  $2^{(\log(1/\epsilon))^{0.9}}$  we would need  $\kappa \leq (\log(1/\epsilon))^{-0.1}$ , but having  $\kappa$  be this small means that the  $O_{d,\kappa}(1)$  factor will make  $O_{d,\kappa}(1) \cdot (1/\epsilon)^\kappa$  much bigger than  $2^{(\log(1/\epsilon))^{0.9}}$ .
  - If  $d$  is larger than  $(\log(1/\epsilon))^{1/3}$ , then the first expression  $\exp(2^{O(d\sqrt{\log(d/\epsilon)})})$  is less than  $\exp(2^{d^3})$ , whereas the second expression is still at least  $\exp(O_{d,\kappa}(1) \cdot (1/\epsilon)^\kappa)$ . Irrespective of the value of  $\kappa$ , the  $O_{d,\kappa}(1)$  in this second expression asymptotically dominates  $\exp(2^{d^3})$ .
- This concludes the proof of Theorem 2.  $\blacktriangleleft$

---

#### References

- 1 Noga Alon, Gregory Gutin, and Michael Krivelevich. Algorithms with large domination ratio. *J. Algorithms*, 50(1):118–134, 2004.

- 2 Aline Bonami. Etude des coefficients Fourier des fonctions de  $L^p(G)$ . *Ann. Inst. Fourier (Grenoble)*, 20(2):335–402, 1970.
- 3 Anindya De, Ilias Diakonikolas, and Rocco A. Servedio. Deterministic approximate counting for juntas of degree-2 polynomial threshold functions. In *Proceedings of the 29th Annual Conference on Computational Complexity (CCC)*, pages 229–240. IEEE, 2014.
- 4 Anindya De and Rocco A. Servedio. Efficient deterministic approximate counting for low-degree polynomial threshold functions. In *Proceedings of the 46th Annual Symposium on Theory of Computing (STOC)*, pages 832–841, 2014.
- 5 Ilias Diakonikolas, Daniel M. Kane, and Jelani Nelson. Bounded independence fools degree-2 threshold functions. In *Proc. 51st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 11–20, 2010.
- 6 Ilias Diakonikolas, Rocco A. Servedio, Li-Yang Tan, and Andrew Wan. A regularity lemma and low-weight approximators for low-degree polynomial threshold functions. *Theory of Computing*, 10:27–53, 2014.
- 7 Leonard Gross. Logarithmic Sobolev inequalities. *Amer. J. Math.*, 97(4):1061–1083, 1975.
- 8 Prahladh Harsha, Adam Klivans, and Raghu Meka. Bounding the sensitivity of polynomial threshold functions. *Theory of Computing*, 10(1):1–26, 2014. URL: <http://www.theoryofcomputing.org/articles/v010a001>.
- 9 Valentine Kabanets, Daniel M Kane, and Zhenjian Lu. A polynomial restriction lemma with applications. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 615–628, 2017.
- 10 Valentine Kabanets and Zhenjian Lu. Satisfiability and Derandomization for Small Polynomial Threshold Circuits. In *Proceedings of the 22nd International Conference on Randomization and Computation (RANDOM)*, volume 116, pages 46:1–46:19, 2018.
- 11 Daniel Kane.  $k$ -independent Gaussians fool polynomial threshold functions. In *Proceedings of the 26th Conference on Computational Complexity (CCC)*, pages 252–261, 2011.
- 12 Daniel Kane. A small PRG for polynomial threshold functions of Gaussians. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 257–266, 2011.
- 13 Daniel Kane. A structure theorem for poorly anticoncentrated Gaussian chaoses and applications to the study of polynomial threshold functions. In *Proceedings of the 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 91–100, 2012.
- 14 Daniel Kane. A pseudorandom generator for polynomial threshold functions of Gaussians with subpolynomial seed length. In *Proceedings of the 29th Annual Conference on Computational Complexity (CCC)*, pages 217–228, 2014.
- 15 Daniel Kane. A polylogarithmic PRG for degree 2 threshold functions in the Gaussian setting. In *Proceedings of the 30th Conference on Computational Complexity (CCC)*, pages 567–581, 2015.
- 16 Daniel Kane and Raghu Meka. A PRG for Lipschitz functions of polynomials with applications to sparsest cut. In *Proceedings of the 45th Annual Symposium on Theory of Computing (STOC)*, pages 1–10, 2013.
- 17 Daniel Kane and Sankeerth Rao. A PRG for Boolean PTF of degree 2 with seed length subpolynomial in  $\epsilon$  and logarithmic in  $n$ . In *Proceedings of the 33rd Computational Complexity Conference (CCC)*, pages 2:1–2:24, 2018.
- 18 Daniel M. Kane. The Correct Exponent for the Gotsman-Linial Conjecture. *computational complexity*, 23:151–175, 2014.
- 19 Raghu Meka and David Zuckerman. Pseudorandom generators for polynomial threshold functions. *SIAM Journal on Computing*, 42(3):1275–1301, 2013.
- 20 Elchannan Mossel, Ryan O’Donnell, and Krzysztof Oleszkiewicz. Noise stability of functions with low influences: Invariance and optimality. *Annals of Mathematics*, 171:295–341, 2010.
- 21 Ryan O’Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014. Available at <http://analysisofbooleanfunctions.net/>.

- 22 Ryan O'Donnell, Rocco A. Servedio, and Li-Yang Tan. Fooling Gaussian PTFs via Local Hyperconcentration. In *Proceedings of the 51st Annual Symposium on Theory of Computing (STOC)*, 2020. to appear.
- 23 Rocco A. Servedio and Li-Yang Tan. Luby-Velickovic-Wigderson Revisited: Improved Correlation Bounds and Pseudorandom Generators for Depth-Two Circuits. In *Proceedings of the 22nd International Conference on Randomization and Computation (RANDOM)*, pages 56:1–56:20, 2018.
- 24 Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012.

## A Proof of Fact 6

Let  $p : \{-1, 1\}^n \rightarrow \mathbb{R}$  be a degree- $d$  multilinear polynomial normalized so that  $\mathbf{Var}[p]$  (which is equal to  $\sum_{\emptyset \neq S, |S| \leq d} \hat{p}(S)^2$ ) equals 1, and let  $\mathcal{D}$  be a  $4d$ -wise uniform distribution over  $\{-1, 1\}^n$ . We consider the mean-zero random variable  $p(\mathbf{z}) - \mathbf{E}[p(\mathbf{z})] = p(\mathbf{z}) - \hat{p}(\emptyset)$ , where  $\mathbf{z} \leftarrow \mathcal{D}$ . We have that  $\mathbf{E}[p(\mathbf{z})] = 0$ ,  $\mathbf{E}[p(\mathbf{z})^2] = \mathbf{Var}_{\mathbf{z}}[p(\mathbf{z})] = \mathbf{Var}_{\mathbf{x} \leftarrow \{-1, 1\}^n}[p(\mathbf{x})] = 1$ , and by Corollary 4 (derandomized (2,4)-Hypercontractivity), we further have that  $\mathbf{E}[p(\mathbf{z})^4] \leq 9^d \mathbf{E}[p(\mathbf{z})^2]^2 = 9^d$ . Fact 6 now follows from the following simple fact:

► **Fact 21** ([1], Lemma 3.2). *Let  $A$  be a real valued random variable satisfying  $\mathbf{E}[A] = 0$ ,  $\mathbf{E}[A^2] = 1$  and  $\mathbf{E}[A^4] \leq b$ . Then  $\Pr[A \geq 1/(4\sqrt{b})] \geq 1/(4^4/3b)$ .*

## B Proof of Lemma 13 and Lemma 14

### B.1 Proof of Lemma 13 (derandomized Lemma 5.2 of [8]: large critical index)

We express  $p(x)$  as  $q(x) + r(x) + \mathbf{E}[p]$ , where

$$q(x) = \sum_{S \not\subseteq K} \hat{p}(S) \chi_S(x) \quad \text{and} \quad r(x) = \sum_{\substack{S \subseteq K \\ S \neq \emptyset}} \hat{p}(S) \chi_S(x).$$

[8]'s Lemma 5.2 follows from the following two claims:

1. For every constant  $c$  there is a sufficiently large constant  $C_1$  such that if  $K := 2^{C_1 d} \log(1/\delta)/\tau^2$ , then

$$\mathbf{E}_{\rho \leftarrow \{-1, 1\}^K} [\mathbf{Var}(p_\rho)] \leq \delta \cdot 2^{-cd} \cdot \mathbf{E}_{\rho \leftarrow \{-1, 1\}^K} [r(\rho)^2]. \quad (9)$$

(This is [8]'s Claim 5.6.) Consequently, by Markov's inequality, for all constants  $c$  and  $c'$  we can again choose  $C_1$  to be sufficiently large to ensure that

$$\Pr_{\rho \leftarrow \{-1, 1\}^K} \left[ \mathbf{Var}(p_\rho) \leq \delta \cdot 2^{-cd} \cdot \mathbf{E}_{\rho \leftarrow \{-1, 1\}^K} [r(\rho)^2] \right] \geq 1 - 2^{-c'd}. \quad (10)$$

2. There are constants  $b$  and  $b'$  such that

$$\Pr_{\rho \leftarrow \{-1, 1\}^K} \left[ \mathbf{E}[p_\rho]^2 \geq 2^{-bd} \cdot \mathbf{E}_{\rho \leftarrow \{-1, 1\}^K} [r(\rho)^2] \right] \geq 2^{-b'd}, \quad (11)$$

which follows from an application of Lemma 5.4 of [8].

The proof of Lemma 13 follows [8]’s proof of their Lemma 5.2 almost exactly. The only changes are that:

- the functions  $\rho \mapsto \mathbf{Var}(p_\rho)$  and  $\rho \mapsto r(\rho)^2$  are degree  $2d$  polynomials in  $\rho$ , and hence Equations (9) and (10) both hold for  $\rho$  drawn from any  $2d$ -wise independent distribution over  $\{-1, 1\}^K$ ;
- we use our derandomized version of Lemma 5.4 of [8], namely Fact 6, in place of Lemma 5.4 of [8] to deduce that Equation (11) also holds for  $\rho$  drawn from any  $2d$ -wise independent distribution over  $\{-1, 1\}^K$ .

The rest of the proof is unchanged.

## B.2 Proof of Lemma 14 (derandomized Lemma 5.1 of [8]: small critical index)

The proof of Lemma 14 follows [8]’s proof of their Lemma 5.1 almost exactly. The only changes are that

- we use our derandomized version of (2,4)-Hypercontractivity, namely Corollary 4, in place of (2,4)-Hypercontractivity (Lemma 4.3 of [8], which is used in the line immediately following Equation (5.1) of their paper);
- we use our derandomized version of Lemma 5.4 of [8], namely Fact 6, in place of Lemma 5.4 of [8], which is used two lines after Equation (5.2) of their paper.

The rest of the proof is unchanged.

► **Remark 22 (Motivating our notion of regularity).** Recall from Section 1.4 that the works of [6, 18] use a technically slightly different notion of “regularity.” In those works an  $n$ -variable multivariate polynomial  $p$  is considered to be  $\tau$ -regular if for every  $i \in [n]$  we have that  $\text{Inf}_i(p) \leq \tau \cdot \sum_{i=1}^n \text{Inf}_i(p)$ . Intuitively, we may view this as a notion of “regularity-in- $\ell_\infty$ ”, and the notion used in the current paper and in [8, 19] as a notion of “regularity-in- $\ell_2$ ”.

We remark here that the small critical index case (the subject of Lemma 14 and of Lemma 5.1 of [8]) is the reason why we need to work with the [8] notion of regularity-in- $\ell_2$  given in Definition 8 rather than the regularity-in- $\ell_\infty$  notion used in [6, 18]. In more detail, to handle the small critical index case using the regularity-in- $\ell_\infty$  notion, the analysis of [6, 18] uses an exponential tail bound for degree- $d$  polynomials (the “degree- $d$  Chernoff bound”, see Theorem 9.23 of [21]). However, derandomizing this degree- $d$  Chernoff bound requires  $dq$ -wise uniform distributions, where  $q$  depends on the parameters with which the degree- $d$  Chernoff bound is being applied, and it turns out that because of the way that the [6, 18] arguments employ the degree- $d$  Chernoff bound, this can be prohibitively expensive in our context. In contrast, recall from Section 2.1 that derandomizing Theorem 3 and Fact 5 (which is all that is needed to establish a derandomized version of the small critical index case using the regularity-in- $\ell_2$  notion, as outlined above) can be done using only  $4d$ -wise uniformity.



# Improved Product-Based High-Dimensional Expanders

Louis Golowich 

Department of Computer Science, Harvard University, Cambridge, MA, USA

---

## Abstract

---

High-dimensional expanders generalize the notion of expander graphs to higher-dimensional simplicial complexes. In contrast to expander graphs, only a handful of high-dimensional expander constructions have been proposed, and no elementary combinatorial construction with near-optimal expansion is known. In this paper, we introduce an improved combinatorial high-dimensional expander construction, by modifying a previous construction of Liu, Mohanty, and Yang (ITCS 2020), which is based on a high-dimensional variant of a tensor product. Our construction achieves a spectral gap of  $\Omega(\frac{1}{k^2})$  for random walks on the  $k$ -dimensional faces, which is only quadratically worse than the optimal bound of  $\Theta(\frac{1}{k})$ . Previous combinatorial constructions, including that of Liu, Mohanty, and Yang, only achieved a spectral gap that is exponentially small in  $k$ . We also present reasoning that suggests our construction is optimal among similar product-based constructions.

**2012 ACM Subject Classification** Theory of computation → Expander graphs and randomness extractors

**Keywords and phrases** High-Dimensional Expander, Expander Graph, Random Walk

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.38

**Category** RANDOM

**Related Version** *Full Version:* <https://arxiv.org/abs/2105.09358>

**Acknowledgements** The author thanks Salil Vadhan for numerous helpful comments and discussions.

## 1 Introduction

Graphs that are sparse but well connected, called expander graphs, have numerous applications in various areas of computer science (see for example [8]). Recently, there has been much interest in generalizing the notion of expansion to higher-dimensional simplicial complexes, beginning with the work of Lineal and Meshulam [12], Meshulam and Wallach [17], and Gromov [7]. While multiple notions of high-dimensional expansion have been introduced (see the survey by Lubotzky [14]), these notions agree in that random simplicial complexes are not good high-dimensional expanders. In contrast, ordinary random graphs are near-optimal expanders with high probability. Therefore constructions of high-dimensional expanders are of particular interest. Early constructions of high-dimensional expanders, namely Ramanujan complexes [16, 15], were quite mathematically involved, whereas more simple and combinatorial constructions have been introduced recently.

In this paper, we introduce a modification of the high-dimensional expander construction of Liu, Mohanty, and Yang [13], which is based on a sort of high-dimensional tensor product. We then show that this modification gives an exponential improvement in the dependence of the  $k$ th order spectral gap on the dimension  $k$ , and we discuss why our results suggest this modification is optimal among constructions with the same general product structure. These results also address a question posed by Liu et al. [13] pertaining to the limitations of their product-based construction.



© Louis Golowich;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 38; pp. 38:1–38:17



Leibniz International Proceedings in Informatics

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

## 1.1 High-dimensional expanders

A high-dimensional expander is a simplicial complex with certain expansion properties. A **simplicial complex** is a hypergraph with downward-closed hyperedges, called faces. That is, a simplicial complex  $X$  on  $n$  vertices is a collection of faces  $X \subset 2^{[n]}$ , where for any face  $\sigma \in X$ , all subsets of  $\sigma$  are also faces in  $X$ . The **dimension** of a face  $\sigma$  is  $\dim(\sigma) = |\sigma| - 1$ , and the dimension of  $X$  is the maximum dimension of any face in  $X$ . The **1-skeleton** of a simplicial complex is the undirected  $n$ -vertex graph whose edges are given by the 1-dimensional faces. We restrict attention to **pure** simplicial complexes, meaning that every face is contained in a face of maximal dimension.

We consider the notion of high-dimensional expansion introduced by Kaufman and Mass [9], which requires rapid mixing for the high-order “up-down” or “down-up” random walks on simplicial complexes. The  $k$ -dimensional up-down walk on a simplicial complex specifies transition probabilities for a walk that alternates between faces of dimension  $k$  and  $k + 1$ . The 0-dimensional up-down walk is an ordinary lazy random walk on the 1-skeleton of the simplicial complex. Therefore 1-dimensional expanders are just ordinary (spectral) expander graphs. The spectral gaps of higher-order walks are difficult to bound directly, so they are instead typically bounded using a line of work [9, 5, 10, 1], which has shown that a large spectral gap for high-order walks is implied by good local expansion, that is, good spectral expansion of a specific set of graphs that describe the local structure of the simplicial complex. Formal definitions of high-order walks and local expansion are provided in Section 2.

We are interested in constructions of infinite families of high-dimensional expanders with bounded degree and spectral gap for all dimensions. Specifically, for any fixed  $H \geq 1$ , a  **$H$ -dimensional expander family** is a family  $\mathcal{X}$  of  $H$ -dimensional simplicial complexes such that there is no finite upper bound on the number of vertices of elements  $X \in \mathcal{X}$ , and the following two properties hold:

1. Bounded degree: There exists some  $\bar{d} > 0$  such that for every  $X \in \mathcal{X}$ , each vertex in  $X$  belongs to at most  $\bar{d}$  faces.
2. Bounded spectral gap: There exists some  $\underline{\nu} > 0$  such that for every  $X \in \mathcal{X}$  and every  $0 \leq k \leq H - 1$ , the  $k$ -dimensional up-down walk on  $X$  has spectral gap  $\geq \underline{\nu}$ .

In general, the spectral gap of the  $k$ -dimensional up down walk cannot be greater than  $\frac{2}{k+2}$  (see for example Proposition 3.3 of [1]). Our goal is to prove good lower bounds for this spectral gap for specific constructions of  $H$ -dimensional expander families. That is, we are interested in the optimal relationship between dimension and spectral gap, and only require an arbitrary upper bound on degree. This goal differs from the study of expander graphs, and specifically Ramanujan graphs, which focuses on the optimal relationship between degree and spectral gap.

While Kaufman and Mass [9] showed that Ramanujan complexes are high-dimensional expanders, multiple more elementary constructions have since been introduced [3, 4, 2, 13, 11, 6]. However, only three of these constructions [13, 11, 6] provide constant-degree high-dimensional expander families of all dimensions. The construction of Kaufman and Oppenheim [11] and the hyper-regular variant introduced by Friedgut and Iluz [6] are based on coset geometries, and achieve near-optimal expansion in all dimensions. In contrast, the construction of Liu et al. [13] is much more elementary, as it consists of a sort of high-dimensional tensor product between an expander graph and a constant-sized complete simplicial complex. However, this construction has suboptimal expansion in high dimensions. Specifically, Alev and Lau [1] showed that the  $k$ -dimensional up-down walk on the  $H$ -dimensional construction of Liu et al. [13] has spectral gap at least  $\frac{c}{(k+2)^{2k+2}}$ , where  $1 \leq c \leq 2$  is a constant depending on  $H$  and on the expander graph used in the construction. Note that this bound has exponential dependence on  $k$ , in contrast to the optimal linear dependence  $\Omega(\frac{1}{k})$ .

## 1.2 Contributions

In this paper, we present a modification of the high-dimensional expander family of Liu et al. [13], for which we show that the spectral gap of the  $k$ -dimensional up-down walk is at least  $\frac{1}{(1+\log H)(k+2)(k+1)}$ . This quadratic dependence on  $k$  provides an exponential improvement compared to the spectral gap bound of  $\frac{c}{(k+2)2^{k+2}}$  for the construction of Liu et al. [13]. We attain this exponential improvement using the same product structure as Liu et al. [13] while adjusting the weights of faces. Our modified construction also yields improved local expansion in high dimensions. For every  $1 \leq k \leq H - 2$ , we show that the 1-skeleton of the link of any  $k$ -dimensional face in our construction has spectral gap at least  $\frac{k+1}{k+2}$ , an improvement over the analogous bound of  $\frac{1}{2}$  for the construction of Liu et al. [13].

The organization of the remainder of this paper is as follows. Section 2 presents preliminary notions, and Section 3 presents our main construction along with some basic properties. In Section 4, we compute the local expansion of the construction, from which a result of Alev and Lau [1] implies rapid mixing of high-order walks. Section 5 discusses potential generalizations and limitations.

## 2 Background and preliminaries

This section provides basic definitions pertaining to simplicial complexes and high-dimensional expanders, as well as relevant past results.

► **Definition 1.** A *simplicial complex*  $X$  on  $n$  vertices is a subset  $X \subset 2^{[n]}$  such that if  $\sigma \in X$ , then all subsets of  $\sigma$  also belong to  $X$ . Let  $X(k) = \{\sigma \in X : |\sigma| = k + 1\}$ , and let the  *$k$ -skeleton* of  $X$  refer to the simplicial complex  $\bigcup_{\ell \leq k} X(\ell)$ . The elements of  $X(k)$  will be referred to as  *$k$ -dimensional faces*. The *dimension* of a simplicial complex is the maximum dimension of any of its faces, and if each face is contained in a face of maximal dimension, then the complex is *pure*. A *balanced weight function*  $m : X \rightarrow \mathbb{R}_+$  on a pure simplicial complex  $X$  is a function such that for every  $-1 \leq k < \dim(X)$  and every  $\sigma \in X(k)$ ,

$$m(\sigma) = \sum_{\tau : \sigma \subset \tau \in X(k+1)} m(\tau).$$

The 1-skeleton of a simplicial complex  $X$  with balanced weight function  $m$  is the undirected weighted graph  $(X(0), X(1), m)$  that has vertices  $[n] = X(0)$ , edges  $X(1)$ , and edge weights  $m(e)$  for  $e \in X(1)$ . The weighted degree of a vertex  $x$  in this graph is given by the weight  $m(x)$ .

This paper will restrict attention to pure weighted simplicial complexes with balanced weight functions. In this case, the faces and weight function may be defined only on faces of maximal dimension, then propagated downwards, and the following useful formula applies.

► **Lemma 2** ([18]). For every  $H$ -dimensional simplicial complex  $X$ , every  $-1 \leq k \leq H$ , and every  $\sigma \in X(k)$ ,

$$m(\sigma) = (H - k)! \sum_{\tau : \sigma \subset \tau \in X(H)} m(\tau).$$

Just as ordinary expander graph families are specified to have bounded degree, we are interested in families of simplicial complexes satisfying an analogous notion:

► **Definition 3.** A family  $\mathcal{X}$  of simplicial complexes has **bounded degree** if there exists some constant  $\bar{d}$  such that for every  $X \in \mathcal{X}$  and every vertex  $x \in X(0)$ , there are at most  $\bar{d}$  faces in  $X$  that contain  $x$ .

The local properties of a simplicial complex are captured by the links of faces, defined below.

► **Definition 4.** For a simplicial complex  $X$  with weight function  $m$ , the **link**  $X_\sigma$  of any  $\sigma \in X$  is the simplicial complex defined by  $X_\sigma = \{\tau \setminus \sigma : \sigma \subset \tau \in X\}$  with weight function  $m_\sigma(\tau \setminus \sigma) = m(\tau)$ .

A common theme in the study of high-dimensional expanders is the “local-to-global” paradigm, which uses bounds on the expansion of the 1-skeletons of links to prove global expansion properties. To state such local-to-global results, it is first necessary to define graph expansion.

► **Definition 5.** For a graph  $G$ , the adjacency matrix is denoted  $M_G$ , the diagonal degree matrix is denoted  $D_G$ , and the random walk matrix is denoted  $W_G = M_G D_G^{-1}$ . The eigenvalues of the random walk matrix are denoted from greatest to least by  $\omega_i(G) = \omega_i(W_G)$ . The **expansion**, or **spectral gap**, of  $G$  is the quantity  $\nu_2(G) = \nu_2(W_G) = 1 - \omega_2(G)$ .

For any  $n$ -vertex graph  $G$ , all eigenvalues  $\omega_i(W_G)$  of the random walk matrix lie in  $[-1, 1]$ , and  $\omega_1(W_G) = 1$ , so  $0 \leq \nu_2(G) \leq 1 + \frac{1}{n-1}$  because  $\sum_i \omega_i(W_G) = \text{Tr}(W_G) \geq 0$ . Graphs with spectral gaps closer to 1 are considered “better” expanders.

We now introduce a notion of expansion for simplicial complexes.

► **Definition 6.** For  $-1 \leq k \leq H - 2$ , the  **$k$ -dimensional local expansion** of a simplicial complex  $X$  with weight function  $m$  is the value

$$\nu^{(k)}(X) = \min_{\sigma \in X(k)} \nu_2(X_\sigma(0), X_\sigma(1), m_\sigma).$$

The **local expansion** of  $X$  is the minimum of the  $k$ -dimensional local expansion over all  $k \geq 0$ , while the **global expansion** equals the  $(-1)$ -dimensional local expansion.

That is, the  $k$ -dimensional local expansion refers to the lowest expansion of the 1-skeleton of the link of any  $k$ -dimensional face. Note that the terminology  $k$ -dimensional local expansion is nonstandard, but will be useful here. The following result shows that good local expansion in higher dimensions implies good local expansion in lower dimensions.

► **Proposition 7 ([18]).** Let  $X$  be a simplicial complex in which all links of dimension  $\geq 1$  are connected. Then for every  $0 \leq k \leq \dim(X) - 2$ ,

$$\nu^{(k-1)}(X) \geq 2 - \frac{1}{\nu^{(k)}(X)}.$$

In particular, Proposition 7 implies that if  $\nu^{(k)}(X) \geq 1 - \epsilon$ , then  $\nu^{(k-1)}(X) \geq 1 - O(\epsilon)$ .

The definition below presents the high-order random walks on simplicial complexes.

► **Definition 8.** Fix a pure  $H$ -dimensional simplicial complex  $X$ . For  $-1 \leq k \leq H - 1$ , define the **up-step random walk operator**  $W_k^\uparrow \in \mathbb{R}^{X(k+1) \times X(k)}$  so that for  $\sigma \in X(k)$ ,  $\tau \in X(k+1)$ ,

$$W_k^\uparrow(\tau, \sigma) = \begin{cases} \frac{m(\tau)}{m(\sigma)}, & \sigma \subset \tau \in X(k+1) \\ 0, & \text{otherwise} \end{cases}$$

For  $0 \leq k \leq H$ , define the **down-step random walk operator**  $W_k^\downarrow \in \mathbb{R}^{X(k-1) \times X(k)}$  so that for  $\sigma \in X(k), \tau \in X(k-1)$ ,

$$W_k^\downarrow(\tau, \sigma) = \begin{cases} \frac{1}{k+1}, & \sigma \supset \tau \in X(k-1) \\ 0, & \text{otherwise.} \end{cases}$$

Define the **up-down and down-up random walk operators** by  $W_k^{\uparrow\downarrow} = W_{k+1}^\downarrow \circ W_k^\uparrow$  and  $W_k^{\downarrow\uparrow} = W_{k-1}^\uparrow \circ W_k^\downarrow$  respectively.

For context with this definition, consider a 1-dimensional simplicial complex  $X$ , which may be viewed as a weighted, undirected graph. Then the up-step operator  $W_0^\uparrow$  moves from a vertex to an adjacent edge with probability proportional to its weight, while the down-step operator  $W_1^\downarrow$  moves from an edge to either of its vertices with probability  $\frac{1}{2}$ . Thus  $W_0^{\uparrow\downarrow} = W_1^\downarrow \circ W_0^\uparrow$  is the ordinary graph lazy random walk operator, with stationary probability  $\frac{1}{2}$ .

The nonzero elements of the spectra of  $W_k^{\uparrow\downarrow}$  and of  $W_{k+1}^{\downarrow\uparrow}$  are identical. Therefore when studying the expansion of these operators, we restrict attention without loss of generality to  $W_k^{\uparrow\downarrow}$ .

The spectral gap  $\nu_2(W_k^{\uparrow\downarrow})$  gives a measure of high-dimensional expansion. Local expansion, defined above, provides another notion of high-dimensional expansion. The following result, which follows the ‘‘local-to-global’’ paradigm, shows that these two notions are closely related.

► **Theorem 9** ([1]). *Let  $X$  be an  $H$ -dimensional simplicial complex, and let  $W_k^{\uparrow\downarrow}$  refer to the up-down walk operator on  $X$ . Then for every  $0 \leq k \leq H-1$ ,*

$$\nu_2(W_k^{\uparrow\downarrow}) \geq \frac{1}{k+2} \prod_{j=-1}^{k-1} \nu^{(j)}(X).$$

Thus if a simplicial complex has good local expansion, then its high-order walks have large spectral gaps. We apply this result to show our main high-dimensional expansion bound.

The bound in Theorem 9 is nearly tight for good local expanders:

► **Proposition 10** ([1]). *Let  $X$  be an  $H$ -dimensional simplicial complex with at least  $2(H+1)$  vertices, and let  $W_k^{\uparrow\downarrow}$  refer to the up-down walk operator on  $X$ . Then for every  $0 \leq k \leq H-1$ ,*

$$\nu_2(W_k^{\uparrow\downarrow}) \leq \frac{2}{k+2}.$$

The main purpose of this paper is to present a construction of high-dimensional expander families, defined below.

► **Definition 11.** *A family  $\mathcal{X}$  of  $H$ -dimensional simplicial complexes is an  **$H$ -dimensional expander family** if the following conditions hold:*

1. *For every  $n \in \mathbb{N}$ , there exists some  $X \in \mathcal{X}$  with  $|X(0)| \geq n$ .*
2.  *$\mathcal{X}$  has bounded degree.*
3. *For every  $0 \leq k \leq H-1$ , there exists some  $\underline{\nu}_{\mathcal{X},k} > 0$  such that for every  $X \in \mathcal{X}$ , the  $k$ -dimensional up-down walk operator  $W_k^{\uparrow\downarrow}$  on  $X$  satisfies  $\nu_2(W_k^{\uparrow\downarrow}) \geq \underline{\nu}_{\mathcal{X},k}$ .*

We are interested in constructing high-dimensional expander families  $\mathcal{X}$  of all dimensions  $H$  with the up-down walk spectral gaps  $\underline{\nu}_{\mathcal{X},k}$  close to the upper bound in Proposition 10. By Theorem 9, item 3 in Definition 11 is implied by a uniform lower bound on the local expansion  $\nu^{(k)}(X)$  of all  $X \in \mathcal{X}$  for every  $-1 \leq k \leq H-1$ . We use this fact in the analysis of our construction.

### 3 Construction

The following definition introduces the simplicial complex  $Z$  with weight function  $m$  considered in this paper. The construction takes as input an  $n$ -vertex graph  $G$ , which will typically be chosen from a family of expanders, as well as a dimension  $H$  and a parameter  $s \geq H + 1$ , the latter of which will typically be taken as  $s = 2H$ . The parameters  $H$  and  $s$  will typically be treated as fixed values, while  $G$  and  $n$  vary, so that the construction provides a family of  $H$ -dimensional simplicial complexes.

► **Definition 12.** Let  $G = (V(G), E(G), w_G)$  be any connected undirected graph on  $n$  vertices with no self-loops. For positive integers  $H$  and  $s$ , define the  $H$ -dimensional simplicial complex  $Z$  with vertex set  $V(G) \times [s]$  such that

$$Z(H) = \{(v_1, b_1), \dots, (v_{H+1}, b_{H+1})\} \subset V(G) \times [s] : \\ \exists \{u, v\} \in E(G) \text{ s.t. } \{v_1, \dots, v_{H+1}\} = \{u, v\} \text{ and } b_1, \dots, b_{H+1} \text{ are all distinct}\}.$$

Define a weight function  $m$  on  $Z$  so that if  $\sigma = \{(v_1, b_1), \dots, (v_{H+1}, b_{H+1})\} \in Z(H)$  is such that  $|\{i : v_i = u\}| = j$  and  $|\{i : v_i = v\}| = H + 1 - j$  for some edge  $\{u, v\} \in E(G)$ , then let

$$m(\sigma) = \frac{w_G(\{u, v\})}{\binom{H-1}{j-1}}.$$

In words, the  $H$ -dimensional faces of  $Z$  are those sets of the form  $\{(v_1, b_1), \dots, (v_{H+1}, b_{H+1})\}$  such that all  $v_i$  are contained in a single edge of  $G$ , and such that all  $b_i$  are distinct. If the above definition is modified so that the condition  $\{v_1, \dots, v_{H+1}\} = \{u, v\}$  is replaced with  $\{v_1, \dots, v_{H+1}\} \subset \{u, v\}$  and so that  $m(\sigma) = 1$  for all  $H$ -dimensional faces  $\sigma$ , then the resulting simplicial complex  $Q$  is exactly the construction of Liu et al. [13]. The difference between the weight functions of  $Z$  and  $Q$  lead to the key insights of this paper. Also note that here  $G$  may be a weighted graph, whereas Liu et al. [13] only considered complexes  $Q$  derived from unweighted graphs.

If  $G$  is unweighted so that  $w_G(\{u, v\}) = 1$  for all  $\{u, v\} \in E(G)$ , then every  $\sigma \in Z$  has  $(H - 1)!m(\sigma) \in \mathbb{N}$ . Thus  $Z$  may be viewed as an unweighted simplicial complex, meaning that all  $H$ -dimensional faces have the same weight, but where multiple copies of faces are permitted.

The simplicial complex  $Z$  may be viewed as a sort of high-dimensional tensor product of the graph  $G$  and the  $s$ -vertex,  $H$ -dimensional complete complex  $\mathcal{K}_{[s]}$ . In particular, the faces in  $Z(H)$  are exactly those  $(H + 1)$ -element subsets of  $V(G) \times [s]$  for which the projection onto the first component gives an edge in  $G$ , and the projection onto the second component gives a face in  $\mathcal{K}_{[s]}(H)$ . For comparison, an analogous property holds for the ordinary graph tensor product  $G_1 \otimes G_2$ , in which the edges are given by those pairs of elements of  $V(G_1) \times V(G_2)$  whose projection onto either component gives an edge in the respective graph  $G_1$  or  $G_2$ .

Note that as with  $Q$ , the simplicial complex  $Z$  has bounded degree with respect to  $n$  if  $G$  has bounded degree. In particular, fix some values  $H$  and  $s$ , and let  $G$  be chosen from a family of bounded degree graphs. For every  $\{u, v\} \in E(G)$  and  $1 \leq j \leq H$ , by definition

$$|\{\sigma = \{(v_1, b_1), \dots, (v_{H+1}, b_{H+1})\} \in Z(H) : |\{i : v_i = u\}| = j\}| = \binom{s}{H+1} \binom{H+1}{j}. \quad (1)$$

It follows by the definition of  $m$  that both the maximum weight and the maximum number of faces containing any face in  $Z$  are bounded by a constant. Furthermore, the number of faces in  $Z$  grows as  $O(n)$ .

An additional consequence of Equation (1) is that while the cardinality of the set on the left hand side, which equals the sum of the weights of the faces in this set under the weight function of  $Q$ , has an exponential dependence on  $j$ , the sum of the weights of the faces in this set under the weight function of  $Z$  is equal to  $\binom{s}{H+1} \binom{H+1}{j} / \binom{H-1}{j-1}$ , which has only a quadratic dependence on  $j$ . This observation provides some initial intuition for the exponential speedup of high-order walks on  $Z$  compared to  $Q$ .

From here on, the weight function  $m$  and the operators  $W_k^\uparrow, W_k^\downarrow, W_k^{\uparrow\downarrow}, W_k^{\downarrow\uparrow}$  will always refer to  $Z$ , unless explicitly stated otherwise.

### 3.1 Main result

Our main result, shown in Section 4, is stated below.

► **Theorem 13** (Restatement of Corollary 19). *Let  $W_k^{\uparrow\downarrow}$  be the up-down walk operator for the simplicial complex  $Z$  of Definition 12. If  $H \geq 2$ ,  $s \geq 2H$ , and  $n \geq 4$ , then for every  $0 \leq k \leq H - 1$ ,*

$$\nu_2(W_k^{\uparrow\downarrow}) \geq \frac{\nu_2(G)}{(1 + \log H)(k + 2)(k + 1)}.$$

In contrast to this quadratic dependence on  $k$ , Alev and Lau [1] showed that the spectral gap of the  $k$ -dimensional up-down walk on  $Q$  is at least  $\frac{c\nu_2(G)}{(k+2)2^{k+2}}$ , where  $c = c(G, H) \in [1, 2]$  depends on the structure of  $G$ . The discussion in Section 3.3 below provides intuition for why this exponential dependence in  $k$  arises for  $Q$ , and how the adjusted weights in  $Z$  yield the improved quadratic dependence.

Theorem 3.1 implies that for any fixed  $H$  with  $s = 2H$ , if  $G$  is chosen from a family of bounded degree expanders with spectral gap  $\nu > 0$ , the resulting simplicial complexes  $Z$  form a family of  $H$ -dimensional expanders with  $k$ -dimensional up-down walk spectral gap at least  $\frac{\nu}{(1+\log H)(k+2)(k+1)}$ . For comparison, an optimal  $H$ -dimensional expander family, as is given by simplicial complexes with optimal local expansion by Theorem 9, achieves a spectral gap of  $\Omega(\frac{1}{k})$  for the  $k$ -dimensional up-down walk.

### 3.2 Decomposition into permutation-invariant subsets

This section formalizes the intuitive notion that the construction of  $Z$  treats elements of  $[s]$  interchangeably, and introduces some notation to reflect this symmetry. For any  $1 \leq k \leq H + 1$ , the set of  $(k - 1)$ -dimensional faces  $Z(k - 1)$  may be decomposed as follows. For any  $\{u, v\} \in E(G)$  and  $0 \leq j \leq k$ , let

$$Z((j, k - j)_{(u, v)}) = \{ \{(v_1, b_1), \dots, (v_k, b_k)\} \in Z(H) : |\{i : v_i = u\}| = j, |\{i : v_i = v\}| = k - j \}$$

be the set of all  $(k - 1)$ -dimensional faces in  $Z$  that contain  $j$  vertices in  $\{u\} \times [s]$  and  $k - j$  vertices in  $\{v\} \times [s]$ . To remove redundancy when  $j = k$ , let

$$Z((k)_u) = Z((k, 0)_{(u, v)}).$$

Then by construction,

$$Z(k - 1) = \bigsqcup_{u \in V(G)} Z((k)_u) \sqcup \bigsqcup_{\{u, v\} \in E(G), 1 \leq j \leq k - 1} Z((j, k - j)_{(u, v)}).$$

This decomposition simply partitions  $Z$  into faces that differ only by permutations of  $[s]$ :

## 38:8 Improved Product-Based High-Dimensional Expanders

► **Lemma 14.** *For every  $1 \leq k \leq H + 1$ , there is a group action of  $S_{[s]} = \{\text{permutations } \pi : [s] \rightarrow [s]\}$  on the set of faces of  $Z$  given by*

$$\pi(\{(v_1, b_1), \dots, (v_k, b_k)\}) = \{(v_1, \pi(b_1)), \dots, (v_k, \pi(b_k))\}.$$

*The orbits of this action are exactly the sets  $Z((j, k - j)_{(u,v)})$ . The action preserves weights, that is,  $m \circ \pi = m$ .*

**Proof.** The construction of  $Z$  directly implies that for all  $\pi$ , if  $\sigma \in Z$  then  $\pi(\sigma) \in Z$ , so the group action on  $Z$  is well defined. Similarly, for all  $\pi$ , the definition of  $Z((j, k - j)_{(u,v)})$  directly implies that  $\pi(Z((j, k - j)_{(u,v)})) = Z((j, k - j)_{(u,v)})$ . For any  $\sigma = \{(v_1, b_1), \dots, (v_k, b_k)\}$ ,  $\sigma' = \{(v'_1, b'_1), \dots, (v'_k, b'_k)\} \in Z((j, k - j)_{(u,v)})$ , if  $\pi \in S_{[s]}$  is any permutation such that  $\pi(\{b_i : v_i = u\}) = \{b'_i : v'_i = u\}$  and  $\pi(\{b_i : v_i = v\}) = \{b'_i : v'_i = v\}$ , then  $\pi(\sigma) = \sigma'$ . Thus  $Z((j, k - j)_{(u,v)})$  is the orbit of  $\sigma$  under the group action. For any  $\pi$ , to verify that  $m \circ \pi = m$ , note that by definition  $m$  is constant over all values of  $Z((j, H + 1 - j)_{(u,v)})$  for any given  $0 \leq j \leq H + 1$  and  $\{u, v\} \in E(G)$ . Thus for all  $\sigma \in Z(H)$ , the characterization of the orbits above implies that  $m(\pi(\sigma)) = m(\sigma)$ . This equality then extends to  $\sigma$  of any dimension by Lemma 2. ◀

Loosely speaking, Lemma 14 says that elements of  $Z((j, k - j)_{(u,v)})$  may be treated interchangeably, which in particular permits the following definition.

► **Definition 15.** *For all  $1 \leq k \leq H + 1$ ,  $0 \leq j \leq k$ , and  $\{u, v\} \in E(G)$ , choose any  $\sigma \in Z((j, k - j)_{(u,v)})$  and define  $w_{(u,v)}^{(j,k-j)} = m(\sigma)$ . This definition does not depend on the choice of  $\sigma \in Z((j, k - j)_{(u,v)})$  by Lemma 14. To avoid redundancy, also define  $w_{(u,v)}^{(k)} = w_{(u,v)}^{(k,0)}$ .*

Note that  $m$  is defined by letting  $w_{(u,v)}^{(j,H+1-j)} = w_G(\{u, v\}) / \binom{H-1}{j-1}$ . A basic property of these weights is that for any  $1 \leq j \leq k - 1$ , the ratio  $w_{(u,v)}^{(j,k-j)} / w_G(\{u, v\})$  is independent of the choice of edge  $\{u, v\} \in E(G)$ , as is shown below.

► **Lemma 16.** *For all  $2 \leq k \leq H + 1$ ,  $1 \leq j \leq k - 1$ , and  $\{u, v\} \in E(G)$ ,*

$$\frac{w_{(u,v)}^{(j,k-j)}}{w_G(\{u, v\})} = (H + 1 - k)! \sum_{\ell=0}^{H+1-k} \binom{s-k}{\ell} \binom{s-k-\ell}{H+1-k-\ell} \cdot \frac{1}{\binom{H-1}{j+\ell-1}}.$$

**Proof.** For any  $\sigma \in Z((j, k - j)_{(u,v)})$ , any  $H$ -dimensional face  $\tau \supset \sigma$  must satisfy  $\tau \in Z((j + \ell, H + 1 - j - \ell)_{(u,v)})$  for some  $0 \leq \ell \leq H + 1 - k$ . Therefore by Lemma 2,

$$\begin{aligned} \frac{m(\sigma)}{w_G(\{u, v\})} &= \frac{(H + 1 - k)! \sum_{\ell=0}^{H+1-k} \sum_{\sigma \subset \tau \in Z((j+\ell, H+1-j-\ell)_{(u,v)})} m(\tau)}{w_G(\{u, v\})} \\ &= (H + 1 - k)! \sum_{\ell=0}^{H+1-k} \binom{s-k}{\ell} \binom{s-k-\ell}{H+1-k-\ell} \cdot \frac{1}{\binom{H-1}{j+\ell-1}}, \end{aligned}$$

where the final equality holds because there are exactly  $\binom{s-k}{\ell} \binom{s-k-\ell}{H+1-k-\ell}$  elements  $\tau \in Z((j + \ell, H + 1 - j - \ell)_{(u,v)})$  such that  $\tau \supset \sigma$ , and for each one  $m(\tau) = w_{(u,v)}^{(j+\ell, H+1-j-\ell)} = w_G(\{u, v\}) / \binom{H-1}{j+\ell-1}$  by definition. ◀



### 3.3 Relative weights of overlapping faces

The proposition below determines the relative weights of faces of  $Z$  that intersect at all but one of their vertices. This result is used in Section 4 to determine the local expansion of  $Z$ .

► **Proposition 17.** *For all  $1 \leq k \leq H$ ,  $1 \leq j \leq k - 1$ , and  $\{u, v\} \in E(G)$ , it holds that*

$$\frac{w_{(u,v)}^{(j+1,k-j)}}{w_{(u,v)}^{(j,k-j+1)}} = \frac{j}{k-j}.$$

Furthermore,

$$\frac{w_{(u)}^{(k+1)}}{\sum_{v \in N(u)} w_{(u,v)}^{(k,1)}} = k \sum_{i=k+1}^H \frac{1}{i}.$$

**Proof.** Both statements are shown using induction. For the first equality, the base case  $k = H$  follows by the definition of  $w_{(u,v)}^{(j,H+1-j)} = w_G(\{u, v\}) / \binom{H-1}{j-1}$ , so that

$$\frac{w_{(u,v)}^{(j+1,H-j)}}{w_{(u,v)}^{(j,H-j+1)}} = \frac{\binom{H-1}{j-1}}{\binom{H-1}{j}} = \frac{j}{H-j}.$$

For the inductive step, assume for some  $1 \leq k \leq H - 1$  that it holds for all  $1 \leq j \leq k$  that  $w_{(u,v)}^{(j+1,k+1-j)} / w_{(u,v)}^{(j,k+2-j)} = j / (k+1-j)$ . For any  $1 \leq j \leq k-1$  and any  $\sigma \in Z((j+1, k-j)_{(u,v)})$ , by definition any  $(k+1)$ -dimensional face  $\tau \supset \sigma$  is obtained from  $\sigma$  by adding either a vertex in  $\{u\} \times ([s] \setminus \Pi_{[s]}(\sigma))$  or in  $\{v\} \times ([s] \setminus \Pi_{[s]}(\sigma))$ . Therefore

$$\begin{aligned} w_{(u,v)}^{(j+1,k-j)} &= m(\sigma) = \sum_{\sigma \subset \tau \in Z(k+1)} m(\tau) \\ &= \sum_{\sigma \subset \tau \in Z((j+2,k-j)_{(u,v)})} m(\tau) + \sum_{\sigma \subset \tau \in Z((j+1,k-j+1)_{(u,v)})} m(\tau) \\ &= (s-k-1)w_{(u,v)}^{(j+2,k-j)} + w_{(u,v)}^{(j+1,k-j+1)}. \end{aligned} \quad (2)$$

Applying the exact same reasoning to a face  $\sigma' \in Z((j, k-j+1)_{(u,v)})$  gives that

$$w_{(u,v)}^{(j,k-j+1)} = m(\sigma') = (s-k-1)w_{(u,v)}^{(j+1,k-j+1)} + w_{(u,v)}^{(j,k-j+2)}.$$

Therefore

$$\begin{aligned} \frac{w_{(u,v)}^{(j+1,k-j)}}{w_{(u,v)}^{(j,k-j+1)}} &= \frac{(s-k-1)w_{(u,v)}^{(j+2,k-j)} + w_{(u,v)}^{(j+1,k-j+1)}}{(s-k-1)w_{(u,v)}^{(j+1,k-j+1)} + w_{(u,v)}^{(j,k-j+2)}} \\ &= \frac{w_{(u,v)}^{(j+2,k-j)} / w_{(u,v)}^{(j+1,k-j+1)} + 1}{1 + w_{(u,v)}^{(j,k-j+2)} / w_{(u,v)}^{(j+1,k-j+1)}} \\ &= \frac{(j+1)/(k-j) + 1}{1 + (k-j+1)/j} \\ &= \frac{j}{k-j}, \end{aligned}$$

completing the inductive step; note that the third equality above holds by the inductive hypothesis.

## 38:10 Improved Product-Based High-Dimensional Expanders

To show the second equality in the proposition statement, first note that the base case  $k = H$  holds immediately as  $w_{(u)}^{(H+1)} = 0$  because the definition of the complex  $Z$  does not include, or equivalently assigns zero weight, to faces in  $Z((H+1)_{(u)})$ . For the inductive step, assume that for some  $1 \leq k \leq H-1$  it holds that  $w_{(u)}^{(k+2)} / \sum_{v \in N(u)} w_{(u,v)}^{(k+1,1)} = (k+1) \sum_{i=k+2}^H \frac{1}{i}$ . For any  $\sigma \in Z((k+1)_{(u)})$ , by definition any  $(k+1)$ -dimensional face  $\tau \supset \sigma$  is obtained from  $\sigma$  by adding either a vertex in  $\{u\} \times ([s] \setminus \Pi_{[s]}(\sigma))$  or in  $\{v\} \times ([s] \setminus \Pi_{[s]}(\sigma))$  for some  $v \in N(u)$ . Therefore

$$\begin{aligned} w_{(u)}^{(k+1)} = m(\sigma) &= \sum_{\sigma \subset \tau \in Z(k+1)} m(\tau) \\ &= \sum_{\sigma \subset \tau \in Z((k+2)_{(u)})} m(\tau) + \sum_{v \in N(u)} \sum_{\sigma \subset \tau \in Z((k+1,1)_{(u,v)})} m(\tau) \\ &= (s-k-1) \left( w_{(u)}^{(k+2)} + \sum_{v \in N(u)} w_{(u,v)}^{(k+1,1)} \right). \end{aligned}$$

Applying (2) with  $j = k-1$  gives that

$$\sum_{v \in N(u)} w_{(u,v)}^{(k,1)} = (s-k-1) \left( \sum_{v \in N(u)} w_{(u,v)}^{(k+1,1)} + \sum_{v \in N(u)} w_{(u,v)}^{(k,2)} \right).$$

Therefore

$$\begin{aligned} \frac{w_{(u)}^{(k+1)}}{\sum_{v \in N(u)} w_{(u,v)}^{(k,1)}} &= \frac{(s-k-1) \left( w_{(u)}^{(k+2)} + \sum_{v \in N(u)} w_{(u,v)}^{(k+1,1)} \right)}{(s-k-1) \left( \sum_{v \in N(u)} w_{(u,v)}^{(k+1,1)} + \sum_{v \in N(u)} w_{(u,v)}^{(k,2)} \right)} \\ &= \frac{w_{(u)}^{(k+2)} / \left( \sum_{v \in N(u)} w_{(u,v)}^{(k+1,1)} \right) + 1}{1 + \left( \sum_{v \in N(u)} w_{(u,v)}^{(k,2)} \right) / \left( \sum_{v \in N(u)} w_{(u,v)}^{(k+1,1)} \right)} \\ &= \frac{(k+1) \sum_{i=k+2}^H \frac{1}{i} + 1}{1 + 1/k} \\ &= k \sum_{i=k+1}^H \frac{1}{i}, \end{aligned}$$

completing the inductive step; note that the third equality above holds by the inductive hypothesis, and because  $w_{(u,v)}^{(k,2)} = w_{(u,v)}^{(k+1,1)} / k$  for all  $v \in N(u)$  as was shown above. ◀

Proposition 17 provides the key insight for understanding why the spectral gap of the up-down walk on  $Z$  has a quadratic dependence in  $k$ , whereas that of  $Q$  has an exponential dependence. For some  $2 \leq k \leq H$ ,  $1 \leq j \leq k-1$ ,  $\{u, v\} \in E(G)$ , consider an element  $\sigma \in Z((j, k-j)_{(u,v)})$ . Let  $\sigma' \sim W_{k-1}^{\uparrow \downarrow} \mathbf{1}_\sigma$  be the random variable for the face obtained by taking one step in the up-down walk starting at  $\sigma$ . Then  $\sigma' \in Z((j', k-j')_{(u,v)})$  for some  $j' \in \{j+1, j, j-1\}$ . Let  $\Pi_{[s]}(\sigma)$  denote the subset of  $[s]$  obtained by projecting the elements of  $\sigma$  to their second components. If  $j' = j+1$ , then the up step must add some vertex in  $\{u\} \times ([s] \setminus \Pi_{[s]}(\sigma))$  and the down step must remove some vertex in  $(\{v\} \times [s]) \cap \sigma$ , while if  $j' = j-1$  then the up step must add some vertex in  $\{v\} \times ([s] \setminus \Pi_{[s]}(\sigma))$  and the down step must remove some vertex in  $(\{u\} \times [s]) \cap \sigma$ . Thus by the definition of the up- and down-step

transition probabilities,

$$\begin{aligned}\Pr[j' = j + 1] &= \frac{w_{(u,v)}^{(j+1,k-j)}}{w_{(u,v)}^{(j+1,k-j)} + w_{(u,v)}^{(j,k-j+1)}} \cdot \frac{k-j}{k+1} \\ \Pr[j' = j - 1] &= \frac{w_{(u,v)}^{(j,k-j+1)}}{w_{(u,v)}^{(j+1,k-j)} + w_{(u,v)}^{(j,k-j+1)}} \cdot \frac{j}{k+1}.\end{aligned}\tag{3}$$

For the construction  $Q$  of Liu et al. [13], these same expressions hold, but  $w_{(u,v)}^{(j+1,k-j)} = w_{(u,v)}^{(j,k-j+1)}$ , so that when  $j$  is close to 1 or close to  $k$ , the transition probabilities in (3) are heavily skewed to push  $j'$  in the direction of  $k/2$ . It is this property that results in an exponential dependence on  $k$  in the  $k$ th order up-down walk on  $Q$ ; the up-down walk becomes “trapped” in the set of faces contained in  $\{u, v\} \times [s]$ , with the transition probabilities pushing away from the “exit routes”  $Z((k)_{(u)})$  and  $Z((k)_{(v)})$ .

To understand why the weight function  $m$  on  $Z$  resolves this issue, observe that for  $Z$ , Proposition 17 implies that both probabilities in (3) equal  $\frac{j(k-j)}{k(k+1)}$ . Therefore the events  $j' = j + 1$  and  $j' = j - 1$  are equally likely. Thus the up-down walk moves across the sets  $Z((j, k-j)_{(u,v)})$  for  $1 \leq j \leq k-1$  as a lazy random walk on an unweighted, undirected,  $(k-1)$ -vertex path. The mixing time for such a walk grows quadratically in  $k$ , thereby providing intuition for the quadratic dependence in  $k$  for the mixing time of  $W_{k-1}^{\uparrow\downarrow}$ .

The intuition described above can be formalized to bound the mixing time of the high-order walks on  $Z$ . However, the following section takes a different approach by computing the local expansion of  $Z$ , which leads to tighter bounds on  $\nu_2(W_k^{\uparrow\downarrow})$ .

## 4 Local and global expansion

This section analyzes the local and global expansion of  $Z$ , which is then used to bound the mixing time of the up-down random walk using Theorem 9.

► **Theorem 18.** *If  $H \geq 2$ ,  $s \geq 2H$ , and  $n \geq 4$ , then for every  $0 \leq k \leq H - 2$ ,*

$$\nu^{(k)}(Z) = \frac{k+1}{k+2}.$$

Furthermore,

$$\nu^{(-1)}(Z) = \frac{\nu_2(G)}{\sum_{\ell=1}^H 1/\ell} \geq \frac{\nu_2(G)}{1 + \log H}.$$

Note that the local expansion  $\nu^{(k)}(Z)$  for  $k \geq 0$  does not depend on  $G$ . This property stems from the fact that the structure of any given link in  $Z$  depends only on the local structure of  $G$ , that is, on the weights of edges adjacent to a single vertex.

For comparison, the construction  $Q$  of Liu et al. [13] has local expansion  $\nu^{(k)}(Q) = \frac{1}{2}$  in each dimension  $k \geq 0$ , and has global expansion  $\nu^{(-1)}(Q)$  approaching  $\frac{1}{2}\nu_2(G)$  as  $H$  grows large. It was posed as an open question in Liu et al. [13] whether  $\frac{1}{2}$  is a natural barrier for local expansion in graph-product-based constructions. Theorem 18 gives a partial answer to this question, as although  $Z$  sacrifices a factor of  $O(\log H)$  in global expansion compared to  $Q$ , for all  $k \geq 1$  the local expansion  $\nu^{(k)}(Z) = \frac{k+1}{k+2}$  is an improvement on  $\nu^{(k)}(Q) = \frac{1}{2}$ . Section 5 provides a discussion suggesting that further improvements in local expansion are not attainable with a similar graph-product-based construction.

## 38:12 Improved Product-Based High-Dimensional Expanders

The result below applies Theorem 9, shown by Alev and Lau [1], to show that the improvement in local expansion of  $Z$  compared to  $Q$  for  $k \geq 1$  results in an exponential improvement with respect to  $k$  of the spectral gap of the  $k$ th order up-down walk.

► **Corollary 19.** *Let  $W_k^{\uparrow\downarrow}$  be the up-down walk operator for the simplicial complex  $Z$ . If  $H \geq 2$ ,  $s \geq 2H$ , and  $n \geq 4$ , then for all  $0 \leq k \leq H - 1$ ,*

$$\nu_2(W_k^{\uparrow\downarrow}) \geq \frac{\nu_2(G)}{(\sum_{\ell=1}^H 1/\ell)(k+2)(k+1)} \geq \frac{\nu_2(G)}{(1 + \log H)(k+2)(k+1)}.$$

**Proof.** Applying Theorem 9 with Theorem 18 gives that

$$\nu_2(W_k^{\uparrow\downarrow}) \geq \frac{1}{k+2} \cdot \frac{\nu_2(G)}{\sum_{\ell=1}^H 1/\ell} \prod_{j=0}^{k-1} \frac{j+1}{j+2} = \frac{\nu_2(G)}{(\sum_{\ell=1}^H 1/\ell)(k+2)(k+1)}. \quad \blacktriangleleft$$

Because by definition  $W_k^{\uparrow\downarrow}$  has self loop probabilities of  $1/(k+2)$ , for all  $i$  it holds that  $\omega_i(W_k^{\uparrow\downarrow}) \geq -k/(k+2)$ . Therefore assuming that  $G$  is chosen from a family of graphs with bounded ratio of maximum degree to minimum degree, then the mixing time of the random walk  $W_k^{\uparrow\downarrow}$  grows as  $O(\frac{k^2(\log H)(\log n)}{\nu_2(G)})$ .

In contrast, the spectral gap of the  $k$ th order random walk on the construction  $Q$  of Liu et al. [13] was shown in Alev and Lau [1] to grow as  $\Omega(\frac{\nu_2(G)}{k2^k})$ , for a mixing time of  $O(\frac{k2^k \log n}{\nu_2(G)})$ .

### 4.1 Proof of Theorem 18

To prove Theorem 18, we first compute the expansion of the 1-skeleton of every link in  $Z$  in the following lemmas.

► **Lemma 20.** *For every  $2 \leq k \leq H - 1$ ,  $1 \leq j \leq k - 1$ , and  $\{u, v\} \in E(G)$ , every face  $\sigma \in Z((j, k - j)_{(u, v)})$  satisfies*

$$\omega_2(Z_\sigma(0), Z_\sigma(1), m_\sigma) = \frac{1}{k+1}.$$

**Proof.** Following the method of Liu et al. [13], the proof will proceed by identifying the 1-skeleton of each link of  $Z$  with a tensor product of two other graphs, whose spectra can be analyzed independently. For a face  $\sigma \in Z((j, k - j)_{(u, v)})$ , the link  $Z_\sigma$  by definition has vertex set  $Z_\sigma(0) = \{u, v\} \times ([s] \setminus \Pi_{[s]}(\sigma))$  and edge set

$$Z_\sigma(1) = \{\tau \setminus \sigma : \sigma \subset \tau \in Z(k+1)\} = \{(v_1, b_1), (v_2, b_2)\} \subset Z_\sigma(0) : b_1 \neq b_2\}.$$

For such an edge  $\{(v_1, b_1), (v_2, b_2)\}$ , let  $\tau = \sigma \cup \{(v_1, b_1), (v_2, b_2)\}$ , so that the edge's weight  $m(\tau)$  may be one of three possible values:

- If  $v_1 = v_2 = u$ , then  $\tau \in Z((j+2, k-j)_{(u, v)})$ , so  $m(\tau) = w_{(u, v)}^{(j+2, k-j)}$ .
- If  $v_1 = u, v_2 = v$ , then  $\tau \in Z((j+1, k-j+1)_{(u, v)})$ , so  $m(\tau) = w_{(u, v)}^{(j+1, k-j+1)}$ .
- If  $v_1 = v_2 = v$ , then  $\tau \in Z((j, k-j+2)_{(u, v)})$ , so  $m(\tau) = w_{(u, v)}^{(j, k-j+2)}$ .

Therefore letting  $P$  be the 2-vertex graph with adjacency matrix

$$M_P = \begin{pmatrix} w_{(u, v)}^{(j+2, k-j)} & w_{(u, v)}^{(j+1, k-j+1)} \\ w_{(u, v)}^{(j+1, k-j+1)} & w_{(u, v)}^{(j, k-j+2)} \end{pmatrix},$$

then the graph  $(Z_\sigma(0), Z_\sigma(1), m_\sigma)$  described above is exactly given by  $P \otimes K_{[s] \setminus \Pi_{[s]}(\sigma)}$ , where  $K_V$  denotes the complete graph without self-loops on vertex set  $V$ . By Proposition 17, it holds

that  $w_{(u,v)}^{(j+2,k-j)}/w_{(u,v)}^{(j+1,k-j+1)} = (j+1)/(k-j)$  and  $w_{(u,v)}^{(j+1,k-j+1)}/w_{(u,v)}^{(j,k-j+2)} = j/(k-j+1)$ , so the random walk matrix of  $P$  is given by

$$W_P = \frac{1}{k+1} \begin{pmatrix} j+1 & j \\ k-j & k-j+1 \end{pmatrix},$$

whose second eigenvalue is  $1/(k+1)$ , with eigenvector  $(1, -1)^T$ . Thus because 1 is the only positive eigenvalue of  $W_{K_{[s] \setminus \Pi_{[s]}(\sigma)}}$ , it follows that the second eigenvalue of  $W_P \otimes W_{K_{[s] \setminus \Pi_{[s]}(\sigma)}}$  is  $1/(k+1)$ , as desired.  $\blacktriangleleft$

► **Lemma 21.** *If  $s \geq 2H$ , then for every  $1 \leq k \leq H-1$  and  $u \in V(G)$ , every face  $\sigma \in Z((k)_{(u)})$  satisfies*

$$\omega_2(Z_\sigma(0), Z_\sigma(1), m_\sigma) = \frac{1}{k+1}.$$

**Proof.** The proof will proceed similarly to that of Lemma 20. Consider a face  $\sigma \in Z((k)_{(u)})$ , and let  $N(u) = \{v_1, \dots, v_d\}$ . The link  $Z_\sigma$  then has vertex set  $Z_\sigma(0) = (\{u\} \cup N(u)) \times ([s] \setminus \Pi_{[s]}(\sigma))$  and edge set

$$Z_\sigma(1) = \{\tau \setminus \sigma : \sigma \subset \tau \in Z(k+1)\} = \{\{(v, b_1), (v', b_2)\} \subset Z_\sigma(0) : b_1 \neq b_2, |\{v, v'\} \setminus \{u\}| \leq 1\}.$$

For such an edge  $\{(v, b_1), (v', b_2)\}$ , let  $\tau = \sigma \cup \{(v, b_1), (v', b_2)\}$ , so that there are three possible cases for the edge's weight  $m(\tau)$ :

- If  $v = v' = u$ , then  $\tau \in Z((k+2)_{(u)})$ , so  $m(\tau) = w_{(u)}^{(k+2)}$ .
- If  $v = u, v' = v_i$  for some  $i$ , then  $\tau \in Z((k+1, 1)_{(u, v_i)})$ , so  $m(\tau) = w_{(u, v_i)}^{(k+1, 1)}$ .
- If  $v = v' = v_i$  for some  $i$ , then  $\tau \in Z((k, 2)_{(u, v_i)})$ , so  $m(\tau) = w_{(u, v_i)}^{(k, 2)}$ .

Therefore letting  $S$  be the  $(d+1)$ -vertex star graph with adjacency matrix

$$M_S = \begin{pmatrix} w_{(u)}^{(k+2)} & w_{(u, v_1)}^{(k+1, 1)} & w_{(u, v_2)}^{(k+1, 1)} & \dots & w_{(u, v_d)}^{(k+1, 1)} \\ w_{(u, v_1)}^{(k+1, 1)} & w_{(u, v_1)}^{(k, 2)} & 0 & \dots & 0 \\ w_{(u, v_2)}^{(k+1, 1)} & 0 & w_{(u, v_2)}^{(k, 2)} & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ w_{(u, v_d)}^{(k+1, 1)} & 0 & \dots & 0 & w_{(u, v_d)}^{(k, 2)} \end{pmatrix},$$

it follows that the graph  $(Z_\sigma(0), Z_\sigma(1), m_\sigma)$  is exactly given by  $S \otimes K_{[s] \setminus \Pi_{[s]}(\sigma)}$ . Let  $x = w_{(u)}^{(k+2)} + \sum_{i=1}^d w_{(u, v_i)}^{(k+1, 1)}$ , so that by Proposition 17, the random walk matrix of  $S$  is therefore

$$W_S = \begin{pmatrix} w_{(u)}^{(k+2)}/x & k/(k+1) & k/(k+1) & \dots & k/(k+1) \\ w_{(u, v_1)}^{(k+1, 1)}/x & 1/(k+1) & 0 & \dots & 0 \\ w_{(u, v_2)}^{(k+1, 1)}/x & 0 & 1/(k+1) & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ w_{(u, v_d)}^{(k+1, 1)}/x & 0 & \dots & 0 & 1/(k+1) \end{pmatrix},$$

Let  $\delta_i \in \mathbb{R}^{d+1}$  denote the  $i$ th basis vector, so that by this expression for  $W_S$ , every vector in the codimension-2 subspace  $\text{span}\{\vec{1}, \delta_1\}^\perp$  is an eigenvector with eigenvalue  $1/(k+1)$ . The

### 38:14 Improved Product-Based High-Dimensional Expanders

final eigenvector in  $\text{span}\{\vec{1}\}^\perp$  is then given by  $(w_{(u)}^{(k+2)} - x, w_{(u,v_1)}^{(k+1,1)}, \dots, w_{(u,v_d)}^{(k+1,1)})^T$ , with eigenvalue

$$\frac{1}{k+1} - \frac{x - w_{(u)}^{(k+2)}}{x} = \frac{1}{k+1} - \frac{1}{1 + (k+1) \sum_{\ell=k+2}^H 1/\ell},$$

where the equality above holds by Proposition 17. Thus  $\omega_2(S) = 1/(k+1)$ . Because  $s \geq 2H$  and  $k \leq H-1$ , it follows that  $|[s] \setminus \Pi_{[s]}(\sigma)| = s - k \geq H+1$ . Therefore the eigenvalues of  $W_{K_{[s] \setminus \Pi_{[s]}(\sigma)}}$  are 1 and  $-1/(s-k-1)$ , with  $0 \leq 1/(s-k-1) \leq 1/H \leq 1/(k+1)$ , so it follows that all eigenvalues of  $W_{K_{[s] \setminus \Pi_{[s]}(\sigma)}}$  that do not equal 1 must have absolute value at most  $1/(k+1)$ . Therefore it follows from  $\omega_2(S) = 1/(k+1)$  that  $\omega_2(S \otimes K_{[s] \setminus \Pi_{[s]}(\sigma)}) = 1/(k+1)$ , as desired.  $\blacktriangleleft$

► **Lemma 22.** For  $1 \leq i \leq n$ , let

$$\tilde{\omega}_i(G) = \frac{1}{\sum_{\ell=1}^H 1/\ell} \omega_i(G) + \left(1 - \frac{1}{\sum_{\ell=1}^H 1/\ell}\right) \quad (4)$$

denote the eigenvalues of a lazy random walk on  $G$ . Then

$$\omega_2(Z(0), Z(1), m) = \max\{\tilde{\omega}_2(G), -\tilde{\omega}_n(G)/(s-1)\}.$$

**Proof.** Consider any  $\tau = \{(u, b_1), (v, b_2)\} \in Z(1)$ . If  $u = v$  then  $\tau \in Z((2)_{(u)})$  so that  $m(\tau) = w_{(u)}^{(2)}$ , while if  $u \neq v$  then  $\tau \in Z((1,1)_{(u,v)})$  so that  $m(\tau) = w_{(u,v)}^{(1,1)}$ . Therefore define  $\tilde{G}$  to be the undirected graph with  $V(\tilde{G}) = V(G)$ ,  $E(\tilde{G}) = E(G) \cup V(G)$ , and with edge weight function  $w_{\tilde{G}}(\cdot)$  given for  $\{u, v\} \in E(G)$  by  $w_{\tilde{G}}(\{u, v\}) = w_{(u,v)}^{(1,1)}$  and  $w_{\tilde{G}}(\{u\}) = w_{(u)}^{(2)}$ . Then the graph  $(Z(0), Z(1), m)$  is exactly given by  $\tilde{G} \otimes K_{[s]}$ . Let  $W_{\tilde{G}}$  denote the random walk matrix of  $\tilde{G}$ , and let  $W'_{\tilde{G}}$  denote  $W_{\tilde{G}}$  with all diagonal entries zeroed out, and let  $W''_{\tilde{G}}$  denote  $W_{\tilde{G}}$  with all non-diagonal entries zeroed out, so that  $W_{\tilde{G}} = W'_{\tilde{G}} + W''_{\tilde{G}}$ . Then for any  $u \in V(G)$ ,

$$W''_{\tilde{G}}(u, u) = \frac{w_{(u)}^{(2)}}{w_{(u)}^{(2)} + \sum_{v \in N(u)} w_{(u,v)}^{(1,1)}} = \frac{1}{1 + \frac{\sum_{v \in N(u)} w_{(u,v)}^{(1,1)}}{w_{(u)}^{(2)}}} = \frac{\sum_{\ell=2}^H \frac{1}{\ell}}{\sum_{\ell=1}^H \frac{1}{\ell}},$$

where the final equality holds by Proposition 17. Therefore  $W''_{\tilde{G}} = (\sum_{\ell=2}^H \frac{1}{\ell} / \sum_{\ell=1}^H \frac{1}{\ell}) I$ . Furthermore, for any  $v \neq u$ ,

$$\begin{aligned} \left(\sum_{\ell=1}^H \frac{1}{\ell}\right) W'_{\tilde{G}}(v, u) &= \frac{w_{(u)}^{(2)} + \sum_{v' \in N(u)} w_{(u,v')}^{(1,1)}}{\sum_{v' \in N(u)} w_{(u,v')}^{(1,1)}} \cdot \frac{w_{(u,v)}^{(1,1)}}{w_{(u)}^{(2)} + \sum_{v' \in N(u)} w_{(u,v')}^{(1,1)}} \\ &= \frac{w_{(u,v)}^{(1,1)}}{\sum_{v' \in N(u)} w_{(u,v')}^{(1,1)}} \\ &= \frac{w_G(\{u, v\})}{\sum_{v' \in N(u)} w_G(\{u, v'\})} \\ &= W_G(v, u), \end{aligned}$$

where the first equality above holds by Proposition 17, and the third equality by Lemma 16. Thus in summary,

$$W_{\tilde{G}} = W'_{\tilde{G}} + W''_{\tilde{G}} = \frac{1}{\sum_{\ell=1}^H 1/\ell} W_G + \left(1 - \frac{1}{\sum_{\ell=1}^H 1/\ell}\right) I,$$

so  $W_{\tilde{G}}$  is simply a lazy instance of the random walk  $W_G$ , and in particular for all  $1 \leq i \leq n$ ,

$$\omega_i(\tilde{G}) = \frac{1}{\sum_{\ell=1}^H 1/\ell} \omega_i(G) + \left(1 - \frac{1}{\sum_{\ell=1}^H 1/\ell}\right) = \tilde{\omega}_i(G).$$

Because the eigenvalues of  $K_{[s]}$  are 1 and  $-1/(s-1)$ , it follows that

$$\omega_2(\tilde{G} \otimes K_{[s]}) = \max\{\omega_2(\tilde{G}), -\omega_n(\tilde{G})/(s-1)\},$$

as desired. ◀

**Proof of Theorem 18.** For every  $0 \leq k \leq H-2$ , each  $\sigma \in Z(k)$  by definition either lies in  $Z((j, k+1-j)_{(u,v)})$  or in  $Z((k+1)_{(u)})$  for some  $1 \leq j \leq k$  and  $\{(u, v)\} \in E(G)$ . Therefore Lemma 20 and Lemma 21 together imply that the link of every  $\sigma \in Z(k)$  has expansion  $\nu_2(Z_\sigma(0), Z_\sigma(1), m_\sigma) = \frac{k+1}{k+2}$ , so

$$\nu^{(k)}(Z) = \frac{k+1}{k+2}.$$

For the global expansion statement, letting  $\tilde{\omega}_i(G)$  be defined as in (4), then by Lemma 22,

$$\nu^{(-1)}(Z) = \nu_2(Z(0), Z(1), m) = 1 - \max\{\tilde{\omega}_2(G), -\tilde{\omega}_n(G)/(s-1)\}.$$

Now because  $n \geq 4$ ,  $H \geq 2$ , and  $s \geq 4$  by assumption, then  $\omega_2(G) \geq -1/3$  and  $\sum_{\ell=1}^H 1/\ell \geq 3/2$ , which implies that  $\tilde{\omega}_2(G) \geq 1/9$  and  $\tilde{\omega}_n(G) \geq -1/3$ , and thus  $\tilde{\omega}_2(G) \geq -\tilde{\omega}_n(G)/(s-1)$ , so

$$\nu^{(-1)}(Z) = 1 - \tilde{\omega}_2(G) = \frac{\nu_2(G)}{\sum_{\ell=1}^H 1/\ell} \geq \frac{\nu_2(G)}{1 + \log H}. \quad \blacktriangleleft$$

## 5 Discussion and future directions

Given the constructions of ordinary expanders using graph products (e.g. [19]), it seems natural to investigate simplicial complex product constructions that yield high-dimensional expanders. From this perspective, the construction  $Q$  of Liu et al. [13] is quite interesting, as it may be viewed as a form of tensor product between a graph  $G$  and the complete simplicial complex on  $s$  vertices. The principal drawback with  $Q$  lies in the exponential dependence of the spectral gap  $\Omega(\frac{\nu_2(G)}{k^{2k}})$  of the up-down walk on the dimension  $k$ . By reducing this dependence to quadratic with a spectral gap of  $\Omega(\frac{\nu_2(G)}{k^2 \log H})$ , the construction  $Z$  greatly improves the mixing time of high-dimensional up-down walks, while maintaining the product-based nature of the construction. However, the optimal spectral gap of the  $k$ th order up-down walk grows as  $\Omega(\frac{1}{k})$ , which is achieved by Ramanujan complexes and by the constructions based on coset geometries of Kaufman and Oppenheim [11] and Friedgut and Iluz [6]. It is therefore natural to ask whether the construction  $Z$  can be further optimized to reduce the quadratic dependence on  $k$  to linear. Below, we suggest a negative answer to this question, by analyzing the implications for local expansion.

The determination of the optimal local expansion for any “graph-product-based construction” was posed as an open question by Liu et al. [13], although no formal definition for a graph-product-based construction was proposed. For concreteness, fix a dimension  $H$ , and let a graph-product-based construction be one such as  $Z$  that takes as input a graph  $G$ , and outputs an  $H$ -dimensional simplicial complex with the same faces as  $Q$ , but with

an arbitrary weight function.<sup>1</sup> The following reasoning suggests that no such construction can improve upon the  $k$ -dimensional local expansion  $\nu^{(k)}(Z) = \frac{k+1}{k+2}$  of  $Z$ . For if some graph-product-based construction  $Z'$  were to satisfy  $\nu^{(k)}(Z') > \frac{k+1+\epsilon}{k+2+\epsilon}$  for some  $\epsilon > 0$  when  $G$  is chosen from a family of  $d$ -regular expanders, then inductively applying Proposition 7 would imply that  $\nu^{(j)}(Z') > \frac{j+1+\epsilon}{j+2+\epsilon}$  for all  $j \leq k$ , so that  $\nu^{(-1)}(Z') > \frac{\epsilon}{1+\epsilon}$ . But as observed in Section 4, the product-based structure ensures that the link of any face of dimension  $\geq 0$  obtains its structure from the structure of the neighborhood of a single vertex in  $G$ , while the entire 1-skeleton of  $Z'$  inherits the global structure of  $G$ . Thus if  $G$  is instead chosen from a family of  $d$ -regular graphs with sufficiently poor expansion, then the 1-skeleton of  $Z'$  will now inherit this poor expansion, so  $\nu^{(-1)}(Z') < \frac{\epsilon}{1+\epsilon}$ . But all  $d$ -regular graphs have the same local structure in the neighborhood of a vertex, so it will still hold that  $\nu^{(k)}(Z') > \frac{k+1+\epsilon}{k+2+\epsilon}$ , and therefore  $\nu^{(-1)}(Z') > \frac{\epsilon}{1+\epsilon}$ , a contradiction.

With  $k$ -dimensional local expansion  $\nu^{(k)} = \frac{k+1}{k+2}$ , the bound from Theorem 9 on the spectral gap of the  $k$ -dimensional up-down walk is at best  $\frac{1}{(k+2)(k+1)}$ . Indeed, a quadratic dependence on  $k$  in mixing time also arose in the discussion in Section 3.3, which shows how for a given edge  $\{u, v\} \in G$ , the  $k$ -dimensional up-down walk on a graph-product-based construction such as  $Z$  proceeds within the faces  $\bigcup_{j=1}^k Z((j, k+1-j)_{(u,v)})$  analogously to a random walk on a  $k$ -vertex path, which has a mixing time of  $\Omega(k^2)$ .<sup>2</sup> These observations suggest that no graph-product-based construction obtains better than a quadratic dependence on  $k$  in the spectral gap of the  $k$ -dimensional up-down walk. It is an interesting open problem to develop alternative combinatorial constructions of high-dimensional expanders that attain the optimal up-down walk spectral gap of  $\Omega(\frac{1}{k})$ .

---

## References

- 1 Vedat Levi Alev and Lap Chi Lau. Improved analysis of higher order random walks and applications. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 1198–1211, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3357713.3384317.
- 2 Michael Chapman, Nati Linial, and Yuval Peled. Expander Graphs — Both Local and Global. *Combinatorica*, 40(4):473–509, August 2020. doi:10.1007/s00493-019-4127-8.
- 3 David Conlon. Hypergraph expanders from Cayley graphs. *Israel Journal of Mathematics*, 233(1):49–65, 2019. doi:10.1007/s11856-019-1895-1.
- 4 David Conlon, Jonathan Tidor, and Yufei Zhao. Hypergraph expanders of all uniformities from Cayley graphs. *Proceedings of the London Mathematical Society*, 121(5):1311–1336, 2020. doi:10.1112/plms.12371.
- 5 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. ISSN: 0272-5428. doi:10.1109/FOCS.2017.94.
- 6 Ehud Friedgut and Yonatan Iluz. Hyper-regular graphs and high dimensional expanders. *arXiv:2010.03829 [math]*, October 2020. arXiv: 2010.03829. URL: <http://arxiv.org/abs/2010.03829>.
- 7 Mikhail Gromov. Singularities, Expanders and Topology of Maps. Part 2: from Combinatorics to Topology Via Algebraic Isoperimetry. *Geometric and Functional Analysis*, 20(2):416–526, August 2010. doi:10.1007/s00039-010-0073-8.

---

<sup>1</sup> The reasoning presented here also applies to more general constructions.

<sup>2</sup> For graph-product-based constructions other than  $Z$ , the analogous path graph may have arbitrary edge weights. We believe that such paths have mixing time  $\Omega(k^2)$ , but we have not proven such a bound.



- 8 Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006. doi:10.1090/S0273-0979-06-01126-8.
- 9 Tali Kaufman and David Mass. High Dimensional Random Walks and Colorful Expansion. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:27, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITCS.2017.4.
- 10 Tali Kaufman and Izhar Oppenheim. High Order Random Walks: Beyond Spectral Gap. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018)*, volume 116 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 47:1–47:17, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.47.
- 11 Tali Kaufman and Izhar Oppenheim. High dimensional expanders and coset geometries. *arXiv:1710.05304 [math]*, 2020. arXiv: 1710.05304 version: 3. URL: <http://arxiv.org/abs/1710.05304>.
- 12 Nathan Linial and Roy Meshulam. Homological Connectivity Of Random 2-Complexes. *Combinatorica*, 26(4):475–487, August 2006. doi:10.1007/s00493-006-0027-9.
- 13 Siqi Liu, Sidhant Mohanty, and Elizabeth Yang. High-Dimensional Expanders from Expanders. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:32, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITCS.2020.12.
- 14 Alexander Lubotzky. High Dimensional Expanders. In *Proceedings of the International Congress of Mathematicians (ICM 2018)*, pages 705–730. World Scientific, June 2018. doi:10.1142/9789813272880\_0027.
- 15 Alexander Lubotzky, Beth Samuels, and Uzi Vishne. Explicit constructions of Ramanujan complexes of type  $\tilde{A}_d$ . *European Journal of Combinatorics*, 26(6):965–993, August 2005. doi:10.1016/j.ejc.2004.06.007.
- 16 Alexander Lubotzky, Beth Samuels, and Uzi Vishne. Ramanujan complexes of type  $\tilde{A}_d$ . *Israel Journal of Mathematics*, 149(1):267–299, December 2005. doi:10.1007/BF02772543.
- 17 R. Meshulam and N. Wallach. Homological connectivity of random  $k$ -dimensional complexes: Homological Connectivity of Random Complexes. *Random Structures & Algorithms*, 34(3):408–417, May 2009. doi:10.1002/rsa.20238.
- 18 Izhar Oppenheim. Local Spectral Expansion Approach to High Dimensional Expanders Part I: Descent of Spectral Gaps. *Discrete & Computational Geometry*, 59(2):293–330, 2018. doi:10.1007/s00454-017-9948-x.
- 19 Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy Waves, the Zig-Zag Graph Product, and New Constant-Degree Expanders. *The Annals of Mathematics*, 155(1):157, January 2002. doi:10.2307/3062153.



# Improved Bounds for Coloring Locally Sparse Hypergraphs

Fotis Iliopoulos 

Institute for Advanced Study, Princeton, NJ, USA  
Princeton University, NJ, USA

---

## Abstract

We show that, for every  $k \geq 2$ , every  $k$ -uniform hypergraph of degree  $\Delta$  and girth at least 5 is efficiently  $(1 + o(1))(k - 1)(\Delta / \ln \Delta)^{1/(k-1)}$ -list colorable. As an application we obtain the currently best deterministic algorithm for list-coloring random hypergraphs of bounded average degree.

**2012 ACM Subject Classification** Theory of computation → Generating random combinatorial structures; Theory of computation → Pseudorandomness and derandomization; Theory of computation → Random search heuristics; Theory of computation → Random network models

**Keywords and phrases** hypergraph coloring, semi-random method, locally sparse, random hypergraphs

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.39

**Category** RANDOM

**Related Version** *Full Version:* <https://arxiv.org/pdf/2004.02066.pdf>

**Funding** This material is based upon work directly supported by the IAS Fund for Math and indirectly supported by the National Science Foundation Grant No. 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 National Science Foundation. This work is also supported by the Simons Collaboration on Algorithms and Geometry

**Acknowledgements** The author is grateful to Dimitris Achlioptas, Irit Dinur and anonymous reviewers for detailed comments and feedback.

## 1 Introduction

In hypergraph coloring one is given a hypergraph  $H(V, E)$  and the goal is to find an assignment of one of  $q$  colors to each vertex  $v \in V$  so that no hyperedge is monochromatic. In the more general *list-coloring* problem, a list of  $q$  allowed colors is specified for each vertex. A graph is  $q$ -list-colorable if it has a list-coloring no matter how the lists are assigned to each vertex. The *list chromatic number*,  $\chi_\ell(H)$ , is the smallest  $q$  for which  $H$  is  $q$ -list colorable.

Hypergraph coloring is a fundamental constraint satisfaction problem with several applications in computer science and combinatorics, that has been studied for over 60 years. In this paper we consider the task of coloring locally sparse hypergraphs and its connection to coloring sparse random hypergraphs.

A hypergraph is *k-uniform* if every hyperedge contains exactly  $k$  vertices. An  $i$ -cycle in a  $k$ -uniform hypergraph is a collection of  $i$  distinct hyperedges spanned by at most  $i(k - 1)$  vertices. We say that a  $k$ -uniform hypergraph has girth at least  $g$  if it contains no  $i$ -cycles for  $2 \leq i < g$ . Note that if a  $k$ -uniform hypergraph has girth at least 3 then every two of its hyperedges have at most one vertex in common.

The main contribution of this paper is to prove the following theorem.



© Fotis Iliopoulos;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 39; pp. 39:1–39:16



Leibniz International Proceedings in Informatics

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

► **Theorem 1.** *Let  $H$  be any  $k$ -uniform hypergraph,  $k \geq 2$ , of maximum degree  $\Delta$  and girth at least 5. For all  $\epsilon > 0$ , there exist a positive constant  $\Delta_{\epsilon,k}$  such that if  $\Delta \geq \Delta_{\epsilon,k}$ , then*

$$\chi_\ell(H) \leq (1 + \epsilon)(k - 1) \left( \frac{\Delta}{\ln \Delta} \right)^{\frac{1}{k-1}}. \quad (1)$$

Furthermore, if  $H$  is a hypergraph on  $n$  vertices then there exists a deterministic algorithm that constructs such a coloring in time polynomial in  $n$ .

Theorem 1 is interesting for a number of reasons. First, it generalizes a well-known result of Kim [20] for coloring graphs of degree  $\Delta$  and girth 5, and it implies the classical theorem of Ajtai, Komlós, Pintz, Spencer and Szemerédi [4] regarding the independence number of  $k$ -uniform hypergraphs of degree  $\Delta$  and girth 5. The latter is a seminal result in combinatorics, with applications in geometry and coding theory [21, 22, 24]. Second, Theorem 1 is tight up to a constant [8]. Note also that, without the girth assumption, the best possible bound [11] on the chromatic number of  $k$ -uniform hypergraphs is  $O(\Delta^{1/(k-1)})$ , i.e., it is asymptotically worse than the one of Theorem 1. For example, there exist graphs of degree  $\Delta$  whose chromatic number is exactly  $\Delta + 1$ . Third, when it applies, Theorem 1 improves upon a result of Frieze and Mubayi [14] regarding the chromatic number of simple hypergraphs, who showed (1) with an unspecified large leading constant (of order at least  $\Omega(k^4)$ ). Finally, Theorem 1 can be used to provide the currently best *deterministic* algorithm for list-coloring random  $k$ -uniform hypergraphs of bounded average degree. We discuss the connection between locally sparse hypergraphs and sparse random hypergraphs with respect to the task of coloring in the following section.

## 1.1 Application to coloring pseudo-random hypergraphs

The random  $k$ -uniform hypergraph  $H(k, n, p)$  is obtained by choosing each of the  $\binom{n}{k}$   $k$ -element subsets of a vertex set  $V$  ( $|V| = n$ ) independently with probability  $p$ . The chosen subsets are the hyperedges of the hypergraph. Note that for  $k = 2$  we have the usual definition of the random graph  $G(n, p)$ . We say that  $H(k, n, p)$  has a certain property  $A$  *almost surely* or *with high probability*, if the probability that  $H \in H(k, n, p)$  has  $A$  tends to 1 as  $n \rightarrow \infty$ .

In this paper we are interested in  $H(k, n, d/\binom{n}{k-1})$ , i.e., the family of random  $k$ -uniform hypergraphs of bounded average degree  $d$ . Specifically, we use Theorem 1 to prove the following theorem.

► **Theorem 2.** *For any constants  $\delta \in (0, 1)$ ,  $k \geq 2$ , there exists  $d_{\delta,k} > 0$  such that for every constant  $d \geq d_{\delta,k}$ , almost surely, the random hypergraph  $H(k, n, d/\binom{n}{k-1})$  can be  $(1 + \delta)(k - 1)(d/\ln d)^{1/(k-1)}$ -list-colored by a deterministic algorithm whose running time is polynomial in  $n$ .*

► **Remark 3.** Note that, for  $k, d$  constants, a very standard argument reveals that distribution  $H(k, n, d/\binom{n}{k-1})$  is essentially equivalent to  $\mathbb{H}(k, n, kdn)$ , namely the uniform distribution over  $k$ -uniform hypergraphs with  $n$  vertices and exactly  $kdn$  hyperedges. Thus, Theorem 2 extends to that model as well.

We note that previous approaches [3, 23, 31] for list-coloring random  $k$ -uniform hypergraphs of bounded average degree  $d$  are either randomized, or require significantly larger lists of colors per vertex in order to succeed. Indeed, to the best of our knowledge, current deterministic approaches require lists of size at least  $O(k^4(d/\ln d)^{1/(k-1)})$ . Moreover, it is believed that *all* efficient algorithms (including randomized ones) require lists of size at least  $(1 + o(1))(k - 1)d/\ln d)^{1/(k-1)}$ , as this bound corresponds to the so-called *shattering*

threshold [1, 7, 15] for coloring sparse random hypergraphs, which is also often referred to as the “algorithmic barrier” [1]. This threshold arises in a plethora of random constraint satisfaction problems, and it corresponds to a precise phase transition in the geometry set of solutions. In all of these problems, we are not aware of any efficient algorithm that works beyond the algorithmic barrier, despite the fact that solutions exist for constraint-densities larger than the one in which the shattering phenomenon appears. We refer the reader to [1, 33] for further details.

In order to prove Theorem 2, we show that random  $k$ -uniform hypergraphs of bounded average degree  $d$  can essentially be treated as hypergraphs of girth 5 and maximum degree  $d$  for the purposes of list-coloring, and then apply Theorem 1. In particular, we identify a pseudo-random family of hypergraphs which we call *girth-reducible*, and show that almost all  $k$ -uniform hypergraphs of bounded average degree belong in this class. Then we show that girth-reducible hypergraphs can be colored efficiently using Theorem 1.

Formally, a  $k$ -uniform hypergraph  $H$  is  $\kappa$ -degenerate if the induced subhypergraph of all subsets of its vertex set has a vertex of degree at most  $\kappa$ . The *degeneracy* of a hypergraph  $H$  is the smallest value of  $\kappa$  for which  $H$  is  $\kappa$ -degenerate. Note that it is known that  $\kappa$ -degenerate hypergraphs are  $(\kappa + 1)$ -list colorable and that the degeneracy of a hypergraph can be computed efficiently by an algorithm that repeatedly removes minimum degree vertices. Indeed, to list-color a  $\kappa$ -degenerate hypergraph we repeatedly find a vertex with (remaining) degree at most  $\kappa$ , assign to it a color that does not appear in any of its neighbors so far, and remove it from the hypergraph. Clearly, if the lists assigned to each vertex are of size at least  $\kappa + 1$  this procedure always terminates successfully.

► **Definition 4.** For  $\delta \in (0, 1)$ , we say that a  $k$ -uniform hypergraph  $H(V, E)$  of average degree  $d$  is  $\delta$ -girth-reducible if its vertex set can be partitioned in two sets,  $U$  and  $V \setminus U$ , such that:

- (a)  $U$  contains all cycles of length at most 4, and all vertices of degree larger than  $(1 + \delta)d$ ;
- (b) subhypergraph  $H[U]$  is  $\left(\frac{d}{\ln d}\right)^{\frac{1}{k-1}}$ -degenerate;
- (c) every vertex in  $V \setminus U$  has at most  $\delta \left(\frac{d}{\ln d}\right)^{\frac{1}{k-1}}$  neighbors in  $U$ .

In words, a hypergraph is  $\delta$ -girth-reducible if its vertex set can be seen as the union of two parts: A “low-degeneracy” part, which contains all vertices of degree more than  $(1 + \delta)d$  and all cycles of lengths at most 4, and a “high-girth” part, which induces a hypergraph of maximum degree at most  $(1 + \delta)d$  and girth 5. Moreover, each vertex in the “high-girth” part has only a few neighbors in the “low-degeneracy” part.

Note that given a  $\delta$ -girth-reducible hypergraph we can efficiently find the promised partition  $(U, V \setminus U)$  as follows. We start with  $U := U_0$ , where  $U_0$  is the set of vertices that either have degree at least  $(1 + \delta)d$ , or they are contained in a cycle of length at most 4. Let  $\partial U$  denote the vertices in  $V \setminus U$  that violate property (c). While  $\partial U \neq \emptyset$ , update  $U$  as  $U := U \cup \partial U$ . The correctness of the process lies in the fact that in each step we add to the current  $U$  a set of vertices that must be in the low-degeneracy part of the hypergraph. Observe also that this process allows us to efficiently check whether a hypergraph is  $\delta$ -girth-reducible.

We prove the following theorem regarding the list-chromatic number of girth-reducible hypergraphs.

► **Theorem 5.** For any constants  $\delta \in (0, 1)$  and  $k \geq 2$ , there exists  $d_{\delta,k} > 0$  such that if  $H$  is a  $\delta$ -girth-reducible,  $k$ -uniform hypergraph of average degree  $d \geq d_{\delta,k}$ , then

$$\chi_\ell(H) \leq (1 + \epsilon)(k - 1) \left(\frac{d}{\ln d}\right)^{\frac{1}{k-1}},$$

where  $\epsilon = 4\delta = O(\delta)$ . Furthermore, if  $H$  is a hypergraph on  $n$  vertices then there exists a deterministic algorithm that constructs such a coloring in time polynomial in  $n$ .

**Proof of Theorem 5.** Let  $\epsilon = 4\delta$ . Given lists of colors of size  $(1 + \epsilon)(k - 1) \left(\frac{d}{\ln d}\right)^{\frac{1}{k-1}}$  for each vertex of  $H$ , we first color the vertices of  $U$  using the greedy algorithm which exploits the low degeneracy of  $H[U]$ . Now each vertex in  $V - U$  has at most  $\delta \left(\frac{d}{\ln d}\right)^{\frac{1}{k-1}}$  forbidden colors in its list as it has at most that many neighbors in  $U$ . We delete these colors from the list. Observe that if we manage to properly color the induced subgraph  $H[V \setminus U]$  using colors from the updated lists, then we are done since every hyperedge with vertices both in  $U$  and  $V \setminus U$  will be automatically “satisfied”, i.e., it cannot be monochromatic. Notice now that the updated list of each vertex still contains at least  $(1 + 3\delta)(k - 1) \left(\frac{d}{\ln d}\right)^{\frac{1}{k-1}}$  colors, for sufficiently large  $d$ . Since the induced subgraph  $H[V \setminus U]$  is of girth at least 5 and of maximum degree at most  $(1 + \delta)d$ , it is efficiently  $(1 + \delta)(k - 1) \left(\frac{(1 + \delta)d}{\ln((1 + \delta)d)}\right)^{\frac{1}{k-1}}$ -list-colorable for sufficiently large  $d$  per Theorem 1. This concludes the proof since  $(1 + \delta)(1 + \delta)^{\frac{1}{k-1}} < (1 + 3\delta)$ . ◀

Moreover, we show that girth-reducibility is a pseudo-random property which is admitted by almost all sparse  $k$ -uniform hypergraphs.

► **Theorem 6.** *For any constants  $\delta \in (0, 1)$ ,  $k \geq 2$ , there exists  $d_{\delta,k} > 0$  such that for every constant  $d \geq d_{\delta,k}$ , almost surely, the random hypergraph  $H(k, n, d/\binom{n}{k-1})$  is  $\delta$ -girth-reducible.*

Theorem 6 follows by simple, although somewhat technical, considerations on properties of sparse random hypergraphs, which are mainly inspired by the results of Alon, Krivelevich and Sudakov [6] and Łuczak [25]. Observe that combining Theorem 6 with Theorem 5 immediately implies Theorem 2.

Overall, the task of coloring locally sparse hypergraphs is inherently related to the average-case complexity of coloring. In particular, in this section we showed that Theorem 1 implies a *robust* algorithm for hypergraph coloring, namely a deterministic procedure that applies to worst-case  $k$ -uniform hypergraphs, while at the same time using a number of colors that is only a  $(k - 1)$ -factor away from the algorithmic barrier for random instances (matching it for  $k = 2$ ). We remark that this application is inspired by recent results that study the connection between local sparsity and efficient randomized algorithms for coloring sparse regular random graphs [26, 2, 10].

## 1.2 Technical overview

The intuition behind the proof of Theorem 1 comes from the following observation, which we explain in terms of graph coloring for simplicity. Let  $G$  be a triangle-free graph of degree  $\Delta$ , and assume that each of its vertices is assigned an arbitrary list of  $q$  colors. Fix a vertex  $v$  of  $G$ , and consider the random experiment in which the neighborhood of  $v$  is properly list-colored randomly. Since  $G$  contains no triangles, this amounts to assigning to each neighbor of  $v$  a color from its list randomly and independently. Assuming that  $q \geq q^* := (1 + \epsilon)\Delta/\ln \Delta$ , the expected number of *available* colors for  $v$ , i.e., the colors from the list of  $v$  that do not appear in any of its neighbors, is at least  $q(1 - 1/q)^\Delta = \omega(\Delta^{\epsilon/2})$ . In fact, a simple concentration argument reveals that the number of available colors for  $v$  in the end of this experiment is at least  $\Delta^{\epsilon/2}$  with probability that goes to 1 as  $\Delta$  grows. To put it differently, as long as  $q \geq q^*$ , the vast majority of valid ways to list-color the neighborhood of  $v$  “leaves enough room” to color  $v$  without creating any monochromatic edges.

A completely analogous observation regarding the ways to properly color the neighborhood of a vertex can be made for  $k$ -uniform hypergraphs. In order to exploit it we employ the so-called *semi-random method*, which is the main tool behind some of the strongest graph coloring results, e.g., [16, 17, 18, 19, 27, 32], including the one of Kim [20]. The idea is to

gradually color the hypergraph in iterations until we reach a point where we can finish the coloring with a simple, e.g., greedy, algorithm. In its most basic form, each iteration consists of the following simple procedure (using graph vertex coloring as a canonical example): Assign to each vertex a color chosen uniformly at random; then uncolor any vertex that receives the same color as one of its neighbors. Using the Lovász Local Lemma [11] and concentration inequalities, one typically shows that, with positive probability, the resulting partial coloring has useful properties that allow for the continuation of the argument in the next iteration. (In fact, using the Moser-Tardos algorithm [29] this approach yields efficient, and often times deterministic [9], algorithms.) Specifically, one keeps track of certain parameters of the current partial coloring and makes sure that, in each iteration, these parameters evolve almost as if the coloring was totally random. For example, recalling the heuristic experiment of the previous paragraph, one of the parameters we would like to keep track of in our case is a lower bound on the number of available colors of each vertex in the hypergraph: If this parameter evolves “randomly” throughout the process, then the vertices that remain uncolored in the end are guaranteed to have a non-trivial number of available colors.

Applications of the semi-random method tend to be technically intense and this is even more so in our case, where we have to deal with constraints of large arity. Large constraints introduce several difficulties, but the most important one is that our algorithm has to control many parameters that interact with each other. Roughly, in order to guarantee the properties that allow for the continuation of the argument in the next iteration, for each uncolored vertex  $v$ , each color  $c$  in the list of  $v$ , and each integer  $r \in [k - 1]$ , we should keep track of a lower bound on the number of adjacent to  $v$  hyperedges that have  $r$  uncolored vertices and  $k - 1 - r$  vertices colored  $c$ . Clearly, these parameters are not independent of each other throughout the process, and so the main challenge is to design and analyze a coloring procedure in which all of them, simultaneously, evolve essentially randomly.

### 1.3 Organization of the paper

The paper is organized as follows. In Section 2 we present the necessary background. In Section 3 we present the algorithm and state the key lemmas for the proof of Theorem 1. (The proofs of these lemmas can be found in the full version of our paper). In Section 4 we prove Theorem 6.

## 2 Background and preliminaries

In this section we give some background on the technical tools that we will use in our proofs.

### 2.1 The Lovász Local Lemma

We will find useful the so-called *lopsided* version of the Lovász Local Lemma [11, 12].

► **Theorem 7.** *Consider a set  $\mathcal{B} = \{B_1, B_2, \dots, B_m\}$  of (bad) events. For each  $B \in \mathcal{B}$ , let  $D(B) \subseteq \mathcal{B} \setminus \{B\}$  be such that  $\Pr[B \mid \bigcap_{C \in S} \overline{C}] \leq \Pr[B]$  for every  $S \subseteq \mathcal{B} \setminus (D(B) \cup \{B\})$ . If there is a function  $x : \mathcal{B} \rightarrow (0, 1)$  satisfying*

$$\Pr[B] \leq x(B) \prod_{C \in D(B)} (1 - x(C)) \quad \text{for all } B \in \mathcal{B} \quad , \quad (2)$$

*then the probability that none of the events in  $\mathcal{B}$  occurs is at least  $\prod_{B \in \mathcal{B}} (1 - x(B)) > 0$ .*

## 39:6 Coloring Locally Sparse Hypergraphs

In particular, we will need the following two corollaries of Theorem 7. For their proofs, the reader is referred to Chapter 19 in [28].

► **Corollary 8.** Consider a set  $\mathcal{B} = \{B_1, \dots, B_m\}$  of (bad) events. For each  $B \in \mathcal{B}$ , let  $D(B) \subseteq \mathcal{B} \setminus \{B\}$  be such that  $\Pr[B \mid \bigcap_{C \in S} \overline{C}] \leq \Pr[B]$  for every  $S \subseteq \mathcal{B} \setminus (D(B) \cup \{B\})$ . If for every  $B \in \mathcal{B}$ :

(a)  $\Pr[B] \leq \frac{1}{4}$ ;

(b)  $\sum_{C \in D(B)} \Pr[C] \leq \frac{1}{4}$ ,

then the probability that none of the events in  $\mathcal{B}$  occurs is strictly positive.

► **Corollary 9.** Consider a set  $\mathcal{B} = \{B_1, B_2, \dots, B_m\}$  of (bad) events such that for each  $B \in \mathcal{B}$ :

(a)  $\Pr[B] \leq p < 1$ ;

(b)  $B$  is mutually independent of a set of all but at most  $\Delta$  of the other events.

If  $4p\Delta \leq 1$  then with positive probability, none of the events in  $\mathcal{B}$  occur.

## 2.2 Talagrand's inequality

We will also need the following version of Talagrand's inequality [30] whose proof can be found in [28].

► **Theorem 10.** Let  $X$  be a non-negative random variable, not identically 0, which is determined by  $n$  independent trials  $T_1, \dots, T_n$ , and satisfying the following for some  $c, r > 0$ :

1. changing the outcome of any trial can affect  $X$  by at most  $c$ , and
2. for any  $s$ , if  $X \geq s$  then there is a set of at most  $w$  trials whose outcomes certify that  $X \geq s$ ,

then for any  $0 \leq t \leq \mathbb{E}[X]$ ,

$$\Pr[|X - \mathbb{E}[X]| > t + 60c\sqrt{w\mathbb{E}[X]}] \leq 4e^{-\frac{t^2}{8c^2w\mathbb{E}[X]}}.$$

## 3 List-coloring high-girth hypergraphs

In this section we describe the algorithm of Theorem 1 and state the key lemmas behind its analysis. As we already explained, our approach is based on the semi-random method. For an excellent exposition both of the method and Kim's result the reader is referred to [28].

We assume without loss of generality that  $\epsilon < \frac{1}{10}$ . Also, it will be convenient to define the parameter  $\delta := (1 + \epsilon)(k - 1) - 1$ , so that the list of each vertex initially has at least  $(1 + \delta)\left(\frac{\Delta}{\ln \Delta}\right)^{\frac{1}{k-1}}$  colors.

We analyze each iteration of our procedure using a probability distribution over the set of (possibly improper) colorings of the uncolored vertices of  $H$  where, additionally, each vertex is either activated or deactivated. We call a pair of coloring and activation bits assignments for the uncolored vertices of hypergraph  $H$  a *state*.

Let  $V_i$  denote the set of uncolored vertices in the beginning of the  $i$ -th iteration. (Initially, all vertices are uncolored.) For each  $v \in V_i$  we denote by  $L_v = L_v(i)$  the list of colors of  $v$  in the beginning of the  $i$ -th iteration. Further, we say that a color  $c \in L_v$  is *available* for  $v$  in a state  $\sigma$  if assigning  $c$  to  $v$  does not cause any hyperedge whose initially uncolored vertices are all activated in  $\sigma$  to be monochromatic.

For each vertex  $v$ , color  $c \in L_v$  and iteration  $i$ , we define a few quantities of interest that our algorithm will attempt to control. Let  $\ell_i(v)$  be the size of  $L_v$ . Further, for each  $r \in [k]$ , let  $D_{i,r}(v, c)$  denote the set of hyperedges  $h$  that contain  $v$  and (i) exactly  $r$  vertices  $\{u_1, \dots, u_r\} \subseteq h \setminus \{v\}$  are uncolored and  $c \in L_{u_j}$  for every  $j \in [r]$ ; (ii) the rest  $k - 1 - r$  vertices other than  $v$  are colored  $c$ . We define  $t_{i,r}(v, c) = |D_{i,r}(v, c)|$ .



As it is common in the applications of the semi-random method, we will not attempt to keep track of the values of  $\ell_i(v)$  and  $t_{i,r}(v, c)$ ,  $r \in [k - 1]$ , for every vertex  $v$  and color  $c$  but, rather, we will focus on their extreme values. In particular, we will define appropriate  $L_i, T_{i,r}$  such that we can show that, for each  $i$ , the following property holds at the beginning of iteration  $i$ :

*Property  $P(i)$ :* For each vertex  $v \in V_i$ , color  $c \in L_v$  and  $r \in [k - 1]$ ,

$$\begin{aligned}\ell_i(v) &\geq L_i, \\ t_{i,r}(v, c) &\leq T_{i,r}.\end{aligned}$$

As a matter of fact, it would be helpful for our analysis (though not necessary) if the inequalities defined in  $P(i)$  were actually tight. Given that  $P(i)$  holds, we can always enforce this stronger property in a straightforward way as follows. First, for each vertex  $v$  such that  $\ell_i(v) > L_i$  we choose arbitrarily  $\ell_i(v) - L_i$  colors from its list and remove them. Then, for each vertex  $v$  and color  $c \in L_i$  such that  $t_{i,r}(v, c) < T_{i,r}$  we add to the hypergraph  $T_{i,r} - t_{i,r}(v, c)$  new hyperedges of size  $r + 1$  that contain  $v$  and  $r$  new “dummy” vertices. (As it will be evident from the proof, we can always assume that  $L_i, T_{i,r}$  are integers, since our analysis is robust to replacing  $L_i, T_{i,r}$  with  $\lfloor L_i \rfloor$  and  $T_{i,r}$  with  $\lceil T_{i,r} \rceil$ .) We assign each dummy vertex a list of  $L_i$  colors:  $L_i - 1$  of them are new and do not appear in the list of any other vertex, and the last one is  $c$ .

► **Remark 11.** Dummy vertices are only useful for the purposes of our analysis and can be removed at the end of the iteration. Indeed, one could use the technique of “equalizing coin flips” instead. For more details see e.g., [28].

Overall, without loss of generality, at each iteration  $i$  our goal will be to guarantee that  $P(i + 1)$  holds assuming  $Q(i)$ .

*Property  $Q(i)$ :* For each vertex  $v \in V_i$ , color  $c \in L_v$  and  $r \in [k - 1]$ ,

$$\begin{aligned}\ell_i(v) &= L_i, \\ t_{i,r}(v, c) &= T_{i,r}.\end{aligned}$$

**An iteration.** For the  $i$ -th iteration we will apply the Local Lemma with respect to the probability distribution induced by assigning to each vertex  $v \in V_i$  a color chosen uniformly at random from  $L_v$  and activating  $v$  with probability  $\alpha = \frac{K}{\ln \Delta}$ , where  $K = (100k^{3k})^{-1}$ . That is, we will apply the Moser-Tardos algorithm in a configuration space consisting of  $2|V_i|$  variables corresponding to the color and activation bit of each variable in  $V_i$ . (We will define the family of bad events for each iteration shortly.)

When the execution of the Moser-Tardos algorithm terminates, we will uncolor some of the vertices in  $V_i$ , to get a new partial coloring. In particular, the partial coloring of the hypergraph, set  $V_{i+1}$ , and the lists of colors for each uncolored vertex in the beginning of iteration  $i + 1$  are induced as follows. Let  $\sigma$  be the output state of the application of the Moser-Tardos algorithm in the  $i$ -th iteration. The list of each vertex  $v$ ,  $L_v(i + 1)$ , is induced from  $L_v(i)$  by removing every non-available color  $c \in L_v(i)$  for  $v$  in  $\sigma$ . We obtain the partial coloring  $\phi$  for the hypergraph and set  $V_{i+1}$  for the beginning of iteration  $i + 1$  by removing the color from every vertex  $v \in V_i$  which is either deactivated or is assigned a non-available for it color in  $\sigma$ .

Overall, the  $i$ -th iteration of our algorithm can be described at a high-level as follows:

1. Apply the Moser-Tardos algorithm to the probability space induced by assigning to each vertex  $v \in V_i$  a color chosen uniformly at random from  $L_v(i)$ , and activating  $v$  with probability  $\alpha$ .

## 39:8 Coloring Locally Sparse Hypergraphs

2. Let  $\sigma$  be the output state of the Moser-Tardos algorithm.
3. For each vertex  $v \in V_i$ , remove any non-available color  $c \in L_v(i)$  in  $\sigma$  to get a list  $L_v(i+1)$ .
4. Uncolor every vertex  $v \in V_i$  that has either received a non-available color or is deactivated in  $\sigma$ , to get a new partial coloring  $\phi$ .

**Controlling the parameters of interest.** Next we describe the recursive definitions for  $L_i$  and  $T_{i,r}$  which, as we already explained, will determine the behavior of the parameters  $\ell_i(v)$  and  $t_{i,r}(v, c)$ , respectively.

Initially,  $L_1 = (1 + \delta) \left(\frac{\Delta}{\ln \Delta}\right)^{\frac{1}{k-1}}$ ,  $T_{1,k-1} = \Delta$  and  $T_{1,r} = 0$  for every  $r \in [k-2]$ . Letting

$$\text{Keep}_i = \prod_{r=1}^{k-1} \left(1 - \left(\frac{\alpha}{L_i}\right)^r\right)^{T_{i,r}}, \quad (3)$$

we define

$$L_{i+1} = L_i \cdot \text{Keep}_i - L_i^{2/3}, \quad (4)$$

$$\begin{aligned} T_{i+1,r} &= \sum_{j=r}^{k-1} \binom{j}{r} T_{i,j} \cdot (\text{Keep}_i (1 - \alpha \text{Keep}_i))^r \left(\frac{\alpha \text{Keep}_i}{L_i}\right)^{j-r} \\ &\quad + 3k^r \alpha^{-r+1} L_i^r \sum_{\ell=1}^{k-1} \frac{T_{i,\ell}}{L_i^{2\ell} (\ln \Delta)^{2\ell}} + \left(\sum_{j=r}^{k-1} \binom{j}{r} \alpha^{j-r} \frac{T_{i,j}}{L_i^{j-r}}\right)^{2/3}. \end{aligned} \quad (5)$$

To get some intuition for the recursive definitions (4), (5), observe that  $\text{Keep}_i$  is the probability that a color  $c \in L_v(i)$  is present in  $L_v(i+1)$  as well. Note further that this implies that the expected value of  $\ell_{i+1}(v, c)$  is  $L_i \cdot \text{Keep}_i$ , a fact which motivates (4). Calculations of similar flavor for  $\mathbb{E}[t_{i+1,r}(v, c)]$  motivate (5).

**The key lemmas.** We are almost ready to state the main lemmas that will guarantee that our procedure eventually reaches a partial list-coloring of  $H$  with favorable properties that will allow us to extend it to a full list-coloring. Before doing so, we need to settle a subtle issue that has to do with the fact that  $t_{i+1,r}(v, c)$  is not sufficiently concentrated around its expectation. To see this, notice for example that  $t_{i+1,1}(v, c)$  drops to zero if  $v$  is assigned  $c$ . (Similarly, for  $r \in \{2, \dots, k-1\}$ , if  $v$  is assigned  $c$  then  $t_{i+1,r}(v, c)$  can be affected by a large amount.) To deal with this problem we will focus instead on variable  $t'_{i+1,r}(v, c)$ , i.e., the number of hyperedges  $h$  that contain  $v$  and (i) exactly  $k-r-1$  vertices of  $h \setminus \{v\}$  are colored  $c$  in the end of iteration  $i$ ; (ii) the rest  $r$  vertices of  $h \setminus \{v\}$  did not retain their color during iteration  $i$  and, further,  $c$  would be available for them if we ignored the color assigned to  $v$ . Observe that if  $c$  is not assigned to  $v$  then  $t_{i+1,r}(v, c) = t'_{i+1,r}(v, c)$  and  $t'_{i+1,r}(v, c) \geq t_{i+1,r}(v, c)$  otherwise.

The first lemma that we prove estimates the expected value of the parameters at the end of the  $i$ -th iteration. Its proof can be found in the full version of our paper.

► **Lemma 12.** *Let  $S_i = \sum_{\ell=1}^{k-1} \frac{T_{i,\ell}}{L_i^{2\ell} (\ln \Delta)^{2\ell}}$  and  $Y_{i,r} = \sum_{j=r}^{k-1} \frac{T_{i,j}}{L_i^j}$ . If  $Q(i)$  holds and for all  $1 < j < i$ ,  $r \in [k-1]$ ,  $L_j \geq (\ln \Delta)^{20(k-1)}$ ,  $T_{i,r} \geq (\ln \Delta)^{20(k-1)}$ , then, for every vertex  $v \in V_{i+1}$  and color  $c \in L_v$ :*

(a)  $\mathbb{E}[\ell_{i+1}(v)] = \ell_i(v) \cdot \text{Keep}_i$ ;

(b)

$$\begin{aligned} \mathbb{E}[t'_{i+1,r}(v, c)] &\leq \sum_{j=r}^{k-1} \binom{j}{r} T_{i,j}(v, c) \cdot (\text{Keep}_i (1 - \alpha \text{Keep}_i))^r \left(\frac{\alpha \text{Keep}_i}{L_i}\right)^{j-r} \\ &\quad + 3k^r \alpha^{-r+1} L_i^r S_i + O(Y_i). \end{aligned}$$

The next step is to prove strong concentration around the mean for our random variables per the following lemma. Its proof can be found in the full version of our paper.

► **Lemma 13.** *If  $Q(i)$  holds and  $L_i, T_{i,r} \geq (\ln \Delta)^{20(k-1)}$ ,  $r \in [k-1]$ , then for every vertex  $v \in V_{i+1}$  and color  $c \in L_v$ ,*

- (a)  $\Pr \left[ |\ell_{i+1}(v) - \mathbb{E}[\ell_{i+1}(v)]| < L_i^{2/3} \right] < \Delta^{-\ln \Delta}$ ;
- (b)  $\Pr \left[ t'_{i+1,r}(v, c) - \mathbb{E}[t'_{i+1,r}(v, c)] > \frac{1}{2} \left( \sum_{j=r}^{k-1} \binom{j}{r} \alpha^{j-r} \frac{T_{i,j}}{L_i^{j-r}} \right)^{2/3} \right] < \Delta^{-\ln \Delta}$ .

Armed with Lemmas 12, 13, a straightforward application of the symmetric Local Lemma, i.e., Corollary 9, reveals the following.

► **Lemma 14.** *With positive probability,  $P(i)$  holds for every  $i$  such that for all  $1 < j < i$ :  $L_j, T_{j,r} \geq (\ln \Delta)^{20(k-1)}$  and  $T_{j,k-1} \geq \frac{1}{10k^2} L_j^{k-1}$ .*

The proof of Lemma 14 can be found in the full version of our paper.

In analyzing the recursive equations (4), (5), it would be helpful if we could ignore the “error terms”. The next lemma shows that this is indeed possible. Its proof can be found in the full version of our paper.

► **Lemma 15.** *Define  $L'_1 = (1 + \delta) \left( \frac{\Delta}{\ln \Delta} \right)^{\frac{1}{k-1}}$ ,  $T'_{1,k-1} = \Delta$ ,  $T'_{1,r} = 0$  for  $r \in [k-2]$ , and recursively define*

$$\begin{aligned} L'_{i+1} &= L'_i \cdot \text{Keep}_i, \\ T'_{i+1,r} &= \sum_{j=r}^{k-1} \binom{j}{r} \left( T'_{i,j} \cdot (\text{Keep}_i \cdot (1 - \alpha \text{Keep}_i))^r \left( \frac{\alpha \text{Keep}_i}{L'_i} \right)^{j-r} \right) \\ &\quad + 3k^r \alpha^{-r+1} L_i^r \sum_{\ell=1}^{k-1} \frac{T_{i,\ell}}{L_i^{2\ell} (\ln \Delta)^{2\ell}}. \end{aligned}$$

*If for all  $1 < j < i$ ,  $L_j \geq (\ln \Delta)^{20(k-1)}$ ,  $T_{j,r} \geq (\ln \Delta)^{20(k-1)}$  for every  $r \in [k-1]$ , and  $T_{j,k-1} \geq \frac{L_j^{k-1}}{10k^2}$ , then*

- (a)  $|L_i - L'_i| \leq (L'_i)^{\frac{5}{6}}$ ;
- (b)  $|T_{i,r} - T'_{i,r}| \leq (T'_{i,r})^{\frac{100r}{100r+1}}$ .

► **Remark 16.** Note that  $\text{Keep}_i$  in Lemma 15 is still defined in terms of  $L_i, T_{i,r}$  and not  $L'_i, T'_{i,r}$ . Note also that in the definition of  $T'_{i+1,r}$ , the second summand is a function of  $T_{i,\ell}, L_i, \ell \in [r-1]$ , and not  $T'_{i,\ell}, L'_i$ .

Using Lemma 15 we are able to prove the following in the full version of our paper.

► **Lemma 17.** *There exists  $i^* = O(\ln \Delta \ln \ln \Delta)$  such that*

- (a) *For all  $1 < i \leq i^*$ ,  $T_{i,r} > (\ln \Delta)^{20(k-1)}$ ,  $L_i \geq \Delta^{\frac{\epsilon/3}{(k-1)(1+\epsilon/2)}}$ , and  $T_{i,k-1} \geq \frac{1}{10k^2} L_i^{k-1}$ ;*
- (b)  *$T_{i^*+1,r} \leq \frac{1}{10k^2} L_{i^*+1}^r$ , for every  $r \in [k-1]$  and  $L_{i^*+1} \geq \Delta^{\frac{\epsilon/3}{(k-1)(1+\epsilon/2)}}$ .*

Lemmas 14, 17 and 18 imply Theorem 1.

► **Lemma 18.** *Let  $\sigma$  be the state promised by Lemma 17. Given  $\sigma$ , we can find a full list-coloring of  $H$  in polynomial time in the number of vertices of  $H$ .*

**Proof of Theorem 1.** We carry out  $i^*$  iterations of our procedure. If  $P(i)$  fails to hold for any iteration  $i$ , then we halt. By Lemmas 14 and 17,  $P(i)$  (and, therefore,  $Q(i)$ ) holds with positive probability for each iteration and so it is possible to perform  $i^*$  iterations. Further, the fact that our LLL application is within the scope of the so-called *variable setting* [29] implies that the deterministic version of the Moser-Tardos algorithm [29, 9] applies and, thus, we can perform  $i^*$  iterations in polynomial time.

After  $i^*$  iterations we can apply the algorithm of Lemma 18 and complete the list-coloring of the input hypergraph.  $\blacktriangleleft$

### 3.1 Proof of Lemma 18

Let  $\mathcal{U}_\sigma$  denote the set of uncolored vertices in  $\sigma$ , and  $\mathcal{U}_\sigma(h)$  the subset of  $\mathcal{U}_\sigma$  that belongs to a hyperedge  $h$ . Our goal is to color the vertices in  $\mathcal{U}_\sigma$  to get a full list-coloring.

Towards that end, let  $L_v = L_v(\sigma)$  denote the list of colors for  $v$  at  $\sigma$ , and  $D_r(v, c) := D_{i^*+1, r}(v, c)$  the set of hyperedges (of size  $t_{i^*+1, r}(v, c)$ ) with  $r$  uncolored vertices in  $\sigma$  whose vertices “compete” for  $c$  with  $v$ , and recall the conclusion of Lemma 17. Let  $\mu$  be the probability distribution induced by giving each vertex  $v \in \mathcal{U}_\sigma$  a color from  $L_v$  uniformly at random. For every hyperedge  $h$  and color  $c \in \bigcap_{u \in h} L_u$  we define  $A_{h, c}$  to be the event that all vertices of  $h$  are colored  $c$ . Let  $\mathcal{A}$  be the family of these (bad) events, and observe that for every  $A_{h, c} \in \mathcal{A}$ :

$$\mu(A_{h, c}) \leq \frac{1}{\prod_{v \in \mathcal{U}_\sigma(h)} |L_v(\sigma)|} < \frac{1}{4}$$

for large enough  $\Delta$ , since  $L_{i^*+1} = L_{i^*+1}(\Delta) \xrightarrow{\Delta \rightarrow \infty} \infty$ .

Moreover, let  $I(A_{h, c})$  denote the set of all bad events  $A_{h', c'}$ , where  $h' \neq h$ , such that either  $\mathcal{U}_\sigma(h) \cap \mathcal{U}_\sigma(h') = \emptyset$ , or  $c'$  is not in the list of colors of the (necessarily unique) uncolored vertex that  $h$  and  $h'$  share. Notice that conditioning on any the non-occurrence of any set  $S \subseteq I(A_{h, c})$  does not increase the probability of  $A_{h, c}$ .

Let  $D(A_{h, c}) := \mathcal{A} \setminus I(A_{h, c})$ . Lemma 18 follows from Corollary 8 (and can be made constructive using the deterministic version of the Moser-Tardos algorithm [29, 9]) as, for every  $A_{h, c} \in \mathcal{A}$ :

$$\begin{aligned} \sum_{A \in D(A_{h, c})} \mu(A) &\leq \sum_{v \in \mathcal{U}_\sigma(h)} \sum_{c' \in L_v} \sum_{r=1}^{k-1} \sum_{h' \in D_r(v, c')} \mu(A_{h', c'}) \\ &= \sum_{v \in \mathcal{U}_\sigma(h)} \sum_{c' \in L_v} \sum_{i=1}^{k-1} \sum_{h' \in D_r(v, c')} \frac{1}{\prod_{u \in \mathcal{U}_\sigma(h')} |L_u|} \\ &\leq \max_{v \in \mathcal{U}_\sigma(h)} \frac{k}{|L_v|} \sum_{c' \in L_v} \sum_{r=1}^{k-1} \frac{|D_r(v, c')|}{L_{i^*+1}^r} \end{aligned} \quad (6)$$

$$\leq \frac{k}{10k^2} \max_{v \in \mathcal{U}_\sigma(h)} \frac{L_{i^*+1}^r \cdot |L_v|}{|L_v| \cdot L_{i^*+1}^r} \quad (7)$$

$$\leq \frac{1}{10} < \frac{1}{4}, \quad (8)$$

for large enough  $\Delta$ , concluding the proof. Note that in (6) we used the facts that every hyperedge has at most  $k$  vertices and  $L_{i^*+1} \geq \Delta^{\frac{\epsilon/3}{(k-1)(1+\epsilon/2)}}$ , and in (7) we used the fact that  $|D_r(v, c')| \leq T_{i^*+1}^r \leq \frac{1}{10k^2} L_{i^*+1}^r$ .

**4 A sufficient pseudo-random property for coloring**

In this section we present the proof of Theorem 6. To do so, we build on ideas of Alon, Krivelevich and Sudakov [6] and show that the random hypergraph  $H(k, n, d/\binom{n}{k-1})$  almost surely admits a few useful features.

The first lemma we prove states that all subgraphs of  $H(k, n, d/\binom{n}{k-1})$  with not too many vertices are sparse and, therefore, of small degeneracy.

► **Lemma 19.** *For every constant  $k \geq 2$ , there exists  $d_k > 0$  such that for any constant  $d \geq d_k$ , the random hypergraph  $H(k, n, d/\binom{n}{k-1})$  has the following property almost surely: Every  $s \leq nd^{-\frac{1}{k-1}}$  vertices of  $H$  span fewer than  $s \left(\frac{d}{(\ln d)^2}\right)^{\frac{1}{k-1}}$  hyperedges. Therefore, any subhypergraph of  $H$  induced by a subset  $V_0 \subset V$  of size  $|V_0| \leq nd^{-\frac{1}{k-1}}$ , is  $k \left(\frac{d}{(\ln d)^2}\right)^{\frac{1}{k-1}}$ -degenerate.*

**Proof.** Letting  $r = \left(\frac{d}{(\ln d)^2}\right)^{\frac{1}{k-1}}$ , we see that the probability that there exists a subset  $V_0 \subset V$  which violates the statement of the lemma is at most

$$\begin{aligned} \sum_{i=r^{\frac{1}{k-1}}}^{nd^{-\frac{1}{k-1}}} \binom{n}{i} \binom{\binom{i}{k}}{\binom{i}{r}} \left(\frac{d}{\binom{n}{k-1}}\right)^{ri} &\leq \sum_{i=r^{\frac{1}{k-1}}}^{nd^{-\frac{1}{k-1}}} \left[ \frac{en}{i} \left(\frac{ei^{k-1}}{r}\right)^r \left(\frac{d}{\binom{n}{k-1}}\right)^{r} \right]^i \tag{9} \\ &\leq \sum_{i=r^{\frac{1}{k-1}}}^{nd^{-\frac{1}{k-1}}} \left[ e^{1+\frac{1}{k-1}} (k-1) \left(\frac{d}{r}\right)^{\frac{1}{k-1}} \left(\frac{ei^{k-1}d}{r\binom{n}{k-1}}\right)^{r-\frac{1}{k-1}} \right]^i \\ &= o(1), \end{aligned}$$

for sufficiently large  $d$ . Note that in the lefthand side of (9) we used the fact that any subset of vertices of size  $s < r^{\frac{1}{k-1}}$  cannot violate the assertion of the lemma, since it can span at most  $s^k < rs$  hyperedges. In deriving the final inequality we used that for any pair of integers  $\alpha, \beta$ , we have that  $\binom{\alpha}{\beta} \geq \left(\frac{\alpha}{\beta}\right)^\beta$ . ◀

Next we show that, as far as the number of vertices of  $H(k, n, d/\binom{n}{k-1})$  that have a constant degree  $c$  is concerned, the degree of each vertex of  $H$  is essentially a Poisson random variable with mean  $d$ .

► **Lemma 20.** *For constants  $c \geq 1$ ,  $k \geq 2$  and  $d$ , let  $X_c$  denote the number of vertices of degree  $c$  in  $H(k, n, d/\binom{n}{k-1})$ . Then, for  $c = O(1)$ , with high probability,*

$$X_c \leq \frac{d^c e^{-d}}{c!} n \left( 1 + O\left(\frac{\log n}{\sqrt{n}}\right) \right).$$

### 39:12 Coloring Locally Sparse Hypergraphs

**Proof.** The lemma follows from standard ideas for estimation of the degree distribution of random graphs (see for example the proof of Theorem 3.3 in [13] for the case  $k = 2$ ). In particular, assume that the vertices of  $H(k, n, d/\binom{n}{k-1})$  are labeled  $1, 2, \dots, n$ . Then,

$$\begin{aligned} \mathbb{E}[X_c] &= n \Pr[\deg(1) = c] \\ &= n \binom{\binom{n-1}{k-1}}{c} \left( \frac{d}{\binom{n}{k-1}} \right)^c \left( 1 - \frac{d}{\binom{n}{k-1}} \right)^{\binom{n-1}{k-1} - c} \\ &\leq n \frac{\binom{\binom{n-1}{k-1}}{c}}{c!} \left( 1 + O\left( \frac{c^2}{\binom{n-1}{k-1}} \right) \right) \left( \frac{d}{\binom{n}{k-1}} \right)^c \exp\left( - \left( \binom{n-1}{k-1} - c \right) \frac{d}{\binom{n}{k-1}} \right) \\ &\leq n \frac{d^c e^{-d}}{c!} \left( 1 + O\left( \frac{1}{n^{k-1}} \right) \right). \end{aligned}$$

To show concentration of  $X_c$  around its expectation, we will use Chebyshev's inequality. In order to do so, we need to estimate  $\Pr[\deg(1) = \deg(2) = c]$ . For  $\ell \in \{0, \dots, c\}$ , let  $E_{1,2}^\ell$  denote the event that there exist exactly  $\ell$  hyperedges that contain both vertices 1 and 2. Then, letting  $p = \frac{d}{\binom{n}{k-1}}$ , we see that

$$\begin{aligned} \Pr[\deg(1) = \deg(2) = c] &\leq \sum_{\ell=0}^c \Pr[E_{1,2}^\ell] \left( \binom{\binom{n-1}{k-1}}{c-\ell} p^c (1-p)^{\binom{n-1}{k-1} - c} \right)^2 \\ &= \sum_{\ell=0}^c \binom{\binom{n-2}{k-2}}{\ell} p^\ell (1-p)^{\binom{n-2}{k-2} - \ell} \left( \binom{\binom{n-1}{k-1}}{c-\ell} p^c (1-p)^{\binom{n-1}{k-1} - c} \right)^2 \\ &= \Pr[\deg(1) = c] \cdot \Pr[\deg(2) = c] \left( 1 + O\left( \frac{1}{n^{k-1}} \right) \right). \end{aligned}$$

Therefore,

$$\begin{aligned} \text{Var}[X_c] &= \sum_{i=1}^n \sum_{j=1}^n (\Pr[\deg(i) = c, \deg(j) = c] - \Pr[\deg(i) = c] \Pr[\deg(j) = c]) \\ &\leq \sum_{i \neq j=1} O\left( \frac{1}{n^{k-1}} \right) + \mathbb{E}[X_c] = An, \end{aligned}$$

for some constant  $A = A(c, d)$ .

Finally, applying the Chebyshev's inequality, we obtain that, for any  $t > 0$ ,

$$\Pr[|X_c - \mathbb{E}[X_c]| \geq t\sqrt{n}] \leq \frac{A}{t^2},$$

and, thus, the proof is concluded by choosing  $t = \log n$ . ◀

Lemma 20 implies the following useful corollary.

► **Corollary 21.** *For any constants  $\delta \in (0, 1)$ ,  $k \geq 2$ ,  $d > 0$ , let  $X = X(\delta, k, d)$  denote the random variable equal to the number of vertices in  $H(k, n, d/\binom{n}{k-1})$  whose degree is in  $[(1 + \delta)d, 3(k-1)^{k-1}d]$ . There exists a constant  $d_\delta > 0$  such that if  $d \geq d_\delta$  then, almost surely,  $X \leq \frac{n}{d^2}$ .*

**Proof.** Let  $X_r$  denote the number of vertices of degree  $r$  in  $H(k, n, d/\binom{n}{k-1})$ . Since  $k, d$  are constants, using Lemma 20 and Stirling's approximation we see that, almost surely,

$$\begin{aligned} \sum_{r=(1+\delta)d}^{3(k-1)^{k-1}d} X_r &\leq n \left(1 + O\left(\frac{\log n}{\sqrt{n}}\right)\right) \sum_{r=(1+\delta)d}^{3(k-1)^{k-1}d} \frac{d^r e^{-d}}{r!} \\ &\leq n(1+\delta) \sum_{r=(1+\delta)d}^{3(k-1)^{k-1}d} \frac{d^r e^{-d}}{\sqrt{2\pi r} \left(\frac{r}{e}\right)^r} \leq \frac{n}{d^2}, \end{aligned}$$

for sufficiently large  $d$  and  $n$ . ◀

Using Lemma 19 and Corollary 21 we show that, almost surely, only a small fraction of vertices of  $H(k, n, d/\binom{n}{k-1})$  have degree that significantly exceeds its average degree.

► **Lemma 22.** *For every constants  $k \geq 2$  and  $\delta \in (0, 1)$ , there exists  $d_{k,\delta} > 0$  such that for any constant  $d \geq d_{k,\delta}$ , all but at most  $\frac{2n}{d^2}$  vertices of the random hypergraph  $H(k, n, d/\binom{n}{k-1})$  have degree at most  $(1+\delta)d$ , almost surely.*

**Proof.** Corollary 21 implies that the number of vertices with degree in the interval  $[(1+\delta)d, 3(k-1)^{k-1}d]$  is at most  $\frac{n}{d^2}$ , for sufficiently large  $d$ .

Suppose now there are more than  $\frac{n}{d^2}$  vertices with degree at least  $3(k-1)^{k-1}d$ . Denote by  $S$  a set containing exactly  $\frac{n}{d^2}$  such vertices. According to Lemma 19, almost surely, the induced subhypergraph  $H[S]$  has at most

$$e(H[S]) \leq \left(\frac{d}{(\ln d)^2}\right)^{\frac{1}{k-1}} |S| = \frac{n}{d^{2-\frac{1}{k-1}} (\ln d)^{\frac{2}{k-1}}}$$

hyperedges. Therefore, the number of hyperedges between the sets of vertices  $S$  and  $V \setminus S$  is at least

$$3(k-1)^{k-1}d|S| - ke(H[S]) \geq \frac{2.9(k-1)^{k-1}n}{d} =: N.$$

However, the probability that  $H(k, n, d/\binom{n}{k-1})$  contains such a subhypergraph is at most

$$\binom{n}{\frac{n}{d^2}} \binom{\frac{n^k}{d^2}}{N} \left(\frac{d}{\binom{n}{k-1}}\right)^N \leq (ed^2)^{\frac{n}{d^2}} \left(\frac{n^k e}{d^2 N} \cdot \frac{d}{\binom{n}{k-1}}\right)^N = o(1),$$

for sufficiently large  $d$ . Note that in deriving the final equality we used that for any pair of integers  $\alpha, \beta$ , we have that  $\binom{\alpha}{\beta} \geq \left(\frac{\alpha}{\beta}\right)^\beta$ . Therefore, almost surely there are at most  $\frac{n}{d^2}$  vertices in  $G$  with degree greater than  $3(k-1)^{k-1}d$ , concluding the proof. ◀

Finally, we show that the neighborhood of a typical vertex of  $H(k, n, d/\binom{n}{k-1})$  is locally tree-like.

► **Lemma 23.** *For every constants  $k \geq 2, \delta \in (0, 1)$ , almost surely, the random hypergraph  $H(k, n, d/\binom{n}{k-1})$  has a subset  $U \subseteq V(H)$  of size at most  $n^{1-\delta}$  such that the induced hypergraph  $H[V \setminus U]$  is of girth at least 5.*

## 39:14 Coloring Locally Sparse Hypergraphs

**Proof.** Let  $Y_2, Y_3, Y_4$ , denote the number of 2-, 3- and 4-cycles in  $H(n, k, d/\binom{n}{k-1})$ , respectively. A straightforward calculation reveals that for  $i \in \{2, 3, 4\}$ :

$$\begin{aligned} \mathbb{E}[Y_i] &\leq \sum_{s=1}^{i(k-1)} \binom{n}{s} \binom{\binom{s}{k-1}}{i} \left( \frac{d}{\binom{n}{k-1}} \right)^i \\ &\leq i(k-1)n^{i(k-1)} \left( \frac{(i(k-1))^{k-1} e^2 (k-1)^{k-1}}{in^{k-1}} \right)^i = O(1). \end{aligned}$$

By Markov's inequality this implies that  $Y_2 + Y_3 + Y_4 \leq n^{1-\sqrt{\delta}}$  almost surely. Denote by  $U$  the union of all 2-, 3- and 4- cycles in  $H$ . Then the induced subhypergraph  $H[V \setminus U]$  has girth at least 5 and, almost surely,  $|U| \leq n^{1-\delta}$ . ◀

We are now ready to prove Theorem 6.

**Proof of Theorem 6.** Our goal will be to find a subset  $U \subset V$  of size  $|U| \leq nd^{-\frac{1}{k-1}}$  that (i) contains all cycles of length at most 4 and every vertex of degree more than  $(1+\delta)d$ ; and (ii) such that, every vertex  $v$  in  $V \setminus U$  has at most  $9k^2 \left( \frac{d}{(\ln d)^2} \right)^{\frac{1}{k-1}} = o\left( \left( \frac{d}{\ln d} \right)^{\frac{1}{k-1}} \right)$  neighbors in  $U$ . Note that in this case, according to Lemma 19,  $H[U]$  is  $k \left( \frac{d}{(\ln d)^2} \right)^{\frac{1}{k-1}}$ -degenerate, concluding the proof assuming  $d$  is sufficiently large. A similar idea has been used in [5, 6, 25].

Towards that end, let  $U_1$  be the set of vertices of degree more than  $(1+\delta)d$ , and  $U_2$  the set of vertices that are contained in a 2-,3- or a 4-cycle. Notice that  $U_1, U_2$ , can be found in polynomial time and, according to Lemmas 22 and 23, the size of  $U_0 := |U_1 \cup U_2|$  is at most  $\frac{3n}{d^2}$  for sufficiently large  $n$  and  $d$ .

We now start with  $U := U_0$  and as long as there exists a vertex  $v \in V \setminus U$  having at least  $9k^2 \left( \frac{d}{(\ln d)^2} \right)^{\frac{1}{k-1}}$  neighbors in  $U$  we do the following. Let  $S_v = \{u_1, u_2, \dots, u_N\}$  be the neighbors of  $v$  in  $U$ . We choose an arbitrary hyperedge  $h$  that contains  $v$  and  $u_1$  and update  $U$  and  $S_v$  by defining  $U := U \cup h$  and  $S_v := S_v \setminus h$ . We keep repeating this operation until  $S_v$  is empty.

This process terminates with  $|U| < nd^{-\frac{1}{k-1}}$  because, otherwise, we would get a subset  $U \subset V$  of size  $|U| = nd^{-\frac{1}{k-1}}$  spanning more than

$$\frac{1}{k} \left( \frac{n}{d^{\frac{1}{k-1}}} - |U_0| \right) \times 9k^2 \left( \frac{d}{(\ln d)^2} \right)^{\frac{1}{k-1}} \times \frac{1}{k} > \frac{n}{d^{\frac{1}{k-1}}} \times \left( \frac{d}{(\ln d)^2} \right)^{\frac{1}{k-1}}$$

hyperedges, for sufficiently large  $d$ . According to Lemma 19 however,  $H$  does not contain any such set almost surely. ◀

---

## References

- 1 Dimitris Achlioptas and Amin Coja-Oghlan. Algorithmic barriers from phase transitions. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 793–802. IEEE Computer Society, 2008. doi: 10.1109/FOCS.2008.11.
- 2 Dimitris Achlioptas, Fotis Iliopoulos, and Alistair Sinclair. Beyond the lovász local lemma: Point to set correlations and their algorithmic applications. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 725–744. IEEE Computer Society, 2019. doi: 10.1109/FOCS.2019.00049.



- 3 Dimitris Achlioptas and Michael Molloy. The analysis of a list-coloring algorithm on a random graph. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 204–212. IEEE, 1997.
- 4 Miklós Ajtai, János Komlós, Janos Pintz, Joel Spencer, and Endre Szemerédi. Extremal uncrowded hypergraphs. *Journal of Combinatorial Theory, Series A*, 32(3):321–335, 1982.
- 5 Noga Alon and Michael Krivelevich. The concentration of the chromatic number of random graphs. *Combinatorica*, 17(3):303–313, 1997.
- 6 Noga Alon, Michael Krivelevich, and Benny Sudakov. List coloring of random and pseudo-random graphs. *Combinatorica*, 19(4):453–472, 1999.
- 7 Peter Ayre, Amin Coja-Oghlan, and Catherine Greenhill. Hypergraph coloring up to condensation. *Random Structures & Algorithms*, 54(4):615–652, 2019.
- 8 Tom Bohman, Alan Frieze, and Dhruv Mubayi. Coloring  $h$ -free hypergraphs. *Random Structures & Algorithms*, 36(1):11–25, 2010.
- 9 Karthekeyan Chandrasekaran, Navin Goyal, and Bernhard Haeupler. Deterministic algorithms for the Lovász local lemma. *SIAM J. Comput.*, 42(6):2132–2155, 2013. doi: 10.1137/100799642.
- 10 Ewan Davies, Ross J Kang, François Pirot, and Jean-Sébastien Sereni. An algorithmic framework for coloring locally sparse graphs. *arXiv preprint arXiv:2004.07151*, 2020.
- 11 Paul Erdős and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In *Infinite and finite sets (Colloq., Keszthely, 1973; dedicated to P. Erdős on his 60th birthday)*, Vol. II, pages 609–627. Colloq. Math. Soc. János Bolyai, Vol. 10. North-Holland, Amsterdam, 1975.
- 12 Paul Erdős and Joel Spencer. Lopsided Lovász local lemma and latin transversals. *Discrete Applied Mathematics*, 30(2-3):151–154, 1991. doi:10.1016/0166-218X(91)90040-4.
- 13 Alan Frieze and Michał Karoński. *Introduction to random graphs*. Cambridge University Press, 2016.
- 14 Alan Frieze and Dhruv Mubayi. Coloring simple hypergraphs. *Journal of Combinatorial Theory, Series B*, 103(6):767–794, 2013.
- 15 Marylou Gabrié, Varsha Dani, Guilhem Semerjian, and Lenka Zdeborová. Phase transitions in the  $q$ -coloring of random hypergraphs. *Journal of Physics A: Mathematical and Theoretical*, 50(50):505002, 2017.
- 16 A. Johansson. Asymptotic choice number for triangle free graphs, 1996.
- 17 A. Johansson. The choice number of sparse graphs. *Unpublished manuscript*, 1996.
- 18 Jeff Kahn. Asymptotics of the chromatic index for multigraphs. *Journal of Combinatorial Theory, Series B*, 68(2):233–254, 1996.
- 19 Jeff Kahn. Asymptotics of the list-chromatic index for multigraphs. *Random Structures & Algorithms*, 17(2):117–156, 2000.
- 20 Jeong Han Kim. On Brooks’ theorem for sparse graphs. *Combinatorics, Probability and Computing*, 4(2):97–132, 1995.
- 21 János Komlós, János Pintz, and Endre Szemerédi. A lower bound for heilbronn’s problem. *Journal of the London Mathematical Society*, 2(1):13–24, 1982.
- 22 Alexandr Kostochka, Dhruv Mubayi, Vojtěch Rödl, and Prasad Tetali. On the chromatic number of set systems. *Random Structures & Algorithms*, 19(2):87–98, 2001.
- 23 Michael Krivelevich and Benny Sudakov. The chromatic numbers of random hypergraphs. *Random Structures & Algorithms*, 12(4):381–403, 1998.
- 24 Hanno Lefmann. Sparse parity-check matrices over  $\text{GF}(q)$ . *Combinatorics, Probability and Computing*, 14(1-2):147–169, 2005.
- 25 Tomasz Łuczak. The chromatic number of random graphs. *Combinatorica*, 11(1):45–54, 1991.
- 26 Michael Molloy. The list chromatic number of graphs with small clique number. *Journal of Combinatorial Theory, Series B*, 134:264–284, 2019.
- 27 Michael Molloy and Bruce Reed. A bound on the total chromatic number. *Combinatorica*, 18(2):241–280, 1998.

## 39:16 Coloring Locally Sparse Hypergraphs

- 28 Michael Molloy and Bruce Reed. *Graph colouring and the probabilistic method*, volume 23 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 2002.
- 29 Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász local lemma. *J. ACM*, 57(2):Art. 11, 15, 2010. doi:10.1145/1667053.1667060.
- 30 Michel Talagrand. Concentration of measure and isoperimetric inequalities in product spaces. *Publications Mathématiques de l'Institut des Hautes Etudes Scientifiques*, 81(1):73–205, 1995.
- 31 Van H Vu. On the choice number of random hypergraphs. *Combinatorics Probability and Computing*, 9(1):79–95, 2000.
- 32 Van H Vu. A general upper bound on the list chromatic number of locally sparse graphs. *Combinatorics, Probability and Computing*, 11(1):103–111, 2002.
- 33 Lenka Zdeborová and Florent Krzakala. Phase transitions in the coloring of random graphs. *Physical Review E*, 76(3):031131, 2007.

# Smoothed Analysis of the Condition Number Under Low-Rank Perturbations

Rikhav Shah ✉

University of California at Berkeley, CA, USA

Sandeep Silwal ✉

Massachusetts Institute of Technology, Cambridge, MA, USA

---

## Abstract

Let  $M$  be an arbitrary  $n$  by  $n$  matrix of rank  $n - k$ . We study the condition number of  $M$  plus a *low-rank* perturbation  $UV^T$  where  $U, V$  are  $n$  by  $k$  random Gaussian matrices. Under some necessary assumptions, it is shown that  $M + UV^T$  is unlikely to have a large condition number. The main advantages of this kind of perturbation over the well-studied dense Gaussian perturbation, where every entry is independently perturbed, is the  $O(nk)$  cost to store  $U, V$  and the  $O(nk)$  increase in time complexity for performing the matrix-vector multiplication  $(M + UV^T)x$ . This improves the  $\Omega(n^2)$  space and time complexity increase required by a dense perturbation, which is especially burdensome if  $M$  is originally sparse. Our results also extend to the case where  $U$  and  $V$  have rank larger than  $k$  and to symmetric and complex settings. We also give an application to linear systems solving and perform some numerical experiments. Lastly, barriers in applying low-rank noise to other problems studied in the smoothed analysis framework are discussed.

**2012 ACM Subject Classification** Mathematics of computing → Numerical analysis; Theory of computation → Randomness, geometry and discrete structures

**Keywords and phrases** Smoothed analysis, condition number, low rank noise

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.40

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2009.01986>

**Funding** *Sandeep Silwal*: Research supported by the NSF Graduate Research Fellowship under Grant No. 1122374.

**Acknowledgements** We thank Sushruth Reddy and Samson Zhou for helpful conversations. We also thank Piotr Indyk and Arsen Vasilyan for helpful feedback on a draft of the paper.

## 1 Introduction

The smoothed analysis framework as introduced by Spielman and Teng aims to explain the performance of algorithms on real world inputs through a hybrid of worst-case and average case analysis [25]. In this framework, we are given an arbitrary input that is then perturbed randomly according to some specified noise model. We apply this framework to study the condition number of a matrix perturbed with low-rank Gaussian noise. The condition number is of interest since it influences the behavior of many algorithms in numerical linear algebra, both in theory and in practice.

To give context to our result, recall that the condition number of a  $n \times n$  matrix  $M$  with singular values  $s_1(M) \geq \dots \geq s_n(M)$  is defined as the ratio  $s_1(M)/s_n(M)$ . Generally, a condition number is “well behaved” if  $s_1(M)/s_n(M) = n^{O(1)}$ . It can be shown that under very mild and natural conditions, we have  $s_1(M) \leq n^{O(1)}$ . For instance, this readily follows from Proposition 6 if the entries are not too large compared to the size of  $M$  or if the entries have sufficient tail concentration far from the origin. Since our random variables are



© Rikhav Shah and Sandeep Silwal;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 40; pp. 40:1–40:21



Leibniz International Proceedings in Informatics

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

Gaussians, this easily holds. Therefore, the bulk of the work lies in controlling the *smallest* singular value  $s_n(M)$ . Extending a result of Edelman [11], Sankar, Spielman and Teng showed the following result in [22]:

► **Theorem 1.** *There is a constant  $C > 0$  such that the following holds. Let  $M$  be an arbitrary matrix and let  $N_n$  be a random matrix whose entries are iid Gaussian. Let  $M_n = M + N_n$ . Then for any  $t > 0$ ,*

$$\mathbb{P}(s_n(M_n) \leq t) \leq Cn^{1/2}t.$$

Later, Tao and Vu generalized the above result where the entries of  $N_n$  are independent copies of a general class of random variables that have mean zero and bounded variance [32, 27].

## 1.1 Motivation for low-rank noise

The main drawback of these results is that *every* entry of  $M$  must be perturbed by independent noise. This means that if such a perturbation was carried out in practice, we would need to first draw  $n^2$  random numbers and store them. This is more problematic if  $M$  is sparse to begin with and stored in a data structure utilized for sparse matrices. These observations lead us to ask if we can achieve well-conditioned matrices with **less randomness and less space**. Our results demonstrate the answer is yes by replacing the dense Gaussian ensemble  $N_n$  with a *low-rank* matrix.

To further motivate our work, we note that in the context of smoothed analysis, Theorem 1 is used to explain the phenomenon that matrices encountered in practice frequently have “well behaved” condition number. For instance, many matrices can arise out of empirical observations or measurements which can be subject to some inherent noise.

Similarly, low-rank noise is also natural and arises in many scenarios. Low-rank noise has been studied as a noise model in least squares [33], compressed sensing [8, 18, 10], and imaging [15, 26] to name a few applications. In addition, low-rank noise also arises in many applied sciences model, for instance, see the examples in [1] and references therein where examples are given for the eigenvalue problem  $Mx = x + Ex$ , for a low rank matrix  $E$ , arising in scientific modelling. Furthermore, one of the most frequent properties that matrices in data science possess is having low rank (see [6, 7, 30, 19] and references within). Thus for these matrices, the traditional smoothed analysis viewpoint of having a dense perturbation cannot apply due to their low rank requirement.

Hence, an additional motivation of our work is that studying low-rank noise is a natural step in smoothed analysis which we initiate.

## 1.2 Our results

As stated before, we replace the dense Gaussian perturbation  $N_n$  with a low-rank perturbation. Our main result is the following.

► **Theorem 2** (Theorem 13 simplified). *Let  $1 \leq k \leq n/2$  and  $M$  be a matrix of rank  $n - k$ . Let  $U, V$  be  $n \times k$  matrices with i.i.d.  $\mathcal{N}(0, 1)$  entries. Then*

$$\mathbb{P}\left(s_n(M + UV^T) \leq \frac{\varepsilon}{nk} s_{n-k}(M)\right) \leq C\sqrt{\varepsilon} + \exp(-cn)$$

so long as  $s_{n-k}(M) < n$  for absolute constants  $C, c > 0$ .

Theorem 2 roughly states that if we add a rank  $k$  random perturbation to a rank  $n - k$  matrix, then the smallest  $k$  singular values of the matrix improve. The advantage of our approach is that the matrices  $U, V$  can be stored separately from  $M$  using  $O(nk)$  space. This is especially useful in the case that  $M$  is sparse to begin with and is stored using a data structure optimized for sparse matrices. Furthermore, a matrix vector product operation  $(M + UV^T)x$  can be computed in  $\text{Time}(Mx) + O(nk)$  time where  $\text{Time}(Mx)$  is the time required to compute  $Mx$ . For instance, when  $k = O(1)$ , the extra increase in space and time complexity is only  $O(n)$ . This is a significant improvement in both the space required to store a dense Gaussian random matrix  $G$  and computing  $(M + G)x$  which are both  $\Omega(n^2)$ . We prove Theorem 2 in Section 3 and discuss the dependence on the terms  $s_{n-k}(M)$  and  $\sqrt{\varepsilon}$  which we show is unavoidable (see remarks 17, 18).

Theorem 2 can be generalized in a variety of ways. First, our result carries over to the case where we pick the columns of  $U, V$  to be from a rotationally invariant distribution, such as uniform vectors on the unit sphere. We show that our result also carries over to the case where  $M$  is symmetric and we pick  $U = V$  to preserve symmetry.

It is natural to ask if a broader family of random variables can be used in Theorem 2. In Section 3.1 we show that our result *cannot hold* if we pick the entries of  $U, V$  to be from the Rademacher distribution. This is in contrast to the dense perturbation case where Gaussian random variables can be replaced with a wide variety of other distributions such as sub-Gaussian random variables (which include Rademachers). On the other hand, we can get an analogous statement to Theorem 13 if we allow for complex Gaussian perturbations.

► **Theorem 3** (Theorem 15 simplified). *Let  $1 \leq k \leq n/2$  and  $M$  be a matrix of rank  $n - k$ . Let  $U, V$  be  $n \times k$  complex matrices with real and imaginary parts in each entry drawn independently from  $\mathcal{N}(0, 1/2)$ . Then*

$$\mathbb{P}\left(s_n(M + UV^T) \leq \frac{\varepsilon}{nk} s_{n-k}(M)\right) \leq C\varepsilon + \exp(-cn)$$

so long as  $s_{n-k}(M) < n$  for absolute constants  $C, c > 0$ .

A further natural question to consider is if the low-rank noise model can be studied in other problems in smoothed analysis. In Section B, we highlight the challenges that arise when applying low-rank random perturbations to other well studied problems in smoothed analysis such as the simplex method and  $k$ -means clustering. We show that current analysis methods that work for dense random perturbations for these problems do not carry over to the low-rank case due to the lack of independence.

Lastly, we note that Theorem 2 requires that if the input matrix  $M$  has rank  $n - k$ , then perturbation has rank exactly  $k$ . This condition can be relaxed in a couple of ways; first, if we add a perturbation of rank less than  $k$  then the matrix will be singular so there is nothing to study in this case. On the other hand, adding a rank  $k' > k$  perturbation to a rank  $n - k$  matrix can be thought of as adding a rank  $(k' - k)$  perturbation to a full rank matrix since the original matrix plus a rank  $k$  perturbation will be full rank with probability 1. Then as explained further in Section 4, we can obtain the following result in this case.

► **Theorem 4** (Theorem 20 simplified). *Let  $M$  be a  $n \times n$  real matrix with  $\text{rank}(M) = n$ , smallest singular value  $s_n$ , and  $U, V \in \mathbb{R}^{n \times k}$  have independent  $\mathcal{N}(0, 1)$  coordinates. Then for all  $\varepsilon \in (0, 1)$ ,*

$$\mathbb{P}\left(s_n(M + UV^T) \leq \frac{\varepsilon}{\sqrt{n}}\right) \leq C \left( \sqrt{\varepsilon} \left(1 + \frac{2nk}{s_n}\right) + \frac{1}{(2nk)^{9/4}} + \exp(-cnk) \right).$$

The second way to circumvent Theorem 2 is with the use of Weyl’s perturbation inequality. To see how it applies, consider the case of  $k = 1$ . Decompose  $M = s_n(M)\ell_n r_n^T + M'$  where  $\ell_n, r_n$  are the left and right singular vectors associated with  $s_n(M)$ . Then we can view  $M + uv^T$  as a random perturbation of  $M'$  (which has rank  $n - 1$ ), plus matrix  $s_n(M)\ell_n r_n^T$  whose operator norm is at most  $s_n$ . We can then apply Theorem 2 to  $M'$  to bound  $s_n(M' + UV^T)$  in terms of  $s_{n-1}(M') = s_{n-1}(M)$ , and then incur an additional additive  $s_n(M)$  error by Weyl’s inequality. Since our ideal use case is when  $s_n(M)$  is already negligible, the final bound that we get is comparable to the bound from Theorem 2.

Finally in Section 3.2, we discuss an application of low-rank perturbations to solving large sparse linear systems and in Section A, we present numerical evidence for our low-rank error model.

► **Remark 5.** Note that Theorem 1 and the works of Tao and Vu in [32, 27] both prove a statement of the form  $\mathbb{P}(s_n(M + E) \leq n^{-A}) \leq n^{-B}$  where  $E$  is the perturbation and  $A, B$  are parameters that depend on the random variables comprising the perturbation. Our statements in Theorem 2 is also of similar flavor since it shows that  $\mathbb{P}(s_n(M + E) \leq s_{n-k}(M) n^{-A}) \leq n^{-B}$ . Since the theorems of Sankar, Spielman, and Teng and Tao and Vu have found other applications in smoothed analysis, such as in the analysis of the simplex method and beyond, we envision that our theorem could also find similar applications. We discuss barriers in applying the low-rank noise model to other smoothed analysis problems in Section B.

### 1.3 Previous techniques and our approach

In summary, it is difficult to apply previous techniques in our case since we have *shared randomness* across different rows/columns of the matrix. In more detail, all of the previous techniques used to bound the singular values of a random matrix rely on the controlling the distance between a row to the span of the other rows. To see why this is relevant, imagine a singular matrix. In such a case, it is clear that there must exist a row that lies in the span of the other rows. Therefore, controlling the distance from a row to the span of the other rows gives control over the smallest singular value.

Controlling this geometric quantity boils down to understanding the dot product between a row and the normal vector of the hyperplane spanned by the other rows. Crucially if the rows are independent, we can treat the normal vector of the hyperplane as fixed so this question reduces to the well known Erdos-Littlewood-Offord anti-concentration inequality and its variants which are used in previous works such as [28, 32, 27].

To be more precise, lets consider a high level overview of the proof of Sankar, Spielman, and Teng’s Theorem 1. Fix a vector  $x$  and note that from the identity  $s_n(M_n) = \|M_n^{-1}\|$ , it suffices to give a tail bound on  $\|M_n^{-1}x\|$ . By applying an orthogonal rotation and using the rotational invariance of the Gaussian, we can say that

$$\|M_n^{-1}x\| = \|M_n^{-1}e_1\| = \|c_1\|$$

where  $e_1$  is the first basis vector and  $c_1$  is the first column of  $M_n^{-1}$ . From the equation  $M_n \cdot M_n^{-1} = I$ , it follows that  $\|c_1\| = 1/|w^T r_1|$  where  $r_1$  is the first row of  $M_n$  and  $w^T$  is the normal vector of the span of the rows  $r_2, \dots, r_n$ . Therefore, the proof reduces to understanding the dot product between a random vector  $r_1$ , and another independent vector  $w$ . In the more general case of Tao and Vu [32, 27], more elaborate dot product estimates using the Erdos-Littlewood-Offord inequality are needed.

In our case, if we add a rank 1 perturbation to a matrix, randomness is shared across *all* rows. Therefore, we cannot reduce our problem to understanding the dot product between a random vector and another independent vector since fixing a normal hyperplane of a span

of a subset of rows automatically gives information about the rows not considered in the span due to the shared randomness. Hence, it is tricky to apply the spectrum of existing techniques in our case.

To overcome these barriers, we use a completely different method to prove Theorem 13. We first reduce our problem to adding noise to a diagonal matrix by using rotational invariance. Then we employ linear algebraic tools (rather than probabilistic tools), to get an “explicit” representation of the inverse of a matrix after adding rank  $k$  noise. After arriving at an explicit representation of the inverse, we are able to compute a probabilistic bound on the smallest singular value. Our proof crucially uses the fact that our low-rank perturbations have Gaussian entries whereas the proofs of the dense perturbations carry over in other distributional settings as well. This is not a flaw of our method since it is simply not possible to prove an analogue of our theorems even if the entries of the low-rank perturbations come from sub-Gaussian distributions. We elaborate this point further in Section 3.1.

For Theorem 20 where we add a rank  $k$  random noise to a rank  $n$  matrix, we carefully adapt the geometric ideas utilized in previous approaches as explained above. However to get around the shared randomness between rows, we have to perform some careful conditioning which allows us better control the behavior of the random normal vector  $w$ .

#### 1.4 Why would the perturbed matrix even be full rank?

We briefly address the question of why we even expect low-rank perturbations to improve the condition number. Consider the case where  $D$  is a diagonal matrix of rank  $n - 1$  and we add a random rank 1 Gaussian perturbation  $uv^T$ . Recall the matrix determinant lemma which states that

$$\det(D + uv^T) = \det(D) + v^T \text{adj}(D)u$$

where  $\text{adj}(D)$  is the adjugate matrix of  $D$ . In our case, we can assume that the first  $n - 1$  entries on the diagonal of  $D$  are given by  $s_1(D), \dots, s_{n-1}(D)$  while the last entry is 0. Then, the adjugate matrix is the all zeros matrix except the bottom rightmost entry which is  $s_1(D) \cdots s_{n-1}(D)$ . Therefore,

$$\det(D) + v^T \text{adj}(D)u = (u_n v_n)(s_1(D) \cdots s_{n-1}(D))$$

which is non-zero with probability 1 since  $u_n v_n \neq 0$  with probability 1. Thus, adding a random rank 1 perturbation results in  $D$  not being singular which motivates the question of studying the smallest singular value after a random rank 1 (and more generally low-rank) perturbation.

#### 1.5 Related works

The smoothed analysis framework has been applied to a variety of problems, most notably in analyzing optimization problems such as  $k$ -means [2, 3], the perceptron algorithm [5], and the simplex method [25, 9]. In all of these results, the goal is to show that after an input instance of the problem is suitably perturbed, the algorithm or heuristic runs in polynomial time (the time may depend on the properties of the noise). For a survey of results, see [29, 24, 16] and references within. The analysis used tends to be very problem specific and also heavily dependent on the type of noise added which for a vast majority of cases are dense Gaussian noise.

### Zero preserving noise

The work that is closest in spirit to our work is the zero preserving noise model studied by Spielman, and Teng. It was shown in [22] that if  $M$  is a *symmetric* matrix, then adding an independent Gaussian random variable  $x_{ij}$  to each entry  $M_{ij}$  such that  $i \neq j$ ,  $M_{ij} \neq 0$ , and satisfying  $x_{ij} = x_{ji}$  along with a Gaussian perturbation along the diagonal results in a “well behaved” condition number. However, the main drawback of this result is that it only holds for symmetric matrices and even in this case, a dense perturbation is required if  $M$  is dense to begin with.

### Other works

There are works that use sparse Gaussian perturbations, i.e, their perturbation model is a Gaussian times a Bernoulli random variable with a small parameter. If the Bernoulli random variable has sufficiently small parameter, then with high probability, most of the entries in the perturbation will be zero. The downsides of these methods are that many random variables still need to be drawn and it is not clear if they can show the resulting matrix is well conditioned. For example, Theorem 3.6 in [20] only shows that the singular values of the resulting matrix are *separated* from each other, not that the matrix is well conditioned. In fact, the study of these types of random matrices where entries are sub-Gaussian random variables multiplied by independent Bernoulli variables is still lacking. For instance, the smallest singular value of such family of random matrices was only recently resolved in the highly technical paper of Basak and Rudelson [4].

## 1.6 Notation

We use capital letters as  $A, M$  to denote matrices and lower case letters such as  $x$  for vectors. For a vector  $x$ , the norm  $\|x\|$  is always the Euclidean norm whereas for a matrix  $A$ , the norm  $\|A\|$  always refers to the operator norm (the largest singular value). For a matrix  $A$ , let  $A_S$  denote the sub-matrix of  $A$  which includes the  $i$ th row of  $A$  if and only if  $i \in S$ . The relation  $a \lesssim b$  denotes that  $a$  is less than or equal to  $b$  up to some fixed positive constant and similarly,  $a \simeq b$  denotes that  $a$  and  $b$  are equal up to some fixed positive constant. Unless otherwise indicated, variables  $C, c, C_1, C_2, \dots$  denote positive constants.

## 2 Preliminaries

In this section we enumerate some useful results. First, we recall a classical estimate of the operator norm of a random matrix of Seginer [23]. The following proposition essentially shows that the top singular value of a random matrix is well behaved under mild assumptions. Alternatively, one can also bound the top singular value by the frobenius norm if the random variables populating the matrix have sufficient tail concentration.

► **Proposition 6.** *Let  $M$  be a random  $n \times n$  matrix with entries  $m_{ij}$ . Then,*

$$\mathbb{E}\|M\| = O \left( \mathbb{E} \max_{1 \leq i \leq n} \sqrt{\sum_{j=1}^n m_{ij}^2} + \mathbb{E} \max_{1 \leq j \leq n} \sqrt{\sum_{i=1}^n m_{ij}^2} \right).$$

Next we establish tail bounds for the smallest and largest singular values of real and complex Gaussian matrices.



► **Lemma 7** (Theorem 1 reformulated.). Let  $G \in \mathbb{R}^{k \times k}$  with all entries chosen i.i.d. from  $\mathcal{N}(0, 1)$ . Then

$$\mathbb{P}\left(s_k(G) \leq t_1/\sqrt{k}\right) < Ct_1.$$

for some absolute constant  $C$ .

► **Lemma 8** (Theorem 1.1 in [28]). Let  $G \in \mathbb{C}^{k \times k}$  with all entries chosen with i.i.d. real and imaginary parts from  $\mathcal{N}(0, 1/2)$ . Then

$$\mathbb{P}\left(s_k(G) \leq t_1/\sqrt{k}\right) < t_1^2.$$

► **Lemma 9** (Proposition 2.3 in [21]). Let  $G \in \mathbb{R}^{(n-k) \times k}$  for  $k \leq n/2$  with all entries chosen i.i.d. from  $\mathcal{N}(0, 1)$ . Then

$$\mathbb{P}\left(s_1(G) \geq t_2\sqrt{n-k}\right) < C_1 e^{-C_2 t_2^2 n}.$$

for  $t_2$  larger than some absolute constant, and  $C_1, C_2$  absolute constants.

► **Lemma 10.** Let  $G \in \mathbb{C}^{(n-k) \times k}$  for  $k \leq n/2$  with all entries chosen with i.i.d. real and imaginary parts from  $\mathcal{N}(0, 1/2)$ . Then

$$\mathbb{P}\left(s_1(G) \geq t_2\sqrt{n-k}\right) < 2C_1 e^{-C_2 t_2^2 n}.$$

for  $t_2, C_1, C_2$  as in Lemma 9.

**Proof.** Decompose  $G = A + iB$  and bound  $s_1(G) \leq s_1(A) + s_1(B)$ . Then

$$\begin{aligned} \mathbb{P}\left(s_1(G) \geq t_2\sqrt{2(n-k)}\right) &\leq \mathbb{P}\left(s_1(A) + s_1(B) \geq t_2\sqrt{2(n-k)}\right) \\ &\leq \mathbb{P}\left(s_1(A) \geq \frac{t_2\sqrt{2(n-k)}}{2}\right) + \mathbb{P}\left(s_1(B) \geq \frac{t_2\sqrt{2(n-k)}}{2}\right) \\ &\leq \mathbb{P}\left(s_1(\sqrt{2}A) \geq t_2\sqrt{n-k}\right) + \mathbb{P}\left(s_1(\sqrt{2}B) \geq t_2\sqrt{n-k}\right) \\ &\leq 2C_1 e^{-C_2 t_2^2 n} \end{aligned}$$

where the last inequality follows by Lemma 9 since  $\sqrt{2}A$  and  $\sqrt{2}B$  have real i.i.d.  $\mathcal{N}(0, 1)$  entries. ◀

The following lemma bounds the smallest singular value of a block matrix.

► **Lemma 11.** Let

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

be an  $n \times n$  matrix. Then

$$s_n(M)^{-1} \leq \|A^{-1}\| + \|(M/A)^{-1}\| (1 + \|A^{-1}B\|) (1 + \|CA^{-1}\|)$$

where  $(M/A) = D - CA^{-1}B$  is the Schur complement of  $A$ .

**Proof.** We first use the Schur formula for the inverse of a block matrix:

$$M^{-1} = \begin{bmatrix} A^{-1} + A^{-1}B(M/A)^{-1}CA^{-1} & A^{-1}B(M/A)^{-1} \\ (M/A)^{-1}CA^{-1} & (M/A)^{-1} \end{bmatrix}.$$

The norm of  $M^{-1}$  is upper bounded by the sum of the norms of each of its blocks.

$$\begin{aligned} s_n(M)^{-1} = \|M^{-1}\| &\leq \|A^{-1}\| + \|A^{-1}B\| \|(M/A)^{-1}\| \|CA^{-1}\| \\ &\quad + \|A^{-1}B\| \|(M/A)^{-1}\| \\ &\quad + \|(M/A)^{-1}\| \|CA^{-1}\| \\ &\quad + \|(M/A)^{-1}\| \\ &= \|A^{-1}\| + \|(M/A)^{-1}\| (1 + \|A^{-1}B\|) (1 + \|CA^{-1}\|). \end{aligned} \quad \blacktriangleleft$$

Lastly, we recall that Gaussians are sufficiently anti-concentrated.

► **Proposition 12.** *Let  $x \sim \mathcal{N}(0, 1)$ . Then,  $\mathbb{P}(|x| \leq \varepsilon) = \Theta(\varepsilon)$  for  $\varepsilon$  sufficiently small.*

### 3 Proof of main theorems

The goal of this section is to prove the following theorem and its complex and symmetric analogs.

► **Theorem 13.** *Let  $M$  be an arbitrary matrix of rank  $n - k \geq n/2$ . Let  $U, V$  be  $n \times k$  matrices with i.i.d.  $\mathcal{N}(0, 1)$  entries. Then*

$$\mathbb{P}\left(s_n(M + UV^T) \leq \frac{t_1^2}{k} \min\left(\frac{1}{2}, \frac{s_{n-k}(M)}{4t_2^2(n-k)}\right)\right) \leq C_1 t_1 + C_2 \exp(-C_3 t_2^2 n) \quad (1)$$

for  $t_1 \leq C_4$  and  $t_2 \geq C_5$  for some absolute constants  $C_i, 1 \leq i \leq 5$ .

Our strategy to prove Theorem 13 will reduce general  $M$  to the case of  $M$  nonnegative and diagonal, then express  $s_n(M + UV^T)$  in terms of the singular values of  $M$  and certain sub-matrices of  $U$  and  $V$ , and finally apply tail bounds to said singular values. We start by proving a lemma that allows us to reduce to the case of  $M$  nonnegative and diagonal. As stated in Section 1.3, this is a completely different proof strategy than the one used in previous works.

► **Lemma 14.** *Let  $D = \text{diag}(s_n(M), \dots, s_1(M))$ . Let  $U, V$  be as in Theorem 13. Then the distributions of  $s_n(M + UV^T)$  and of  $s_n(D + UV^T)$  are identical.*

**Proof.** Let  $LDR^T = M$  be the singular value decomposition of  $M$ . Then

$$M + UV^T = LDR^T + UV^T = L(D + L^T UV^T R)R^T.$$

Left- and right- multiplication by unitary matrices preserves singular values so

$$s_n(M + UV^T) = s_n(D + L^T UV^T R).$$

Finally,  $U$  and  $V$  are rotationally invariant, so  $L^T U$  and  $R^T V$  are distributed just as  $U$  and  $V$  are. ◀

Now we proceed to the main proof.

**Proof of Theorem 13.** For any matrix  $T$ , recall that  $T_S$  denotes the sub-matrix of  $T$  which includes the  $i$ th row of  $T$  if and only if  $i \in S$ . Lemma 14 shows that we may assume  $M$  is nonnegative and diagonal without loss of generality. We may write  $M$  and  $M + UV^T$  in block form as

$$M = \begin{bmatrix} 0 & 0 \\ 0 & M' \end{bmatrix} \quad \text{and} \quad M + UV^T = \begin{bmatrix} U_{[k]}V_{[k]}^T & U_{[k]}V_{[n]\setminus[k]}^T \\ U_{[n]\setminus[k]}V_{[k]}^T & M' + U_{[n]\setminus[k]}V_{[n]\setminus[k]}^T \end{bmatrix}$$

where  $M'$  has no zeros on the diagonal. We can use Lemma 11 to upper bound  $s_n(M + UV^T)$ . The factor corresponding to the Schur complement is

$$\left\| \left( M' - U_{[n]\setminus[k]} \left( I - V_{[k]}^T (U_{[k]}V_{[k]}^T)^{-1} U_{[k]} \right) V_{[n]\setminus[k]}^T \right)^{-1} \right\| = \|M'^{-1}\| = s_{n-k}(M)^{-1}$$

since  $I - V_{[k]}^T (U_{[k]}V_{[k]}^T)^{-1} U_{[k]} = 0$ . This is one of the key insights of our proof. Then the resulting bound is

$$\begin{aligned} & s_n(M + UV^T)^{-1} \\ & \leq \frac{1}{s_k(U_{[k]})s_k(V_{[k]}^T)} + \frac{1}{s_{n-k}(M)} \left( 1 + \|(U_{[k]}V_{[k]}^T)^{-1}U_{[k]}V_{[n]\setminus[k]}^T\| \right) \left( 1 + \|U_{[n]\setminus[k]}V_{[k]}^T(U_{[k]}V_{[k]}^T)^{-1}\| \right) \\ & = \frac{1}{s_k(U_{[k]})s_k(V_{[k]}^T)} + \frac{1}{s_{n-k}(M)} \left( 1 + \|V_{[k]}^{-1}V_{[n]\setminus[k]}^T\| \right) \left( 1 + \|U_{[k]}^{-1}U_{[n]\setminus[k]}\| \right) \\ & \leq \frac{1}{s_k(U_{[k]})s_k(V_{[k]}^T)} + \frac{1}{s_{n-k}(M)} \left( 1 + \|V_{[k]}^{-1}\| \|V_{[n]\setminus[k]}^T\| \right) \left( 1 + \|U_{[k]}^{-1}\| \|U_{[n]\setminus[k]}\| \right) \\ & = \frac{1}{s_k(U_{[k]})s_k(V_{[k]}^T)} + \frac{1}{s_{n-k}(M)} \left( 1 + \frac{s_1(V_{[n]\setminus[k]})}{s_k(V_{[k]})} \right) \left( 1 + \frac{s_1(U_{[n]\setminus[k]})}{s_k(U_{[k]})} \right). \end{aligned}$$

Denote events

$$\begin{aligned} \mathcal{E}_1 &= \left( s_1(U_{[n]\setminus[k]}) \leq t_2\sqrt{n-k} \quad \text{and} \quad s_1(V_{[n]\setminus[k]}) \leq t_2\sqrt{n-k} \right), \\ \mathcal{E}_2 &= \left( s_k(U_{[k]}) \geq t_1/\sqrt{k} \quad \text{and} \quad s_k(V_{[k]}) \geq t_1/\sqrt{k} \right). \end{aligned}$$

Conditioning on  $\mathcal{E}_1$  and  $\mathcal{E}_2$ , the above bound becomes

$$s_n(M + UV^T)^{-1} \leq \frac{1}{s_{n-k}(M)} \left( 1 + \frac{t_2}{t_1} \sqrt{(n-k)k} \right)^2 + \frac{k}{t_1^2}.$$

For sufficiently large  $n$  (specifically  $n \geq 6\frac{t_1^2}{k t_2^2}$ ), this becomes

$$s_n(M + UV^T)^{-1} \leq \frac{k}{t_1^2} \left( \frac{2t_2^2(n-k)}{s_{n-k}(M)} + 1 \right) \leq \frac{2k}{t_1^2} \max \left( \frac{2t_2^2(n-k)}{s_{n-k}(M)}, 1 \right)$$

Taking the reciprocal of both sides yields

$$s_n(M + UV^T) \geq \frac{t_1^2}{2k} \min \left( \frac{s_{n-k}(M)}{2t_2^2(n-k)}, 1 \right)$$

The probability that this bound is violated is upper bounded by the probability that at least one of  $\mathcal{E}_1$  or  $\mathcal{E}_2$  fail. We may upper bound this quantity using the union bound:

$$\begin{aligned} \mathbb{P}(\neg\mathcal{E}_1 \vee \neg\mathcal{E}_2) &\leq \mathbb{P}(\neg\mathcal{E}_1) + \mathbb{P}(\neg\mathcal{E}_2) \\ &\leq \mathbb{P}(s_1(U_{[n]\setminus[k]}) \geq t_2\sqrt{n-k}) + \mathbb{P}(s_1(V_{[n]\setminus[k]}) \geq t_2\sqrt{n-k}) \\ &\quad + \mathbb{P}(s_k(U_{[k]}) \leq t_1/\sqrt{k}) + \mathbb{P}(s_k(V_{[k]}) \leq t_1/\sqrt{k}) \\ &\leq 2C_1t_1 + 2C_2e^{-C_3t_2^2n}. \end{aligned}$$

where the last step follows by applying Lemmas 7 and 9 twice each. The factors of 2 can be subsumed into the constants  $C_1$  and  $C_2$  giving the final result.  $\blacktriangleleft$

## 40:10 Smoothed Analysis of the Condition Number Under Low-Rank Perturbations

► **Theorem 15.** Let  $M, t_1, t_2, C_2, C_3$  be as in theorem 13. Let  $U, V$  be  $n \times k$  complex matrices with real and imaginary parts drawn independently from  $\mathcal{N}(0, 1/2)$ . Then

$$\mathbb{P}\left(s_n(M + UV^T) \leq \frac{t_1^2}{k} \min\left(\frac{1}{2}, \frac{s_{n-k}(M)}{4t_2^2(n-k)}\right)\right) \leq 2t_1^2 + C_2 \exp(-C_3 t_2^2 n).$$

Note that the first term on the righthand side is  $2t_1^2$  rather than  $C_1 t_1$  as it was in Theorem 13.

**Proof.** The only place the proof differs from the proof of Theorem 13 is in the upper bound on  $\mathbb{P}(-\mathcal{E}_1)$ . Instead of  $C_1 t_1$ , it is simply  $t_1^2$  by Lemma 8. ◀

► **Remark 16.** Theorems 13 and 15 hold when instead of sampling  $U$  and  $V$  independently, simply set  $U = V$ .

**Proof.** The proof follows almost exactly as before with only a single modification: In Lemma 14, the left- and right- singular vectors of symmetric matrices are the same so  $L = R$  (so  $L^T U = R^T V$ ). Optionally, one may note that events  $\mathcal{E}_1$  and  $\mathcal{E}_2$  are redundant, so one reduces the bound on  $\mathbb{P}(-\mathcal{E}_1 \vee -\mathcal{E}_2)$  by a factor of 2. ◀

► **Remark 17.** Let us briefly mention why the term  $s_{n-k}(M)$  is unavoidable in the statement of Theorem 13. For simplicity, consider  $k = 1$  and suppose that  $M$  is of rank  $n - 1$  and suppose its smallest nonzero singular value is equal to  $\delta$ . After adding a rank 1 term  $uv^T$  to  $M$ , its rank is  $n$  with probability 1. However, if we consider the limit  $\delta \rightarrow 0$ , then  $M + uv^T$  approaches a rank  $n - 1$  matrix meaning  $s_n(M + uv^T) \rightarrow 0$ . Hence, any concentration bound such as (1) must depend on the term  $s_{n-1}$ .

► **Remark 18.** The term  $t_1$  on the right hand side of (1) is also unavoidable. For simplicity, consider the case that  $M \in \mathbb{R}^{n \times n}$  is a diagonal matrix and all the entries are non zero except the last diagonal entry and consider a rank 1 symmetric update. Now after a symmetric rank 1 update, the perturbed matrix is  $M + gg^T$  where  $g \in \mathbb{R}^n$ . The smallest singular value of a symmetric matrix is equivalent to the smallest absolute eigenvalue. From the Raleigh quotient characterization of eigenvalues, we have that

$$\mathbb{P}(s_n(M + gg^T) \leq t^2) \geq \mathbb{P}(|e_n^T(M + gg^T)e_n| \leq t^2).$$

Using the fact that  $Me_n = 0$ , we have

$$\mathbb{P}(|e_n^T(M + gg^T)e_n| \leq t^2) = \mathbb{P}(z^2 \leq t^2)$$

where  $z \in \mathbb{R}$  is a standard normal. Finally,  $\mathbb{P}(z^2 \leq t^2) = \mathbb{P}(|z| \leq t) = \Theta(t)$  from Proposition 12 for  $t$  sufficiently small.

### 3.1 Sub-Gaussian perturbations

Just as Tao and Vu generalized Theorem 1 to the case where more general types of random perturbations beyond Gaussian are used, it is of interest to generalize Theorem 13 to the case where  $U, V$  are from a general family of distributions. A standard choice are mean zero sub-Gaussian distributions since they encompass well known distributions such as the standard Gaussian and  $\pm 1$  (Rademacher) random variables. Surprisingly, we show in this case that we cannot state a general statement like Theorem 13 unless extra assumptions about the fixed matrix  $M$  is made.

► **Lemma 19.** *Let  $u, v \in \mathbb{R}^n$  be vectors with i.i.d. entries that are  $\pm 1$  (Rademacher) with equal probability. There exists a rank  $n - 1$  matrix  $M$  such that with constant probability,  $M + uv^T$  is singular.*

**Proof.** As in the proof of Theorem 13, let  $M = LDR$  and we can say  $s_n(M + uv^T) = s_n(D + (L^T u)(v^T R))$ . In the case that  $u$  and  $v$  are Gaussian, rotational invariance implies that  $L^T u$  and  $v^T R$  are distributed as  $u, v$  respectively. However, this is no longer the case if  $u, v$  have entries coming from general sub-Gaussian distributions, such as  $\pm 1$ . Here, the properties of  $L, R$  can have substantial impact on  $s_n(M + uv^T)$ .

Suppose that the top left entry of  $D$  is 0. Then, if the first row of  $L$  is *sparse*, i.e. has  $O(1)$  non zero entries, then it is possible that the first coordinate of  $L^T u$ ,  $(L^T u)_1$ , is 0 with constant probability and hence the first row of  $D + (L^T u)(v^T R)$  is all 0 which implies that  $M + uv^T$  is still rank  $n - 1$  with constant probability. ◀

Therefore, a general statement such as Theorem 13 in the case of sub-Gaussian distributions is impossible unless extra assumptions are made about the input matrix  $M$ . However, we note that in the  $k = 1$  case, if we *assume* every row of  $L, R$  are *dense* (say have at least a constant fraction of non-zero entries), then the proof of Theorem 13 carries through in the  $\pm 1$  case since the two estimates we need (corresponding to the events  $\mathcal{E}_1$  and  $\mathcal{E}_2$  respectively) are the concentration of the norms of  $L^T u, v^T R$  and each entry being anti-concentrated from 0 which follows from Erdos-Littlewood-Offord type results. It is not clear when such an assumption is natural.

## 3.2 Application to linear systems

We briefly highlight the importance of the condition number in solving systems of linear equations. If we are interested in solving the system  $Ax = b$  where  $A \in \mathbb{R}^{n \times n}$  then the condition number of  $A$  influences both the stability and runtime of linear systems solving. Much of this material is standard and can be found in [31].

**Stability:** If  $\tilde{x}$  denotes the result computed by numerical algorithms to the equation  $Ax = b$  then it is known that the relative error quantity  $\|x - \tilde{x}\|/\|x\|$  satisfies

$$\frac{\|x - \tilde{x}\|}{\|x\|} = O\left(\varepsilon_{\text{machine}} \cdot \frac{s_1(A)}{s_n(A)}\right)$$

where  $\varepsilon_{\text{machine}}$  is the machine precision.

**Runtime:** One of the most widely used algorithms for solving systems of linear equations, especially large sparse ones that arise often in practice, is the conjugate gradient descent method. If the conjugate gradient descent method is run for  $k$  steps, then its convergence scales roughly as

$$\left(\frac{\sqrt{s_1(A)/s_n(A)} - 1}{\sqrt{s_1(A)/s_n(A)} + 1}\right)^k \approx \left(1 - \frac{2}{\sqrt{s_1(A)/s_n(A)}}\right)^k.$$

Therefore, a larger the condition number means more steps of the conjugate gradient descent method are required.

The usefulness of our low-rank error model is further supported by the conjugate gradient descent method. As mentioned previously, this iterative method is mainly used for large sparse systems. Thus, a low-rank perturbation that only requires additional linear space and incurs an additive linear increase in cost per iteration is desirable compared to a dense perturbation which makes the original problem infeasible for large systems.

#### 4 Perturbation beyond rank $k$

In this section, we deal with the case that we add a rank  $k'$  perturbation to a rank  $n - k$  matrix for  $k' > k$ . In such a case, we simply ignore a rank  $k$  portion of the noise and imagine that we are adding a rank  $(k' - k)$  perturbation to a general full-rank matrix. This is valid since the original rank  $k$  matrix plus the rank  $k$  part of the noise will be full rank with probability 1. Our result then is the following.

► **Theorem 20.** *Let  $M$  be a  $n \times n$  real matrix with  $\text{rank}(M) = n > 10$  with smallest singular value  $s_n$ . Let  $U, V \in \mathbb{R}^{n \times k}$  have independent  $\mathcal{N}(0, 1)$  coordinates. Then,*

$$\mathbb{P}(s_n(M + UV^T) \leq 1/t) \leq C \left( \frac{\sqrt{n}}{x_2 \cdot t} \left( 1 + \frac{x_1 \cdot x_3^{1/2} \cdot \sqrt{nk}}{s_n(M)} \right) + \exp(-x_1^2/4) + (2/\pi)^{k/2} x_2^k + \exp(-c x_3) \right) \quad (2)$$

for all  $t > 0$ ,  $x_1 \geq 3\sqrt{2\log(2nk)}$ ,  $x_3 \geq nk$ , and  $x_2 \leq 1$ .

We obtain the following corollary under some natural parameter settings.

► **Corollary 21.** *Let  $M$  be a  $n \times n$  real matrix with  $\text{rank}(M) = n$ , and  $U, V \in \mathbb{R}^{n \times k}$  have independent  $\mathcal{N}(0, 1)$  coordinates. Then for all  $\varepsilon \in (0, 1)$ ,*

$$\mathbb{P}(s_n(M + UV^T) \leq \varepsilon/\sqrt{n}) \leq C \left( \sqrt{\varepsilon} \left( 1 + \frac{3nk\sqrt{\log(2nk)}}{s_n(M)} \right) + \frac{1}{(2nk)^{9/4}} + \exp(-cnk) \right). \quad (3)$$

**Proof.** Set  $t = \sqrt{n}/\varepsilon$ ,  $x_1 = 3\sqrt{\log(2nk)}$ ,  $x_2 = \sqrt{\varepsilon}$ ,  $x_3 = nk$ . ◀

By setting  $\varepsilon$  appropriately, we recover the common “theme” of  $\mathbb{P}(s_n(M + UV^T) \leq n^{-A}) \leq n^{-B}$  as in the case of Theorem 1 and the works of Tao and Vu [32, 27].

We now proceed to prove Theorem 20. Denote  $A = M + UV^T$  and note that  $\text{rank}(A) = n$  with probability 1 so  $A^{-1}$  exists. We observe that (2) reduces to bounding  $\mathbb{P}(\|A^{-1}\| \geq t)$ . Our proof is adapted from the proof of Theorem 1. However, we need to perform a careful conditioning argument to prove the most important part of the argument which is presented in Lemma 23.

We begin by handling the case of a single vector.

► **Lemma 22.** *For any unit vector  $y$ , we have*

$$\mathbb{P}(\|A^{-1}y\|_2 \geq t) \leq C \left( \frac{1}{x_2 \cdot t} \left( 1 + \frac{x_1 \cdot x_3^{1/2} \cdot \sqrt{nk}}{s_n(M)} \right) + \exp(-x_1^2/4) + (2/\pi)^{k/2} x_2^k + \exp(-c x_3) \right)$$

for all  $t > 0$ ,  $x_1 \geq 2\sqrt{\log n}$ ,  $x_3 \geq nk$ , and  $x_2 \leq 1$ .

**Proof.** Let  $Q$  be a rotation that takes  $y$  to  $e_n$  and denote  $QA$  as  $A'$ . Then,

$$\|A^{-1}y\|_2 = \|A^{-1}Q^T e_n\|_2 = \|(QA)^{-1}e_n\|_2 = \|c_n\|_2.$$

where  $c_n$  be the last column of  $(QA)^{-1}$ . From the identity  $A'A'^{-1} = I$ , we have that  $c_n$  is orthogonal to the first  $n - 1$  rows of  $A'$  and has dot product 1 with the last row of  $A'$ . Hence,

$$\|c_n\|_2 = \frac{1}{|\langle w, r^n \rangle|}$$

where  $r^i$  is the  $i$ th row of  $A'$  and  $w$  is the unique unit vector orthogonal to the span of  $\{r^1, \dots, r^{n-1}\}$  (up to to sign). This means that

$$\mathbb{P}(\|A^{-1}y\|_2 \geq t) = \mathbb{P}(\|c_n\|_2 \geq t) = \mathbb{P}(|\langle w, r^n \rangle| \leq 1/t).$$

The last row  $r^n$  of  $A'$  is the sum of the last rows of  $QM$  and  $QUV^T$ . Note that the inner product of  $w$  with the last row of  $QM$  is some fixed parameter; denote it  $r$ . Then  $QU$  is distributed as  $U$  by the rotational invariance of the normal distribution, so the last row of  $QUV^T$  is distributed as  $Vu^n$  where  $u^n \in \mathbb{R}^k$  is a vector of independent Gaussians. Therefore,  $\langle w, r^n \rangle$  is distributed as  $\langle w, Vu^n \rangle + r$ , so it suffices to bound the Levy concentration of  $\langle w, Vu^n \rangle$ . Specifically, we want to show that

$$\sup_{r \in \mathbb{R}} \mathbb{P}(|\langle w, Vu^n \rangle + r| < 1/t) \leq C \left( \frac{1}{x_2 \cdot t} \left( 1 + \frac{x_1 \cdot x_3^{1/2} \cdot \sqrt{nk}}{s_n(M)} \right) + \exp(-x_1^2/4) + (2/\pi)^{k/2} x_2^k + \exp(-c x_3) \right)$$

where the probability is over the realization of  $u^n$  and  $V$ . This readily follows from Lemma 23.  $\blacktriangleleft$

**Proof of Theorem 20.** Let  $s$  be a unit vector chosen uniformly at random from  $\mathcal{S}^{n-1}$ . By Lemma 22, we have

$$\mathbb{P}_{A,s}(\|A^{-1}s\|_2 \geq t/\sqrt{n}) \leq C \left( \frac{1}{x_2 \cdot t/\sqrt{n}} \left( 1 + \frac{x_1 \cdot x_3^{1/2} \cdot \sqrt{nk}}{s_n(M)} \right) + \exp(-x_1^2/4) + (2/\pi)^{k/2} x_2^k + \exp(-c x_3) \right).$$

Now with probability 1, there exists a unique  $y$  such that  $\|A^{-1}y\|_2 = \|A^{-1}\|$ . From Lemma 24, we see that

$$\|A^{-1}s\|_2 \geq \|A^{-1}(y^T s)y\|_2 = |y^T s| \|A^{-1}\|.$$

Therefore we have

$$\begin{aligned} \mathbb{P}_{A,s}(\|A^{-1}s\|_2 \geq t/\sqrt{n}) &\geq \mathbb{P}_{A,s}(\|A^{-1}\| \geq t \text{ and } |s^T y| \geq 1/\sqrt{n}) \\ &= \mathbb{P}(\|A^{-1}\| \geq t) \cdot \mathbb{P}_{A,s}(|s^T y| \geq 1/\sqrt{n} \mid \|A^{-1}\| \geq t). \end{aligned}$$

By the rotational invariance of  $s$ , we have that

$$\mathbb{P}(\|A^{-1}\| \geq t) \cdot \mathbb{P}_{A,s}(|s^T y| \geq 1/\sqrt{n} \mid \|A^{-1}\| \geq t) = \mathbb{P}(\|A^{-1}\| \geq t) \cdot \mathbb{P}(|s^T e_1| \geq 1/\sqrt{n}).$$

From Lemma 25, we have that

$$\mathbb{P}(|s^T e_1| \geq 1/\sqrt{n}) \geq \mathbb{P}(|Z| \geq 1) - 1/n$$

where  $Z \sim \mathcal{N}(0, 1)$ . Altogether, it follows that

$$\begin{aligned} \mathbb{P}(\|A^{-1}\| \geq t) &\leq \frac{\mathbb{P}_{A,s}(\|A^{-1}s\|_2 \geq t/\sqrt{n})}{\mathbb{P}(|Z| \geq 1) - 1/n} \\ &\leq C \left( \frac{1}{x_2 \cdot t/\sqrt{n}} \left( 1 + \frac{x_1 \cdot x_3^{1/2} \cdot \sqrt{nk}}{s_n(M)} \right) + \exp(-x_1^2/4) + (2/\pi)^{k/2} x_2^k + \exp(-c x_3) \right) \end{aligned}$$

for  $n > 10$  as desired.  $\blacktriangleleft$

**► Lemma 23.** Let  $U, V \in \mathbb{R}^{n \times k}$  have independent  $\mathcal{N}(0, 1)$  coordinates. Let  $M \in \mathbb{R}^{n \times n}$  be a matrix with singular values  $s_1 \geq \dots \geq s_n$  and let  $w$  be a vector perpendicular to the first  $n-1$  rows of  $M + UV^T$ . Then for  $x_1 \geq 3\sqrt{\log(2nk)}$ ,  $x_3 \geq nk$ , and  $x_2 \leq 1$ , and all  $t > 0$ ,

$$\sup_{r \in \mathbb{R}} \mathbb{P}(|\langle w, Vu^n \rangle - r| < 1/t) \leq C \left( \frac{1}{x_2 \cdot t} \left( 1 + \frac{x_1 \cdot x_3^{1/2} \cdot \sqrt{nk}}{s_n(M)} \right) + \exp(-x_1^2/4) + (2/\pi)^{k/2} x_2^k + \exp(-c x_3) \right)$$

for some  $C, c > 0$  where  $u^n$  is the last row of  $U$ .

## 40:14 Smoothed Analysis of the Condition Number Under Low-Rank Perturbations

**Proof.** Let  $m^1, \dots, m^n$  be the rows of  $M$  and let  $u^1, \dots, u^n$  be the rows of  $U$ . Then the rows of  $A = M + UV^T$  are given by  $m^i + Vu^i$ . Let  $y$  be the unit vector orthogonal to the span of  $\{m^1, \dots, m^{n-1}\}$ . Consider the following three events:

- $\mathcal{E}_1 =$  event that every entry of  $U$  is at most  $x_1$  in absolute value,
- $\mathcal{E}_2 =$  event that  $\|V^T y\|$  is at least  $x_2$ ,
- $\mathcal{E}_3 =$  event that  $\|V\|^2$  is at most  $x_3$ ,

for  $x_1 \geq 3\sqrt{\log(2nk)}$ ,  $x_3 \geq nk$ , and  $x_2 \leq 1$ . Denote  $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3$ . We now show each of these occurs with high probability. By a standard concentration bound, the maximum of  $nk$  i.i.d. standard Gaussians is strongly concentrated around  $\sqrt{2\log(2nk)}$ . In particular,  $\mathcal{E}_1$  happens with probability at least  $1 - 2\exp(-x_1^2/4)$ . Next, each coordinate of  $V^T y$  is distributed as  $\mathcal{N}(0, 1)$ , and  $\|V^T y\| \geq \|V^T y\|_\infty$ , so we may upper bound  $\mathbb{P}(\mathcal{E}_2^c)$  by  $\mathbb{P}(|g| < x_2)^k$  where  $g$  is  $\mathcal{N}(0, 1)$ . This means  $\mathcal{E}_2$  occurs with probability at least  $1 - (2/\pi)^{k/2} x_2^k$ . Lastly, the event  $\mathcal{E}_3$  happens with probability at least  $1 - \exp(-\Omega(x_3))$  by Lemma 9.

Now fix some realization of  $V$  and  $U$  such that  $\mathcal{E}$  occurs. Suppose for some parameter  $z < \frac{\sqrt{3}}{2} \frac{s_n}{x_1 \cdot \sqrt{nk}}$  that we have  $\|V^T w\| < z$ . We will find a statement which this assumption implies, then take the contrapositive to obtain a lower bound on  $\|V^T w\|$ . From definition of  $w$ , we know

$$\langle w, m^i \rangle + \langle w, Vu^i \rangle = 0$$

for  $1 \leq i \leq n-1$ . We may apply Cauchy-Schwarz to  $\langle w, Vu^i \rangle = \langle V^T w, u^i \rangle$  and use event  $\mathcal{E}_1$  to bound  $\|u^i\|$  and obtain

$$|\langle w, m^i \rangle| \leq \|V^T w\| \cdot \|u^i\| \leq z \cdot x_1 \cdot \sqrt{k}.$$

Decompose  $w = w^\parallel + w^\perp$  where  $w^\parallel$  is in the span of  $m^1, \dots, m^{n-1}$  and  $w^\perp$  is in the orthogonal complement. Write  $w^\parallel = \sum_{i=1}^{n-1} \alpha_i m^i$  for some coefficients  $\alpha_i$ . Then we have

$$\begin{aligned} \|w^\parallel\|^2 &= |\langle w^\parallel, w \rangle| \leq \sum_{i=1}^{n-1} |\alpha_i| \cdot |\langle m^i, w \rangle| \\ &\leq \|\alpha\|_1 \cdot z \cdot x_1 \cdot \sqrt{k} \\ &\leq \|\alpha\| \cdot z \cdot x_1 \cdot \sqrt{nk} \\ &\leq \|w^\parallel\| \frac{z \cdot x_1 \cdot \sqrt{nk}}{s_n} \end{aligned}$$

where  $\alpha$  is the vector of  $\alpha_i$ 's, and the last step follows since  $M$  is non-singular, making  $\alpha$  the unique solution to  $M^T \alpha = w^\parallel$ . Now, note that  $y$  and  $w^\perp$  are parallel, so

$$\|w^\perp\| \|V^T y\| = \|V^T w^\perp\| \leq \|V^T w\| + \|V^T w^\parallel\| \leq z + \|V\| \|w^\parallel\| \leq z \left( 1 + \frac{x_1 \cdot x_3^{1/2} \cdot \sqrt{nk}}{s_n} \right)$$

where the last step follows from  $\mathcal{E}_3$ . From  $z < \frac{\sqrt{3}}{2} \frac{s_n}{x_1 \cdot \sqrt{nk}}$ , we have

$$\|w^\perp\| = \sqrt{1 - \|w^\parallel\|^2} = \sqrt{1 - \left( \frac{z \cdot x_1 \cdot \sqrt{nk}}{s_n} \right)^2} > \frac{1}{2}.$$

Moving  $\|w^\perp\|$  to the right hand side and using  $\mathcal{E}_2$  and the above bound, we arrive at

$$x_2 \leq \|V^T y\| < 2z \left( 1 + x_1 \cdot x_3^{1/2} \cdot \sqrt{nk}/s_n \right).$$



We have thus established the syllogism

$$\|V^T w\| \leq z \leq \frac{\sqrt{3}}{2} \frac{s_n}{x_1 \cdot \sqrt{nk}} \implies \frac{1}{2} \frac{x_2 s_n}{s_n + x_1 \cdot x_3^{1/2} \cdot \sqrt{nk}} < z.$$

By taking the contrapositive and setting  $z = \frac{1}{2} \frac{x_2 s_n}{s_n + x_1 \cdot x_3^{1/2} \cdot \sqrt{nk}}$ , we see that one of the inequalities on the left must fail. It isn't the second one since  $x_2 < \sqrt{3}$ ,  $x_3^{1/2} > 1$ , and  $s_n > 0$ . We therefore conclude our selection of  $z$  lower bounds  $\|V^T w\|$ .

This leads to sufficient anti-concentration. For any fixed vector  $x$ , the distribution of  $\langle x, u^n \rangle$  is the same as  $\mathcal{N}(0, \|x\|^2)$ . So,  $\langle V^T w, u^n \rangle$  for random  $V$  is a mixture of Gaussians, each of which have variance at least  $z^2$  when we condition on  $\mathcal{E}$ . The Levy concentration is thus easily bounded by

$$\sup_{r \in \mathbb{R}} \mathbb{P} \left( |\langle w, V u^n \rangle + r| \leq \frac{1}{t} \mid \mathcal{E} \right) \leq \frac{\sqrt{2/\pi}}{z \cdot t}.$$

The desired bound then follows by incorporating a probability bound on the complement of  $\mathcal{E}$ .  $\blacktriangleleft$

The following lemmas follow easily from basic properties of the SVD and Gaussian random variables so we omit their proof.

► **Lemma 24.** Consider a  $n \times n$  matrix  $M$  and  $u \in \mathcal{S}^{n-1}$  such that  $\|M\| = \|Mu\|_2$ . Then for every  $v \in \mathbb{R}^n$ , we have

$$\|Mv\|_2 \geq |u^T v| \|M\|.$$

► **Lemma 25.** Let  $x$  be a uniformly random vector in  $\mathcal{S}^{n-1}$  and  $Z \sim \mathcal{N}(0, 1)$ . Then for every  $c > 0$ ,

$$\mathbb{P} \left( |x^T e_1| \geq \sqrt{\frac{c}{n}} \right) \geq \mathbb{P}(|Z| \geq \sqrt{c}) - \frac{1}{n}.$$

---

## References

- 1 Thomas J Anastasio, Andrea K Barreiro, and Jared C Bronski. A geometric method for eigenvalue problems with low-rank perturbations. *The Royal Society Publishing.*, 4, September 2017. doi:10.1098/rsos.170390.
- 2 D. Arthur and S. Vassilvitskii. Worst-case and smoothed analysis of the icp algorithm, with an application to the k-means method. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 153–164, 2006.
- 3 David Arthur, Bodo Manthey, and Heiko Röglin. Smoothed analysis of the k-means method. *J. ACM*, 58(5), 2011. doi:10.1145/2027216.2027217.
- 4 Anirban Basak and Mark Rudelson. Invertibility of sparse non-hermitian matrices. *Advances in Mathematics*, 310:426–483, 2017. doi:10.1016/j.aim.2017.02.009.
- 5 Avrim Blum and John Dunagan. Smoothed analysis of the perceptron algorithm for linear programming. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, page 905–914, USA, 2002. Society for Industrial and Applied Mathematics.
- 6 E. J. Candes and Y. Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010. doi:10.1109/JPROC.2009.2035722.
- 7 E. J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006. doi:10.1109/TIT.2005.862083.

- 8 Zhuo Chen and D. Ellis. Speech enhancement by sparse, low-rank, and dictionary spectrogram decomposition. *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–4, 2013.
- 9 Daniel Dadush and Sophie Huiberts. A friendly smoothed analysis of the simplex method. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, page 390–403, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3188745.3188826.
- 10 Minh Dao, Yuanming Suo, Sang Chin, and Trac Tran. Structured sparse representation with low-rank interference. *Conference Record - Asilomar Conference on Signals, Systems and Computers*, 2015:106–110, April 2015. doi:10.1109/ACSSC.2014.7094407.
- 11 Alan Edelman. Eigenvalues and condition numbers of random matrices. *SIAM Journal on Matrix Analysis and Applications*, 9(4):543–560, 1988. doi:10.1137/0609045.
- 12 Friedrich Eisenbrand and Santosh S. Vempala. Geometric random edge. *Math. Program.*, 164(1-2):325–339, 2017. doi:10.1007/s10107-016-1089-0.
- 13 Noam D. Elkies (<https://mathoverflow.net/users/14830/noam-d-elkies>). Smallest non-zero eigenvalue of a (0,1) matrix. MathOverflow. URL:<https://mathoverflow.net/q/157554> (version: 2017-04-13). arXiv:<https://mathoverflow.net/q/157554>.
- 14 V. Klee, G.J. Minty, and WASHINGTON UNIV SEATTLE Dept. of MATHEMATICS. *HOW GOOD IS THE SIMPLEX ALGORITHM*. Defense Technical Information Center, 1970. URL: <https://books.google.com/books?id=R8430AAACAAJ>.
- 15 X. Liu, M. Tanaka, and M. Okutomi. Practical signal-dependent noise parameter estimation from a single noisy image. *IEEE Transactions on Image Processing*, 23(10):4361–4371, 2014. doi:10.1109/TIP.2014.2347204.
- 16 Bodo Manthey and Heiko Raglin. Smoothed analysis: Analysis of algorithms beyond worst case. *it - Information Technology*, 53(6):280–286, 2011. doi:10.1524/itit.2011.0654.
- 17 Bodo Manthey and Rianne Veenstra. Smoothed analysis of the 2-opt heuristic for the tsp: Polynomial bounds for gaussian noise. In Leizhen Cai, Siu-Wing Cheng, and Tak-Wah Lam, editors, *Algorithms and Computation*, pages 579–589, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- 18 Morteza Mardani, Gonzalo Mateos, and G.B. Giannakis. Recovery of low-rank plus compressed sparse matrices with application to unveiling traffic anomalies. *IEEE Transactions on Information Theory*, 59, April 2012.
- 19 Sean O’Rourke, Van Vu, and Ke Wang. Random perturbation of low rank matrices: Improving classical bounds. *Linear Algebra and its Applications*, 540:26–59, 2018. doi:10.1016/j.laa.2017.11.014.
- 20 Richard Peng and Santosh Vempala. Solving sparse linear systems faster than matrix multiplication. In *Proceedings of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA ’21, page 504–521, USA, 2021. Society for Industrial and Applied Mathematics.
- 21 Mark Rudelson and Roman Vershynin. Smallest singular value of a random rectangular matrix. *Communications on Pure and Applied Mathematics*, 62(12):1707–1739, 2009. doi:10.1002/cpa.20294.
- 22 Arvind Sankar, Daniel A. Spielman, and Shang-Hua Teng. Smoothed analysis of the condition numbers and growth factors of matrices. *SIAM Journal on Matrix Analysis and Applications*, 28(2):446–476, 2006. doi:10.1137/S0895479803436202.
- 23 Yoav Seginer. The expected norm of random matrices. *Combinatorics, Probability and Computing*, 9:149–166, March 2000. doi:10.1017/S096354830000420X.
- 24 Daniel A. Spielman and Shang hua Teng. Smoothed analysis: an attempt to explain the behavior of algorithms in practice. *COMMUN. ACM*, 2009.
- 25 Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463, 2004. doi:10.1145/990308.990310.

- 26 Moeller Steen, Sebastian Weingartner, and Mehmet Akcakaya. Multi-scale locally low-rank noise reduction for high-resolution dynamic quantitative cardiac mri. *Annual International Conference of the IEEE Engineering in Medicine and Biology Society.*, 2017:1473–1476, July 2017. doi:10.1109/EMBC.2017.8037113.
- 27 Terence Tao and Van Vu. Smooth analysis of the condition number and the least singular value. *Mathematics of Computation*, 79(272):2333–2352, 2010. URL: <http://www.jstor.org/stable/20779147>.
- 28 Vu Tau. Random matrices: the distribution of the smallest singular values. *Geometric and Functional Analysis*, 20:260–297, March 2010. doi:10.1007/s00039-010-0057-8.
- 29 Shang-Hua Teng. Smoothed analysis of algorithms and heuristics. In Lusheng Wang, editor, *Computing and Combinatorics*, pages 10–11, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- 30 C Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *Int Journal of Computer Vision*, 9:137–154, 1992. doi:10.1007/BF00129684.
- 31 Lloyd N. Trefethen and David Ill. Bau. *Numerical linear algebra*. SIAM Society for Industrial and Applied Mathematics, 2000.
- 32 Van H. Vu and Terence Tao. The condition number of a randomly perturbed matrix. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, page 248–255, New York, NY, USA, 2007. Association for Computing Machinery. doi:10.1145/1250790.1250828.
- 33 Karl Werner and Magnus Jansson. Parameter estimation for reduced-rank multivariate linear regressions in the presence of correlated noise. In *Conference Record of the Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 2101–2105 Vol.2, December 2003.
- 34 Wikipedia contributors. Product distribution — Wikipedia, the free encyclopedia, 2020. [Online; accessed 18-August-2020]. URL: [https://en.wikipedia.org/w/index.php?title=Product\\_distribution&oldid=970273508](https://en.wikipedia.org/w/index.php?title=Product_distribution&oldid=970273508).

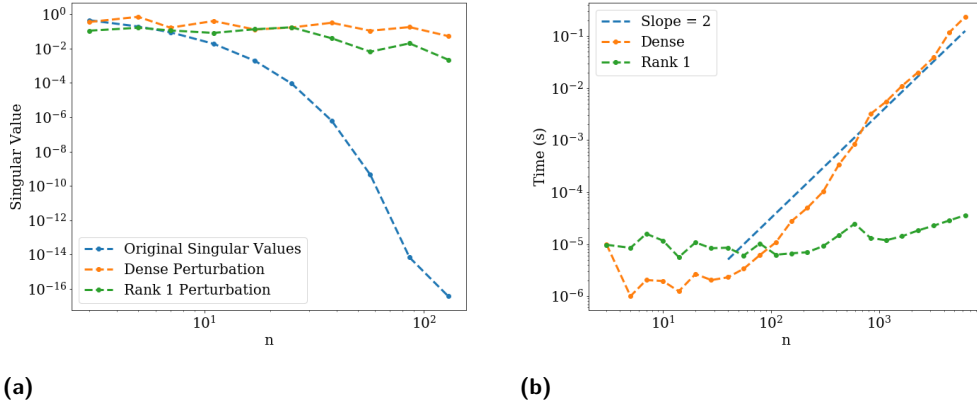
## A Numerical experiments

In this section, we numerically demonstrate our theoretical results by giving an example of a sparse family of  $n$  by  $n$  matrices that are “poorly” conditioned and whose condition number improves significantly after adding a random Gaussian rank 1 perturbation. We show that this perturbation results in an improvement comparable to what is achieved after adding a dense Gaussian matrix while maintaining a low time complexity for matrix vector product operations.

Our family of  $n$  by  $n$  matrices will be constructed as follows:  $M_n$  will have ones on the anti-diagonal and the first and third off-diagonals above the anti-diagonal. For example,  $M_7$  is displayed below.

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

It is shown in [13] that  $M_n$  is ill-conditioned by showing that the magnitude of the smallest eigenvalue of  $M_n$  is of the order  $O(n/C^n)$  where  $C \approx 1.47$  which implies that the smallest singular value of  $M_n$  is also at most  $O(n/C^n)$ . The second smallest eigenvalue on the other hand is a constant.



**Figure 1** (a) Smallest singular values of the original matrix compared against dense and rank 1 perturbations. (b) Time taken to perform a matrix vector product after a dense perturbation and a rank 1 perturbation. The cost for the dense perturbation has a quadratic scaling (slope = 2).

In Figure 1 (a), we show the smallest singular value of  $M_n$  for a range of  $n$  along with the smallest singular values after a dense and rank 1 perturbation. As we can see in the log-log plot, the original values are decaying exponentially while the smallest singular value after the rank 1 perturbation is within a few orders of magnitude of the corresponding value after a dense perturbation. In Figure 1 (b), we show the time taken to perform a matrix vector product after a dense and a rank 1 perturbation. For this task, we used the popular numerical libraries NumPy and SciPy. Since  $M_n$  is sparse, it can be represented in a special sparse format to speed up computations. In the case of a rank 1 perturbation, we only need to store two additional vectors and a matrix vector product  $(M + uv^T)x$  can be performed as

$$(M_n + uv^T)x = M_n x + (v^T x) \cdot u.$$

However, in the case of a dense perturbation, we need to store a dense matrix  $G$  and perform the matrix vector product operation with a vector and a dense matrix resulting in a much slower operation than in the rank 1 case. Indeed, note that the slope of the “Dense” curve in Figure 1 (b) is close to 2 signifying a quadratic increase in time. Overall, we see that in this case, a rank 1 update results in a comparable improvement of the condition number of  $M_n$  while greatly improving the cost to perform a fundamental matrix operation.

## **B** Low-rank noise model for other problems in smoothed analysis

In this section, we outline some of the challenges that arise when applying the rank 1 noise model in other popular problems studied in smoothed analysis. While not a comprehensive survey of all problems, we focus on two of the most studied applications of this framework outside of the condition number. These are the simplex method and  $k$ -means. For these problems, the standard noise model is the dense one where every entry of the input matrix or input set of points respectively, is independently perturbed by a random Gaussian. We highlight some of the challenges that arise when trying to carry out existing proof techniques for these problems using rank 1 noise. This ultimately shows that new ideas are required to bypass the lack of independence as we did for the condition number.

## B.1 Simplex method

The simplex method is one of the most famous applications of the smoothed analysis framework. The goal is to solve a linear program of the form  $\max c^T x$  subject to  $Ax \leq b$  using the simplex method where the entries of  $A \in \mathbb{R}^{m \times n}$  have been perturbed by random noise. Recall that the simplex method operates by moving among the vertices of the polytopes defined by the constrained matrix  $A$ . The geometric operation of moving from one vertex to another is called a pivot operation and the most commonly analyzed pivot operation with respect to smoothed analysis is the shadow vertex pivot method.

Without getting into technical details that will lead us too far afield, we note that the shadow vertex pivoting method requires us to calculate the following bound: let  $a_i$  for  $1 \leq i \leq m$  denote the rows of the matrix  $A$  and let  $W$  be a fixed two dimensional subspace. We wish to bound

$$\mathbb{E}[|\text{edges}(\text{conv}(a_1, \dots, a_m) \cap W)|]$$

where  $\text{conv}(a_1, \dots, a_m)$  is the convex hull of the rows (see [9] for more information).

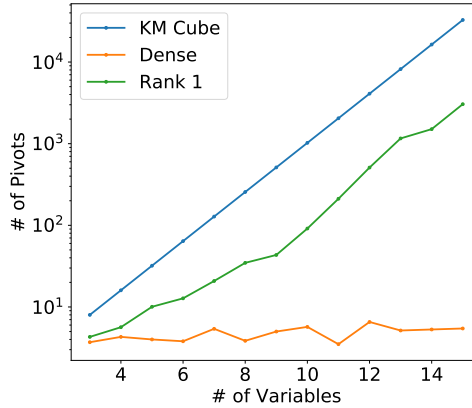
To calculate the above bound, we essentially need to understand the probability that  $a_j^T \theta \leq t$  for a range of values of  $j$  and some  $t \in \mathbb{R}$  (here  $\theta$  represents the normal vector of the line connecting some two points  $a_i, a_k$ . For the pair  $a_i, a_k$  to be on the convex hull, we need the rest of the points to be on one side of the line). In the case that we add independent noise across the rows, this bound is possible to compute due to independence across  $a_j$ . However, in the case that we add rank 1 noise  $u^T v$  (here  $u \in \mathbb{R}^m, v \in \mathbb{R}^n$ ) to  $A$ , these probabilities become intractable using existing methods since  $a_j$  satisfying  $a_j^T \theta \leq t$  gives us information about all other  $a_{j'}$  for  $j' \neq j$  since randomness is shared across the rows.

Nevertheless, it is possible to get a weak result for the smoothed analysis of the simplex method in our low-rank noise model by using a different pivoting operation. It is shown in [12] that if the rows satisfy a certain *geometric* property, then using a random pivoting rule results in an expected polynomial number of steps for the simplex method to converge.

The geometric property is the following: For any  $I \subseteq [m]$ , and  $j \in [m]$ , if  $a_j$  is not in the span generated by  $a_i, i \in I$ , then the distance from  $a_j$  to this span is at least  $\delta$ . We note that the bound on the expected number of steps depends polynomially on  $1/\delta$  and other parameters. This geometric property reduces to a singular value estimate as follows. For simplicity, lets focus on  $j = 1$  and  $I = \{2, \dots, n-1\}$ . As in Section 1.3, it follows that  $\|A_{[n]}^{-1} e_1\|$  is equal to  $1/|w^T a_1|$  where  $w$  is the normal vector of the span of the rows  $a_2, \dots, a_n$ . Thus, if  $s_n(A_{[n]})$  is “not too small” then  $\|A_{[n]}^{-1} e_1\|$  cannot be “too large” and consequently, the distance from  $a_1$  to the span of  $a_2, \dots, a_n$  is “not too small” (we are intentionally leaving our specific relations for a high level overview). The caveat is that we need the geometric property to hold between  $a_1$  and *every* set of  $n-1$  other vectors. However, since the bound of Theorem 13 only gives us an inverse polynomial probability, we cannot afford the union bound of  $\binom{m}{n}$  unless  $m = n + C$  for some constant  $C$ , which is not a realistic scenario.

Lastly, empirical evidence shows that rank 1 perturbation may not be a suitable if the *original* simplex method (the Dantzig simplex method) is used. In Figure 2, we use the Klee-Minty lower bound [14] for the Dantzig simplex method and add either a dense Gaussian or a rank 1 perturbation to the constraint matrix. We then plot the average number of pivot steps taken over twenty independent trials. It can be seen that a rank 1 perturbation only slightly improves over the exponential number of pivot steps required by the Dantzig simplex method whereas dense perturbations help greatly.

We conclude our discussion with a major open problem.



■ **Figure 2** The Dantzig simplex method applied to the Klee-Minty lower bound and its random perturbations.

► **Open Problem 26.** *Is there a pivoting rule for the simplex method that runs in expected polynomial time if we add random rank 1 noise to the constraint matrix?*

## B.2 *k*-means clustering

Recall that in the *k*-means problem, we are given a set  $X$  of  $n$  points in  $\mathbb{R}^d$  and our goal is to partition the points into  $k$  sets  $S_i$  to minimize the objective

$$\sum_{i=1}^k \sum_{x \in S_i} \|x - \mu_i\|^2$$

where  $\mu_i$  is the mean of the points in  $S_i$ . A common heuristic for this problem, also confusingly known as the *k*-means algorithm, or Lloyd’s method, is to randomly pick an initial set of  $k$  centers, assign each point in  $X$  to its closest center, update the means accordingly, and repeat until convergence. In the smoothed analysis framework, it was shown that if each point in  $X$  is perturbed by an independent Gaussian vector then convergence happens in polynomially many steps [3]. The existing analysis all crucially rely on the following geometric lemma.

► **Lemma 27.** *Let  $x \in \mathbb{R}^d$  be drawn according to a  $d$ -dimensional Gaussian distribution of standard deviation  $\sigma$ , and let  $B$  be the  $d$ -dimensional ball of radius  $\varepsilon$  centered at the origin. Then  $\mathbb{P}(x \in B) \leq (\varepsilon/\sigma)^d$ .*

This lemma roughly states that the probability of a random Gaussian being in any ball of radius  $\varepsilon$  is at most  $\varepsilon^d$ , and is used to union bound over exponentially many events in the smoothed analysis of *k*-means.

Surprisingly, this lemma *does not* hold in our “rank 1” setting. More precisely, we can prove the following probabilistic bound which is a major impediment to understanding the smoothed complexity of the *k*-means problem with rank 1 noise.

► **Lemma 28.** *Let  $x \in \mathbb{R}^d$  be drawn according to a standard  $d$ -dimensional Gaussian distribution and let  $y \in \mathbb{R}$  be a scalar standard Gaussian random variable. If  $B$  is the  $d$ -dimensional ball of radius  $\varepsilon$  centered at the origin then  $\mathbb{P}(yx \in B) = O(\varepsilon/\sqrt{d})$ .*

Note that  $yx \in \mathbb{R}^d$ . We are considering random variables of this form because if a rank 1 perturbation was added to  $X$ , then each row is perturbed by a random vector of the form  $yx \in \mathbb{R}^d$ . Lemma 28 roughly states that the probability that the random vector  $yx$  is in any ball of radius  $\varepsilon$  only weakly depends on the dimension  $d$ . In particular, we do not get an exponentially small probability afforded by Lemma 27 that enables us to union bound over exponentially many events as in the arguments for the smoothed analysis of  $k$ -means under the standard noise model.

The intuition for Lemma 28 is as follows. First, note that from standard Gaussian concentration, we have  $\|x\| \approx \sqrt{d}$ . Treating this as fixed for now, this means that  $y\|x\|$  is approximately distributed as a scalar Gaussian distribution with variance  $d$ . Therefore, from Proposition 12, it follows that  $\mathbb{P}(|y\|x\| \leq \varepsilon) = \Theta(\varepsilon/\sqrt{d})$ . We now formalize this argument.

**Proof.** Note that  $\|x\|_2^2$  is a chi-squared variable with  $d$  degrees of freedom. From [34], we know that the density of the product  $Z = \|yx\|_2^2 = y^2\|x\|_2^2$  is given by

$$f_Z(z) \simeq \frac{1}{2^{d/2}\Gamma(d/2)} \int_0^\infty \left(x^{d/2-2}e^{-x/2}\right) \left(\frac{1}{\sqrt{z/x}}e^{-z/(2x)}\right) dx.$$

Therefore,

$$\mathbb{P}(\|yx\|_2^2 \leq \varepsilon^2) \simeq \frac{1}{2^{d/2}\Gamma(d/2)} \int_0^{\varepsilon^2} \int_0^\infty \left(x^{d/2-2}e^{-x/2}\right) \left(\frac{1}{\sqrt{z/x}}e^{-z/(2x)}\right) dx dz.$$

We now switch the order of summation which is valid since the integrand is positive. From the definition of the error function, we can check that

$$\int_0^{\varepsilon^2} \frac{1}{\sqrt{z/x}}e^{-z/(2x)} dz \simeq x \cdot \operatorname{erf}(\varepsilon/\sqrt{x}).$$

We now use the estimate  $\operatorname{erf}(t) \leq 2t$  which holds for all  $t \geq 0$ . This gives us

$$\int_0^\infty x^{d/2-1}e^{-x/2}\operatorname{erf}(\varepsilon/\sqrt{x}) dx \lesssim \varepsilon \int_0^\infty x^{d/2-3/2}e^{-x/2} dx = \varepsilon 2^{d/2-1/2}\Gamma(d/2 - 1/2).$$

Finally, noting that  $\Gamma(d/2 - 1/2)/\Gamma(d/2) \lesssim 1/\sqrt{d}$  gives us our desired probability bound. ◀

Note that the above bound is the best that we can hope for. Indeed, we can say that  $\|x\|_2^2 = \Omega(d)$  with probability  $1/2$  so conditioning on this event, we have that  $\Pr(|y\|x\|_2 \leq \varepsilon) = \Omega(\varepsilon/\sqrt{d})$ . We note that Lemma 27 is also required for the smoothed analysis of other Euclidean problems such as a local search heuristic for Euclidean TSP [17].





# Matroid Intersection: A Pseudo-Deterministic Parallel Reduction from Search to Weighted-Decision

Sumanta Ghosh  

Indian Institute of Technology Bombay, India

Rohit Gurjar  

Indian Institute of Technology Bombay, India

---

## Abstract

We study the matroid intersection problem from the parallel complexity perspective. Given two matroids over the same ground set, the problem asks to decide whether they have a common base and its search version asks to find a common base, if one exists. Another widely studied variant is the weighted decision version where with the two matroids, we are given small weights on the ground set elements and a target weight  $W$ , and the question is to decide whether there is a common base of weight at least  $W$ . From the perspective of parallel complexity, the relation between the search and the decision versions is not well understood. We make a significant progress on this question by giving a pseudo-deterministic parallel (NC) algorithm for the search version that uses an oracle access to the weighted decision.

The notion of pseudo-deterministic NC was recently introduced by Goldwasser and Grossman [19], which is a relaxation of NC. A pseudo-deterministic NC algorithm for a search problem is a randomized NC algorithm that, for a given input, outputs a fixed solution with high probability. In case the given matroids are linearly representable, our result implies a pseudo-deterministic NC algorithm (without the weighted decision oracle). This resolves an open question posed by Anari and Vazirani [2].

**2012 ACM Subject Classification** Mathematics of computing → Matroids and greedoids; Mathematics of computing → Probabilistic algorithms

**Keywords and phrases** Linear Matroid, Matroid Intersection, Parallel Complexity, Pseudo-deterministic NC

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.41

**Category** RANDOM

**Related Version** *Full Version:* <https://eccc.weizmann.ac.il/report/2021/121/>

**Acknowledgements** We thank the anonymous reviewers for pointing towards the relevant literature on lattice families.

## 1 Introduction

Most often, a search problem can be efficiently solved using an oracle for a closely related decision problem. For example, if you have access to a decision oracle that tells you whether a given graph has a perfect matching, you can efficiently construct a perfect matching in a given graph using the decision oracle. Such search-to-decision reductions usually involve self-reducibility and make a linear number of oracle calls sequentially. However such reductions do not fit into the framework of parallel complexity, where one can make multiple oracle calls in parallel, but wants poly-logarithmic time complexity. For a more detailed discussion on the difference in parallel complexity of search and decision problems, see [25].



© Sumanta Ghosh and Rohit Gurjar;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 41; pp. 41:1–41:16

Leibniz International Proceedings in Informatics

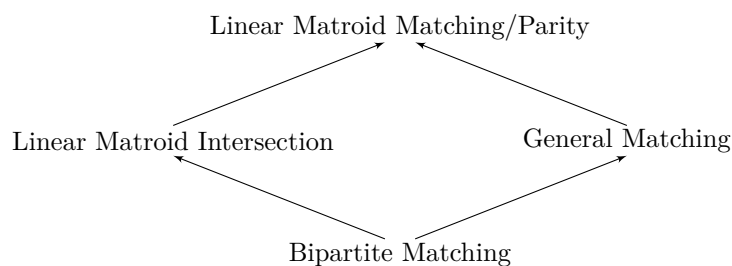


Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Graph matching and related problems like linear matroid intersection and linear matroid matching were one of the first problems to be studied from the parallel complexity perspective [26, 5]. The decision versions of these problems ask to decide the existence of the respective combinatorial substructures:

- Matching: Does a given graph contain a perfect matching – a set of disjoint edges that cover all the vertices in the graph?
- Linear Matroid Intersection: Given two sets of  $m$  vectors each, is there a set of indices  $B \subseteq [m]$  that corresponds to a basis set in each of the two sets?
- Linear Matroid Matching/Parity: Given a set of pairs of vectors, is there a subset of pairs whose union will give a basis for the union of all pairs?

The search versions of these problems ask for constructing the respective combinatorial substructures (if one exists). The matching problem in bipartite graphs is a special case of all the three problems above (see Figure 1). A bipartite graph is a graph whose vertices can be partitioned into two parts such that every edge connects a vertex from one part to one in the other part. Even in the special case of bipartite matching, the questions of the exact parallel complexity of decision and search and whether decision and search are equivalent in a parallel sense still remain unresolved.



■ **Figure 1** Reductions among the four problems.  $A \rightarrow B$  represents that problem  $A$  reduces to problem  $B$ .

The first efficient randomized parallel algorithms for the three decision problems above followed from the results of Lovász [26]. Lovász gave randomized algorithms for these problems by first reducing these decision questions to testing whether the determinant of a certain symbolic matrix is nonzero, as a polynomial. Then he used the fact that the zeroness of a polynomial can be tested efficiently by just evaluating it at a random point [32, 36, 12, 29]. Hence, the questions were basically reduced to computing determinant of a randomly generated matrix. Interestingly, there are efficient parallel (NC) algorithms for computing the determinant of a matrix [3, 7, 4]. An NC algorithm is one which uses polynomially many parallel processors and takes only polylogarithmic time. Thus, the algorithms of Lovász [26] can be viewed as randomized parallel (RNC) algorithms for the three decision problems. However, this did not imply any parallel algorithms for the search versions.

Randomized parallel (RNC) algorithms for the search versions of these problems were obtained some years later [24, 27, 28]. However, these results did not go via a parallel search-to-decision reduction. Instead, they gave randomized parallel (RNC) reductions from the search version to a variant of the decision problem, namely *weighted decision*. For example, the weighted decision version for perfect matchings asks: given a graph with *small* weights on edges and a target weight  $W$ , is there a perfect matching of weight at most  $W$  (or at least  $W$ ). Here the weight of a perfect matching is defined to be the sum of the

weights of the edges in the perfect matching. It turns out that Lovász's RNC algorithms can be appropriately modified to solve the weighted decision versions as well, when the given weights are small. The search-to-weighted-decision reductions together with the weighted decision algorithms implied randomized parallel search algorithms for the three problems. We elaborate a bit on the reductions.

### Reductions from search to weighted-decision

Karp, Upfal and Wigderson [24] do not explicitly talk about weights, but their reduction is from finding a perfect matching to a subroutine that can be viewed as weighted decision with 0-1 weights on the edges. From the perspective of our current investigation, the result of Mulmuley, Vazirani, and Vazirani [27] is much more interesting. They showed that using the weighted decision oracle, one can compute a perfect matching with just two rounds of parallel calls to the oracle. The crucial ingredient in their algorithm was the powerful Isolation Lemma which states that if the edges of a graph are assigned random weights from a polynomially bounded range uniformly and independently then with high probability, there is a *unique* minimum weight perfect matching in the graph. Once we have such a weight assignment, we can first find the minimum weight  $w^*$  of a perfect matching by calling the weighted decision oracle for each possible target value  $W$  in a polynomially bounded range. Then for each edge  $e$  in parallel, delete  $e$  and ask the oracle if there is a perfect matching of weight at most  $w^*$ . The answer will be no if and only if  $e$  is a part of the unique minimum weight perfect matching. Thus, in two rounds of polynomially many parallel oracle calls, we can compute the unique minimum weight perfect matching.

The amazing thing about the Isolation Lemma is that it applies to not just the family of perfect matchings in a graph, but to arbitrary families of subsets. Thus, the above described search-to-weighted-decision reduction of [27] can be made to work for any problem that admits a similar self-reducibility property. Narayan, Saran, and Vazirani [28] used the same Isolation Lemma based reduction to give RNC algorithms for the search versions of linear matroid intersection and linear matroid matching.

### Derandomization

Since the work of Lovász [26], it has been a big open question to derandomize these results i.e., to find deterministic parallel (NC) algorithms for these problems. While derandomization results have been obtained for the matching problem in many special classes of graphs [11, 35, 10, 20, 1], the question remains open even for bipartite graphs. Only recently, there was a significant progress made when a quasi-NC algorithm was obtained for finding a perfect matching in a bipartite graph [16, 15]. A quasi-NC algorithm runs in polylogarithmic time but can use quasipolynomially ( $2^{\log^{O(1)} n}$ ) many parallel processors, so this result brought the problem quite close to the class NC. Similar quasi-NC algorithms were later obtained for linear matroid intersection [22] and matching in general graphs [34] as well.

In the quest of understanding the deterministic parallel complexity of these problems, an interesting question one can ask is whether there is a deterministic parallel (NC) search-to-decision reduction. An easier question would be to ask for an NC reduction from search to weighted-decision, i.e., derandomizing the reductions of [27, 24, 28] described above. Soon after the quasi-NC result for bipartite matching [16], Goldwasser and Grossman [19] started quite an interesting line of enquiry, where they answered the above question positively for bipartite matching. They observed that the quasi-NC algorithm can be modified to give a deterministic parallel (NC) search-to-weighted-decision reduction for bipartite matching. Their main result was, what they call, a pseudo-deterministic NC algorithm for bipartite matching, which followed from this reduction.

### Pseudo-determinism

The notion of pseudo-deterministic algorithms was introduced by Gat and Goldwasser [17] which is applicable only for search problems. For a given instance of a search problem, a randomized algorithm can possibly give different outputs for different choices of the random seed. Pseudo-deterministic algorithms are randomized algorithms which give a fixed output for a given input with high probability. Note that the earlier described RNC algorithm of [27] for matching is not pseudo-deterministic because for a given graph, it will output different perfect matchings for different possibilities of the randomly chosen weight assignment.

It is not hard to see that if one gives a *deterministic* reduction from a search problem to a decision problem that is known to have a randomized algorithm, then one immediately gets a pseudo-deterministic algorithm for the search problem (see [18, Theorem 2.2]). That is why the NC search-to-weighted-reduction for bipartite matching [19] implied a pseudo-deterministic NC algorithm for bipartite matching, i.e., an RNC algorithm that, for a given graph, outputs the same perfect matching with high probability. One interesting implication of this result is that if one finds an NC algorithm for the weighted-decision of bipartite matching, one will get an NC algorithm for the search version as well.

A natural question arises: can we similarly modify the quasi-NC algorithms for linear matroid intersection [22] and matching in general graphs [34] into NC search-to-weighted-decision reductions, and thus, get pseudo-deterministic NC algorithms for the search versions? It looks quite possible because one can extract out an abstract framework from [19] for converting these quasi-NC algorithms into pseudo-deterministic NC algorithms. But as we discuss below, a straightforward application of this framework does not work out for linear matroid intersection or matching in general graphs. A key step in [19] is to *compute a succinct description* of the set of all (possibly exponentially many) minimum weight perfect matchings in a weighted bipartite graph in NC, given the weighted-decision oracle. However, it is not immediately clear how to solve the analogous question in NC for linear matroid intersection or matching in general graphs. Interestingly, in an earlier work in a different context, Cygan, Gabow, and Sankowski [9] had already solved this question for matching in general graphs. They had designed a procedure based on LP duality to compute a succinct description of the set of all minimum weight perfect matchings, via the weighted-decision oracle. Moreover, as observed in [30], this procedure can also be parallelized using standard techniques. Armed with this heavy hammer, Anari and Vazirani [2] give an NC search-to-weighted-decision reduction, and thus, get a pseudo-deterministic NC algorithm for perfect matching in general graphs. Anari and Vazirani [2] put it as an open question to obtain similar results for linear matroid intersection. In this work, we take up this challenge.

### Our contributions

In the setting of linear matroid intersection, the analogue of a perfect matching is referred as a *common base* – a set of indices that corresponds to a basis in both the sets of vectors. For the weighted version, it is well understood how to succinctly describe the set of minimum or maximum weight common bases, i.e., the minimizing/maximizing face of the common base polytope; see e.g., [31, Chapter 41]. Any face of the common base polytope is characterized by its tight sets. Suppose that  $M_1$  and  $M_2$  are two matroids over the same ground set  $E$ . Then, a subset  $S$  of  $E$  is called a tight set for a maximizing face (of the common base polytope), if for some matroid  $M_i$  the following holds: for every maximum weight common base  $B$ , the set  $S \cap B$  spans the set  $S$ . Note that the number of tight sets of a maximizing face can be exponentially large. However, they are known to have succinct representations. We give a randomized NC algorithm to compute a succinct and unique representation for the tight sets of a maximizing face.

► **Theorem 1** (Informal version of Theorem 5). *There exists a randomized NC algorithm to compute a succinct and unique description for the tight sets of a maximizing face of the common base polytope, given the weighted-decision oracle.*

For a maximizing face of the common base polytope, all the tight sets for some matroid  $M_i$  forms a lattice family, and our description for tight sets is motivated by the succinct representation of lattice families based on the partial order of its prime subsets (also known as irreducible subsets). We construct a digraph in bottom-up fashion, using bases from the maximizing face of the common base polytope, such that it contains the necessary information regarding the tight sets of maximizing face. From this digraph we shall be able to compute the succinct description. See Section 3.1 for more details. Here, we would like to mention that the succinct representation of lattice families using the partial order of its prime subsets is well known and has been used in multiple previous algorithms [31, Chapter 49], [23, 14, 6]. However, all these applications do not fall in the category of parallel computation.

Note that the uniqueness of the description is important because then this RNC algorithm is by default pseudo-deterministic, as there is only one possible output. Once we have designed this heavy hammer, it is relatively easier to combine the procedure of [22] with the abstract framework provided by [19] and obtain a pseudo-deterministic NC search-to-weighted-decision reduction. This leads to our first main result.

► **Theorem 2.** *The search version of the linear matroid intersection problem has a pseudo-deterministic NC algorithm.*

### General Matroid Intersection

Our main technical contributions are applicable to not just linear matroid intersection but also to matroid intersection. In the general matroid intersection problem, instead of two sets of vectors, we are given two matroids on the same ground set and the goal is to find a set of elements that forms a base in each of the two matroids. In this problem, the matroids are not given explicitly but only via a independence or rank oracle. Thus, it does not makes sense to talk about NC or RNC algorithms for this problem. One can however consider a parallel oracle model where we can make polynomially many queries to the oracle in parallel (see [25]). To the best of our knowledge, there is no such parallel algorithm known for the decision or the search version of matroid intersection, even with sub-linear number of rounds of parallel oracle calls. This makes the question all the more interesting whether decision and search are equivalent in a parallel sense.

Interestingly, the search-to-weighted-decision reduction of [28] applies to general matroid intersection as well and can be said to be in RNC. Our results make a significant progress on this question by giving a pseudo-deterministic NC reduction from search to weighted decision. Formally, we can show the following.

► **Theorem 3.** *There is a pseudo-deterministic NC algorithm for finding a common base of two matroids  $M_1$  and  $M_2$  on the same ground set  $E$ , provided that the algorithm has an oracle access to the following decision question: given two matroids with polynomially bound (in  $|E|$ ) weights on the ground set elements and a target weight  $W$ , is there a common base of weight at least  $W$ ? Furthermore, the oracle calls need to be made only for the following pairs of matroids:  $\langle M_1, M_2 \rangle$ ,  $\langle M_1, M_1 \rangle$ , and  $\langle M_2, M_2 \rangle$ .*

Note that in the above theorem, as there is no explicit input, the ground set size is taken as the input size.

## Discussion

There are many natural open questions that are highlighted by our work. The big question is whether there is an NC algorithm for linear matroid intersection. Going to the more general setting, is there some kind of parallel algorithm for matroid intersection? Another question which can generate some new ideas is whether there is an NC reduction from search to decision for linear matroid intersection. For general matroid intersection, it would be interesting to find a parallel search to decision reduction even with the use of randomization.

The third question mentioned in the beginning, that is, linear matroid matching is completely open, in the sense that not even a quasi-NC algorithm is known for it. Given the wide applicability of the Isolation Lemma, the randomized parallel search-to-weighted-decision reduction of Mulmuley, Vazirani, and Vazirani [27] would work for any combinatorial problem with an appropriate self-reducibility property, including NP-hard problems like maximum independent set. An intriguing meta-question is – what is the most general setting where we can find deterministic or pseudo-deterministic parallel search-to-weighted-decision reductions.

## 2 Previous works

We start by briefly describing the techniques of previous works [28, 22, 19] that will be helpful in both comprehending as well as describing our work. Wherever these works talk about a minimization problem, we will describe it in terms of maximization, just for convenience. We will be using the following notations for the two versions of the matroid intersection problem.

- **search-MI**: Given two matroids on a common ground set, compute a common base.
- **weighted-decision-MI**: Given two matroids on the same ground set, polynomially bounded weights on the ground set elements, and a target weight  $W$ , is there a common base of weight at least  $W$ ?

Whenever we are in a setting where the matroids are not given explicitly, we will consider the ground set size as the input size.

The result of Narayanan, Saran, and Vazirani [28] can be interpreted as an RNC reduction from search-MI to weighted-decision-MI. The first step of this reduction is to assign weights to the ground set elements, randomly and independently from a small range. Then from the Isolation Lemma [27], one can say that there is a unique maximum weight common base of the two matroids, with high probability. Here, the *weight of a common base* is defined to be sum of the weights of the elements in the common base. We can first find the maximum weight  $w^*$  of a common base by calling the weighted-decision-MI oracle for each possible target value  $W$  in a small range. Then for each ground set element  $e$  in parallel, increase its weight by one and find out the new maximum weight. The maximum weight increases if and only if  $e$  is a part of the unique maximum weight common base. This way we can find the unique maximum weight common base.

Note that the uniqueness property is crucial for this construction and that is the only place where randomness is needed. And this construction is not pseudo-deterministic because for different choices of random weights, we will get a different maximum weight common base. There has been several efforts to *deterministically* construct a weight assignment in NC that *isolates* a common base, i.e., ensures unique maximum weight common base, but this goal has not been achieved till now. A recent work [22] came quite close to this goal and constructed an isolating weight assignment in quasi-NC. This work generalizes the ideas used to do the same for bipartite matching in [16]. We build on their ideas to construct an isolating weight assignment in pseudo-deterministic NC. We first give a brief description of their result.

### Isolating a common base in quasi-NC

Suppose that  $M_1 = (E, \mathcal{I}_1)$  and  $M_2 = (E, \mathcal{I}_2)$  are two matroids over the same ground set  $E$  where  $\mathcal{B}_1$  and  $\mathcal{B}_2$  are the family of bases of  $M_1$  and  $M_2$ , respectively. Let  $m = |E|$  and  $r_1$  and  $r_2$  be the rank functions of the matroids. The main idea of [22] is to isolate a common base in  $\log m$  rounds, where in each round they significantly reduce the set of maximum weight common bases, and finally bring it down to just one maximum weight common base. In each of these rounds, they deterministically propose a set of  $\text{poly}(m)$  weight assignments, one of which will do the desired reduction in the set of maximum weight common bases. In a round, they have no way of figuring out which one out of these  $\text{poly}(m)$  weight assignments will do the job. So, they have to try all  $\text{poly}(m)^{\log m}$  combinations of these weight assignments. Moreover, for any particular combination, they have to combine the  $\log m$  weight assignments on different scales, which means their weights become as large as  $\text{poly}(m)^{\log m}$ . Due to these two factors, their construction is in quasi-NC and not in NC.

To measure the progress in each round, they need a succinct way to describe the current set of maximum weight common bases. The most convenient way to understand the set of maximum weight common bases is through the *common base polytope*. The common base polytope  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$  is a polytope formed by taking convex hull of the 0-1 indicator vectors of the sets in  $\mathcal{B}_1 \cap \mathcal{B}_2$ . For any weight assignment  $\mathbf{w} \in \mathbb{R}^E$ , the weight of a common base  $B$  is defined as a linear function, and thus, one can obtain the maximum weight common bases by maximizing the function  $\sum_{e \in E} \mathbf{w}_e \mathbf{x}_e$  over  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$ . In particular, the set of maximum weight common bases will always be the set of corners of a face of  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$ .

Edmonds [13] gave a nice description of  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$  using the rank functions  $r_1$  and  $r_2$ . He showed that a point  $\mathbf{x} \in \mathbb{R}^E$  is in  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$  if and only if it satisfies the following constraints:

$$\begin{aligned} \mathbf{x}_e &\geq 0 \quad \forall e \in E, & (1) \\ \mathbf{x}(S) = \sum_{e \in S} \mathbf{x}_e &\leq r_i(S) \quad \forall S \subset E, i = 1, 2, & (2) \\ \mathbf{x}(E) = \sum_{e \in E} \mathbf{x}_e &= r_1(E) = r_2(E). & (3) \end{aligned}$$

The construction in [22] crucially uses the description of the common base polytope  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$ . In terms of the polytope, their construction of the weight assignment is such that in each round, the maximum weight face of  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$  gets significantly smaller and after  $\log m$  rounds, the maximum weight face is simply a corner point. The key notions they introduced to measure the improvement in each iteration are *cycles* with respect to a face and their *circulations* with respect to a weight assignment.

Suppose that  $F$  is a face of  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$ . A subset  $S$  of  $E$  is called a *tight set* of  $M_i$  with respect to  $F$  if the corresponding inequality in (2) is tight for  $F$  i.e, for all  $\mathbf{x} \in F$ ,  $\mathbf{x}(S) = r_i(S)$ . Then [22] showed that for every face  $F$ , we have two partitions of  $E$ , denoted by  $\text{partition}_1[F]$  and  $\text{partition}_2[F]$ , such that every tight set of  $M_i$  with respect to  $F$  is a union of the sets from  $\text{partition}_i[F]$ . The partitions of  $E$  naturally induce a bipartite graph, denoted by  $\mathcal{G}[F]$ , with the left vertex set  $\text{partition}_1[F]$ , the right vertex set  $\text{partition}_2[F]$  and the edge set  $E$ : the edge corresponding to an element  $e \in E$  is incident on the vertex corresponding to a set  $v \in \text{partition}_i[F]$  if and only if  $e \in v$ . A sequence of distinct elements  $(e_1, \dots, e_k)$  from  $E$  is called a *cycle* with respect to  $F$  if it forms a cycle in the graph  $\mathcal{G}[F]$ .

Let  $\mathcal{C}_F$  denotes the set of cycles with respect to a face  $F$  of  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$ . Then [22] showed that for face  $F$ , if  $\mathcal{C}_F = \emptyset$  then  $F$  is a corner point of the polytope  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$ . Their idea was to keep eliminating cycles via appropriate modification of the weight assignment and

get smaller and smaller maximizing face of  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$  to eventually reach a corner point. For a weight assignment  $\mathbf{w}$  on  $E$ , define the *circulation* for a (even length) cycle as the absolute value of the difference of weights in the two sets of alternating edges. Let  $C$  be a cycle, say with respect to  $F = P(\mathcal{B}_1 \cap \mathcal{B}_2)$ , and let  $\mathbf{w}$  be a weight assignment such that the circulation of  $C$  is non-zero w.r.t.  $\mathbf{w}$ . Then they showed that the cycle  $C$  does not appear in the maximizing face with respect to  $\mathbf{w}$ . Now if the weight assignment  $\mathbf{w}$  gives non-zero circulation to *all* the cycles in  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$ , then all the cycles in the maximizing face  $F$  will be eliminated, i.e.  $\mathcal{C}_F = \emptyset$ , and  $F$  will be a corner. However, with polynomially bounded weights, one cannot expect to give nonzero circulation to all the cycles at once, since the number of cycles can be exponentially large.

One of the key ideas in [22, 16] was to eliminate the cycles in rounds. In each round, they double the length of the eliminated cycles and reach to face of a smaller dimension. Thus, in  $\log m$  rounds, one can eliminate all the cycles and reach a corner point of  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$ . They used the fact that if in a graph all the cycles have length greater than  $2^i$ , then there are at most  $m^4$  many cycles of length at most  $2^{i+1}$  [33]. This implies that, at each iteration, we have to give nonzero circulation to at most  $m^4$  many cycles. Using a hashing technique (for example see [16, Lemma 2.3]), one can give nonzero circulation for each of these  $m^4$  many cycles. Formally, Gurjar and Thierauf [21, Lemma 3.11 and 3.12] showed the following property for faces  $F$  (of  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$ ) having no cycle of length  $\leq r$ .

► **Lemma 4.** *Let  $F$  be a face of the polytope  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$  such that  $\mathcal{C}_F$  has no cycle of length  $r$ . Then one can construct a set of  $O(m^6)$  many weight assignments with weights bounded by  $O(m^6)$  in NC such that one of the weight assignment will give nonzero circulation to all the cycles in  $\mathcal{C}_F$  of length at most  $2r$ .*

Now, as described earlier, we consider all possible combinations of weight assignments from different rounds to get a family of  $\text{poly}(m^{\log m})$  many weight assignments with weights bounded by  $\text{poly}(m^{\log m})$  such that for any two matroids on a ground set of size  $m$ , at least one weight assignment isolates a common base.

In this paper, we give a *pseudo-deterministic NC* reduction from search-MI to weighted-decision-MI. This line of work was started by Goldwasser and Grossman [19]. One can extract an abstract framework from [19] with the following two steps to get a pseudo-deterministic NC search-to-weighted-decision reduction: 1) Like [16, 22], an iterative approach of designing an isolating weight assignment family, 2) Succinct representation of the maximum weight faces of the underlying polytope with an RNC algorithm to compute it, assuming the oracle access to the weighted decision. For example, a face of the bipartite matching polytope is completely described by the set edges that participate in some perfect matching in that face, and [19] gives an NC algorithm to compute it using the respective weighted decision oracle.

The faces of the perfect matching polytope for general graphs are more complicated than their bipartite counterpart. Here, any face is described by a maximal laminar family of tight odd cuts. The work of [8, 30] give an NC procedure, with the oracle access to the weight decision problem, to compute a maximal laminar family of tight odd cuts. This result supplies the second ingredient of the [19] framework, which helped [2] give an NC reduction from search to weighted decision for general perfect matching.

Our reduction also follows the abstract framework of [19]. We use the iterative approach developed by [22]. On top of that, we need an RNC algorithm (using the oracle access to weighted-decision-MI) to compute a succinct representation for a maximum weight face of the common base polytope  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$ . However, none of the previous ideas help to answer this question, and we need something completely new.



### 3 Proof techniques

In this section, we briefly describe the proof ideas of our results. Our proofs strongly rely on some structural properties of lattice families over finite sets. Therefore, we briefly discuss the necessary notations and facts about lattice families. For a finite set  $E$ , a family of subsets  $\mathcal{L}$  of  $E$  is called a *lattice family* over  $E$  if it is closed under set union and intersection and for every element  $a \in E$  there exists a set in  $\mathcal{L}$  containing  $a$ . For every element  $a \in E$  there exists a *unique* smallest set in  $\mathcal{L}$  containing  $a$ . Such sets are called as *prime sets* of  $\mathcal{L}$ . All the sets in a lattice family can be written as a union of its prime sets. Every lattice family  $\mathcal{L}$  over  $E$  induces a *unique partition*  $\mathcal{P}$  of  $E$  such that every set in  $\mathcal{L}$  is a disjoint union of sets in  $\mathcal{P}$ . Moreover, the sets in  $\mathcal{P}$  can be written as a sequence  $(S_1, \dots, S_\ell)$  with the following property: for all  $k \in [\ell]$ ,  $\cup_{j=1}^k S_j$  is in  $\mathcal{L}$ . A family  $\mathcal{L}' \subseteq \mathcal{L}$  is called a *sublattice* of  $\mathcal{L}$ , if  $\mathcal{L}'$  is also a lattice family over  $E$ . The partition  $\mathcal{P}$  is a refinement of the partition  $\mathcal{P}'$  induced by  $\mathcal{L}'$ , that is for all  $S \in \mathcal{P}'$ , the sets in  $\mathcal{P}$  having a nonempty intersection with  $S$  form a partition of  $S$ . For proof of these properties, one can see Section 4.2 of the full version.

#### 3.1 Proof Idea of Theorem 1

We discuss a succinct representation for the maximum weight face of the common base polytope and an RNC algorithm to compute it. First, we define some notations. Suppose that  $M_1 = (E, \mathcal{I}_1)$  and  $M_2 = (E, \mathcal{I}_2)$  are two matroids with  $\mathcal{B}_1$  and  $\mathcal{B}_2$  as their family of the bases and  $r_1$  and  $r_2$  as the rank functions, respectively. Let  $m = |E|$ . Let  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$  be the common base polytope of  $M_1$  and  $M_2$  defined by the equations (1), (2), (3), and  $F$  be a face of  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$ . Then a subset  $S$  of  $E$  is called a *tight set* for  $M_i$  (with respect to  $F$ ) if for all  $\mathbf{x} \in F$ ,  $\mathbf{x}(S) = r_i(S)$ . For all  $i \in [2]$ , let  $\text{tight-sets}_i[F]$  denote the family of all tight sets for  $M_i$  with respect to the face  $F$ . Edmonds [13] showed that for all  $i \in [2]$ ,  $\text{tight-sets}_i[F]$  forms a lattice family over  $E$ .

Suppose that  $\mathbf{w}$  is a weight assignment on  $E$ . Let  $F_{\mathbf{w}}$  be the maximizing face of the common base polytope  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$ , with respect to  $\mathbf{w}$ . The face  $F_{\mathbf{w}}$  can be uniquely represented by  $\text{tight-sets}_1[F_{\mathbf{w}}]$  and  $\text{tight-sets}_2[F_{\mathbf{w}}]$ . However, we can not compute them explicitly with our limited computational resources since the size of each family can be exponentially large. On the other hand, since  $\text{tight-sets}_i[F_{\mathbf{w}}]$  is a lattice family over  $E$ , each  $\text{tight-sets}_i[F_{\mathbf{w}}]$  has a succinct representation using partial order defined on its prime sets. More specifically, one can define a pre-order  $\preceq_i$  (that is, reflexive and transitive) on  $E$  as follows: for all  $a, b \in E$ ,  $a \preceq_i b$  if and only if in  $\text{tight-sets}_i[F_{\mathbf{w}}]$ , the prime set containing  $b$  is a subset of the prime set containing  $a$ . The pre-order  $\preceq_i$  gives a succinct representation of  $\text{tight-sets}_i[F_{\mathbf{w}}]$ , that is for every  $S \subseteq E$ ,  $S$  is in  $\text{tight-sets}_i[F_{\mathbf{w}}]$  if and only if  $S$  is transitively closed under  $\preceq_i$ . Such succinct representation for lattice families is well known (see [31, Chapter 49]<sup>1</sup>). For any  $a \in E$ , the transitive closure of  $a$  in  $\preceq_i$  is same as the prime set in  $\text{tight-sets}_i[F_{\mathbf{w}}]$  containing  $a$ . Also, the collection of all maximal subsets of  $E$  which are symmetric under  $\preceq_i$  is same as the partition  $E$  induced by  $\text{tight-sets}_i[F_{\mathbf{w}}]$ . If one consider the digraph representation of  $\prec_i$ , (that is  $(a, b)$  is an edge if and only if  $a \prec_i b$ ) then in  $\text{tight-sets}_i[F_{\mathbf{w}}]$ , the prime set containing  $a$  is same the set of vertices reachable from  $a$  in the digraph and the partition of  $E$  induced by  $\text{tight-sets}_i[F_{\mathbf{w}}]$  is same as the set of strongly connected components. Thus, the prime sets of  $\text{tight-sets}_i[F_{\mathbf{w}}]$  contain all the information

<sup>1</sup> Our definition of  $\preceq_i$  is exactly opposite to the definition used [31, Chapter 49], that is according to their definition,  $a \preceq_i b$  if and only if the prime set containing  $a$  is a subset of the prime set containing  $b$ .

regarding it. In our context, we compute the following succinct objects related to  $F_{\mathbf{w}}$ :  $\text{prime-sets}_i[F_{\mathbf{w}}]$  and  $\text{partition}_i[F_{\mathbf{w}}]$  for all  $i \in [2]$ , where  $\text{prime-sets}_i[F]$  be the set of all primes sets of the lattice family  $\text{tight-sets}_i[F_{\mathbf{w}}]$  and  $\text{partition}_i[F_{\mathbf{w}}]$  denote the partition of  $E$  induced by  $\text{tight-sets}_i[F]$ . Recall from [22, Section 3.3] that the cycles of the bipartite graph induced by  $\text{partition}_1[F_{\mathbf{w}}]$  and  $\text{partition}_2[F_{\mathbf{w}}]$  define the cycles with respect to the face  $F_{\mathbf{w}}$ . And, the tight constraints coming from sets in  $\text{prime-sets}_i[F_{\mathbf{w}}]$  serve as a basis for all the tight constraints from  $\text{tight-sets}_i[F_{\mathbf{w}}]$ . Here, we would like to mention that basis forming families of tight sets are well studied (see [31]). However, to best of our knowledge, no efficient parallel algorithm is known to compute them. Also, the succinct representation of lattices using the partial order of its prime sets has been widely used to design algorithms for different optimization problems. For example, computing optimal stable matching [23], problems in computational geometry [14, 6], submodular function minimization [31, Chapter 49]. However, the context of these applications are very different from parallel computation.

With the above two objects, we also need the following characterization: for all  $i \in [2]$ , there exists a function  $N_i^{F_{\mathbf{w}}}$  from  $\text{partition}_i[F_{\mathbf{w}}]$  to  $\mathbb{Z}_{\geq 0}$  such that a base  $B \in \mathcal{B}_1 \cap \mathcal{B}_2$  is in the face  $F_{\mathbf{w}}$  if and only if for all  $i \in [2]$  and  $S \in \text{partition}_i[F_{\mathbf{w}}]$ , we have  $|S \cap B| = N_i^{F_{\mathbf{w}}}(S)$ . Here, we would like to mention that both the notions of partition and the function  $N_i^{F_{\mathbf{w}}}$  and the criteria we just mentioned were already introduced in [22], but were a bit weaker in the following ways: Our criteria is an exact characterization, however, they showed it for one direction. Our partition has an additional ‘‘chain property’’ ensured by the structural properties of the lattice families. All these additional points will be useful in our proofs. For details see Section 4.5 of the full version and [22, Section 3.2].

Now we briefly discuss about our RNC algorithm to compute  $\text{prime-sets}_i[F_{\mathbf{w}}]$  and  $\text{partition}_i[F_{\mathbf{w}}]$  for all  $i \in [2]$ . One important point is that our algorithm is equipped with the oracle access to **weighted-decision-MI**. Our idea is the following: We first compute a random vertex, equivalently a random base,  $B$  in the face  $F_{\mathbf{w}}$ . The base  $B$  can be computed in RNC using the oracle access to **weighted-decision-MI**. Then iteratively construct a chain of subsets of bases from  $F_{\mathbf{w}}$

$$\{B\} = \mathcal{B}_0 \subseteq \mathcal{B}_1 \subseteq \dots \subseteq \mathcal{B}_\ell$$

such that the minimal face containing  $\mathcal{B}_\ell$  is same as  $F_{\mathbf{w}}$  and  $\ell = \lceil \log m \rceil$ . Next we briefly discuss how to construct the set  $\mathcal{B}_j$  from  $\mathcal{B}_{j-1}$  and compute  $\text{prime-sets}_i[F_{\mathbf{w}}]$  and  $\text{partition}_i[F_{\mathbf{w}}]$  from the set of common bases  $\mathcal{B}_\ell$ .

For all  $j \in \{0, \dots, \ell\}$ , let  $F_j$  denotes the minimal face containing  $\mathcal{B}_j$ . For all  $j \in [\ell]$ , the set  $\mathcal{B}_j$  contains the elements in  $\mathcal{B}_{j-1}$  with the following extra elements: For all  $i \in [2]$ ,  $A \in \text{partition}_i[F_{j-1}]$ , we add a common base  $B_{ij}^{(A)}$  (if it exists) from the face  $F_{\mathbf{w}}$  with the property

$$|A \cap B_{ij}^{(A)}| \neq N_i^{F_{j-1}}(A). \quad (4)$$

We know that for all  $i \in [2]$   $A \in \text{partition}_i[F_{j-1}]$ , every base in  $F_{j-1}$  contains exactly  $N_i^{F_{j-1}}(A)$  many elements from  $A$ . However, our property on  $B_{ij}^{(A)}$  says that we want a base from  $F_{\mathbf{w}}$  which violates that condition, and if exists, we can compute such a base in RNC using the oracle access to **weighted-decision-MI**. Next, we discuss how to compute  $\text{partition}_i[F_j]$  in NC. Note that, after computing  $\text{partition}_i[F_j]$ ,  $N_i^{F_j}$  can be computed in NC by computing  $|B \cap A|$ , for some  $B \in \mathcal{B}_j$ , in parallel for all  $A \in \text{partition}_i[F_j]$ .

The set families  $\text{tight-sets}_i[F_j]$  for all  $i \in [2]$  form lattice families over  $E$ , and given  $B_j$ , we are interested to compute  $\text{prime-sets}_i[F_j]$  and  $\text{partition}_i[F_j]$  in NC. As we mentioned earlier, every lattice family has a digraph representation based on the partial order on primes

sets of lattice family. Given this digraph representation of  $\text{tight-sets}_i[F_j]$ , one can compute  $\text{prime-sets}_i[F_j]$  and  $\text{partition}_i[F_j]$  in NC. However, given  $\mathcal{B}_j$ , it is not clear how to construct the digraph representation of the lattice family  $\text{tight-sets}_i[F_j]$  in NC. We show that, instead of this digraph, it would sufficient for us if we work with a subgraph  $G_i[\mathcal{B}_j]$  defined as follows: the vertex set is same as the ground set  $E$  and for all  $a, b \in E$ ,  $(a, b)$  is an edge of  $G_i[\mathcal{B}_j]$  if and only if

*there exists a base  $B \in \mathcal{B}_j$  such that  $b \in B$  and  $(B \setminus \{b\}) \cup \{a\}$  is also a base of  $M_i$ .*

More specifically, we prove that for every  $a \in E$  the prime set in  $\text{tight-sets}_i[F_j]$  containing  $a$  is same as the set of vertices reachable from  $a$  in  $G_i[F_j]$  and  $\text{partition}_i[F_j]$  is same as the set of strongly connected components in  $G_i[\mathcal{B}_j]$ . Using this characterization, we can compute  $\text{prime-sets}_i[F_j]$  and  $\text{partition}_i[F_j]$  in NC, given the graph  $G_i[\mathcal{B}_j]$ . For more details see Section 6 of the full version. Also, using the weighted decision oracle we can compute  $G_i[\mathcal{B}_j]$  in NC. Thus, given  $\mathcal{B}_j$ ,  $\text{prime-sets}_i[F_j]$  and  $\text{partition}_i[F_j]$  are computable in NC. Here, we would like to mention that constructing directed graphs using base exchange property is a well known technique in matroid literature and has been used in various contexts. For example, one can see the augmenting path based algorithm for matroid intersection in [31, Section 41.2], and some other context in [31, Section 40.3]. The definition of  $G_i[\mathcal{B}_j]$  is very close to the definition used in the second example.

At very high level, this part of our algorithm is doing exactly the opposite of the idea used to construct isolating weight assignment family in [16, 22, 34]. They start from a face of the polytope and iteratively move to the subfaces of smaller dimensions until a corner point is reached. On the other hand, we are starting from a corner point of the face and iteratively reaching bigger faces until we cover the whole face.

Now we give a very brief overview of the correctness of our algorithm. For all  $j \in \{0, 1, \dots, \ell\}$ , since  $F_j$  is a subface of  $F_{\mathbf{w}}$ ,  $\text{tight-sets}_i[F_{\mathbf{w}}]$  is a sublattice of  $\text{tight-sets}_i[F_j]$  for all  $i \in [2]$ . Therefore  $\text{partition}_i[F_j]$  is a refinement of  $\text{partition}_i[F_{\mathbf{w}}]$ , that is for all  $S \in \text{partition}_i[F_{\mathbf{w}}]$ , the sets in  $\text{partition}_i[F_j]$  having nonempty intersection with  $S$  create a partition of  $S$ . Let  $\mathcal{W}_{ij}^{(S)}$  denote the family of sets in  $\text{partition}_i[F_j]$  which have nonempty intersection with  $S \in \text{partition}_i[F_{\mathbf{w}}]$ . As we move from  $(j-1)$ th iteration to  $j$ th iteration, our algorithm satisfies the following property: either the size of the smallest sets in  $\mathcal{W}_{ij}^{(S)}$  satisfying the equation 4 becomes double, or if no such set exists in  $\mathcal{W}_{ij}^{(S)}$ , it becomes equal to  $\{S\}$ . Thus, after  $\ell$ th iteration,  $\text{partition}_i[F_\ell]$  becomes equal to  $\text{partition}_i[F_{\mathbf{w}}]$  for all  $i \in [2]$ . This leads us to prove that  $F_\ell = F_{\mathbf{w}}$ . Therefore,  $\text{prime-sets}_i[F_\ell]$  is also same as  $\text{prime-sets}_i[F_{\mathbf{w}}]$ . In Algorithm 1, we describe all the steps to compute  $\text{prime-sets}_i[F_{\mathbf{w}}]$  and  $\text{partition}_i[F_{\mathbf{w}}]$  for all  $i \in [2]$ . For the detail analysis of the correctness and time complexity of our algorithm see Section 7 of the full version. From the above discussion, we can conclude that

► **Theorem 5.** *Let  $M_1 = (E, \mathcal{I}_1)$  and  $M_2 = (E, \mathcal{I}_2)$  be two matroids with  $\mathcal{B}_1$  and  $\mathcal{B}_2$  be the family of bases, respectively. Let  $\mathbf{w}$  be a weight assignment on  $E$  with polynomially bounded weights, and  $F_{\mathbf{w}}$  be the maximizing face of  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$  with respect to  $\mathbf{w}$ . Then, given  $M_1$ ,  $M_2$  and  $\mathbf{w}$  as inputs, Algorithm 1 computes  $\text{prime-sets}_i[F_{\mathbf{w}}]$  and  $\text{partition}_i[F_{\mathbf{w}}]$  for all  $i \in [2]$  in randomized NC, provided that the algorithm has an oracle access to  $\text{weighted-decision-MI}$ . Furthermore, for all positive integer  $c$ , the success probability of the algorithm can be made at least  $1 - \frac{1}{m^c}$ , where  $m = |E|$ .*

■ **Algorithm 1** Computing prime sets and partitions corresponding to a max-weight face.

**Input:** Two matroids  $M_1 = (E, \mathcal{I}_1)$ ,  $M_2 = (E, \mathcal{I}_2)$ , and a weight assignment  $\mathbf{w} : E \rightarrow \mathbb{Z}_{\geq 0}$ .

**Output:**  $\text{prime-sets}_i[F_{\mathbf{w}}]$  and  $\text{partition}_i[F_{\mathbf{w}}]$  for all  $i \in [2]$ , where  $F_{\mathbf{w}}$  denotes the max-weight face.

**Assumption:** Oracle access to weighted-decision-MI.

```

1: Compute a base  $B$  in  $F_{\mathbf{w}}$ .
2:  $\mathcal{B}_0 \leftarrow \{B\}$ .
3: for all  $i \in [2]$  do in parallel
4:   Compute the graph  $G_i[\mathcal{B}_0]$ .
5:   Let  $F_0$  be the minimal face containing  $\mathcal{B}_0$ .
6:   Compute  $\text{prime-sets}_i[F_0]$ ,  $\text{partition}_i[F_0]$  and  $N_i^{F_0}$ .
7: end for
8: for  $j \leftarrow 1$  to  $\lceil \log m \rceil$  do
9:    $\mathcal{B}_j \leftarrow \mathcal{B}_{j-1}$ .
10:  for all  $i \in [2]$  do in parallel
11:    for all  $A \in \text{partition}_i[F_{j-1}]$  do in parallel
12:      If exists, compute a base  $B_{ij}^{(A)}$  in  $F_{\mathbf{w}}$  such that
      
$$|A \cap B_{ij}^{(A)}| \neq N_i^{F_{j-1}}(A).$$

13:       $\mathcal{B}_j \leftarrow \mathcal{B}_j \cup \{B_{ij}^{(A)}\}$ .
14:    end for
15:  end for
16:  for all  $i \in [2]$  do in parallel
17:    Let  $F_j$  be the minimal face containing  $\mathcal{B}_j$ .
18:    Compute the graph  $G_i[\mathcal{B}_j]$ .
19:    Compute  $\text{prime-sets}_i[F_j]$ ,  $\text{partition}_i[F_j]$  and  $N_i^{F_j}$ .
20:  end for
21: end for
22: return  $\text{prime-sets}_i[F_{\ell}]$  and  $\text{partition}_i[F_{\ell}]$  for  $i \in [2]$  and  $\ell = \lceil \log m \rceil$ .

```

### 3.2 Proof idea of Theorem 3

In this section, we give a proof overview of Theorem 3, which states that there is a pseudo-deterministic NC algorithm for the matroid intersection search problem that uses the weighted-decision oracle. Since the weighted-decision for *linear* matroid intersection can be solved in RNC [28], we get a pseudo-deterministic NC algorithm for the search version of linear matroid intersection, that is, Theorem 2.

Suppose that  $M_1 = (E, \mathcal{I}_1)$  and  $M_2 = (E, \mathcal{I}_2)$  are two matroids with  $\mathcal{B}_1$  and  $\mathcal{B}_2$  as the family of bases, respectively. Let  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$  be the common base polytope of  $M_1$  and  $M_2$ . Let  $\mathbf{w}_0$  be a weight assignment defined as  $\mathbf{w}_0(a) = 1$  for all  $a \in E$ . Then the maximizing face of  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$  with respect to  $\mathbf{w}_0$  is the polytope itself. Let  $m = |E|$  and  $\ell = \lceil \log m \rceil$ . Now our idea is the following: We start from the weight assignment  $\mathbf{w}_0$  and inductively construct a sequence of weight assignments

$$\mathbf{w}_0, \mathbf{w}_1, \dots, \mathbf{w}_{\ell}$$

such that for all  $j \in \{0, 1, \dots, \ell\}$ , the weights in  $\mathbf{w}_j$  are bounded by  $O(m)$  and the length of the shortest cycle with respect to the face  $F_j$  is greater than  $2^j$  where  $F_j$  denotes the

maximizing face with respect to  $\mathbf{w}_j$ . Therefore, the face  $F_\ell$  does not have any cycle, and from [22], it has a unique base. Now using the oracle access to **weighted-decision-MI** the base in  $F_\ell$  can be computed in NC. Next we discuss how to construct  $\mathbf{w}_j$  from  $\mathbf{w}_{j-1}$ .

For all  $j \in \{0, 1, \dots, \ell\}$ , let  $\mathcal{C}_{F_j}$  denotes the set of all cycles with respect to the face  $F_j$ . From the induction hypothesis, for some  $j$ , all the cycles in  $\mathcal{C}_{F_j}$  have length greater than  $2^j$ . Then from [22], there are at most  $m^4$  many cycles of length at most  $2^{j+1}$ . Let  $\mathcal{W}$  be a polynomially large family of weight assignments with polynomially bounded weights such that one of the weight assignments in  $\mathcal{W}$  gives nonzero circulation to all the cycles in  $\mathcal{C}_{F_j}$  of length at most  $2^{j+1}$ . There are well known NC constructions of such a family  $\mathcal{W}$  (see e.g., [16, Lemma 2.3]). For each  $\mathbf{w} \in \mathcal{W}$  we do the following in parallel: combine  $\mathbf{w}_j$  and  $\mathbf{w}$  in decreasing order of precedence. Let  $\mathbf{w}'$  be the combined weight and  $F_{\mathbf{w}'}$  is the maximizing face with respect to it. Now using our RNC algorithm discussed in the previous section, compute  $\text{prime-sets}_i[F_{\mathbf{w}'}]$  and  $\text{partition}_i[F_{\mathbf{w}'}]$  for all  $i \in [2]$ . Now, construct the bipartite graph  $\mathcal{G}[F_{\mathbf{w}'}]$  from  $\text{partition}_1[F_{\mathbf{w}'}]$  and  $\text{partition}_2[F_{\mathbf{w}'}]$  as defined in the description of [22]. The length of the shortest cycles in  $\mathcal{G}[F_{\mathbf{w}'}]$  can be computed in NC. Thus, in NC, we can compute the lexicographically smallest weight assignment  $\mathbf{w} \in \mathcal{W}$  such that the length of the shortest cycles in  $\mathcal{G}[F_{\mathbf{w}'}]$  is greater than  $2^{j+1}$ .

■ **Algorithm 2** Pseudo-deterministic NC algorithm for computing a common base of two matroids.

**Input:** Two matroids  $M_1 = (E, \mathcal{I}_1)$  and  $M_2 = (E, \mathcal{I}_2)$ .

**Output:** A common base of  $M_1$  and  $M_2$ , if exists.

**Assumption:** Oracle access to **weighted-decision-MI**.

```

1:  $\mathbf{w}_0 \leftarrow \mathbf{1}$ .
2: for  $j \leftarrow 1$  to  $\lceil \log m \rceil$  do
3:   Compute a family of weight assignments  $\mathcal{W}$  as promised by Lemma 4.
4:   for all  $\mathbf{w} \in \mathcal{W}$  do in parallel
5:     Combine  $\mathbf{w}_{j-1}$  and  $\mathbf{w}$  with descending order in precedence.
6:     For a  $\mathbf{w} \in \mathcal{W}$ , let  $\mathbf{w}'$  be the combined weight.
7:     Let  $F_{\mathbf{w}'}$  be the maximizing face of  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$  with respect to  $\mathbf{w}'$ .
8:     For all  $i \in [2]$ , compute  $\text{prime-sets}_i[F_{\mathbf{w}'}]$  and  $\text{partition}_i[F_{\mathbf{w}'}]$  using Algorithm 1.
9:     Let  $\mathcal{G}[F_{\mathbf{w}'}]$  be the bipartite graph induced by  $\text{partition}_1[F_{\mathbf{w}'}]$  and  $\text{partition}_2[F_{\mathbf{w}'}]$ .
10:    Compute the length of shortest cycle of  $\mathcal{G}[F_{\mathbf{w}'}]$ .
11:   end for
12:   Take some fixed ordering on  $\mathcal{W}$ , like lexicographic ordering.
13:   Take the smallest  $\mathbf{w}$  such that the length of the shortest cycle in  $\mathcal{G}[F_{\mathbf{w}'}] > 2^j$ .
14:    $\mathbf{w}_j \leftarrow \sum_{i=1}^2 \sum_{S \in \text{prime-sets}_i[F_{\mathbf{w}'}]} \mathbf{1}_S$ .
15: end for
16: Compute the unique common base maximizing  $\mathbf{w}_{\lceil \log m \rceil}$  and output.
```

Next we show how to compute  $\mathbf{w}_{j+1}$  from  $\mathbf{w}'$  such that weights in  $\mathbf{w}_{j+1}$  are bounded by  $O(m)$ . Define  $\mathbf{w}_{j+1}$  as the following:

$$\mathbf{w}_{j+1} = \sum_{i=1}^2 \sum_{S \in \text{prime-sets}_i[F_{\mathbf{w}'}}} \mathbf{1}_S,$$

where  $\mathbf{1}_S \in \mathbb{R}^E$  denotes the indicator vector for the set  $S$ . From the definition, it is clear that weights are bounded by  $2m$ , and can be computed in NC from  $\text{prime-sets}_1[F_{\mathbf{w}'}]$  and  $\text{prime-sets}_2[F_{\mathbf{w}'}]$ . Using the description of  $P(\mathcal{B}_1 \cap \mathcal{B}_2)$ , we can show that every point  $\mathbf{x}$  in

the maximizing face  $F_{j+1}$  must satisfy  $\mathbf{x}(S) = r_i(S)$  for all  $i \in [2]$ ,  $S \in \text{prime-sets}_i[F_{\mathbf{w}'}]$ . This implies that  $\text{prime-sets}_i[F_{\mathbf{w}'}]$  is a subset of  $\text{tight-sets}_i[F_{j+1}]$ . Thus  $\text{tight-sets}_i[F_{\mathbf{w}'}]$  is a subset of  $\text{tight-sets}_i[F_{j+1}]$  since all the sets in a lattice family can be written as a union of its prime sets. This helps us to show that  $F_{\mathbf{w}'}$  is same as  $F_{j+1}$ . Also, one can verify that each step of our algorithm as has a unique answer, therefore it is pseudo-deterministic. In Algorithm 2, we describe the steps of our pseudo-deterministic NC reduction from search-MI to weighted-decision-MI. For the proof of correctness and time complexity analysis of our algorithm, see Section 8 of the full version.

---

## References

- 1 Manindra Agrawal, Thanh Minh Hoang, and Thomas Thierauf. The polynomially bounded perfect matching problem is in  $\text{NC}^2$ . In *24th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 4393 of *Lecture Notes in Computer Science*, pages 489–499. Springer Berlin Heidelberg, 2007. doi:10.1007/978-3-540-70918-3\_42.
- 2 Nima Anari and Vijay V. Vazirani. Matching is as easy as the decision problem, in the NC 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 54:1–54:25. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ITCS.2020.54.
- 3 Stuart J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18(3):147–150, 1984.
- 4 Allan Borodin, Stephen Cook, and Nicholas Pippenger. Parallel computation for well-endowed rings and space-bounded probabilistic machines. *Information and Control*, 58(1-3):113–136, July 1984.
- 5 Allan Borodin, Joachim von zur Gathen, and John Hopcroft. Fast parallel matrix and GCD computations. *Information and Control*, 52(3):241–256, 1982.
- 6 Kevin Buchin, David Eppstein, Maarten Löffler, Martin Nöllenburg, and Rodrigo I. Silveira. Adjacency-preserving spatial treemaps. *J. Comput. Geom.*, 7(1):100–122, 2016. doi:10.20382/jocg.v7i1a6.
- 7 Laszlo Csanky. Fast parallel matrix inversion algorithms. *SIAM Journal on Computing*, 5(4):618–623, 1976. doi:10.1137/0205040.
- 8 Marek Cygan, Harold N. Gabow, and Piotr Sankowski. Algorithmic applications of baurstrassen’s theorem: Shortest cycles, diameter and matchings. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 531–540. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.72.
- 9 Marek Cygan, Harold N. Gabow, and Piotr Sankowski. Algorithmic applications of baurstrassen’s theorem: Shortest cycles, diameter, and matchings. *J. ACM*, 62(4), 2015. doi:10.1145/2736283.
- 10 Elias Dahlhaus and Marek Karpinski. Matching and multidimensional matching in chordal and strongly chordal graphs. *Discrete Applied Mathematics*, 84(1–3):79–91, 1998.
- 11 Samir Datta, Raghav Kulkarni, and Sambuddha Roy. Deterministically isolating a perfect matching in bipartite planar graphs. *Theory of Computing Systems*, 47:737–757, 2010. doi:10.1007/s00224-009-9204-8.
- 12 Richard A. Demillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, 1978.
- 13 Jack Edmonds. Submodular functions, matroids, and certain polyhedra. In *Combinatorial Structures and Their Applications, Gordon and Breach, New York*, pages 69–87, 1970.
- 14 David Eppstein, Elena Mumford, Bettina Speckmann, and Kevin Verbeek. Area-universal and constrained rectangular layouts. *SIAM J. Comput.*, 41(3):537–564, 2012. doi:10.1137/110834032.

- 15 Stephen Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite perfect matching is in quasi-NC. *SIAM Journal on Computing*, 0(0):STOC16–218–STOC16–235, 2019. doi:10.1137/16M1097870.
- 16 Stephen A. Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite perfect matching is in quasi-nc. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA*, pages 754–763, 2016.
- 17 Eran Gat and Shafi Goldwasser. Probabilistic search algorithms with unique answers and their cryptographic applications. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:136, 2011. URL: <http://eccc.hpi-web.de/report/2011/136>.
- 18 Oded Goldreich, Shafi Goldwasser, and Dana Ron. On the possibilities and limitations of pseudodeterministic algorithms. In Robert D. Kleinberg, editor, *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, pages 127–138. ACM, 2013. doi:10.1145/2422436.2422453.
- 19 Shafi Goldwasser and Ofer Grossman. Bipartite perfect matching in pseudo-deterministic NC. 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 87:1–87:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.87.
- 20 Dima Grigoriev and Marek Karpinski. The matching problem for bipartite graphs with polynomially bounded permanents is in NC (extended abstract). In *28th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 166–172, 1987. doi:10.1109/SFCS.1987.56.
- 21 Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Identity testing for constant-width, and any-order, read-once oblivious arithmetic branching programs. *Theory of Computing*, 13(2):1–21, 2017.
- 22 Rohit Gurjar and Thomas Thierauf. Linear matroid intersection is in quasi-nc. In *49th Annual ACM Symposium on Theory of Computing*, pages 821–830, 2017.
- 23 Robert W. Irving, Paul Leather, and Dan Gusfield. An efficient algorithm for the "optimal" stable marriage. *J. ACM*, 34(3):532–543, 1987. doi:10.1145/28869.28871.
- 24 Richard M. Karp, Eli Upfal, and Avi Wigderson. Constructing a perfect matching is in random NC. *Combinatorica*, 6(1):35–48, 1986.
- 25 Richard M. Karp, Eli Upfal, and Avi Wigderson. The complexity of parallel search. *Journal of Computer and System Sciences*, 36(2):225–253, 1988. doi:10.1016/0022-0000(88)90027-X.
- 26 László Lovász. On determinants, matchings, and random algorithms. In *FCT*, volume 79, pages 565–574, 1979.
- 27 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7:105–113, 1987. doi:10.1007/BF02579206.
- 28 H. Narayanan, Huzur Saran, and Vijay V. Vazirani. Randomized parallel algorithms for matroid union and intersection, with applications to arborescences and edge-disjoint spanning trees. *SIAM J. Comput.*, 23(2):387–397, 1994. doi:10.1137/S0097539791195245.
- 29 Øystein Ore. Über höhere Kongruenzen. *Norsk Mat. Forenings Skrifter Ser. I*, 7(15):27, 1922.
- 30 Piotr Sankowski. NC algorithms for weighted planar perfect matching and related problems. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 97:1–97:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.97.
- 31 Alexander Schrijver. *Combinatorial optimization : polyhedra and efficiency. Vol. B. , Matroids, trees, stable sets. chapters 39-69*. Algorithms and combinatorics. Springer-Verlag, Berlin, Heidelberg, New York, N.Y., et al., 2003. URL: <http://opac.inria.fr/record=b1124843>.
- 32 J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.
- 33 Ashok Subramanian. A polynomial bound on the number of light cycles in an undirected graph. *Information Processing Letters*, 53(4):173–176, 1995. doi:10.1016/0020-0190(94)00202-A.

## 41:16 Pseudo-Deterministic NC Reduction from search-MI to weighted-decision-MI

- 34 Ola Svensson and Jakub Tarnawski. The matching problem in general graphs is in quasi-nc. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 696–707, 2017. doi:10.1109/FOCS.2017.70.
- 35 Raghunath Tewari and N. V. Vinodchandran. Green’s theorem and isolation in planar graphs. *Information and Computation*, 215:1–7, 2012. doi:10.1016/j.ic.2012.03.002.
- 36 Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation, EUROSAM '79*, pages 216–226, 1979.



# On the Probabilistic Degree of an $n$ -Variate Boolean Function

Srikanth Srinivasan<sup>1</sup> ✉ 🏠

Department of Computer Science, Aarhus University, Aarhus, Denmark

S. Venkitesh ✉ 🏠

Department of Mathematics, IIT Bombay, Mumbai, India

---

## Abstract

Nisan and Szegedy (CC 1994) showed that any Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  that depends on all its input variables, when represented as a real-valued multivariate polynomial  $P(x_1, \dots, x_n)$ , has degree at least  $\log n - O(\log \log n)$ . This was improved to a tight  $(\log n - O(1))$  bound by Chiarelli, Hatami and Saks (Combinatorica 2020). Similar statements are also known for other Boolean function complexity measures such as Sensitivity (Simon (FCT 1983)), Quantum query complexity, and Approximate degree (Ambainis and de Wolf (CC 2014)).

In this paper, we address this question for *Probabilistic degree*. The function  $f$  has probabilistic degree at most  $d$  if there is a random real-valued polynomial of degree at most  $d$  that agrees with  $f$  at each input with high probability. Our understanding of this complexity measure is significantly weaker than those above: for instance, we do not even know the probabilistic degree of the OR function, the best-known bounds put it between  $(\log n)^{1/2-o(1)}$  and  $O(\log n)$  (Beigel, Reingold, Spielman (STOC 1991); Tarui (TCS 1993); Harsha, Srinivasan (RSA 2019)).

Here we can give a near-optimal understanding of the probabilistic degree of  $n$ -variate functions  $f$ , *modulo* our lack of understanding of the probabilistic degree of OR. We show that if the probabilistic degree of OR is  $(\log n)^c$ , then the minimum possible probabilistic degree of such an  $f$  is at least  $(\log n)^{c/(c+1)-o(1)}$ , and we show this is tight up to  $(\log n)^{o(1)}$  factors.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Probabilistic computation; Theory of computation  $\rightarrow$  Circuit complexity; Theory of computation  $\rightarrow$  Complexity classes; Theory of computation  $\rightarrow$  Pseudorandomness and derandomization

**Keywords and phrases** truly  $n$ -variate, Boolean function, probabilistic polynomial, probabilistic degree

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.42

**Category** RANDOM

**Funding** *Srikanth Srinivasan*: Supported by a startup grant from Aarhus University.

*S. Venkitesh*: Supported by the Senior Research Fellowship of the Human Resource Development Group, Council of Scientific and Industrial Research, Government of India.

## 1 Introduction

### 1.1 Background and motivation

Representing Boolean functions  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  by polynomials is a tried-and-tested technique that has found uses in many areas of Theoretical Computer Science. In particular, such representations have led to important results in Complexity theory [8, 9], Learning theory [19, 11], and Algorithm Design [29].

---

<sup>1</sup> On leave from Department of Mathematics, IIT Bombay.



© Srikanth Srinivasan and S. Venkitesh;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 42; pp. 42:1–42:20



Leibniz International Proceedings in Informatics

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

There are many different kinds of polynomial representations that are useful in various applications. The most straightforward way to represent a Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  by a polynomial is by finding a  $P \in \mathbb{R}[x_1, \dots, x_n]^2$  such that  $P(a) = f(a)$  for all  $a \in \{0, 1\}^n$ . It is a standard fact (say by Möbius Inversion or polynomial interpolation) that any  $f$  has such a representation<sup>3</sup> has degree at most  $n$ , and the smallest degree of such a  $P$  is called the *degree of  $f$*  (or sometimes the *Fourier degree of  $f$*  because of its close relation to the Fourier spectrum of  $f$  [22]), and denoted  $\deg(f)$ .

The degree of  $f$  is an important notion of *complexity* of the function  $f$  and is closely related to a slew of combinatorial measures of Boolean function complexity such as *Sensitivity*, *Decision Tree complexity*, *Quantum Query complexity*, etc. (see, e.g., the survey of Buhrman and de Wolf [10] for a nice introduction). Given a complexity measure  $\mu(\cdot)$  (such as  $\deg(\cdot)$ ) on Boolean functions, a natural question to ask is the following.

► **Question 1.** *How small can  $\mu(f)$  be for a function  $f$  on  $n$  variables?*

To make this question interesting, one must exclude trivial functions like the constant functions, and more generally, functions that depend on just a small subset of their input variables. This brings us to the following definition.

► **Definition 2** (Truly  $n$ -variate Boolean function). *We say that a Boolean function  $f(x_1, \dots, x_n)$  depends on its input variable  $x_i$ , or equivalently that  $x_i$  is influential for  $f$ , if there is an input  $a$  such that flipping the value of the  $i$ th variable at  $a$  changes the value of  $f$  (in this case, we also say that  $x_i$  is influential for  $f$  at  $a$ ). We say that a Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  is truly  $n$ -variate<sup>4</sup> if it depends on all its  $n$  variables.*

A number of results have addressed questions regarding how small complexity measures can be for truly  $n$ -variate Boolean functions.

1. Motivated by problems in Learning theory and PRAM lower bounds, Nisan and Szegedy [21] showed that any truly  $n$ -variate function has degree at least  $\log n - O(\log \log n)$ . Recently, this was improved to  $\log n - O(1)$  by Chiarelli, Hatami and Saks [13]. There are standard examples of Boolean functions (see, e.g., the *Addressing function* defined below) for which this is tight.
2. Ambainis and de Wolf [4] studied the same question for the *approximate degree of  $f$* , which is defined to be the minimum degree of a polynomial  $P$  such that  $|P(a) - f(a)| < 1/3$  for all  $a \in \{0, 1\}^n$ . This complexity measure is closely related to the *quantum query complexity of  $f$*  [10]. Ambainis and de Wolf [4] showed that any truly  $n$ -variate function has approximate degree (and also quantum query complexity)  $\Omega(\log n / \log \log n)$ . They also constructed variants of the Addressing function for which this bound is tight up to constant factors.
3. Such results are also known for more combinatorial complexity measures, such as the *sensitivity* of a Boolean function  $f$ , which is defined as follows. The sensitivity of  $f$  at a point  $a \in \{0, 1\}^n$  is the number of input variables to  $f$  that are influential for  $f$  at  $a$ . The sensitivity of  $f$  is the maximum sensitivity of  $f$  at any input. Simon [25] showed that any truly  $n$ -variate  $f$  has sensitivity at least  $\log n - O(\log \log n)$ . This is also tight up to the  $O(\log \log n)$  additive term (say, for the Addressing function).

<sup>2</sup> We can represent  $f$  as a polynomial over any field, but in this paper, we will work over the reals.

<sup>3</sup> The representation is in fact *unique* if we restrict  $P$  to be *multilinear*, i.e. that no variable has degree more than 1 in  $f$ .

<sup>4</sup> Such functions are also called *non-degenerate* Boolean functions in the literature [25].

We address Question 1 for another well-known polynomial-degree measure called the *Probabilistic degree*. We define this notion first.

► **Definition 3** (Probabilistic polynomial and Probabilistic degree). *Given a Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  and an  $\varepsilon \geq 0$ , an  $\varepsilon$ -error probabilistic polynomial for  $f$  is a random polynomial  $\mathbf{P}$  (with some distribution having finite support) over  $\mathbb{R}[x_1, \dots, x_n]$ <sup>5</sup> such that for each  $a \in \{0, 1\}^n$ ,*

$$\Pr_{\mathbf{P}}[\mathbf{P}(a) \neq f(a)] \leq \varepsilon.$$

(Note that  $\mathbf{P}(a)$  need not be Boolean when  $\mathbf{P}(a) \neq f(a)$ .)

We say that the degree of  $\mathbf{P}$ , denoted  $\deg(\mathbf{P})$ , is at most  $d$  if the probability distribution defining  $\mathbf{P}$  is supported on polynomials of degree at most  $d$ . Finally, we define the  $\varepsilon$ -error probabilistic degree of  $f$ , denoted  $\text{pdeg}_{\varepsilon}(f)$ , to be the least  $d$  such that  $f$  has an  $\varepsilon$ -error probabilistic polynomial of degree at most  $d$ .

In the special case  $\varepsilon = 1/3$ , we omit the subscript in the notation and simply use  $\text{pdeg}(f)$ .

The probabilistic degree is a fundamentally important and well-studied complexity measure of Boolean functions. It was implicitly introduced (in the finite field setting) in a celebrated result of Razborov [23], who showed how to use it to construct low-degree polynomial approximations to small-depth circuits, and hence prove strong circuit lower bounds. The real-valued version was first studied by Beigel, Reingold and Spielman [7] and Tarui [27] who were motivated by other circuit lower bound questions and oracle separations. This measure has since found other applications in complexity theory [5, 8], Pseudorandom generator constructions [9], Learning theory [11], and Algorithm design [29, 1]. Further, in many of these applications (e.g. [5, 9, 1]) we *need* real-valued approximations.

Despite this, however, our understanding of probabilistic degree is much less developed than the other measures above. For instance, near-optimal lower bounds of  $n^{1-o(1)}$  on the probabilistic degree of an explicit Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  were proved only recently by Viola [28], and are only known for a function in the complexity class  $\text{E}^{\text{NP}}$ ; in comparison, the Parity function has degree and approximate degree  $n$ , which is the largest possible. Another example is the OR function on  $n$  variables. It is trivial to estimate the degree of OR (which is  $n$ ) and well-known that its approximate degree is  $\Theta(\sqrt{n})$  [21, 15]. However, its probabilistic degree (over the reals) remains unknown: the best known upper bound is  $O(\log n)$  due to independent results of Beigel et al. [7] and Tarui [27], while the best lower bound is  $(\log n)^{1/2-o(1)}$  due to Harsha and the first author [16]. This indicates that we need better tools to understand probabilistic degree in general and over the reals in particular. This is one of the motivations behind this paper.

Another motivation is to understand the contrast between the setting of real-valued probabilistic polynomials and polynomials over constant-sized finite fields. At a high level, this helps us understand the contrast between circuit complexity classes  $\text{AC}^0$  and  $\text{AC}^0[p]$ , as the former class of circuits has low-degree probabilistic polynomials over the reals [7, 27], while the latter does not [26]. It is easy to show that there are truly  $n$ -variate Boolean functions of constant degree over finite fields (e.g., the parity function is a linear polynomial over the field  $\mathbb{F}_2$ ). It is interesting to ask to what extent such phenomena fail over the reals.

A final motivating reason is to understand more precisely the relationships between probabilistic degree and other complexity measures such as approximate degree. A recent conjecture of Golovnev, Kulikov and Williams [14] shows that porting results for approximate

<sup>5</sup> This can also be defined over other fields.

degree to probabilistic degree would have interesting consequences for De Morgan formula lower bounds. By proving results such as the one in this paper, we hope to be able to prove such connections and hopefully uncover others.

With these motivations in mind, we address Question 1 in the setting of Probabilistic degree. That is, what is the lowest possible probabilistic degree of a truly  $n$ -variate Boolean function? As far as we know, this question has not been addressed before. Putting together Simon's bound on the sensitivity of a truly  $n$ -variate function with known probabilistic degree lower bounds [16], one can show a lower bound of  $(\log \log n)^{1/2-o(1)}$ . This is quite far from the best known upper bounds of  $O(\log n)$ , which hold for say the OR function [7, 27] and the Addressing function defined below in Section 1.3.

## 1.2 Results

Our aim is to prove a result characterizing the minimum possible probabilistic degree of a truly  $n$ -variate Boolean function. However, the gap even just in our understanding of the OR function (as mentioned above) tells us that this may not yet be within reach. What we are able to do is to give a near-complete characterization modulo the gap between known upper and lower bounds for  $\text{pdeg}(\text{OR})$ . *Moreover, the answer is non-trivial: it is not simply  $\text{pdeg}(\text{OR})$ .*

More precisely, our results are the following. Below,  $\text{OR}_n$  denotes the OR function on  $n$  variables. We assume that we have bounds of the form  $\text{pdeg}(\text{OR}_n) = (\log n)^{c \pm o(1)}$  for some  $c > 0$ .

► **Theorem 4.** *Assume that  $\text{pdeg}(\text{OR}_n) \geq (\log n)^{c-o(1)}$  for some  $c > 0$  and all large enough  $n \in \mathbb{N}$ . Then, any truly  $n$ -variate Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  satisfies  $\text{pdeg}(f) \geq (\log n)^{c/(c+1)-o(1)}$ .*

► **Theorem 5.** *Assume that  $\text{pdeg}(\text{OR}_n) \leq (\log n)^{c+o(1)}$  for some  $c > 0$  and all large enough  $n \in \mathbb{N}$ . Then, there exists a truly  $n$ -variate Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  such that  $\text{pdeg}(f) \leq (\log n)^{c/(c+1)+o(1)}$ .*

Thus, we get close-to-matching lower and upper bounds for truly  $n$ -variate Boolean functions assuming close-to-matching lower and upper bounds for the OR function. However, the above statements also imply *unconditional* lower and upper bounds on the probabilistic degrees of truly  $n$ -variate Boolean functions. Using known results that yield  $(\log n)^{(1/2)-o(1)} \leq \text{pdeg}(\text{OR}_n) \leq O(\log n)$  [7, 27, 16], we get

► **Corollary 6.** *Any truly  $n$ -variate Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  satisfies  $\text{pdeg}(f) \geq (\log n)^{(1/3)-o(1)}$ .*

► **Corollary 7.** *There exists a truly  $n$ -variate Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  such that  $\text{pdeg}(f) \leq (\log n)^{(1/2)+o(1)}$ .*

► **Remark 8.** The reader may wonder why we assume lower and upper bounds of the form  $(\log n)^{c \pm o(1)}$  for  $\text{pdeg}(\text{OR}_n)$ . This is because the gaps between the known upper and lower bounds are  $(\log n)^{\Omega(1)}$ , and so it makes sense to use a characterization that shrinks this gap to something relatively insignificant. Furthermore, the best known lower bound on  $\text{pdeg}(\text{OR}_n)$  is of the form  $(\log n)^{1/2-o(1)}$  [16] (more precisely, it is  $\Omega((\log n)/(\log \log n)^{3/2})$ ).

If we instead assume a more precise characterization  $\text{pdeg}(\text{OR}_n) = \Theta((\log n)^c)$ , then going through the proofs of the above theorems would yield a sharper lower bound of  $\Omega((\log n)^{c/(c+1)}/(\log \log n)^2)$  for any truly  $n$ -variate Boolean function and a better upper bound of  $O((\log n)^{c/(c+1)})$  for some truly  $n$ -variate Boolean function.

### 1.3 Proof Outline

Our proof is motivated by two important examples. The first of these is the  $\text{OR}_n$  function which has probabilistic degree at most  $O(\log n)$  by results of [7, 27] and at least  $(\log n)^{(1/2)-o(1)}$  by [16]. The second is the *Addressing function*, which we now define.

The Addressing function  $\text{Addr}_r$  has  $n = r + 2^r$  variables. We think of the input variables as being divided into two parts: there are  $r$  “addressing” variables  $y_1, \dots, y_r$  and  $2^r$  “addressed” variables  $\{z_a \mid a \in \{0, 1\}^r\}$  (the latter part of the input is thus indexed by elements of  $\{0, 1\}^r$ ). On an input  $(a, A) \in \{0, 1\}^r \times \{0, 1\}^{2^r}$ , the output of the function is defined to be  $A_a$  (i.e. the  $a$ th co-ordinate of the vector  $A$ ). The Addressing function satisfies  $\deg(\text{Addr}_r) = r + 1 = O(\log n)$ . This example is quite relevant to this line of work: in particular, it implies that the results of Nisan and Szegedy [21] and Chiarelli et al. [13] stated above are tight, and is also a tight example for Simon’s theorem [25].

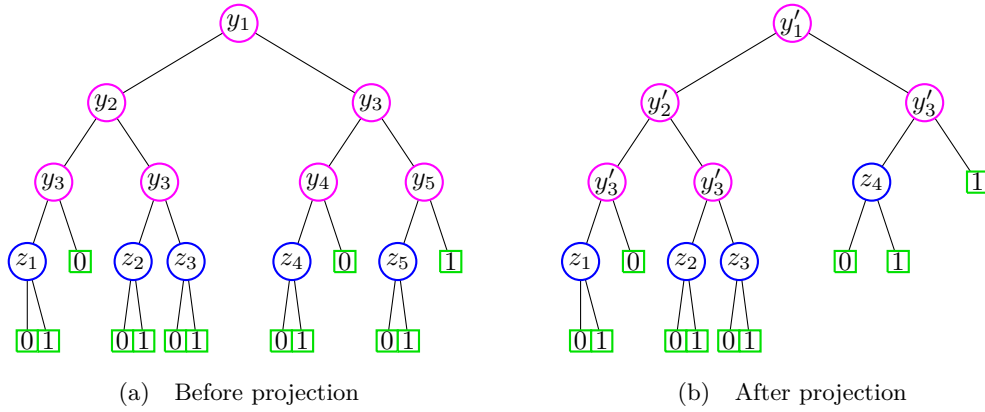
We now describe the upper and lower bound proofs, starting with the less technical upper bound.

#### The Upper Bound

Given that we have two natural families of truly  $n$ -variate functions that have degree  $O(\log n)$ , one may suspect that this is the best possible. Indeed this was also our initial conjecture. However, using the ideas of Ambainis and de Wolf [4], we can do better. Ambainis and de Wolf showed that there are truly  $n$ -variate Boolean functions that have *approximate degree*  $O(\log n / \log \log n)$ . Their construction<sup>6</sup> uses a modified Addressing function, where the addressing variables are present in an “encoded” form. While this blows up the size of the first part of the input, this does not affect  $n$  much as the addressing variables take up only a small part of the input. On the other hand, the advantage is that the “decoding” procedure can be performed approximately by a suitable low-degree polynomial: a proof of this uses two famous Quantum algorithms, the *Bernstein-Vazirani algorithm* and *Grover search*, along with the fact that efficient Quantum algorithms yield approximating low-degree polynomials [6]. Putting things together yields an improved approximate degree bound for some  $n$ -variate  $f$ .

We show how to port their construction to the probabilistic degree setting. The first observation is that Grover search, which is essentially an algorithm for computing  $\text{OR}_n$ , is much more “efficient” in the probabilistic degree setting, as  $\text{pdeg}(\text{OR}_n) = O(\log n)$ , while its approximate degree is  $\Omega(\sqrt{n})$  [21]. The second observation is that the Bernstein-Vazirani algorithm, which can be thought of as a decoding algorithm for a suitable error-correcting code, can be replaced by polynomial interpolation. This gives a good idea of why we should also be able to use a similar construction in the probabilistic degree setting. In fact, the better probabilistic degree upper bound for  $\text{OR}_n$  implies that we should be able to get a *better* bound than what is possible for approximate degree. Indeed this is true. By a similar construction, we show that we can construct a truly  $n$ -variate  $f$  with probabilistic degree  $O(\sqrt{\log n})$  *unconditionally*, which is quite a bit better than previous results for any of the above degree measures. If we assume, moreover, that  $\text{pdeg}(\text{OR}_n) \leq (\log n)^{c+o(1)}$ , the same construction yields a function with probabilistic degree  $(\log n)^{c/(c+1)+o(1)}$ .

<sup>6</sup> They actually give two, slightly different, constructions. We use the second one here.



■ **Figure 1** The function  $f(y_1, \dots, y_5, z_1, \dots, z_5)$  is defined by the decision tree on the left (we assume that the left child corresponds to the queried variable taking value 0). When  $z_1, \dots, z_5$  are set i.u.a.r. to  $\mathbf{b}_1, \dots, \mathbf{b}_5$ , we get a random function  $F(y_1, \dots, y_5)$  such that  $F(00000) = \mathbf{b}_1, F(01000) = \mathbf{b}_2, F(01100) = \mathbf{b}_3, F(10000) = \mathbf{b}_4, F(10100) = \mathbf{b}_5$ . After a projection that maps  $y_1 \mapsto y'_1; y_2 \mapsto y'_2; y_3, y_4, y_5 \mapsto y'_3$ , we get the function  $f'$  computed by the tree on the right. This reduces the number of addressing variables to 3. But also note that the variable  $z_5$  is no longer relevant as the path leading to it is inconsistent with the projection. So the number of addressed variables falls to 4.

**The Lower Bound**

Given that the upper bound construction uses the Addressing function as well as the OR function, it is only natural that the lower bound would use the lower bounds for these two families of functions. Our hypothesis already assumes a lower bound of  $(\log n)^{c-o(1)}$  for  $\text{pdeg}(\text{OR}_n)$ . For the addressing function  $\text{Addr}_r$  described above, one can prove an  $\Omega(r) = \Omega(\log n)$  lower bound in the following way. We observe that by setting the  $2^r$  addressed variables uniformly at random, we obtain a uniformly random function on the  $r$  addressing variables. By a counting argument, one can show that a uniformly random Boolean function  $F$  on  $r$  variables has probabilistic degree  $\Omega(r)$  with high probability. In particular, as setting some input variables to constants can only reduce probabilistic degree, this implies that  $\text{pdeg}(\text{Addr}_r) = \Omega(r) = \Omega(\log n)$ . Note that this is tight, as  $\text{deg}(\text{Addr}_r) = r + 1$ .

Our aim is to generalize the above lower bounds enough to prove a lower bound for any truly  $n$ -variate  $f$ . The first informal observation is that the  $\text{OR}_n$  function is the “simplest” function on  $n$  variables to have sensitivity  $n$ . Therefore, it is intuitive that any Boolean function with sensitivity  $n$  should have probabilistic degree at least that of the  $\text{OR}_n$  function. We show that this is true, up to  $(\log n)^{o(1)}$  factors. More generally, we show that any Boolean function  $f$  with sensitivity  $s$  has probabilistic degree at least that of the OR function on  $s$  variables (up to  $(\log s)^{o(1)}$  factors). The proof of this is in the contrapositive: we use a probabilistic degree upper bound for  $f$  to construct a probabilistic polynomial for  $\text{OR}_s$ . The ideas behind this go back to a sampling argument used in the works of Beigel et al. and Tarui [7, 27]. Viewing this argument more abstractly, we can use this to construct a reduction from  $\text{OR}_s$  to  $f$  (for any  $f$  of sensitivity  $s$ ) in the probabilistic degree setting.

The above argument implies a strong lower bound for any  $n$ -variate  $f$  with large sensitivity. In particular, it implies that if  $f$  has sensitivity at least  $s = n^{\Omega(1)}$ , then its probabilistic degree is almost that of the OR function. We now consider the case of functions with small sensitivity (specifically when  $s = n^{o(1)}$ ), which is the most technical part of the proof. By a recent breakthrough result of Huang [18], we also know that  $f$  also has a *decision tree* (we

refer the reader to [10] for the definition of Decision trees) of depth  $d = \text{poly}(s) = n^{o(1)}$ .<sup>7</sup> The prototypical example of such an  $f$  is the Addressing function which has a decision tree of depth  $r + 1 = \lfloor \log n \rfloor + 1$ , which we argued a lower bound for above. The idea, in general, is to find a copy of something like an Addressing function “inside” the function  $f$ .

We illustrate how this argument works by considering a special case of the problem, which is only a small variant of the Addressing function. Assume that a truly  $n$ -variate function  $f$  is computed by a decision tree  $T$  of depth  $d = \text{poly}(\log n)$ . Note that as the function depends on all its variables, each of the underlying  $n$  variables appear in the tree  $T$ . To make things even simpler, assume that we have  $n/2$  “addressing” variables  $y_1, \dots, y_{n/2}$  and  $n/2$  “addressed” variables  $z_1, \dots, z_{n/2}$ . The tree reads  $d - 1$  addressing variables among  $y_1, \dots, y_{n/2}$  in some (possibly adaptive) fashion and then possibly queries one addressed variable, the value of which is output. (See Figure 1 (a).)

How do we argue a lower bound on  $\text{pdeg}(f)$ ? We could try to proceed as above and set the addressed variables  $z_1, \dots, z_{n/2}$  as random to obtain a random function  $\mathbf{F}$  in  $y_1, \dots, y_{n/2}$ . However, this function is not *uniformly* random, as it is sampled using only  $n/2$  random bits, while the number of functions in  $n/2$  variables is  $2^{2^{n/2}}$ . Nevertheless, we can observe that the function  $\mathbf{F}$  does take independent random values at at least  $n/2$  distinct inputs, those which are consistent with  $n/2$  distinct paths in  $T$  leading to the various addressed variables. (See Figure 1 (a).) We could try to lower bound  $\text{pdeg}(\mathbf{F})$  as above.

This leads to the following general question: given a random function  $\mathbf{F} : \{0, 1\}^r \rightarrow \{0, 1\}$  that takes independent and random values at  $M$  distinct inputs in  $\{0, 1\}^r$ , what can we say about the probabilistic degree of  $\mathbf{F}$ ? By a more general counting argument, we are able to show that with high probability, the probabilistic degree of  $\mathbf{F}$  is at least  $\Omega(\log M / \log r)$ . This is easily seen to be tight in the case that  $X$  is, say, a Hamming ball of radius  $R \leq r^{1-\Omega(1)}$ . (In the case that  $M = 2^r$ , this leads to a bound of  $\Omega(r / \log r)$ , nearly matching the claim for random functions that we mentioned above. A tight bound can be obtained in the same way but is harder to state for general  $M$ .)

Given this bound for random functions, we can try to use it in the case of the function  $f$  above. Unfortunately, in this case, both parameters  $r$  and  $M$  are  $n/2$ , and hence we do not get any non-trivial bound. However, we show that we can still reduce to a case where a non-trivial bound is possible (this is where the depth of  $T$  comes in). More precisely, we reduce the number of addressing variables by *projecting* the  $n/2$  addressing variables to a smaller set of  $r'$  variables  $Y' = \{y'_1, \dots, y'_{r'}\}$ . That is, we randomly set each variable  $Y$  to a uniformly random variable in  $Y'$  to get a different function in the variables  $Y' \cup Z$ . This has the effect of reducing the number of addressing variables to  $r'$ . But there is also a potential problem: the projection could also render some of the addressed variables irrelevant, as the paths that lead to them become inconsistent. (See Figure 1 (b).)

Nevertheless, if we choose  $r'$  large enough (something like  $r' = 4d^2$  is enough by the Birthday paradox), the variables of each path are sent to *distinct* variables in  $Y'$  with high probability, which implies that each addressed variable remains relevant with high probability. In particular, there is a projection that maps  $f$  to an “Addressing function” with only  $\text{poly}(\log n)$  addressing variables and  $\Omega(n)$  addressed variables. Now applying the argument for random functions, we get a probabilistic degree lower bound of  $\Omega(\log n / \log \log n)$  for this

<sup>7</sup> Strictly speaking, we do not need to use Huang’s result as we could also use the known polynomial relationship between the decision tree height and the *block sensitivity* of a function [20]. But it is notationally easier to work with sensitivity.

function, nearly matching what we obtained for the Addressing function. As projections do not increase probabilistic degree, the same bound holds for  $f$ , concluding the proof in this special case.<sup>8</sup>

A similar argument can be carried out in the general case by first carefully partitioning the variables into the addressing and addressed variables. We do this by looking at the structure of the decision tree  $T$ . These details are postponed to the formal proof. In general, this argument yields a lower bound of  $\Omega(\log n / \log s)$  on the probabilistic degree of a truly  $n$ -variate function  $f$  with sensitivity at most  $s$ .

Using this lower bound along with the previous lower bound for functions of sensitivity at least  $s$ , and optimizing our choice of  $s$ , yields a lower bound of  $(\log n)^{c/(c+1)-o(1)}$  for any truly  $n$ -variate function  $f$ .

## 2 Preliminaries

### Functions, Restrictions, Projections

Throughout, we work with real-valued functions  $f : \{0, 1\}^n \rightarrow \mathbb{R}$ . Boolean functions (i.e. functions mapping  $\{0, 1\}^n$  to  $\{0, 1\}$ ) are also treated as real-valued. We use boldface notation to denote random variables. A *random function*  $\mathbf{F}$  is a probability distribution over functions.

A *restriction* on  $n$  variables is a map  $\rho : [n] \rightarrow \{0, 1, *\}$ . Given a function  $f : \{0, 1\}^n \rightarrow \mathbb{R}$  and a restriction  $\rho$  on  $n$  variables, we have a natural restricted function  $f_\rho$  defined by setting the  $i$ th input variable to  $f$  to 0, 1 or leaving it as is, depending on whether  $\rho(i)$  is 0, 1 or  $*$  respectively. Note that the function  $f_\rho$  now depends on  $|\rho^{-1}(*)|$  many variables. However, we sometimes also treat  $f_\rho$  as a function of all the original variables that only *depends* on (a subset of) the variables indexed by  $\rho^{-1}(*)$ .

A *projection* from  $n$  variables to  $m$  variables is a map  $\nu : [n] \rightarrow [m]$ . Given a function  $f : \{0, 1\}^n \rightarrow \mathbb{R}$  and a projection  $\nu$  from  $n$  variables to  $m$  variables, we get a function  $f|_\nu : \{0, 1\}^m \rightarrow \mathbb{R}$  by identifying variables of  $f$  that map to the same image under  $\nu$ .

### Some Boolean functions

For any positive integer  $n$ , we use  $\text{OR}_n$ ,  $\text{AND}_n$  and  $\text{Maj}_n$  to denote the OR, AND and Majority functions on  $n$  variables respectively.

► **Fact 9.** *We have the following simple facts about probabilistic polynomials.*

1. (*Interpolation*) Any function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  has an exact multilinear polynomial representation of degree at most  $n$ , i.e.  $\deg(f) := \text{pdeg}_0(f) \leq n$ .
2. (*Shifts and Restrictions*) Fix any  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  and any  $\varepsilon \geq 0$ . Then the function  $g : \{0, 1\}^n \rightarrow \{0, 1\}$  defined by  $g(x) = f(x \oplus y)$  for a fixed  $y \in \{0, 1\}^n$  has the same probabilistic degree as  $f$ , i.e.,  $\text{pdeg}_\varepsilon(g) = \text{pdeg}_\varepsilon(f)$ .  
If  $g : \{0, 1\}^m \rightarrow \{0, 1\}$  is a restriction or a projection of  $f$ , then  $\text{pdeg}_\varepsilon(g) \leq \text{pdeg}_\varepsilon(f)$ .
3. (*Error reduction [16]*) For any  $\delta < \varepsilon \leq 1/3$  and any Boolean function  $f$ , if  $\mathbf{P}$  is an  $\varepsilon$ -error probabilistic polynomial for  $f$ , then  $\mathbf{Q} = M(\mathbf{P}_1, \dots, \mathbf{P}_\ell)$  is a  $\delta$ -error probabilistic polynomial for  $f$  where  $\ell = O(\log(1/\delta)/\log(1/\varepsilon))$ ,  $M$  is the exact multilinear polynomial for  $\text{Maj}_\ell$  and  $\mathbf{P}_1, \dots, \mathbf{P}_\ell$  are independent copies of  $\mathbf{P}$ . In particular, we have  $\text{pdeg}_\delta(f) \leq \text{pdeg}_\varepsilon(f) \cdot O(\log(1/\delta)/\log(1/\varepsilon))$ .

<sup>8</sup> Random projections of this kind have been used recently to prove important results in circuit complexity [17, 12]. However, as far as we know, they have not been used to prove probabilistic degree lower bounds.



4. (Composition) For any Boolean function  $f$  on  $k$  variables and any Boolean functions  $g_1, \dots, g_k$  on a common set of  $m$  variables, let  $h$  denote the natural composed function  $f(g_1, \dots, g_k)$  on  $m$  variables. For  $\varepsilon, \delta_1, \dots, \delta_k \geq 0$ , let  $\mathbf{P}, \mathbf{Q}_1, \dots, \mathbf{Q}_k$  be probabilistic polynomials for  $f, g_1, \dots, g_k$  respectively with errors  $\varepsilon, \delta_1, \dots, \delta_k$  respectively. Then,  $\mathbf{R} = \mathbf{P}(\mathbf{Q}_1, \dots, \mathbf{Q}_k)$  is a probabilistic polynomial for  $h$  with error at most  $\varepsilon + \sum_i \delta_i$ . In particular, for any  $\varepsilon, \delta > 0$ , we have  $\text{pdeg}_{\varepsilon+k\delta}(h) \leq \text{pdeg}_{\varepsilon}(f) \cdot \max_{i \in [k]} \text{pdeg}_{\delta}(g_i)$ .

We will need the following known upper and lower bounds on  $\text{pdeg}(\text{OR}_n)$ .

► **Theorem 10** ([7, 27]).  $\text{pdeg}_{\varepsilon}(\text{OR}_n) = O(\log n \log(1/\varepsilon))$ .

► **Theorem 11** ([16]).  $\text{pdeg}(\text{OR}_n) \geq (\log n)^{1/2-o(1)}$ .

► **Definition 12** (Some Complexity Measures of Boolean functions). Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  be any Boolean function. We use  $D(f)$  to denote the depth of the smallest Decision Tree computing  $f$ .

For  $a \in \{0, 1\}^n$ , we use  $s(f, a)$  to denote the number of  $b \in \{0, 1\}^n$  that can be obtained by flipping a single bit of  $a$  and satisfying  $f(a) \neq f(b)$ . The Sensitivity of  $f$ , denoted  $s(f)$ , is defined to be the maximum value of  $s(f, a)$  as  $a$  ranges over  $\{0, 1\}^n$ .

Huang [18] proved the following breakthrough result recently.

► **Theorem 13** (Huang's Sensitivity theorem [18]). There is an absolute constant  $c_0 > 0$  such that for all large enough  $n$  and all functions  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ ,  $D(f) \leq s(f)^{c_0}$ .

Strictly speaking, we do not need to use Huang's Sensitivity theorem in what follows as we could also make do with a polynomial relationship between the decision tree height and the *block sensitivity*<sup>9</sup> of  $f$ , which has been known for a long time [20]. However, it is notationally simpler to work with sensitivity.

### 3 The Lower Bound: Proof of Theorem 4

The proof is made up of two lower bounds. We first prove a lower bound on  $\text{pdeg}(f)$  for any function  $f$  that has large sensitivity  $s(f)$ ; this is by a suitable reduction from the case of the OR function. We then prove a lower bound on  $\text{pdeg}(f)$  for any function that depends on all its variables but has small sensitivity; this is by a suitable reduction from a kind of Addressing function. Optimizing over the parameters of the lower bounds will yield the lower bound of the theorem statement.

Throughout this section, we assume that  $\text{pdeg}(\text{OR}_n) \geq (\log n)^{c-o(1)}$  for large enough  $n$ .

#### 3.1 The case of large sensitivity

The main result of this section is the following lower bound on the probabilistic degrees of Boolean functions with large sensitivity.

► **Lemma 14.** Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  be any Boolean function that has sensitivity  $s$ . Then,  $\text{pdeg}(f) \geq (\log s)^{c-o(1)}$ .

<sup>9</sup> The Block sensitivity of a function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  is defined as follows. Given  $a \in \{0, 1\}^n$ , define  $bs(f, a)$  to be the maximum number of pairwise disjoint sets  $B_1, \dots, B_t \subseteq [n]$  such that flipping all the bits indexed by any  $B_i$  in  $a$  results in an input  $b^{(i)}$  such that  $f(a) \neq f(b^{(i)})$ . Then, the block sensitivity of  $f$  is defined to be the maximum value of  $bs(f, a)$  over all inputs  $a \in \{0, 1\}^n$ .

## 42:10 Probabilistic Degree of $N$ -Variate Boolean Function

The above lemma is proved via a probabilistic reduction from the OR function on  $s$  variables to the function  $f$ . This is captured by the following lemma, which shows how a function that has large sensitivity can be used to obtain a probabilistic representation of a large copy of the OR function.

Recall from above that a Boolean function  $h : \{0, 1\}^s \rightarrow \{0, 1\}$  is a restriction of a Boolean function  $g : \{0, 1\}^s \rightarrow \{0, 1\}$  if  $h$  can be obtained by setting some inputs of  $g$  to constants. Though  $h$  no longer depends on the variables that are set to constants, here we still treat  $h$  as a function on all  $s$  variables.

► **Lemma 15.** *Let  $g : \{0, 1\}^s \rightarrow \{0, 1\}$  be any Boolean function such that  $g(0^s) = 0$  and  $g(x) = 1$  for any  $x$  of Hamming weight 1. Then, there exist  $\ell = O(\log s)$  independent random restrictions  $\mathbf{g}_1, \dots, \mathbf{g}_\ell$  of  $g$  such that for any  $a \in \{0, 1\}^s$ ,*

$$\Pr_{\mathbf{g}_1, \dots, \mathbf{g}_\ell} [\text{OR}_\ell(\mathbf{g}_1(a), \dots, \mathbf{g}_\ell(a)) \neq \text{OR}_s(a)] \leq \frac{1}{10}.$$

We interpret the random function  $\text{OR}_\ell(\mathbf{g}_1(a), \dots, \mathbf{g}_\ell(a))$  as a probabilistic representation of the  $\text{OR}_s$  function. The reader may be confused by the fact that the probabilistic representation itself uses an OR function; however, note that this OR function is defined on  $\ell \ll s$  variables and consequently is a much “simpler” function (in particular, for us, what is relevant is that  $\text{pdeg}(\text{OR}_\ell) = O(\log \ell)$  [7, 27] which is much smaller than  $\log s$ ).

The proof of Lemma 15 is closely related to the argument for constructing a probabilistic polynomial for the OR function from [7, 27]. The observation here is that a similar argument can be used to give a probabilistic reduction from  $\text{OR}_s$  to any function  $g$  as above. Due to space constraints, we push the proof of Lemma 15 to the Appendix (Section A). Let us now prove Lemma 14.

**Proof of Lemma 14.** We know that  $f$  has some input of sensitivity  $s$ . Then we note that we may assume  $f(0^n) = 0$  and  $f(0^{j-1}10^{n-j}) = 1$  for  $j \in [s]$ . For let  $a \in \{0, 1\}^n$  such that  $s(f, a) = s$ . If  $f(a) = 1$ , we may replace  $f$  by  $1 - f$ . (Obviously,  $\text{pdeg}_\varepsilon(f) = \text{pdeg}_\varepsilon(1 - f)$ , for all  $\varepsilon \geq 0$ .) So we may assume  $f(a) = 0$ . Now by permuting coordinates if required, we may assume that  $f(\tilde{a}^{(j)}) = 1$ , where  $\tilde{a}^{(j)} := (a_1, \dots, a_{j-1}, 1 - a_j, a_{j+1}, \dots, a_n)$  for all  $j \in [s]$ . Further, if  $a \neq 0^n$ , we may replace  $f$  by  $f'$ , defined as  $f'(x) = f(x \oplus a)$ ,  $x \in \{0, 1\}^n$ . By Fact 9 Item 2,  $\text{pdeg}_\varepsilon(f) = \text{pdeg}_\varepsilon(f')$ , for all  $\varepsilon \geq 0$ . Clearly, we have  $f'(0^n) = 0$  and  $f'(0^{j-1}10^{n-j}) = 1$ , for all  $j \in [s]$ .

So now, by assumption, we have  $f(0^n) = 0$  and  $f(0^{j-1}10^{n-j}) = 1$  for  $j \in [s]$ . Define  $g : \{0, 1\}^s \rightarrow \{0, 1\}$  as  $g(x) = f(x0^{n-s})$ . Then  $g$  satisfies the hypotheses of Lemma 15. Hence, by Lemma 15, there exist  $\ell = O(\log s)$  random restrictions  $\mathbf{g}_1, \dots, \mathbf{g}_\ell$  of  $g$  such that

$$\Pr_{\mathbf{g}_1, \dots, \mathbf{g}_\ell} [\text{OR}_\ell(\mathbf{g}_1(x), \dots, \mathbf{g}_\ell(x)) \neq \text{OR}_s(x)] \leq \frac{1}{10}, \quad \text{for all } x \in \{0, 1\}^s. \quad (1)$$

We use the above representation to devise a probabilistic polynomial for  $\text{OR}_s$ .

Let  $\mathbf{O}$  be any  $(1/10)$ -error probabilistic polynomial for  $\text{OR}_\ell$  and  $\mathbf{G}_1, \dots, \mathbf{G}_\ell$  be any  $(1/10\ell)$ -error probabilistic polynomials for  $\mathbf{g}_1, \dots, \mathbf{g}_\ell$  respectively. Then, by Fact 9 and (1),  $\mathbf{O}(\mathbf{G}_1, \dots, \mathbf{G}_\ell)$  is a  $(1/3)$ -error probabilistic polynomial for  $\text{OR}_s$ .

Note that by Theorem 10, we can choose  $\mathbf{O}$  to have degree at most  $O(\log \ell)$ . Further, by Fact 9, we have  $\text{pdeg}(\mathbf{g}_i) \leq \text{pdeg}(g) \leq \text{pdeg}(f)$  for each  $i \in [\ell]$ . In particular, this implies that we can choose  $\mathbf{G}_i$  to have degree  $O(\text{pdeg}(f) \cdot \log \ell)$  for each  $i \in [\ell]$ . This yields

$$\text{pdeg}(\text{OR}_s) \leq \text{pdeg}(f) \cdot O(\log \ell)^2 = \text{pdeg}(f) \cdot O((\log \log s)^2) = \text{pdeg}(f) \cdot (\log s)^{o(1)}.$$

As  $\text{pdeg}(\text{OR}_s) \geq (\log s)^{c-o(1)}$  by assumption, we get the desired lower bound on  $\text{pdeg}(f)$ . ◀

### 3.2 The case of small sensitivity

We prove the following lemma.

► **Lemma 16.** *Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  be a function of sensitivity at most  $s$  that depends on all its  $n$  variables. Then, we have  $\text{pdeg}(f) = \Omega\left(\frac{\log(n/s^{O(1)})}{\log s}\right)$ .*

The proof of the lemma is in two steps. In the first step, we use a counting argument to prove a lower bound on the probabilistic degrees of random functions  $\mathbf{F} : \{0, 1\}^m \rightarrow \{0, 1\}$  which are chosen from a distribution such that for a large subset  $X \subseteq \{0, 1\}^m$ , the random variables  $\{\mathbf{F}(x) \mid x \in X\}$  are independently and uniformly chosen random bits. In the second step, we show how any  $f$  as in the statement of Lemma 16 can be randomly restricted to a random function  $\mathbf{F}$  where the lower bound for random functions applies.

We now state the lower bound for random functions and use it to prove Lemma 16. The lower bound for random functions uses fairly standard ideas and is proved in the appendix (Section B).

► **Lemma 17 (Random function lower bound).** *The following holds for positive integer parameters  $m, M$  and  $d$  such that  $M > m^{10d}$ . Let  $\mathbf{F} : \{0, 1\}^m \rightarrow \{0, 1\}$  be a random function such that for some  $X \subseteq \{0, 1\}^m$  with  $|X| = M$ , the random variables  $(\mathbf{F}(x))_{x \in X}$  are independent and uniformly distributed random bits. Then, we have*

$$\Pr_{\mathbf{F}}[\text{pdeg}_{1/10}(\mathbf{F}) \leq d] < \frac{1}{10}.$$

Let us see how to use Lemma 17 to prove Lemma 16. This proof again breaks into two smaller steps.

**Step 1.** Show that, after a projection,  $f$  turns into something similar to an addressing function, that we will call a *Pseudoaddressing function*.

**Step 2.** Show that any pseudoaddressing function has large probabilistic degree.

As any projection  $g$  of  $f$  satisfies  $\text{pdeg}(g) \leq \text{pdeg}(f)$  (Fact 9), the above implies a lower bound on  $\text{pdeg}(f)$ , hence proving Lemma 16.

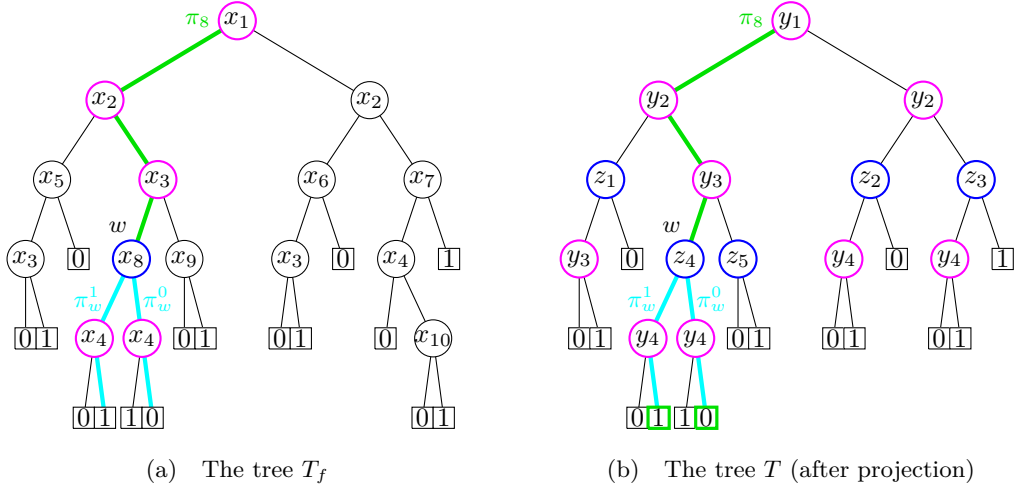
To make the above precise, we need the following definition. We say that a function  $g : \{0, 1\}^{r+t} \rightarrow \{0, 1\}$  is an  $(r, t)$ -*Pseudoaddressing function* if the input variables to  $g$  can be partitioned into two sets  $Y = \{y_1, \dots, y_r\}$  and  $Z = \{z_1, \dots, z_t\}$  and  $g$  can be computed by a decision tree  $T$  with the following properties.

- P1.** For each  $z_j \in Z$ , there are two root-to-leaf paths  $\pi_j^0$  and  $\pi_j^1$  in  $T$  that diverge at a node labeled  $z_j$  and lead to outputs 0 and 1 respectively.
- P2.** All the other nodes on these paths are labeled by variables in  $Y$ , and further these variables take the same values on both paths. In particular,  $\pi_j^0$  and  $\pi_j^1$  differ only on the value of  $z_j$ .

► **Example 18.** Consider the standard Addressing function  $\text{Addr}_r$  on  $n = r + 2^r$  variables as defined in Section 1.3. This function is an  $(r, 2^r)$ -pseudoaddressing function as it can be computed by a decision tree of depth  $r + 1$ , which first queries all the addressing variables to determine  $a \in \{0, 1\}^r$  and then queries and outputs the value of  $z_a$  (the two computational paths querying  $z_a$  give the desired root-to-leaf paths required in the definition above).

In analogy with the Addressing function, given an  $(r, t)$ -pseudoaddressing function as above, we refer to the variables in  $Y$  as the addressing variables and the variables in  $Z$  as the addressed variables.

The two steps of the proof as outlined above can now be formalized as follows.



■ **Figure 2** The decision tree on the left computes a truly 10-variate function  $f(x_1, \dots, x_{10})$ . The paths obtained by concatenating  $\pi_8$  with  $\pi_w^0$  and  $\pi_w^1$  are consistent with each other except for the value of  $x_8$ , the variable queried at node  $w$ . After a projection  $\nu : [10] \rightarrow [9]$  defined by  $\nu(i) = i$  for  $i \leq 9$  and  $\nu(10) = 4$ , we get a tree  $T$ , which computes a  $(4, 5)$ -pseudoaddressing function  $g(y_1, \dots, y_4, z_1, \dots, z_5)$ . Note that each path in  $T$  corresponds to a path in  $T_f$  but not every path in  $T_f$  survives in  $T$  (e.g. the path leading to 0 through the node querying  $x_{10}$  is pruned away, as it is inconsistent with  $\nu$ ).

$\triangleright$  **Claim 19.** Let  $f$  be as in the statement of Lemma 16. Then, there exist  $r \leq s^{O(1)}$  and  $t \geq n/s^{O(1)}$  and a projection  $\nu : [n] \rightarrow [r+t]$  such that  $g = f|_\nu$  is an  $(r, t)$ -pseudoaddressing function.

$\triangleright$  **Claim 20.** Let  $g$  be any  $(r, t)$ -pseudoaddressing function. Then,  $\text{pdeg}(g) = \Omega(\log t / \log r)$ .

As noted above, the above claims immediately imply Lemma 16. We now prove these claims.

**Proof of Claim 19.** We will first outline how to isolate a set of  $n/\text{poly}(s)$  variables that will (almost) be the set of addressed variables. A projection will then be applied to the remaining variables to create the pseudoaddressing function. Let us now see the details.

By Theorem 13, we know that  $f$  has a decision tree  $T_f$  of depth  $d \leq \text{poly}(s)$ . Fix such a tree  $T_f$  of *minimum size*, i.e. with the smallest possible number of leaves. Let  $V = \{x_1, \dots, x_n\}$  denote the input variables of  $f$ .

Any variable  $x_i \in V$  must be queried somewhere in the tree  $T_f$ , as  $f$  depends on all its input variables by assumption. Fix any occurrence of this variable in the decision tree  $T_f$ , and let  $w$  denote the node of  $T_f$  corresponding to this query. (Refer to Figure 2 (a) for an illustration.) Let  $\pi_i$  denote the path from the root of  $T_f$  to  $w$  and let  $T_0$  and  $T_1$  be the subtrees rooted at the left and right children of  $w$ . The decision trees  $T_0$  and  $T_1$  both compute functions of the  $n' < n$  Boolean variables not queried in  $\pi_i$ . Note that these decision trees compute *distinct* functions since otherwise the query made at the vertex  $w$  is unnecessary, and a smaller decision tree than  $T_f$  can be obtained by replacing the subtree rooted at  $w$  by  $T_0$  or by  $T_1$ . This contradicts the minimality of the size of  $T_f$ .

Thus,  $T_0$  and  $T_1$  compute distinct functions. In particular, there is an input  $a \in \{0, 1\}^{n'}$  on which  $T_0$  and  $T_1$  have different outputs; w.l.o.g., assume  $T_0$  and  $T_1$  output 0 and 1 respectively on  $a$ . Let  $\pi_w^0$  and  $\pi_w^1$  be the root-to-leaf paths followed on the input  $a$  in  $T_0$  and

$T_1$  respectively. Note that any variable queried on both  $\pi_w^0$  and  $\pi_w^1$  takes the same value on both paths, as both paths are consistent with the input  $a$ . (Again, see Figure 2 (a) for an example.)

Concatenating each of  $\pi_w^0$  and  $\pi_w^1$  with the path  $\pi_i$  gives us two root-to-leaf paths  $\pi_i^0$  and  $\pi_i^1$  in  $T$  such that

- Q1. The paths  $\pi_i^0$  and  $\pi_i^1$  diverge at the node  $w$  (labelled by variable  $x_i$ ) and lead to outputs 0 and 1 respectively.
- Q2. The two paths agree on all variables other than  $x_i$ , i.e., any other variable that is queried on  $\pi_i^0$  and  $\pi_i^1$  takes the same value on both.

We have such a pair of paths  $\pi_i^0$  and  $\pi_i^1$  for each  $x_i \in V$ . Let  $P_i$  denote the set of all  $j \neq i$  such that  $x_j$  is queried on  $\pi_i^0$  or on  $\pi_i^1$ . Note that  $|P_i| \leq 2d$ .

We claim that we can choose a large subset  $Z' \subseteq [n]$  such that for all  $i \in Z'$ , the set  $P_i$  does not contain any  $j$  where  $j \in Z'$ . To see this, define a graph  $G$  with vertex  $[n]$  and edges between vertices distinct  $i, j \in [n]$  if and only if  $P_i$  contains  $j$  or vice-versa. Since each  $|P_i| \leq 2d$ , it is clear that this graph has average degree at most  $2d$ . By Turán's theorem (see e.g. [3]), this implies that  $G$  has an independent set  $Z'$  of size at least  $n/4d$ . This set  $Z'$  has the required property.

We are now ready to show that the required projection  $\nu$  exists. Let  $r = 10d^2$  and let  $\nu' : [n] \setminus Z' \rightarrow [r]$  be a random map (i.e. the image of each element of the domain is independently and uniformly chosen from  $[r]$ ). We say that an  $i \in Z'$  is *good* if  $\nu'$  is 1-1 on the set  $P_i$ . Let  $\mathcal{G}$  be the set of all good  $i$ , with  $t := |\mathcal{G}|$ . Assume  $\mathcal{G} = \{i_1, \dots, i_t\}$ . We use this to define a random projection  $\nu : [n] \rightarrow [r + t]$  by

$$\nu(i) = \begin{cases} \nu'(i) & \text{if } i \notin Z', \\ 1 & \text{if } i \in Z' \setminus \mathcal{G}, \text{ (here, any } k \in [r] \text{ will do)} \\ r + j & \text{if } i \in \mathcal{G} \text{ and } i = i_j. \end{cases}$$

The random projection defines a random Boolean function  $g$  on  $r + t$  variables. We now show that, with positive probability,  $g$  is an  $(r, n/\text{poly}(s))$ -pseudoaddressing function, where the first  $r$  variables are the addressing variables. This will finish the proof. Note that the projection  $\nu$  applied to the tree  $T_f$  also defines a random decision tree  $T$  computing  $g$ . We will in fact show that  $T$  serves as a witness for the fact that  $g$  is an  $(r, n/\text{poly}(s))$ -pseudoaddressing function (with positive probability).

In fact, this happens whenever  $t = |\mathcal{G}|$  is large enough. More precisely, note that

$$\begin{aligned} \mathbf{E}_{\nu'}[|Z'| - t] &= \sum_{i \in Z'} \Pr_{\nu'}[\nu' \text{ is not 1-1 on } P_i] \leq \sum_{i \in Z'} \sum_{j \neq k \in P_i} \Pr_{\nu'}[\nu'(j) = \nu'(k)] \\ &\leq \sum_{i \in Z'} |P_i|^2 \cdot \frac{1}{r} \leq |Z'| \cdot \frac{(2d)^2}{r} \leq \frac{|Z'|}{2}. \end{aligned}$$

In particular, there is a setting  $\nu'$  of  $\nu'$  such that the corresponding set of good variables  $|\mathcal{G}|$  has size at least  $|Z'|/2$ . Fix this  $\nu'$  and let  $\mathcal{G}, t, \nu, g, T$  be the corresponding fixings of  $\mathcal{G}, t, \nu, g, T$  respectively.

We have  $g = g(y_1, \dots, y_r, z_1, \dots, z_t)$ . Observe that each root-to-leaf path of  $T$  can be identified with a root-to-leaf path of  $T_f$ . Further, a path  $\pi$  of  $T_f$  survives in  $T$  exactly when it is *consistent w.r.t.*  $\nu$ , i.e., if two variables that are set to opposite values in  $\pi$  are not mapped to the same variable by  $\nu$  (see Figure 2 (b) for an example). In particular, if a path  $\pi$  has the property that the variables queried along  $\pi$  are mapped injectively by  $\nu$ , then the path  $\pi$  survives in  $T$ .

## 42:14 Probabilistic Degree of $N$ -Variate Boolean Function

This implies that for any good  $i_j \in \mathcal{G}$ , the corresponding paths  $\pi_{i_j}^0$  and  $\pi_{i_j}^1$  survive in  $T$ . Moreover, as the projection  $\nu$  is injective on the entire set  $P_{i_j}$ , these paths continue to agree with each other on all variables except the variable  $z_j$  queried at the point of their divergence. This gives both properties Q1 and Q2 stated above. As this holds for each  $i_j \in \mathcal{G}$ , we see that  $g$  is indeed an  $(r, t)$ -pseudoaddressing function. Note that  $r = 10d^2 \leq \text{poly}(s)$  and  $t \geq n/4d \geq n/\text{poly}(s)$ . Hence, we have proved the claim.  $\triangleleft$

Proof of Claim 20. The proof is via a reduction to Lemma 17.

Let  $g(y_1, \dots, y_r, z_1, \dots, z_t)$  be an  $(r, t)$ -pseudoaddressing function. Consider the random function  $\mathbf{F}$  on  $\{0, 1\}^r$  obtained by setting the addressed variables  $z_1, \dots, z_t$  to  $\mathbf{b}_1, \dots, \mathbf{b}_t \in \{0, 1\}$  chosen i.u.a.r.. We show that there is an  $X \subseteq \{0, 1\}^r$  of size  $t$  such that the random variables  $(\mathbf{F}(a) : a \in X)$  are independent and uniformly distributed bits. Then, Lemma 17 implies the statement of the claim.

Let us see how  $X$  is defined. Let  $T$  be the decision tree guaranteed for  $g$  by virtue of the fact that it is an  $(r, t)$ -pseudoaddressing function. Further, for any  $z_j$ , let  $\pi_j^0$  and  $\pi_j^1$  be the paths satisfying P1 and P2 above. By P2, we can fix a setting  $a^{(j)} \in \{0, 1\}^r$  to the  $y$ -variables that is consistent with both paths. We set  $X = \{a^{(j)} \mid j \in [t]\}$ .

To analyze  $\mathbf{F}(a^{(j)})$ , note that setting the variables  $z_1, \dots, z_t$  to  $\mathbf{b}_1, \dots, \mathbf{b}_t$  in  $T$  gives us a (random) decision tree  $\mathbf{T}'$  that computes  $\mathbf{F}$ . In particular, the path followed by  $\mathbf{T}'$  on input  $a^{(j)}$  is uniformly chosen among  $\pi_j^0$  and  $\pi_j^1$  depending on the value of  $z_j$ , and hence  $\mathbf{F}(a^{(j)})$  is either  $\mathbf{b}_j$  or  $1 - \mathbf{b}_j$  (exactly which depends on the value of  $z_j$  that is consistent with  $\pi_j^0$  and  $\pi_j^1$ ). In either case, however,  $\mathbf{F}(a^{(j)})$  is a uniformly chosen random bit depending only on  $\mathbf{b}_j$ . Hence, the random variables  $(\mathbf{F}(a^{(j)}) : j \in [t])$  are independent and uniformly distributed.

Thus, Lemma 17 implies that with positive probability,  $\text{pdeg}_{1/10}(\mathbf{F}) = \Omega(\log t / \log r)$ . However, we know by Fact 9 that, as  $\mathbf{F}$  is a restriction of  $g$ ,  $\text{pdeg}_{1/10}(\mathbf{F}) \leq \text{pdeg}_{1/10}(g)$ . Hence, we obtain the same lower bound for  $\text{pdeg}_{1/10}(g)$ . Finally, by error reduction (Fact 9), the same lower bound (up to constant factors) holds for  $\text{pdeg}_{1/3}(g) = \text{pdeg}(g)$ .  $\triangleleft$

### 3.3 Finishing the proof of Theorem 4

Lemma 16 and Lemma 14 imply that

$$\text{pdeg}(f) = \Omega \left( \max \left\{ (\log s)^{c-o(1)}, \frac{\log(n/s^{O(1)})}{\log s} \right\} \right)$$

where  $s$  denotes the sensitivity of  $f$ . The above is minimized for  $s$  so that  $(\log s)^{c+1} = \Theta(\log n)$  (note that this implies that  $s = n^{o(1)}$ ). For this  $s$ , we get

$$\text{pdeg}(f) = \Omega((\log n)^{c/(c+1)-o(1)}) \geq (\log n)^{c/(c+1)-o(1)},$$

proving the theorem.

## 4 The Upper Bound: Proof of Theorem 5

The construction is motivated by and closely follows a construction of Ambainis and de Wolf [2], who used it to prove the existence of a truly  $n$ -variate Boolean function  $f$  whose approximate degree is  $O(\log n / \log \log n)$ . The construction of [2] uses the fact that the approximate degree of the  $\text{OR}_n$  function is  $O(\sqrt{n})$  [15]. Using our assumption that the probabilistic degree of the  $\text{OR}_n$  function is  $(\log n)^{c+o(1)}$  we are able to prove a stronger degree upper bound for probabilistic degree. In particular, Theorem 10 allows us to prove an unconditional upper bound of  $(\log n)^{(1/2)+o(1)}$  on the probabilistic degree of some  $n$ -variable function.

The construction is a variant of the Addressing function, where the addressing bits are replaced by elements of a larger alphabet  $[s]$ , which are themselves presented in an encoded form that allows them to be easily “decoded” by low-degree polynomials. More precisely, we construct the function as follows.

### Construction

Let  $s$  be a power of 2 and let  $H \subseteq \{0, 1\}^s$  be the set of codewords of the Hadamard code. That is, assume  $s = 2^t$  and identify elements of  $\{0, 1\}^s$  with functions  $h : \{0, 1\}^t \rightarrow \{0, 1\}$ . Then  $H$  consists of precisely those elements  $h \in \{0, 1\}^s$  such that  $h$  is a linear function when considered as a mapping from  $\mathbb{F}_2^s$  to  $\mathbb{F}_2$  in the natural way. The set  $H$  contains precisely  $s$  elements, say  $\{h_1, \dots, h_s\}$ .

We define a Boolean function  $f$  on  $n = sr + s^r + 1$  bits as follows. Any input  $a$  is parsed as  $a = (g_1, \dots, g_r, T, b)$ , where  $g_1, \dots, g_r : \{0, 1\}^t \rightarrow \{0, 1\}$ ,  $T : [s]^r \rightarrow \{0, 1\}$  and  $b$  is a single bit. We define  $f$  by

$$f(a) = \begin{cases} T(i_1, \dots, i_r) & \text{if } g_1, \dots, g_r \in H \text{ and } g_1 = h_{i_1}, \dots, g_r = h_{i_r}, \\ b & \text{otherwise.} \end{cases}$$

### Analysis

We have

$$f(g_1, \dots, g_r, T, b) = \sum_{i_1, \dots, i_r \in [s]} 1(g_1 = h_{i_1}, \dots, g_r = h_{i_r}) \cdot T(i_1, \dots, i_r) + (1 - 1(g_1, \dots, g_r \in H)) \cdot b. \quad (2)$$

Here  $1(\mathcal{E})$  for a Boolean predicate  $\mathcal{E}$  takes the value 1 when the Boolean predicate is satisfied and 0 otherwise.

The above implies, in particular, that the function  $f$  is truly  $n$ -variate. To see this, say the variables of  $f$  are

- $x_{j,\alpha}$  ( $j \in [r], \alpha \in \{0, 1\}^t$ ) encoding the entries of the truth tables of  $g_1, \dots, g_r$ . More formally, the variable  $x_{j,\alpha}$  is set to  $g_j(\alpha)$ .
- $y_{i_1, \dots, i_r}$  encoding the entries of  $T$ , and
- $y_0$  which gives the value of  $b$ .

Any variable  $x_{j,\alpha}$  is influential at an input  $(g_1, \dots, g_r, T, b)$  where  $g_1, \dots, g_r$  are  $h_{i_1}, \dots, h_{i_r} \in H$  respectively, and  $b \neq T(i_1, \dots, i_r)$ , which implies that flipping the value of  $x_{j,\alpha}$  at this point changes the output from  $T(i_1, \dots, i_r)$  to  $b$ . The variable  $y_{i_1, \dots, i_r}$  is also influential at the same point. The variable  $y_0$  is influential at any input where not all the  $g_i$  are in  $H$ . Thus, we see that  $f$  is indeed  $n$ -variate.

Now, we will show an upper bound on  $\text{pdeg}(f)$ . This will be done by constructing two polynomials.

- A  $1/3$ -error probabilistic polynomial  $Q(x_{j,\alpha} : j \in [r], \alpha \in \{0, 1\}^t)$  for the Boolean function  $1(g_1, \dots, g_r \in H)$ .
- For each  $i_1, \dots, i_r \in [s]$ , a polynomial  $R_{i_1, \dots, i_r}(x_{j,\alpha} : j \in [r], \alpha \in \{0, 1\}^t)$  such that at input  $(g_1, \dots, g_r) \in H^r$ ,  $R_{i_1, \dots, i_r}(g_1, \dots, g_r) = 1$  if  $g_1 = h_{i_1}, \dots, g_r = h_{i_r}$ , and 0 otherwise. (In other words,  $R_{i_1, \dots, i_r}$  computes a  $\delta$ -function on inputs from  $H^r$ . Note that we do not claim anything if  $(g_1, \dots, g_r) \notin H^r$ .)

## 42:16 Probabilistic Degree of $N$ -Variate Boolean Function

Given the above constructions, the following yields a probabilistic polynomial  $\mathbf{P}$  for  $f$ .

$$\mathbf{P} = \mathbf{Q} \cdot \left( \sum_{i_1, \dots, i_r \in [s]} R_{i_1, \dots, i_r} \cdot y_{i_1, \dots, i_r} \right) + (1 - \mathbf{Q}) \cdot y_0 \quad (3)$$

(The two copies of  $\mathbf{Q}$  are chosen with the same randomness and are *not* independent of each other.) To see that this works, fix any input  $a = (g_1, \dots, g_r, T, b)$ . If  $(g_1, \dots, g_r) \in H^r$ , the term in the parenthesis evaluates to  $T(i_1, \dots, i_r)$  with probability 1. Further,  $\mathbf{Q}(g_1, \dots, g_r)$  evaluates to 1 with probability  $2/3$ . Hence,  $\mathbf{P}(a) = T(i_1, \dots, i_r) = f(a)$  with probability at least  $2/3$ . On the other hand, if  $(g_1, \dots, g_r) \notin H^r$ , then  $\mathbf{Q}(g_1, \dots, g_r)$  evaluates to 0 with probability  $2/3$ . When this event occurs, the first summand evaluates to 0 and the second summand evaluates to  $b$ . Hence,  $\mathbf{P}(a) = b = f(a)$  with probability at least  $2/3$ .

It remains to construct the polynomials  $\mathbf{Q}$  and  $R_{i_1, \dots, i_r}$ . We start with  $\mathbf{Q}$ . Recall that a function  $g : \{0, 1\}^t \rightarrow \{0, 1\}$  lies in  $H$  when it is linear over  $\mathbb{F}_2$ , or equivalently if  $g(\alpha \oplus \beta) \oplus g(\alpha) \oplus g(\beta) = 0$  for every  $\alpha, \beta \in \{0, 1\}^t$ . Thus, the condition that  $g_1, \dots, g_r \in H$  can be rewritten as

$$\bigwedge_{j=1}^r \bigwedge_{\alpha, \beta \in \{0, 1\}^t} (1 \oplus g_j(\alpha \oplus \beta) \oplus g_j(\alpha) \oplus g_j(\beta)).$$

Let  $q(z_1, z_2, z_3)$  be a constant-degree polynomial of 3 Boolean variables that evaluates to  $1 \oplus z_1 \oplus z_2 \oplus z_3$ . Then, the above can be rewritten as  $\bigwedge_{j=1}^r \bigwedge_{\alpha, \beta \in \{0, 1\}^t} q(g_j(\alpha), g_j(\beta), g_j(\alpha \oplus \beta))$ . Thus, we can define the probabilistic polynomial to be

$$\mathbf{Q}(x_{j,\alpha} : j \in [r], \alpha \in \{0, 1\}^t) = \mathbf{Q}_1(q(x_{j,\alpha}, x_{j,\beta}, x_{j,\alpha \oplus \beta}) : j \in [r], \alpha, \beta \in \{0, 1\}^t),$$

where  $\mathbf{Q}_1$  is any probabilistic polynomial for the  $\text{AND}_{r2^{2t}} = \text{AND}_{rs^2}$  function. By assumption,  $\text{pdeg}(\text{OR}_{rs^2})$  and hence, by DeMorgan's laws,  $\text{pdeg}(\text{AND}_{rs^2})$  is at most  $\log(rs^2)^{c+o(1)} = (\log r + \log s)^{c+o(1)}$ .

We now see how to construct  $R_{i_1, \dots, i_r}$  for any fixed  $i_1, \dots, i_r \in [s]$ . Recall the standard fact (see, e.g. [22]) that for  $h_{i_1} \neq h_{i_2} \in H$ , the functions  $\hat{h}_{i_1}, \hat{h}_{i_2} : \{0, 1\}^t \rightarrow \{-1, 1\}$  defined by  $\hat{h}_{i_b}(\alpha) = 1 - 2h_{i_b}(\alpha)$ , for all  $\alpha \in \{0, 1\}^t$  and  $b \in \{1, 2\}$ , are orthogonal to one another, i.e.,  $\sum_{\alpha} \hat{h}_{i_1}(\alpha) \hat{h}_{i_2}(\alpha) = 0$ . Based on this observation, we define the polynomial as follows.

$$R_{i_1, \dots, i_r}(x_{j,\alpha} : j \in [r], \alpha \in \{0, 1\}^t) = \frac{1}{s^r} \prod_{j=1}^r \left( \sum_{\alpha \in \{0, 1\}^t} \hat{h}_{i_j}(\alpha) (1 - 2x_{j,\alpha}) \right).$$

Let us see that this polynomial has the desired properties. Consider input  $(g_1, \dots, g_r) \in H^r$ . Assume  $g_j = h_{i'_j}$  for each  $j \in [r]$ . Then, we have

$$R_{i_1, \dots, i_r}(g_1, \dots, g_r) = \frac{1}{s^r} \prod_{j=1}^r \left( \sum_{\alpha \in \{0, 1\}^t} \hat{h}_{i_j}(\alpha) (1 - 2h_{i'_j}(\alpha)) \right) = \frac{1}{s^r} \prod_{j=1}^r \left( \sum_{\alpha \in \{0, 1\}^t} \hat{h}_{i_j}(\alpha) \hat{h}_{i'_j}(\alpha) \right)$$

and the latter quantity can be seen to be 1 if  $i'_j = i_j$  for all  $j \in [r]$  and 0 otherwise. Thus,  $R_{i_1, \dots, i_r}$  behaves as stipulated. Note that  $\deg(R_{i_1, \dots, i_r}) = r$ .

This concludes the construction of the probabilistic polynomial for  $f$ . The degree of the polynomial thus constructed is at most  $\deg(\mathbf{Q}) + \max_{i_1, \dots, i_r} \deg(R_{i_1, \dots, i_r})$ , which is equal to  $O((\log r + \log s)^{c+o(1)} + r) = O((\log s)^{c+o(1)} + r)$ .

### Parameters

We set  $r = (\log s)^c = t^c$ . This gives a truly  $n$ -variate Boolean function on  $n = O(s^r) = O(2^{t^{1+c}})$  variables with probabilistic degree  $t^{c+o(1)} = (\log n)^{(c/(c+1))+o(1)}$ .



## References

- 1 Josh Alman, Timothy M. Chan, and R. Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 467–476. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.57.
- 2 Josh Alman and Ryan Williams. Probabilistic polynomials and hamming nearest neighbors. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 136–150, 2015. doi:10.1109/FOCS.2015.18.
- 3 Noga Alon and Joel H. Spencer. *The Probabilistic Method, Third Edition*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 2008. ISBN: 978-0-470-17020-5.
- 4 Andris Ambainis and Ronald de Wolf. How low can approximate degree and quantum query complexity be for total boolean functions? *Comput. Complex.*, 23(2):305–322, 2014. doi:10.1007/s00037-014-0083-2.
- 5 James Aspnes, Richard Beigel, Merrick L. Furst, and Steven Rudich. The expressive power of voting polynomials. *Comb.*, 14(2):135–148, 1994. doi:10.1007/BF01215346.
- 6 Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001. doi:10.1145/502090.502097.
- 7 R. Beigel, N. Reingold, and D. Spielman. The perceptron strikes back. In *[1991] Proceedings of the Sixth Annual Structure in Complexity Theory Conference*, pages 286–291, 1991. doi:10.1109/SCT.1991.160270.
- 8 Richard Beigel. The polynomial method in circuit complexity. In *Proceedings of the Eighth Annual Structure in Complexity Theory Conference, San Diego, CA, USA, May 18-21, 1993*, pages 82–95. IEEE Computer Society, 1993. doi:10.1109/SCT.1993.336538.
- 9 Mark Braverman. Polylogarithmic independence fools  $AC^0$  circuits. *J. ACM*, 57(5):28:1–28:10, 2010. doi:10.1145/1754399.1754401.
- 10 Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theor. Comput. Sci.*, 288(1):21–43, 2002. doi:10.1016/S0304-3975(01)00144-X.
- 11 Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In Ran Raz, editor, *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, volume 50 of *LIPICs*, pages 10:1–10:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.CCC.2016.10.
- 12 Xi Chen, Igor Carboni Oliveira, Rocco A. Servedio, and Li-Yang Tan. Near-optimal small-depth lower bounds for small distance connectivity. 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 612–625. ACM, 2016. doi:10.1145/2897518.2897534.
- 13 John Chiarelli, Pooya Hatami, and Michael E. Saks. An asymptotically tight bound on the number of relevant variables in a bounded degree boolean function. *Comb.*, 40(2):237–244, 2020. doi:10.1007/s00493-019-4136-7.
- 14 Alexander Golovnev, Alexander S. Kulikov, and R. Ryan Williams. Circuit depth reductions. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPICs*, pages 24:1–24:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.ITCS.2021.24.
- 15 Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC '96*, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery. doi:10.1145/237814.237866.
- 16 Prahladh Harsha and Srikanth Srinivasan. On polynomial approximations to  $AC^0$ . *Random Structures & Algorithms*, 54(2):289–303, 2019. doi:10.1002/rsa.20786.

- 17 Johan Håstad, Benjamin Rossman, Rocco A. Servedio, and Li-Yang Tan. An average-case depth hierarchy theorem for boolean circuits. *J. ACM*, 64(5):35:1–35:27, 2017. doi:10.1145/3095799.
- 18 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.
- 19 Adam R. Klivans and Rocco A. Servedio. Learning DNF in time  $2^{\tilde{O}(n^{1/3})}$ . *J. Comput. Syst. Sci.*, 68(2):303–318, 2004. doi:10.1016/j.jcss.2003.07.007.
- 20 N. Nisan. Crew prams and decision trees. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, page 327–335, New York, NY, USA, 1989. Association for Computing Machinery. doi:10.1145/73007.73038.
- 21 Noam Nisan and Mario Szegedy. On the degree of boolean functions as real polynomials. *Comput. Complex.*, 4:301–313, 1994. doi:10.1007/BF01263419.
- 22 Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, USA, 2014. doi:10.1017/CB09781139814782.
- 23 A. A. Razborov. Lower bounds on the dimension of schemes of bounded depth in a complete basis containing the logical addition function. *Mat. Zametki*, 41(4):598–607, 623, 1987.
- 24 Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998. ISBN: 978-0-471-98232-6.
- 25 Hans Ulrich Simon. A Tight  $\Omega(\log \log n)$ -Bound on the Time for Parallel RAM's to Compute Nondegenerated Boolean Functions. *Inf. Control.*, 55(1-3):102–106, 1982. doi:10.1016/S0019-9958(82)90477-6.
- 26 Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 77–82. ACM, 1987. doi:10.1145/28395.28404.
- 27 Jun Tarui. Probabilistic polynomials,  $AC^0$  functions and the polynomial-time hierarchy. *Theoretical Computer Science*, 113(1):167–183, 1993. doi:10.1016/0304-3975(93)90214-E.
- 28 Emanuele Viola. New lower bounds for probabilistic degree and  $AC^0$  with parity gates. *Electron. Colloquium Comput. Complex.*, 27:15, 2020. URL: <https://eccc.weizmann.ac.il/report/2020/015>.
- 29 R. Ryan Williams. Faster all-pairs shortest paths via circuit complexity. *SIAM J. Comput.*, 47(5):1965–1985, 2018. doi:10.1137/15M1024524.

## A Proof of Lemma 15

We will only use restrictions  $g'$  of  $g$  obtained by setting some inputs of  $g$  to 0. The basic observation [7, 27] is the following. For any such restriction  $g' = g_\rho$ , the function  $g'$  always agrees with  $\text{OR}_s$  at the all-zero input. Moreover, if  $a$  is non-zero and has weight  $t > 0$ , then  $g'(a) = \text{OR}_s(a) = 1$  as long as exactly  $t - 1$  of the variables that are set to 1 in  $a$  are fixed to 0 by  $\rho$  (this follows from the fact that  $g$  accepts any input of weight exactly 1). While we cannot always choose a *single* restriction that does this for all possible  $a$ , it is possible to choose a *small number of restrictions* randomly such that for each non-zero  $a$ , at least one of them is guaranteed to work with high probability. We now see the details.

For  $i \in [\log s]$ , let  $\mathcal{D}_i$  be the distribution over subsets of  $[s]$  where we pick each element independently to be in the set with probability  $2^{-i}$ . For a (constant) parameter  $p$  to be chosen later, let  $\mathcal{S}_1^i, \dots, \mathcal{S}_p^i$  be independent random subsets picked from distribution  $\mathcal{D}_i$ . Each such set  $\mathcal{S}_j^i$  is associated with the restriction  $\rho_j^i$  where each variable is set to 0 if it does not belong to  $\mathcal{S}_j^i$ , and left alive (i.e. set to  $*$ ) otherwise. Note that  $\mathbf{g}_j^i := g_{\rho_j^i}$  is a random restriction of  $g$ . Also observe that the total number of such restrictions is  $\ell := p \log s = O(\log s)$ . The final probabilistic representation is the OR of all these  $\mathbf{g}_j^i$ s.

We now prove correctness. Consider any  $a \in \{0, 1\}^s$ . The case when  $a = 0^s$  is easy, as each  $g_j^i$  is obtained by setting some inputs of  $g$  to 0 and hence  $g_j^i(a) = 0$  with probability 1. The same is therefore true for the OR of these functions.

Now assume that  $a \neq 0$ . Thus  $|a| = t \in [s]$ . Fix  $i \in [\log s]$  such that  $t \in (2^{i-1}, 2^i]$ . We will show that, with probability at least 0.9, some  $g_j^i$  evaluates to 1. This will finish the proof.

To see this, let  $S \subseteq [s]$  be the set of coordinates where  $a$  takes value 1. Note that  $g_j^i(a) = g(b_j^i)$  where  $b_j^i$  denotes the indicator vector of  $S_j^i \cap S$ . As  $g(b) = 1$  for any input  $b$  of weight 1, we see that  $g_j^i(a) = 1$  if  $|S_j^i \cap S| = 1$ . Hence, we have

$$\Pr_{g_1^i, \dots, g_p^i} [g_1^i(a) = \dots = g_p^i(a) = 0] \leq \Pr_{S_1^i, \dots, S_p^i} \left[ \bigwedge_{j=1}^p |S_j^i \cap S| \neq 1 \right] = \prod_{j=1}^p \Pr_{S_j^i} [|S_j^i \cap S| \neq 1], \quad (4)$$

where the last equality follows from the independence of the  $S_j^i$ s.

Finally, note that for any  $j$ ,

$$\begin{aligned} \Pr_{S_j^i} [|S_j^i \cap S| = 1] &= \sum_{k \in S} \Pr_{S_j^i} \left[ k \in S_j^i \wedge \bigwedge_{k' \in S \setminus k} k' \notin S_j^i \right] \\ &= t \cdot \frac{1}{2^i} \cdot \left( 1 - \frac{1}{2^i} \right)^{t-1} \geq \frac{1}{2} \cdot \left( 1 - \frac{1}{2^i} \right)^{2^{i-1}-1} \geq \frac{1}{2e}, \end{aligned}$$

where the first inequality follows from the fact that  $t \in (2^{i-1}, 2^i]$  and the second from the standard fact that  $(1 - 1/n)^{n-1} \geq 1/e$ . Plugging the above into (4), we get

$$\Pr_{g_1^i, \dots, g_p^i} [g_1^i(a) = \dots = g_p^i(a) = 0] \leq \left( 1 - \frac{1}{2e} \right)^p \leq \frac{1}{10},$$

for a large enough constant  $p$ . In particular, for this  $p$ , the probability that  $\text{OR}_\ell(g_j^i : i \in [\ell], j \in [p])$  evaluates to 0 is at most 1/10, completing the proof of the lemma.

## B Proof of the Random function lower bound (Lemma 17)

The proof is via a counting argument.

We start with a standard observation, which follows from a simple averaging argument. If  $F : \{0, 1\}^m \rightarrow \{0, 1\}$  has (1/10)-error probabilistic degree  $d$ , then for any probability distribution  $\mu$  over  $\{0, 1\}^m$ , there is a polynomial  $P$  of degree at most  $d$  such that

$$\Pr_{a \sim \mu} [P(a) = F(a)] \geq \frac{9}{10}. \quad (5)$$

Conversely, if there is a probability distribution  $\mu$  such that (5) does not hold for any polynomial of degree at most  $d$ , then  $\text{pdeg}(F) > d$ . We will take the hard distribution to be the uniform distribution over  $X$ .

More precisely, call a function  $g : X \rightarrow \{0, 1\}$  *bad* if there is a polynomial  $P$  of degree at most  $d$  that agrees with  $g$  on at least  $9|X|/10 = 9M/10$  points of  $X$ . Let  $\mathcal{B}$  be the set of bad functions. The reasoning above tells us that

$$\Pr_{\mathbf{F}} [\text{pdeg}_{1/10}(\mathbf{F}) \leq d] \leq \Pr_{\mathbf{F}} [\mathbf{F}|_X \in \mathcal{B}] = \frac{|\mathcal{B}|}{2^M}. \quad (6)$$

where for the latter inequality we have used the fact that the random variables  $(\mathbf{F}(x) : x \in X)$  are independently and uniformly distributed. Hence, it will suffice to bound  $|\mathcal{B}|$  to prove the lemma.

## 42:20 Probabilistic Degree of $N$ -Variate Boolean Function

To bound the size of  $\mathcal{B}$ , it will suffice to give a short encoding of each element of  $\mathcal{B}$ . Fix any  $g \in \mathcal{B}$  and a polynomial  $P$  that agrees with  $g$  on a set  $X' \subseteq X$  such that  $|X'| \geq 9M/10$ . Note that  $g$  can be specified by

1. The set  $X'$ .
2. The set of values of  $g$  on  $X \setminus X'$  (in some pre-determined order).
3. A polynomial  $Q$  of degree at most  $d$  that agrees with  $g$  on  $X'$  (specified as a list of coefficients of monomials).

Note that the number of choices for  $X'$  is at most  $\binom{M}{\leq M/10}$ , which is bounded by  $2^{H(1/10)M}$ , where  $H(\cdot)$  denotes the binary entropy function. Further, the number of possibilities for  $g$  on  $X \setminus X'$  is at most  $2^{|X \setminus X'|} \leq 2^{M/10}$ .

It remains to bound the number of possibilities for  $Q$ . A priori, it is not completely clear how to bound the number of  $Q$  as the coefficients of  $Q$  could be arbitrary real numbers. However, we note that if there is a polynomial  $P$  that agrees with  $g$  on  $X'$ , then there is also a  $Q$  that satisfies this property, and furthermore, the coefficients of  $Q$  are rational numbers of small bit complexity.

Formally, we will use the following lemma, which is an easy consequence of [24, Corollary 3.2d].

► **Lemma 21.** *Consider a system of linear equations  $Ax = b$  over the rational numbers, where  $A$  is an  $p \times q$  Boolean matrix, and  $b \in \{0, 1\}^p$ . Then, if the system has a real solution, it has a rational solution that can be specified (as a list of numerator-denominator pairs in binary) by at most  $10q^3$  bits.*

To use the above lemma, consider the problem of finding a polynomial  $Q$  of degree at most  $d$  that agrees with  $g$  at all points in  $X'$ . The coefficients of such a polynomial  $Q$  solve a linear system of  $p := |X'|$  many linear equations in  $q := \binom{m}{\leq d}$  variables. By the existence of the polynomial  $P$ , this system has a solution. Thus by Lemma 21, we know that there is a solution of bit-complexity at most  $10q^3 \leq m^{4d} < M/10$ . Therefore, we may always choose  $Q$  from the set  $\mathcal{Q}$  of polynomials of bit-complexity (as specified above) at most  $M/10$ . Note that  $|\mathcal{Q}| \leq 2^{M/10}$  by definition.

Overall, this gives a complete specification of any given  $g \in \mathcal{B}$ . More precisely, we have given a 1-1 map  $\tau : \mathcal{B} \rightarrow \mathcal{X} \times \mathcal{S} \times \mathcal{Q}$ , where  $\mathcal{X}$  is the collection of subsets of  $X$  of size at least  $9M/10$ ,  $\mathcal{S}$  is the set of Boolean tuples of length  $M/10$ , and  $\mathcal{Q}$  is the set of polynomials of degree at most  $d$  of bit-complexity at most  $M/10$ . Hence,  $|\mathcal{B}| \leq |\mathcal{X}| \cdot |\mathcal{S}| \cdot |\mathcal{Q}| \leq 2^{M \cdot (H(1/10) + 1/10 + 1/10)} \leq 2^{9M/10}$ . Plugging this into (6), we get

$$\Pr_{\mathbf{F}}[\text{pdeg}_{1/10}(\mathbf{F}) \leq d] \leq \frac{2^{9M/10}}{2^M} < \frac{1}{10}.$$

This finishes the proof of the lemma.

# The Swendsen-Wang Dynamics on Trees

**Antonio Blanca** ✉

Pennsylvania State University, University Park, PA, USA

**Zongchen Chen** ✉

Georgia Institute of Technology, Atlanta, GA, USA

**Daniel Štefankovič** ✉

University of Rochester, NY, USA

**Eric Vigoda** ✉

Georgia Institute of Technology, Atlanta, GA, USA

---

## Abstract

The Swendsen-Wang algorithm is a sophisticated, widely-used Markov chain for sampling from the Gibbs distribution for the ferromagnetic Ising and Potts models. This chain has proved difficult to analyze, due in part to the global nature of its updates. We present optimal bounds on the convergence rate of the Swendsen-Wang algorithm for the complete  $d$ -ary tree. Our bounds extend to the non-uniqueness region and apply to all boundary conditions. We show that the spatial mixing conditions known as *Variance Mixing* and *Entropy Mixing*, introduced in the study of local Markov chains by Martinelli et al. (2003), imply  $\Omega(1)$  spectral gap and  $O(\log n)$  mixing time, respectively, for the Swendsen-Wang dynamics on the  $d$ -ary tree. We also show that these bounds are asymptotically optimal. As a consequence, we establish  $\Theta(\log n)$  mixing for the Swendsen-Wang dynamics for *all* boundary conditions throughout the tree uniqueness region; in fact, our bounds hold beyond the uniqueness threshold for the Ising model, and for the  $q$ -state Potts model when  $q$  is small with respect to  $d$ . Our proofs feature a novel spectral view of the Variance Mixing condition inspired by several recent rapid mixing results on high-dimensional expanders and utilize recent work on block factorization of entropy under spatial mixing conditions.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Random walks and Markov chains; Mathematics of computing  $\rightarrow$  Markov processes; Theory of computation  $\rightarrow$  Design and analysis of algorithms

**Keywords and phrases** Markov Chains, mixing times, Ising and Potts models, Swendsen-Wang dynamics, trees

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.43

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2007.08068>

**Funding** *Antonio Blanca*: Research supported in part by NSF grant CCF-1850443.

*Zongchen Chen*: Research supported in part by NSF grant CCF-2007022.

*Daniel Štefankovič*: Research supported in part by NSF grant CCF-2007287.

*Eric Vigoda*: Research supported in part by NSF grant CCF-2007022.

## 1 Introduction

Spin systems are idealized models of a physical system in equilibrium which are utilized in statistical physics to study phase transitions. A phase transition occurs when there is a dramatic change in the macroscopic properties of the system resulting from a small (infinitesimal in the limit) change in one of the parameters defining the spin system. The macroscopic properties of the system manifest with the persistence (or lack thereof) of long-



© Antonio Blanca, Zongchen Chen, Daniel Štefankovič, and Eric Vigoda;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 43; pp. 43:1–43:15



Leibniz International Proceedings in Informatics

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

range influences. There is a well-established mathematical theory connecting the absence of these influences to the fast convergence of Markov chains. In this paper, we study this connection on the regular tree, known as the *Bethe lattice* in statistical physics [7, 27].

The most well-studied example of a spin system is the *ferromagnetic  $q$ -state Potts model*, which contains the *Ising model* ( $q = 2$ ) as a special case. The Potts model is especially important as fascinating phase transitions (first-order vs. second-order) are now understood rigorously in various contexts [5, 21, 20, 17, 18].

Given a graph  $G = (V, E)$ , configurations of the Potts model are assignments of spins  $[q] = \{1, 2, \dots, q\}$  to the vertices of  $G$ . The parameter  $\beta > 0$  (corresponding to the inverse of the temperature of the system) controls the strength of nearest-neighbor interactions, and the probability of a configuration  $\sigma \in [q]^V$  in the *Gibbs distribution* is such that

$$\mu(\sigma) = \mu_G(\sigma) = \frac{e^{-\beta|D(\sigma)|}}{Z}, \quad (1)$$

where  $D(\sigma) = \{\{v, w\} \in E : \sigma(v) \neq \sigma(w)\}$  denotes the set of bi-chromatic edges in  $\sigma$ , and  $Z$  is the normalizing constant known as the *partition function*.

The *Glauber dynamics* is the simplest example of a Markov chain for sampling from the Gibbs distribution; it updates the spin at a randomly chosen vertex in each step. In many settings, as we detail below, the Glauber dynamics converges exponentially slow at low temperatures (large  $\beta$ ) due to the local nature of its transitions and the long-range correlations in the Gibbs distribution. Of particular interest are thus “global” Markov chains such as the *Swendsen-Wang (SW) dynamics* [49, 23], which update a large fraction of the configuration in each step, thus potentially overcoming the obstacles that hinder the performance of the Glauber dynamics, and with steps that can be efficiently parallelized [4].

The SW dynamics utilizes a close connection between the Potts model and an alternative representation known as the *random-cluster model*. The random-cluster model is defined on subsets of edges and is not a spin system as the weight of a configuration depends on the global connectivity properties of the corresponding subgraph. The transitions of the SW dynamics take a spin configuration, transform it to a “joint” spin-edge configuration, perform a step in the joint space, and then map back to a Potts configuration. Formally, from a Potts configuration  $\sigma_t \in [q]^V$ , a transition  $\sigma_t \rightarrow \sigma_{t+1}$  is defined as follows:

1. Let  $M_t = M(\sigma_t) = E \setminus D(\sigma_t)$  denote the set of monochromatic edges in  $\sigma_t$ .
2. Independently for each edge  $e = \{v, w\} \in M_t$ , keep  $e$  with probability  $p = 1 - \exp(-\beta)$  and remove  $e$  with probability  $1 - p$ . Let  $A_t \subseteq M_t$  denote the resulting subset.
3. In the subgraph  $(V, A_t)$ , independently for each connected component  $C$  (including isolated vertices), choose a spin  $s_C$  uniformly at random from  $[q]$  and assign to each vertex in  $C$  the spin  $s_C$ . This spin assignment defines  $\sigma_{t+1}$ .

There are two standard measures of the convergence rate of a Markov chain. The *mixing time* is the number of steps to get within total variation distance  $\leq 1/4$  of its stationary distribution from the worst starting state. The *relaxation time* is the inverse of the spectral gap of the transition matrix of the chain and measures the speed of convergence from a “warm start”. For approximate counting algorithms the relaxation time is quite useful as it corresponds to the “resample” time [30, 37, 35, 34]; see Section 2 for precise definitions and how these two notions relate to each other.

There has been great progress in formally connecting phase transitions with the convergence rate of the Glauber dynamics. Notably, for the  $d$ -dimensional integer lattice  $\mathbb{Z}^d$ , a series of works established that a spatial mixing property known as *strong spatial mixing (SSM)*

implies  $O(n \log n)$  mixing time of the Glauber dynamics [39, 15, 22]. Roughly speaking, SSM says that correlations decay exponentially fast with the distance and is also known to imply optimal mixing and relaxation times of the SW dynamics on  $\mathbb{Z}^d$  [9, 8]. These techniques utilizing SSM are particular to the lattice and do not extend to *non-amenable* graphs (i.e., those whose boundary and volume are of the same order). The  $d$ -ary complete tree, which is the focus of this paper, is the prime example of a non-amenable graph.

On the regular  $d$ -ary tree, there are two fundamental phase transitions: the *uniqueness* threshold  $\beta_u$  and the *reconstruction* threshold  $\beta_r$ . The smaller of these thresholds  $\beta_u$  corresponds to the uniqueness/non-uniqueness phase transition of the Gibbs measure on the infinite  $d$ -ary tree, and captures whether the worst-case boundary configuration (i.e., a fixed configuration on the leaves of a finite tree) has an effect or not on the spin at the root (in the limit as the height of the tree grows). The second threshold  $\beta_r$  is the reconstruction/non-reconstruction phase transition, marking the divide on whether or not a *random* boundary condition (in expectation) affects the spin of the root.

There is a large body of work on the interplay between these phase transitions and the speed of convergence of the Glauber dynamics on the complete  $d$ -ary tree [41, 40, 6], and more generally on bounded degree graphs [44, 28, 13]. Our main contributions in this paper concern instead the speed of convergence of the SW dynamics on trees, how it is affected by these phase transitions, and the effects of the boundary condition.

Martinelli, Sinclair, and Weitz [40, 41] introduced a pair of spatial mixing (decay of correlation) conditions called *Variance Mixing (VM)* and *Entropy Mixing (EM)* which capture the exponential decay of point-to-set correlations. More formally, the VM and EM conditions hold when there exist constants  $\ell > 0$  and  $\varepsilon = \varepsilon(\ell)$  such that, for every vertex  $v \in T$ , the influence of the spin at  $v$  on the spins of the vertices at distance  $\geq \ell$  from  $v$  in the subtree  $T_v$  rooted at  $v$  decays by a factor of at least  $\varepsilon$ . For the case of VM, this decay of influence is captured in terms of the variance of any function  $g$  that depends only on the spins of the vertices in  $T_v$  at distance  $\geq \ell$  from  $v$ ; specifically, when conditioned on the spin at  $v$ , the conditional variance of  $g$  is (on average) a factor  $\varepsilon$  smaller than the unconditional variance; see Definition 7 in Section 3 for the formal definition. EM is defined analogously, with variance replaced by entropy; see Definition 15.

It was established in [40, 41] that VM and EM imply optimal bounds on the convergence rate of the Glauber dynamics on trees. We obtain optimal bounds for the speed of convergence of the SW dynamics under the same VM and EM spatial mixing conditions.

► **Theorem 1.** *For all  $q \geq 2$  and  $d \geq 3$ , for the  $q$ -state ferromagnetic Ising/Potts model on an  $n$ -vertex complete  $d$ -ary tree, Variance Mixing implies that the relaxation time of the Swendsen-Wang dynamics is  $\Theta(1)$ .*

► **Theorem 2.** *For all  $q \geq 2$  and  $d \geq 3$ , for the  $q$ -state ferromagnetic Ising/Potts model on an  $n$ -vertex complete  $d$ -ary tree, Entropy Mixing implies that the mixing time of the Swendsen-Wang dynamics is  $O(\log n)$ .*

The VM condition is strictly weaker (i.e., easier to satisfy) than the EM condition, but, at the moment, EM is known to hold in the same parameter regimes as VM. The relaxation time bound in Theorem 1 is weaker than the mixing time bound in Theorem 2. We also show that the mixing time in Theorem 2 is asymptotically the best possible.

► **Theorem 3.** *For all  $q \geq 2$ ,  $d \geq 3$  and any  $\beta > 0$ , the mixing time of the SW dynamics on an  $n$ -vertex complete  $d$ -ary tree is  $\Omega(\log n)$  for any boundary condition.*

We remark that the mixing time lower bound in Theorem 3 applies to all inverse temperatures  $\beta$  and all boundary conditions.

The VM and EM conditions are properties of the Gibbs distribution induced by a *specific* boundary condition on the leaves of the tree; this contrasts with other standard notions of decay of correlations such as SSM on  $\mathbb{Z}^d$ . This makes these conditions quite suitable for understanding the speed of convergence of Markov chains under different boundary conditions. For instance, [40, 41] established VM and EM for all boundary conditions provided  $\beta < \max\{\beta_u, \frac{1}{2} \ln(\frac{\sqrt{d}+1}{\sqrt{d}-1})\}$  and for the monochromatic (e.g., all-red) boundary condition for all  $\beta$ . Consequently, we obtain the following results.

► **Theorem 4.** *For all  $q \geq 2$  and  $d \geq 3$ , for the  $q$ -state ferromagnetic Ising/Potts model on an  $n$ -vertex complete  $d$ -ary tree, the relaxation time of the Swendsen-Wang dynamics is  $\Theta(1)$  and its mixing time is  $\Theta(\log n)$  in the following cases:*

1. *the boundary condition is arbitrary and  $\beta < \max\left\{\beta_u, \frac{1}{2} \ln\left(\frac{\sqrt{d}+1}{\sqrt{d}-1}\right)\right\}$ ;*
2. *the boundary condition is monochromatic and  $\beta$  is arbitrary.*

Part (i) of this theorem provides optimal mixing and relaxation times bounds for the SW dynamics under arbitrary boundaries throughout the uniqueness region  $\beta < \beta_u$ . In fact,  $\beta_u < \frac{1}{2} \ln(\frac{\sqrt{d}+1}{\sqrt{d}-1})$  when  $q \leq 2(\sqrt{d} + 1)$  and thus our bound extends to the non-uniqueness region for many combinations of  $d$  and  $q$ . We note that while the value of the uniqueness threshold  $\beta_u$  is known, it does not have a closed form (see [31, 10]). In contrast, the reconstruction threshold  $\beta_r$  is not known for the Potts model [47, 43], but one would expect that part (i) holds for all  $\beta < \beta_r$ ; analogous results are known for the Glauber dynamics for other spin systems where more precise bounds on the reconstruction threshold have been established [6, 45, 48].

Previously, only a  $\text{poly}(n)$  bound was known for the mixing time of the SW dynamics for arbitrary boundary conditions [50, 6]. This  $\text{poly}(n)$  bound holds for every  $\beta$ , but the degree of the polynomial bounding the mixing time is quite large (grows with  $\beta$ ); our bound in part (i) is thus a substantial improvement.

In regards to part (ii) of the theorem, we note that our bound holds for all  $\beta$ , including the whole low-temperature region. The only other case where tight bounds for the SW dynamics are known for the full low-temperature regime is on the geometrically simpler complete graph [25, 12].

Previous (direct) analysis of the speed of convergence of the SW dynamics on trees focused exclusively on the special case of the *free* boundary condition [33, 16], where the dynamics is much simpler as the corresponding random-cluster model is trivial (reduces to independent bond percolation); this was used by Huber [33] to establish  $O(\log n)$  mixing time of the SW dynamics for all  $\beta$  for the special case of the free boundary condition.

We comment briefly on our proof methods next; a more detailed exposition of our approach is provided later in this introduction. The results in [40, 41] use the VM and EM condition to deduce optimal bounds for the relaxation and mixing times of the Glauber dynamics; specifically, they analyze its spectral gap and log-Sobolev constant. Their methods do not extend to the SW dynamics. It can be checked, for example, that the log-Sobolev constant for the SW dynamics is  $\Theta(n^{-1})$ , and thus the best possible mixing time one could hope to obtain with such an approach would be  $O(n \log n)$ . For Theorem 2, we utilize instead new tools introduced by Caputo and Parisi [14] to establish a (block) factorization of entropy. This factorization allows to get a handle on the *modified* log-Sobolev constant for the SW dynamics. For Theorem 1, the main novelty in our approach is a new spectral interpretation of the VM condition that facilitates a factorization of variance, similar to the factorization of entropy from [14]. Lastly, the lower bound from Theorem 3 is obtained by adapting the framework of Hayes and Sinclair [32] to the SW setting using recent ideas from [8].



Finally, we mention that part (ii) of Theorem 4 has interesting implications related to the speed of convergence of random-cluster model Markov chains on trees under the wired boundary condition. That is, all the leaves are connected through external or “artificial” wirings. The case of the wired boundary condition is the most studied version of the random-cluster model on trees (see, e.g., [31, 36]) since, as mentioned earlier, the model is trivial under the free boundary. The random-cluster model, which is parameterized by  $p \in (0, 1)$  and  $q > 0$  (see [24, 1] for its definition), is intimately connected to the ferromagnetic  $q$ -state Potts model when  $q \geq 2$  is an integer and  $p = 1 - \exp(-\beta)$ . In particular, there is a variant of SW dynamics for the random-cluster model (by observing the edge configuration after the second step of the chain).

Another standard Markov chain for the random-cluster model is the heat-bath (edge) dynamics, which is the analog of the Glauber dynamics on spins for random-cluster configurations. Our results for the random-cluster dynamics are the following.

► **Theorem 5.** *For all integer  $q \geq 2$ ,  $p \in (0, 1)$ , and  $d \geq 3$ , for the random-cluster model on an  $n$ -vertex complete  $d$ -ary tree with wired boundary condition, the mixing time of the Swendsen-Wang dynamics is  $O(\log n)$ . In addition, the mixing time of the heat-bath edge dynamics for the random-cluster model is  $O(n \log n)$ .*

To prove these results, we use a factorization of entropy in the joint spin-edge space, as introduced in [8]; they cannot be deduced from the mixing time bounds for the Glauber dynamics for the Potts model in [40, 41].

Our final result shows that while random-cluster dynamics mix quickly under the wired boundary condition, there are random-cluster boundary conditions that cause an exponential slowdown for both the SW dynamics and the heat-bath edge dynamics for the random-cluster model.

► **Theorem 6.** *For all  $q \geq 2$ , all  $d \geq 3$ , consider the random-cluster model on an  $n$ -vertex complete  $d$ -ary tree. Then, there exists  $p \in (0, 1)$  and a random-cluster boundary condition such that the mixing times of the Swendsen-Wang dynamics and of the heat-bath edge dynamics is  $\exp(\Omega(\sqrt{n}))$ .*

We prove this result extending ideas from [11]. In particular, we prove a general theorem that allows us to transfer slow mixing results for the edge dynamics on other graphs to the tree, for a carefully constructed tree boundary condition and a suitable  $p$ . To prove this results we use the random-cluster boundary condition to embed an arbitrary graph  $G$  on the tree; a set with bad conductance for the chain on  $G$  is then lifted to the tree. Theorem 6 then follows from any of the known slow mixing results for the edge dynamics [29, 26, 50].

**Our techniques.** Our first technical contribution is a reinterpretation and generalization of the VM condition as a bound on the second eigenvalue of a certain stochastic matrix which we denote by  $P^\uparrow P^\downarrow$ . The matrices  $P^\uparrow$  and  $P^\downarrow$  are distributional matrices corresponding to the distribution at a vertex  $v$  given the spin configuration of the set  $S_v$  of all its descendants at distance at least  $\ell$  and vice versa. These matrices are inspired by the recent results in [2, 3] utilizing high-dimensional expanders; see Section 3 for their precise definitions.

Our new spectral interpretation of the VM condition allows us to factorize it and obtain an equivalent global variant we call *Parallel Variance Mixing (PVM)*. While the VM condition signifies the exponential decay with distance of the correlations between a vertex  $v$  and the set  $S_v$  (and is well-suited for the analysis of local Markov chains), the PVM condition captures instead the decay rate of set-to-set correlations between the set of all the vertices

at a fixed level of the tree and the set of all their descendants at distance at least  $\ell$ . The PVM condition facilitates the analysis of a block dynamics with a constant number of blocks each of linear volume. We call this variant of block dynamics the *tiled block dynamics* as each block consists of a maximal number of non-intersecting subtrees of constant size (i.e., a *tiling*); see Figure 1. We use the PVM condition to show that the spectral gap of the tiled block dynamics is  $\Omega(1)$ , and a generic comparison between the block dynamics and the SW dynamics yields Theorem 1.

Our proof of Theorem 2 follows a similar strategy. We first obtain a global variant of the EM condition, analogous to the PVM condition but for entropy. For this, we use a recent result of Caputo and Parisi [14]. From this global variant of the EM condition we deduce a factorization of entropy into the even and odd subsets of vertices. (The parity of a vertex is that of its distance to the leaves of the tree.) The even-odd factorization of entropy was recently shown in [8] to imply  $O(\log n)$  mixing of the SW on general biparte graphs.

**Paper organization.** The rest of the paper is organized as follows. Section 2 contains some standard definitions and facts we use in our proofs. In Sections 3 and 4 we prove Theorems 1 and 2, respectively. Our general comparison result between the SW dynamics and the block dynamics, our results for the random-cluster model dynamics, and our lower bound for the SW dynamics (Theorem 3) are proved in the full version of this paper [1].

## 2 Preliminaries

We introduce some notations and facts that are used in the remainder of the paper.

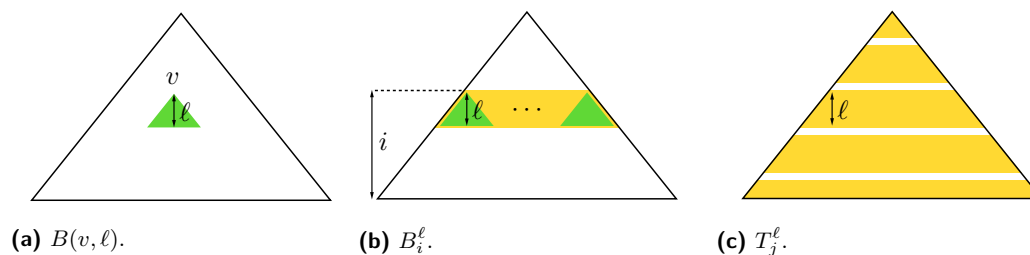
**The Potts model on the  $d$ -ary tree.** For  $d \geq 2$ , let  $\mathbb{T}^d = (\mathbb{V}, \mathbb{E})$  denote the rooted infinite  $d$ -ary tree in which every vertex (including the root) has exactly  $d$  children. We consider the complete finite subtree of  $\mathbb{T}^d$  of height  $h$ , which we denote by  $T = T_h^d = (V(T), E(T))$ . We use  $\partial T$  to denote the external boundary of  $T$ ; i.e., the set of vertices in  $\mathbb{V} \setminus V(T)$  incident to the leaves of  $T$ . We identify subgraphs of  $T$  with their vertex sets. In particular, for  $A \subseteq V(T)$  we use  $E(A)$  for the edges with both endpoints in  $A$ ,  $\partial A$  for the external boundary of  $A$  (i.e., the vertices in  $(T \cup \partial T) \setminus A$  adjacent to  $A$ ), and, with a slight abuse of notation, we write  $A$  also for the induced subgraph  $(A, E(A))$ . When clear from context, we simply use  $T$  for the vertex set  $V(T)$ .

A configuration of the Potts model is an assignment of spins  $[q] = \{1, \dots, q\}$  to the vertices of the graph. For a fixed spin configuration  $\tau$  on the infinite tree  $\mathbb{T}^d$ , we use  $\Omega^\tau = [q]^{T \cup \partial T}$  to denote the set of configurations of  $T$  that agree with  $\tau$  on  $\partial T$ . Hence,  $\tau$  specifies a boundary condition for  $T$ . More generally, for any  $A \subseteq T$  and any  $\eta \in \Omega^\tau$ , let  $\Omega_A^\eta \subseteq \Omega^\tau$  denote the set of configurations of  $T$  that agree with  $\eta$  on  $(T \cup \partial T) \setminus A$ . We use  $\mu_A^\eta$  to denote the Gibbs distribution over  $\Omega_A^\eta$ , so for  $\sigma \in \Omega_A^\eta$  we have

$$\mu_A^\eta(\sigma) := \frac{1}{Z} \exp\left(-\beta \sum_{\{u,v\} \in E(A \cup \partial A)} \mathbb{1}(\sigma_u \neq \sigma_v)\right),$$

where  $Z$  is a normalizing constant (or partition function). For  $\sigma \notin \Omega_A^\eta$ , we set  $\mu_A^\eta(\sigma) = 0$ .

**The tiled block dynamics.** Let  $\mathcal{U} = \{U_1, \dots, U_r\}$  be a collection of subsets (or blocks) such that  $T = \bigcup_i U_i$ . The (heat-bath) block dynamics with blocks  $\mathcal{U}$  is a standard Markov chain for the Gibbs distribution  $\mu_T^\tau$ . If the configuration at time  $t$  is  $\sigma_t$ , the next configuration  $\sigma_{t+1}$  is generated as follows:



■ **Figure 1** An illustration of the sets  $B(v, \ell)$ ,  $B_i^\ell$ , and  $T_j^\ell$ , where  $\ell$  represents the number of levels.

1. Pick an integer  $j \in \{1, 2, \dots, r\}$  uniformly at random;
2. Draw a sample  $\sigma_{t+1}$  from the conditional Gibbs distribution  $\mu_{U_j}^{\sigma_t}$ ; that is, update the configuration in  $U_j$  with a new configuration distributed according to the conditional measure in  $U_j$  given the configuration of  $\sigma_t$  on  $(T \cup \partial T) \setminus U_j$  and the boundary condition  $\tau$ .

We consider a special choice of blocks, where each block is a disjoint union of small subtrees of constant height forming a tiling structure. For  $0 \leq i \leq h + 1$ , let  $L_i$  denote the set of vertices of  $T$  that are of distance exactly  $i$  from the boundary  $\partial T$ ; in particular,  $L_0 = \emptyset$  and  $L_{h+1}$  contains only the root of  $T$ . (It will be helpful to define  $L_i = \emptyset$  for  $i < 0$  or  $i > h + 1$ .) Let  $F_i = \cup_{j \leq i} L_j$  be the set of vertices at distance at most  $i$  from  $\partial T$ ; then  $F_0 = \emptyset$  and  $F_{h+1} = T$ . We further define  $F_i = \emptyset$  for  $i < 0$  and  $F_i = T$  for  $i > h + 1$ . For each  $i \in \mathbb{N}^+$  let

$$B_i^\ell = F_i \setminus F_{i-\ell} = \bigcup_{i-\ell < j \leq i} L_j. \tag{2}$$

In words,  $B_i^\ell$  is the collection of all the subtrees of  $T$  of height  $\ell - 1$  with roots at distance exactly  $i$  from  $\partial T$ ; see Figure 1(b). Finally, for each  $1 \leq j \leq \ell + 1$ , we define

$$T_j^\ell = \bigcup_{0 \leq k \leq \frac{h+\ell-j}{\ell+1}} B_{j+k(\ell+1)}^\ell. \tag{3}$$

The set  $T_j^\ell$  contains all the subtrees of  $T$  whose roots are at distance  $j + k(\ell + 1)$  from  $\partial T$  for some non-negative integer  $k$ ; the height of each subtree (except the top and bottom ones) is  $\ell - 1$ . Also notice that all the subtrees in  $T_j^\ell$  are at (graph) distance at least 2 from each other, and thus they create a tiling pattern over  $T$ . Therefore, we call the block dynamics with blocks  $\mathcal{U} = \{T_1^\ell, \dots, T_{\ell+1}^\ell\}$  the *tilted block dynamics*; see Figure 1(c). The transition matrix of the tiled block dynamics is denoted by  $P_{\text{TB}}$ .

**Mixing and relaxation times.** Let  $P$  be the transition matrix of an ergodic Markov chain over a finite set  $\Phi$  with stationary distribution  $\nu$ . We use  $P^t(X_0, \cdot)$  to denote the distribution of the chain after  $t$  steps starting from  $X_0 \in \Phi$ . The mixing time of  $P$  is defined as  $\tau_{\text{mix}}(P) = \max_{X_0 \in \Phi} \min \{t \geq 0 : \|P^t(X_0, \cdot) - \nu\|_{\text{TV}} \leq 1/4\}$ , where  $\|\cdot\|_{\text{TV}}$  denotes total variation distance.

When  $P$  is reversible, its spectrum is real and we let  $1 = \lambda_1 > \lambda_2 \geq \dots \geq \lambda_{|\Phi|} \geq -1$  denote its eigenvalues ( $1 > \lambda_2$  when  $P$  is irreducible). The *absolute spectral gap* of  $P$  is defined by  $\text{gap}(P) = 1 - \lambda^*$ , where  $\lambda^* = \max\{|\lambda_2|, |\lambda_{|\Phi|}|\}$ . If  $P$  is ergodic (i.e., irreducible and aperiodic), then  $\text{gap}(P) > 0$ , and it is a standard fact that if  $\nu_{\min} = \min_{x \in \Phi} \nu(x)$ , then

$$(\text{gap}(P)^{-1} - 1) \log 2 \leq \tau_{\text{mix}}(P) \leq \text{gap}(P)^{-1} \log(4\nu_{\min}^{-1}); \tag{4}$$

see [38]. The *relaxation time* of the chain is defined as  $\text{gap}(P)^{-1}$ .

**Analytic tools.** We review next some useful tools from functional analysis; we refer the reader to [42, 46] for more extensive background. We can endow  $\mathbb{R}^\Phi$  with the inner product  $\langle f, g \rangle_\nu = \sum_{x \in \Phi} f(x)g(x)\nu(x)$  for two functions  $f, g : \Phi \rightarrow \mathbb{R}$ . The resulting Hilbert space is denoted by  $L_2(\nu) = (\mathbb{R}^\Phi, \langle \cdot, \cdot \rangle_\nu)$  and  $P$  defines an operator from  $L_2(\nu)$  to  $L_2(\nu)$ .

Let  $\mathbf{1} : \Phi \rightarrow \mathbb{R}$  be the constant “all 1” function (i.e.,  $\mathbf{1}(x) = 1 \forall x \in \Phi$ ) and let  $I$  denote the identity mapping over all functions (i.e.,  $If = f$  for all  $f : \Phi \rightarrow \mathbb{R}$ ). We then define:

$$\begin{aligned} \mathbb{E}_\nu(f) &= \sum_{x \in \Phi} f(x)\nu(x) = \langle f, \mathbf{1} \rangle_\nu, \text{ and} \\ \text{Var}_\nu(f) &= \mathbb{E}_\nu(f^2) - \mathbb{E}_\nu(f)^2 = \langle f, (I - \mathbf{1}\nu)f \rangle_\nu \end{aligned}$$

as the expectation and variance of the function  $f$  with respect to (w.r.t.) the measure  $\nu$ . Likewise, for a function  $f : \Omega \rightarrow \mathbb{R}_{\geq 0}$  we define the entropy of  $f$  with respect to  $\nu$  as  $\text{Ent}_\nu(f) = \mathbb{E}_\nu \left[ f \log \left( \frac{f}{\mathbb{E}_\nu(f)} \right) \right]$ .

Often, we will consider  $\nu$  to be the conditional Gibbs distribution  $\mu_A^\eta$  for some  $A \subseteq T$  and  $\eta \in \Omega$ . In those cases, to simplify the notation, we shall write  $\mathbb{E}_A^\eta(f)$  for  $\mathbb{E}_{\mu_A^\eta}(f)$ ,  $\text{Var}_A^\eta(f)$  for  $\text{Var}_{\mu_A^\eta}(f)$ , and  $\text{Ent}_A^\eta(f)$  for  $\text{Ent}_{\mu_A^\eta}(f)$ .

The *Dirichlet form* of a reversible Markov chain with transition matrix  $P$  is defined as

$$\mathcal{E}_P(f, f) = \langle f, (I - P)f \rangle_\nu = \frac{1}{2} \sum_{x, y \in \Phi} \nu(x)P(x, y)(f(x) - f(y))^2, \quad (5)$$

for any  $f : \Phi \rightarrow \mathbb{R}$ . We say  $P$  is *positive semidefinite* if  $\langle f, Pf \rangle_\nu \geq 0$  for all functions  $f : \Phi \rightarrow \mathbb{R}$ . In this case  $P$  has only nonnegative eigenvalues. If  $P$  is positive semidefinite, then the absolute spectral gap of  $P$  satisfies

$$\text{gap}(P) = 1 - \lambda_2 = \inf_{\substack{f: \Phi \rightarrow \mathbb{R} \\ \text{Var}_\nu(f) \neq 0}} \frac{\mathcal{E}_P(f, f)}{\text{Var}_\nu(f)}. \quad (6)$$

### 3 Variance Mixing implies fast mixing: Proof of Theorem 1

We start with the formal definition of the *Variance Mixing (VM)* condition introduced by Martinelli, Sinclair and Weitz [40]. Throughout this section, we consider the Potts model on the  $n$ -vertex  $d$ -ary complete tree  $T = T_h^d$  with a fixed boundary condition  $\tau$ ; hence, for ease of notation we set  $\mu := \mu_T^\tau$  and  $\Omega := \Omega^\tau$ .

For  $v \in T$ , let  $T_v$  denote the subtree of  $T$  rooted at  $v$ . For boundary condition  $\eta \in \Omega$  and a function  $g : \Omega_{T_v}^\eta \rightarrow \mathbb{R}$ , we define the function  $g_v : [q] \rightarrow \mathbb{R}$  as the conditional expectation

$$g_v(a) = \mathbb{E}_{T_v}^\eta [g \mid \sigma_v = a] = \sum_{\sigma \in \Omega_{T_v}^\eta : \sigma_v = a} \mu_{T_v}^\eta(\sigma \mid \sigma_v = a)g(\sigma). \quad (7)$$

In words,  $g_v(a)$  is the conditional expectation of the function  $g$  under the distribution  $\mu_{T_v}^\eta$  given that the root of  $T_v$  (i.e, the vertex  $v$ ) is set to spin  $a \in [q]$ . We also consider the expectation and variance of  $g_v$  w.r.t. the projection of  $\mu_{T_v}^\eta$  on  $v$ . In particular,

$$\begin{aligned} \mathbb{E}_{T_v}^\eta [g_v] &= \sum_{a \in [q]} \mu_{T_v}^\eta(\sigma_v = a)g_v(a) = \mathbb{E}_{T_v}^\eta [g], \text{ and} \\ \text{Var}_{T_v}^\eta [g_v] &= \mathbb{E}_{T_v}^\eta [g_v^2] - \mathbb{E}_{T_v}^\eta [g_v]^2. \end{aligned}$$

For an integer  $\ell \geq 1$ , we define  $B(v, \ell)$  as the set of vertices of  $T_v$  that are at distance less than  $\ell$  from  $v$ ; see Figure 1(a). We say that the function  $g : \Omega_{T_v}^\eta \rightarrow \mathbb{R}$  is independent of the configuration on  $B(v, \ell)$  if for all  $\sigma, \sigma' \in \Omega_{T_v}^\eta$  such that  $\sigma(B(v, \ell)) \neq \sigma'(B(v, \ell))$  and  $\sigma(T_v \setminus B(v, \ell)) = \sigma'(T_v \setminus B(v, \ell))$ , we have  $g(\sigma) = g(\sigma')$ . We can now define VM.

► **Definition 7** (Variance Mixing (VM)). *The Gibbs distribution  $\mu = \mu_T^\tau$  satisfies  $\text{VM}(\ell, \varepsilon)$  if for every  $v \in T$ , every  $\eta \in \Omega$ , and every function  $g : \Omega_{T_v}^\eta \rightarrow \mathbb{R}$  that is independent of the configuration on  $B(v, \ell)$ , we have  $\text{Var}_{T_v}^\eta(g_v) \leq \varepsilon \cdot \text{Var}_{T_v}^\eta(g)$ . We say that the VM condition holds if there exist constants  $\ell$  and  $\varepsilon = \varepsilon(\ell)$  such that  $\text{VM}(\ell, \varepsilon)$  holds.*

The VM condition is a spatial mixing property that captures the rate of decay of correlations, given by  $\varepsilon = \varepsilon(\ell)$ , with the distance  $\ell$  between  $v \in T$  and the set  $T_v \setminus B(v, \ell)$ . To see this, note that, roughly speaking,  $\text{Var}_{T_v}^\eta(g_v)$  is small when  $g_v(a) = \mathbb{E}_{T_v}^\eta[g \mid \sigma_v = a]$  is close to  $g_v(b) = \mathbb{E}_{T_v}^\eta[g \mid \sigma_v = b]$  for every  $a \neq b$ . Since  $g$  is independent of the configuration on  $B(v, \ell)$ , this can only happen if the spin at  $v$ , which is at distance  $\ell$  from  $T_v \setminus B(v, \ell)$ , has only a small influence on the projections of the conditional measures  $\mu_{T_v}^\eta(\cdot \mid \sigma_v = a)$ ,  $\mu_{T_v}^\eta(\cdot \mid \sigma_v = b)$  to  $T_v \setminus B(v, \ell)$ .

It was established in [40, 41] that VM implies optimal mixing of the Glauber dynamics; this was done by analyzing a block dynamics that updates one random block  $B(v, \ell)$  in each step. This block dynamics behaves similarly to the Glauber dynamics since all blocks are of constant size, and there are a linear number of them; see [40, 41] for further details. Our goal here is to establish optimal mixing of global Markov chains, and thus we require a different spatial mixing condition that captures decay of correlations in a more global manner. For this, we introduce the notion of *Parallel Variance Mixing (PVM)*. Recall that for  $0 \leq i \leq h + 1$ ,  $L_i$  is the set all vertices at distance exactly  $i$  from the boundary  $\partial T$ ,  $F_i = \cup_{j \leq i} L_j$ , and  $B_i^\ell = F_i \setminus F_{i-\ell}$ ; see Figures 1(b) and 1(c).

For  $1 \leq i \leq h + 1$ ,  $\eta \in \Omega$  and  $g : \Omega_{F_i}^\eta \rightarrow \mathbb{R}$ , consider the function  $g_{L_i} : [q]^{L_i} \rightarrow \mathbb{R}$  given by

$$g_{L_i}(\xi) = \mathbb{E}_{F_i}^\eta[g \mid \sigma_{L_i} = \xi] = \sum_{\sigma \in \Omega_{F_i}^\eta : \sigma_{L_i} = \xi} \mu_{F_i}^\eta(\sigma \mid \sigma_{L_i} = \xi)g(\sigma),$$

for  $\xi \in [q]^{L_i}$ . That is,  $g_{L_i}(\xi)$  is the conditional expectation of function  $g$  under the distribution  $\mu_{T_v}^\eta$  conditioned on the configuration of the level  $L_i$  being  $\xi$ . Thus, we may consider the expectation and variance of  $g_{L_i}$  w.r.t. the projection of  $\mu_{T_v}^\eta$  to  $L_i$ ; namely,  $\mathbb{E}_{F_i}^\eta[g_{L_i}] = \mathbb{E}_{F_i}^\eta[g]$  and  $\text{Var}_{F_i}^\eta[g_{L_i}] = \mathbb{E}_{F_i}^\eta[g_{L_i}^2] - \mathbb{E}_{F_i}^\eta[g_{L_i}]^2$ . The PVM condition is defined as follows.

► **Definition 8** (Parallel Variance Mixing (PVM)). *The Gibbs distribution  $\mu = \mu_T^\tau$  satisfies  $\text{PVM}(\ell, \varepsilon)$  if for every  $1 \leq i \leq h + 1$ , every  $\eta \in \Omega$ , and every function  $g : \Omega_{F_i}^\eta \rightarrow \mathbb{R}$  that is independent of the configuration on  $B_i^\ell$ , we have  $\text{Var}_{F_i}^\eta(g_{L_i}) \leq \varepsilon \cdot \text{Var}_{F_i}^\eta(g)$ . The PVM condition holds if there exist constants  $\ell$  and  $\varepsilon = \varepsilon(\ell)$  such that  $\text{PVM}(\ell, \varepsilon)$  holds.*

PVM is a natural global variant of VM since  $F_i = \bigcup_{v \in L_i} T_v$  and  $B_i^\ell = \bigcup_{v \in L_i} B(v, \ell)$ . We can show that the two properties are actually equivalent.

► **Theorem 9.** *For every  $\ell \in \mathbb{N}^+$  and  $\varepsilon \in (0, 1)$ , the Gibbs distribution  $\mu$  satisfies  $\text{VM}(\ell, \varepsilon)$  if and only if  $\mu$  satisfies  $\text{PVM}(\ell, \varepsilon)$ .*

In order to show the equivalence between VM and PVM, we introduce a more general spatial mixing condition which we call *General Variance Mixing (GVM)*. We define GVM for general product distributions (see Definition 12) and reinterpret VM and PVM as special cases of this condition. This alternative view of VM and PVM in terms of GVM is quite useful since we can recast the GVM condition as a bound on the spectral gap of a certain Markov chain; this is one key insight in the proof of Theorem 5 and is discussed in detail in Section 3.1.

Now, while VM implies optimal mixing of the Glauber dynamics, we can show that PVM implies a constant bound on the spectral gap of the tiled block dynamics. Recall that this is the heat-bath block dynamics with block collection  $\mathcal{U} = \{T_1^\ell, \dots, T_{\ell+1}^\ell\}$  defined in Section 2.

► **Theorem 10.** *If there exist  $\ell \in \mathbb{N}^+$  and  $\delta \in (0, 1)$  such that  $\mu = \mu_T^\tau$  satisfies  $\text{PVM}(\ell, \varepsilon)$  for  $\varepsilon = \frac{1-\delta}{2(\ell+1)}$ , then the relaxation time of the tiled block dynamics is at most  $2(\ell+1)/\delta$ .*

To prove Theorem 10, we adapt the methods from [40, 41] to our global setting. Our result for the spectral gap of the SW dynamics (Theorem 1) is then obtained through comparison with the tiled block dynamics. We prove the following comparison result between the SW dynamics and a large class of block dynamics, which could be of independent interest.

► **Theorem 11.** *Let  $\mathcal{D} = \{D_1, \dots, D_m\}$  be such that  $D_i \subseteq T$  and  $\cup_{i=1}^m D_i = T$ . Suppose that each block  $D_k$  is such that  $D_k = \cup_{j=1}^{\ell_k} D_{kj}$  where  $\text{dist}(D_{kj}, D_{kj'}) \geq 2$  for every  $j \neq j'$  and let  $\text{vol}(\mathcal{D}) = \max_{k,j} |D_{kj}|$ . Let  $\mathcal{B}_{\mathcal{D}}$  be the transition matrix of the (heat-bath) block dynamics with blocks  $\mathcal{D}$  and let  $\mathbf{SW}$  denote the transition matrix for the SW dynamics. Then,  $\text{gap}(\mathbf{SW}) \geq \exp(-O(\text{vol}(\mathcal{D}))) \cdot \text{gap}(\mathcal{B}_{\mathcal{D}})$ .*

The blocks of the tiled block dynamics satisfy all the conditions in this theorem, and, in addition,  $\text{vol}(\mathcal{D}) = O(1)$ . Hence, combining all the results stated in this section, we see that Theorem 1 from introduction follows.

**Proof of Theorem 1.** Follows from Theorems 9–11. ◀

### 3.1 Equivalence between VM and PVM: Proof of Theorem 9

In this section we establish the equivalence between VM and PVM. We start with the definition of *General Variance Mixing (GVM)*. Let  $\Phi$  and  $\Psi$  be two finite sets and let  $\rho(\cdot, \cdot)$  be an arbitrary joint distribution supported on  $\Phi \times \Psi$ . Denote by  $\nu$  and  $\pi$  the marginal distributions of  $\rho$  over  $\Phi$  and  $\Psi$ , respectively. That is, for  $x \in \Phi$  we have  $\nu(x) = \sum_{y \in \Psi} \rho(x, y)$ , and for  $y \in \Psi$  we have  $\pi(y) = \sum_{x \in \Phi} \rho(x, y)$ . We consider two natural matrices associated to  $\rho$ . For  $x \in \Phi$  and  $y \in \Psi$ , define

$$P^\uparrow(x, y) = \rho(y | x) = \frac{\rho(x, y)}{\nu(x)}, \quad \text{and} \quad P^\downarrow(y, x) = \rho(x | y) = \frac{\rho(x, y)}{\pi(y)}; \quad (8)$$

$P^\uparrow$  is a  $|\Phi| \times |\Psi|$  matrix while  $P^\downarrow$  is a  $|\Psi| \times |\Phi|$  matrix. In addition, observe that  $P^\uparrow P^\downarrow$  and  $P^\downarrow P^\uparrow$  are transition matrices of Markov chains reversible w.r.t.  $\nu$  and  $\pi$ , respectively.

► **Definition 12 (GVM for  $\rho$ ).** *We say that the joint distribution  $\rho$  satisfies  $\text{GVM}(\varepsilon)$  if for every function  $f : \Phi \rightarrow \mathbb{R}$  we have  $\text{Var}_\pi(P^\downarrow f) \leq \varepsilon \cdot \text{Var}_\nu(f)$ .*

One key observation in our proof is that the GVM condition can be expressed in term of the spectral gaps of the matrices  $P^\uparrow P^\downarrow$  and  $P^\downarrow P^\uparrow$ .

► **Lemma 13.** *The joint distribution  $\rho$  satisfies  $\text{GVM}(\varepsilon)$  if and only if  $\text{gap}(P^\uparrow P^\downarrow) = \text{gap}(P^\downarrow P^\uparrow) \geq 1 - \varepsilon$ .*

Before providing the proof of Lemma 13, we recall the definition of the *adjoint operator*. Let  $S_1$  and  $S_2$  be two Hilbert spaces with inner products  $\langle \cdot, \cdot \rangle_{S_1}$  and  $\langle \cdot, \cdot \rangle_{S_2}$  respectively, and let  $K : S_2 \rightarrow S_1$  be a bounded linear operator. The adjoint of  $K$  is the unique operator  $K^* : S_1 \rightarrow S_2$  satisfying  $\langle f, Kg \rangle_{S_1} = \langle K^* f, g \rangle_{S_2}$  for all  $f \in S_1$  and  $g \in S_2$ . When  $S_1 = S_2$ ,  $K$  is called *self-adjoint* if  $K = K^*$ . We can now provide the proof of Lemma 13.

**Proof of Lemma 13.** It is straightforward to check that  $P^\uparrow \mathbf{1} = \mathbf{1}$ ,  $P^\downarrow \mathbf{1} = \mathbf{1}$ ,  $\nu P^\uparrow = \pi$ ,  $\pi P^\downarrow = \nu$ , and that the operator  $P^\uparrow : L_2(\pi) \rightarrow L_2(\nu)$  is the adjoint of the operator  $P^\downarrow : L_2(\nu) \rightarrow L_2(\pi)$ . Hence, both  $P^\uparrow P^\downarrow$  and  $P^\downarrow P^\uparrow$  are positive semidefinite and have the same multiset of non-zero eigenvalues. Now, for  $f : \Phi \rightarrow \mathbb{R}$ , we have

$$\text{Var}_\pi(P^\downarrow f) = \langle P^\downarrow f, (I - \mathbf{1}\pi)P^\downarrow f \rangle_\pi = \langle f, P^\uparrow(I - \mathbf{1}\pi)P^\downarrow f \rangle_\nu = \langle f, P^\uparrow P^\downarrow f \rangle_\nu - \langle f, \mathbf{1}\nu f \rangle_\nu.$$

Therefore,  $\text{Var}_\pi(P^\downarrow f) \leq \varepsilon \cdot \text{Var}_\nu(f)$  holds if and only if

$$\begin{aligned} \langle f, P^\uparrow P^\downarrow f \rangle_\nu - \langle f, \mathbf{1}\nu f \rangle_\nu &\leq \varepsilon \cdot (\langle f, f \rangle_\nu - \langle f, \mathbf{1}\nu f \rangle_\nu) \\ \Leftrightarrow \langle f, (I - P^\uparrow P^\downarrow) f \rangle_\nu &\geq (1 - \varepsilon) \cdot \langle f, (I - \mathbf{1}\nu) f \rangle_\nu \\ \Leftrightarrow \mathcal{E}_{P^\uparrow P^\downarrow}(f, f) &\geq (1 - \varepsilon) \cdot \text{Var}_\nu(f). \end{aligned}$$

The lemma then follows from (6).  $\blacktriangleleft$

We provide next the proof of Theorem 9, which follows from Lemma 13 and interpretations of VM and PVM by GVM. Given  $F = A \cup B \subseteq T$  and  $\eta \in \Omega$ , let  $P^\uparrow = (P_F^\eta)_{A \uparrow B}$  denote the  $q^{|A \setminus B|} \times q^{|B \setminus A|}$  stochastic matrix indexed by the configurations on the sets  $A \setminus B$  and  $B \setminus A$ , such that for  $\xi \in [q]^{A \setminus B}$  and  $\xi' \in [q]^{B \setminus A}$  we have  $P^\uparrow(\xi, \xi') = \mu_F^\eta(\sigma_{B \setminus A} = \xi' \mid \sigma_{A \setminus B} = \xi)$ . In words,  $P^\uparrow$  corresponds to the transition matrix that given the configuration  $\xi$  in  $A \setminus B$  updates the configuration in  $B \setminus A$  from the conditional distribution  $\mu_F^\eta(\cdot \mid \xi)$ . We define in a similar manner the  $q^{|B \setminus A|} \times q^{|A \setminus B|}$  stochastic matrix  $P^\downarrow = (P_F^\eta)_{B \downarrow A}$  where for  $\xi' \in [q]^{B \setminus A}$  and  $\xi \in [q]^{A \setminus B}$  we have  $P^\downarrow(\xi', \xi) = \mu_F^\eta(\sigma_{A \setminus B} = \xi \mid \sigma_{B \setminus A} = \xi')$ .

If we set  $\rho$  to be the marginal of  $\mu_F^\eta$  on  $(A \setminus B) \cup (B \setminus A)$ , then  $\Phi = [q]^{A \setminus B}$ ,  $\Psi = [q]^{B \setminus A}$ , and  $\nu$  and  $\pi$  are the marginals of  $\mu_F^\eta$  on  $A \setminus B$  and  $B \setminus A$ , respectively. Therefore, according to Definition 12, GVM( $\varepsilon$ ) holds for the marginal of  $\mu_F^\eta$  on  $(A \setminus B) \cup (B \setminus A)$  if  $\text{Var}_\pi(P^\downarrow f) \leq \varepsilon \cdot \text{Var}_\nu(f)$  for every function  $f : \Phi \rightarrow \mathbb{R}$ .

Now, note that a function  $g : \Omega_F^\eta \rightarrow \mathbb{R}$  independent of  $B$  only depends on the configuration on  $A \setminus B$ . Thus, for fixed  $\eta$ ,  $g$  induces a function  $f : \Phi \rightarrow \mathbb{R}$ ; in particular,  $\text{Var}_F^\eta(g) = \text{Var}_\nu(f)$ . Moreover, letting  $g_{B \setminus A}(\xi) := \mathbb{E}_F^\eta[g \mid \sigma_{B \setminus A} = \xi]$ , we have  $g_{B \setminus A}(\xi) = P^\downarrow f(\xi)$  for every  $\xi \in \Psi = [q]^{B \setminus A}$ , and so  $\text{Var}_F^\eta(g_{B \setminus A}) = \text{Var}_\pi(P^\downarrow f)$ . Consequently, we arrive at the following equivalences between VM, PVM and GVM.

**► Proposition 14.**

1. The Gibbs distribution  $\mu$  satisfies VM( $\ell, \varepsilon$ ) if and only if for every  $v \in T$  and  $\eta \in \Omega$ , GVM( $\varepsilon$ ) holds for the marginal of  $\mu_{T_v}^\eta$  on  $(T_v \setminus B(v, \ell)) \cup \{v\}$ .
2. The Gibbs distribution  $\mu$  satisfies PVM( $\ell, \varepsilon$ ) if and only if for every  $i$  such that  $1 \leq i \leq h+1$  and  $\eta \in \Omega$ , GVM( $\varepsilon$ ) holds for the marginal of  $\mu_{F_i}^\eta$  on  $(F_i \setminus \cup_{v \in L_i} B(v, \ell)) \cup L_i$ .

To see part 1 simply note that in the notation above, we can set  $F = T_v$ ,  $A = T_v \setminus v$  and  $B = B(v, \ell)$ . For part 2, we set  $F = F_i$ ,  $A = F_{i-1}$  and  $B = B_i^\ell$ .

**Proof of Theorem 9.** From Proposition 14 and Lemma 13, VM( $\ell, \varepsilon$ ) holds if and only if  $\text{gap}(Q_v) \geq 1 - \varepsilon$  for every  $v \in T$  and  $\eta \in \Omega$ , where  $Q_v = (P_{T_v}^\eta)_{B(v, \ell) \downarrow (T_v \setminus v)} (P_{T_v}^\eta)_{(T_v \setminus v) \uparrow B(v, \ell)}$ . Similarly,  $\mu$  satisfies PVM( $\ell, \varepsilon$ ) if and only if  $\text{gap}(Q_{L_i}) \geq 1 - \varepsilon$  for every  $i$  such that  $1 \leq i \leq h+1$  and  $\eta \in \Omega$ , where  $Q_{L_i} = (P_{F_i}^\eta)_{B_i^\ell \downarrow F_{i-1}} (P_{F_i}^\eta)_{F_{i-1} \uparrow B_i^\ell}$ .

Since  $F_i = \bigcup_{v \in L_i} T_v$  and the  $T_v$ 's are at distance at least two from each other,  $\mu_{F_i}^\eta(\sigma_{L_i} = \cdot)$  is a product distribution; in particular  $\mu_{F_i}^\eta(\sigma_{L_i} = \cdot) = \prod_{v \in L_i} \mu_{T_v}^\eta(\sigma_v = \cdot)$  and the chain with transition matrix  $Q_{L_i}$  is a product Markov chain where each component corresponds to  $Q_v$  for some  $v \in L_i$ . A standard fact about product Markov chains, see, e.g., [9, Lemma 4.7], then implies that  $\text{gap}(Q_{L_i}) = \min_{v \in L_i} \text{gap}(Q_v)$  and the result follows.  $\blacktriangleleft$

## 4 Entropy Mixing: Proof of Theorem 2

Let  $E \subseteq T$  denote the set of all *even* vertices of the tree  $T$ , where a vertex is called even if its distance to the leaves is even; let  $O = T \setminus E$  be the set of all odd vertices. We show that EM (i.e., entropy mixing) as defined in [40] implies a factorization of entropy into even and

odd subsets of vertices. This even-odd factorization was recently shown to imply  $O(\log n)$  mixing of the SW dynamics on bipartite graphs [8].

We start with the definition of EM, which is the analog of the VM condition for entropy. Let  $\tau$  be a fixed boundary condition and again set  $\mu := \mu_T^\tau$  and  $\Omega := \Omega^\tau$  for ease of notation. Recall that for  $v \in T$ , we use  $T_v$  for the subtree of  $T$  rooted at  $v$ . Recall that for  $\eta \in \Omega$  and  $g : \Omega_{T_v}^\eta \rightarrow \mathbb{R}$ , we defined the function  $g_v(a) = \mathbb{E}_{T_v}^\eta [g \mid \sigma_v = a]$  for  $a \in [q]$ ; see (7).

► **Definition 15 (Entropy Mixing (EM)).** *The Gibbs distribution  $\mu = \mu_T^\tau$  satisfies  $\text{EM}(\ell, \varepsilon)$  if for every  $v \in T$ , every  $\eta \in \Omega$ , and every function  $g : \Omega_{T_v}^\eta \rightarrow \mathbb{R}$  that is independent of the configuration on  $B(v, \ell)$ , we have  $\text{Ent}_{T_v}^\eta(g_v) \leq \varepsilon \cdot \text{Ent}_{T_v}^\eta(g)$ . The EM condition holds if there exist constants  $\ell$  and  $\varepsilon = \varepsilon(\ell)$  such that  $\text{EM}(\ell, \varepsilon)$  holds.*

Extending our notation from the previous section for the variance functional, for  $A \subseteq T$  and a function  $f : \Omega \rightarrow \mathbb{R}_{\geq 0}$ , we use  $\text{Ent}_A(f)$  for the conditional entropy of  $f$  w.r.t.  $\mu$  given a spin configuration in  $T \setminus A$ ; i.e., for  $\xi \in \Omega$  we have

$$(\text{Ent}_A(f))(\xi) = \text{Ent}_A^\xi(f) = \text{Ent}_\mu[f \mid \sigma_{T \setminus A} = \xi_{T \setminus A}].$$

In particular, we shall write  $\text{Ent}(f) = \text{Ent}_T(f) = \text{Ent}_\mu(f)$ . Notice that  $\text{Ent}_A(f)$  can be viewed as a function from  $[q]^{T \setminus A}$  to  $\mathbb{R}_{\geq 0}$  and  $\mathbb{E}[\text{Ent}_A(f)]$  denotes its mean, averaging over the configuration on  $T \setminus A$ . We state next our even-odd factorization of entropy.

► **Theorem 16.** *If there exist  $\ell \in \mathbb{N}^+$  and  $\varepsilon \in (0, 1)$  such that  $\mu = \mu_T^\tau$  satisfies  $\text{EM}(\ell, \varepsilon)$ , then there exists a constant  $C_{\text{EO}} = C_{\text{EO}}(\ell, \varepsilon)$  independent of  $n$  such that for every function  $f : \Omega \rightarrow \mathbb{R}_{\geq 0}$  we have  $\text{Ent}(f) \leq C_{\text{EO}} (\mathbb{E}[\text{Ent}_E(f)] + \mathbb{E}[\text{Ent}_O(f)])$ .*

Theorem 2 follows immediately.

**Proof of Theorem 2.** By Theorem 16, EM implies the even-odd factorization of entropy, and the results in [8] imply that the mixing time of the SW dynamics is  $O(\log n)$ . ◀

Our main technical contribution in the proof Theorem 2 is thus Theorem 16; namely, that EM implies the even-odd factorization of entropy. To prove Theorem 16, we will first establish entropy factorization for the tiled blocks defined in (3) and (2); see also Figures 1(b) and 1(c). From the tiled block factorization of entropy we then deduce the desired even-odd factorization. This approach is captured by the following two lemmas.

► **Lemma 17.** *If there exist  $\ell \in \mathbb{N}^+$  and  $\varepsilon \in (0, 1)$  such that  $\mu = \mu_T^\tau$  satisfies  $\text{EM}(\ell, \varepsilon)$ , then there exists a constant  $C_{\text{TB}} = C_{\text{TB}}(\ell, \varepsilon)$  independent of  $n$  such that, for every function  $f : \Omega \rightarrow \mathbb{R}_{\geq 0}$ ,  $\text{Ent}(f) \leq C_{\text{TB}} \cdot \sum_{j=1}^{\ell+1} \mathbb{E}[\text{Ent}_{T_j^\ell}(f)]$ .*

► **Lemma 18.** *If for every function  $f : \Omega \rightarrow \mathbb{R}_{\geq 0}$  we have  $\text{Ent}(f) \leq C_{\text{TB}} \cdot \sum_{j=1}^{\ell+1} \mathbb{E}[\text{Ent}_{T_j^\ell}(f)]$ , then there exists  $C_{\text{EO}} = C_{\text{EO}}(C_{\text{TB}}, \ell)$  such that for every function  $f : \Omega \rightarrow \mathbb{R}_{\geq 0}$  we have*

$$\text{Ent}(f) \leq C_{\text{EO}} (\mathbb{E}[\text{Ent}_E(f)] + \mathbb{E}[\text{Ent}_O(f)]).$$

**Proof of Theorem 16.** Follows directly from Lemmas 17 and 18. ◀

We proved a version of Lemma 17 for the variance functional as part of the proof of Theorem 10, and the same argument can then be easily adapted to entropy. We provide next the proof of Lemma 18, which contains the main novelty in our proof of Theorem 16.



**Proof of Lemma 18.** First, we claim that there exists a constant  $C' = C'(\ell)$  such that for every function  $f : \Omega_{B(v,\ell)}^\eta \rightarrow \mathbb{R}_{\geq 0}$  one has the following inequality:

$$\text{Ent}_{B(v,\ell)}^\eta(f) \leq C' \left( \mathbb{E}_{B(v,\ell)}^\eta[\text{Ent}_{B(v,\ell) \cap E}(f)] + \mathbb{E}_{B(v,\ell)}^\eta[\text{Ent}_{B(v,\ell) \cap O}(f)] \right). \quad (9)$$

To deduce (9), consider the even-odd block dynamics  $M$  in  $B(v,\ell)$  with boundary condition  $\eta$  and blocks  $\mathcal{U} = \{E \cap B(v,\ell), O \cap B(v,\ell)\}$ . A simple coupling argument implies that the spectral gap of  $M$  is  $\Omega(1)$ . Then, Corollary A.4 from [19] implies that the log-Sobolev constant  $\alpha(M)$  of  $M$  is  $\Omega(1)$ , which establishes (9) with constant  $C' = O(1/\alpha(M))$ . We note that all bounds and comparisons in this argument are fairly crude, and, in fact, the constant  $C'$  depends exponentially on  $|B(v,\ell)|$ , but it is still independent of  $n$ .

Next, notice that, for any  $\eta \in \Omega$ ,  $\mu_{T_j^\ell}^\eta$  is the product of a collection of distributions on (disjoint) subsets  $B(v,\ell)$ . Lemma 3.2 from [14] allows us to lift the “local” even-odd factorization in each  $B(v,\ell)$  from (9) to a “global” even-odd factorization in  $T_j^\ell$ . Specifically, for every function  $f : \Omega_{T_j^\ell}^\eta \rightarrow \mathbb{R}_{\geq 0}$  we obtain

$$\text{Ent}_{T_j^\ell}^\eta(f) \leq C' \left( \mathbb{E}_{T_j^\ell}^\eta[\text{Ent}_{T_j^\ell \cap E}(f)] + \mathbb{E}_{T_j^\ell}^\eta[\text{Ent}_{T_j^\ell \cap O}(f)] \right).$$

Taking expectation over  $\eta$ , we get

$$\mathbb{E}[\text{Ent}_{T_j^\ell}(f)] \leq C' \left( \mathbb{E}[\text{Ent}_{T_j^\ell \cap E}(f)] + \mathbb{E}[\text{Ent}_{T_j^\ell \cap O}(f)] \right) \leq C' (\mathbb{E}[\text{Ent}_E(f)] + \mathbb{E}[\text{Ent}_O(f)]);$$

the last inequality follows from the fact that  $\text{Ent}_{T_j^\ell}^\eta(f) = \mathbb{E}_E^\eta[\text{Ent}_{T_j^\ell \cap E}(f)] + \mathbb{E}_O^\eta[\text{Ent}_{T_j^\ell \cap O}(f)]$ . Summing up over  $j$ ,

$$\sum_{j=1}^{\ell+1} \mathbb{E}[\text{Ent}_{T_j^\ell}(f)] \leq C'(\ell+1) (\mathbb{E}[\text{Ent}_E(f)] + \mathbb{E}[\text{Ent}_O(f)]),$$

and the result follows by taking  $C_{\text{EO}} = C'(\ell+1)$ . ◀

---

## References

- 1 Blanca A., Chen Z., Štefankovič D., and Vigoda E. The Swendsen-Wang dynamics on trees. *arXiv preprint arXiv:2007.08068*, 2020.
- 2 V. L. Alev and L. C. Lau. Improved analysis of higher order random walks and applications. In *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2020.
- 3 N. Anari, K. Liu, and S. Oveis Gharan. Spectral independence in high-dimensional expanders and applications to the hardcore model. In *Proceedings of the 52nd Annual ACM Symposium on Theory of Computing (STOC)*, 2020.
- 4 B. Awerbuch and Y. Shiloach. New connectivity and MSF algorithms for shuffle-exchange network and PRAM. *IEEE Computer Architecture Letters*, 36(10):1258–1263, 1987.
- 5 V. Beffara and H. Duminil-Copin. The self-dual point of the two-dimensional random-cluster model is critical for  $q \geq 1$ . *Probability Theory and Related Fields*, 153:511–542, 2012.
- 6 N. Berger, C. Kenyon, E. Mossel, and Y. Peres. Glauber dynamics on trees and hyperbolic graphs. *Probability Theory and Related Fields*, 131(3):311–340, 2005.
- 7 H. A. Bethe. Statistical theory of superlattices. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 150(871):552–575, 1935.
- 8 A. Blanca, P. Caputo, D. Parisi, A. Sinclair, and E. Vigoda. Entropy decay in the Swendsen-Wang dynamics on  $\mathbb{Z}^d$ . In *Proceedings of the 53rd Annual ACM Symposium on Theory of Computing (STOC)*, page 1551–1564, 2021.

- 9 A. Blanca, P. Caputo, A. Sinclair, and E. Vigoda. Spatial Mixing and Non-local Markov chains. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1965–1980, 2018.
- 10 A. Blanca, A. Galanis, L. A. Goldberg, D. Štefankovič, E. Vigoda, and K. 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.
- 11 A. Blanca, R. Gheissari, and E. Vigoda. Random-cluster dynamics in  $\mathbb{Z}^2$ : Rapid mixing with general boundary conditions. *Annals of Applied Probability*, 30(1):418–459, 2020.
- 12 A. Blanca and A. Sinclair. Dynamics for the mean-field random-cluster model. *Proceedings of the 19th International Workshop on Randomization and Computation*, pages 528–543, 2015.
- 13 M. Bordewich, C. Greenhill, and V. Patel. Mixing of the Glauber dynamics for the ferromagnetic Potts model. *Random Structures & Algorithms*, 48(1):21–52, 2016.
- 14 P. Caputo and D. Parisi. Block factorization of the relative entropy via spatial mixing, 2020. URL: <https://arxiv.org/abs/2004.10574>.
- 15 F. Cesi. Quasi-factorization of the entropy and logarithmic Sobolev inequalities for gibbs random fields. *Probability Theory and Related Fields*, 120(4):569–584, 2001.
- 16 C. Cooper and A. M. Frieze. Mixing properties of the Swendsen-Wang process on classes of graphs. *Random Structures and Algorithms*, 15(3-4):242–261, 1999.
- 17 M. Costeniuc, R. S. Ellis, and H. Touchette. Complete analysis of phase transitions and ensemble equivalence for the Curie–Weiss–Potts model. *Journal of Mathematical Physics*, 46(6):063301, 2005.
- 18 P. Cuff, J. Ding, O. Loidor, E. Lubetzky, Y. Peres, and A. Sly. Glauber dynamics for the mean-field Potts model. *Journal of Statistical Physics*, 149(3):432–477, 2012.
- 19 P. Diaconis and L. Saloff-Coste. Logarithmic Sobolev inequalities for finite Markov chains. *Annals of Applied Probability*, 6(3):695–750, 1996.
- 20 H. Duminil-Copin, M. Gagnebin, M. Harel, I. Manolescu, and V. Tassion. Discontinuity of the phase transition for the planar random-cluster and Potts models with  $q > 4$ . *Annales de l'ENS*, 2016.
- 21 H. Duminil-Copin, V. Sidoravicius, and V. Tassion. Continuity of the Phase Transition for Planar Random-Cluster and Potts Models with  $1 \leq q \leq 4$ . *Communications in Mathematical Physics*, 349(1):47–107, 2017.
- 22 M. Dyer, A. Sinclair, E. Vigoda, and D. Weitz. Mixing in time and space for lattice spin systems: A combinatorial view. *Random Structure & Algorithms*, 24(4):461–479, 2004.
- 23 R. G. Edwards and A. D. Sokal. Generalization of the Fortuin-Kasteleyn-Swendsen-Wang representation and Monte Carlo algorithm. *Physical Review D*, 38(6):2009–2012, 1988.
- 24 Grimmett G. The random-cluster model. In *Probability on discrete structures*, pages 73–123. Springer, 2004.
- 25 A. Galanis, D. Štefankovič, and E. Vigoda. Swendsen-Wang algorithm on the mean-field Potts model. In *Proceedings of the 19th International Workshop on Randomization and Computation*, pages 815–828, 2015.
- 26 A. Galanis, D. Štefankovič, E. Vigoda, and L. Yang. Ferromagnetic Potts model: Refined #BIS-hardness and related results. *SIAM Journal on Computing*, 45(6):2004–2065, 2016.
- 27 H. O. Georgii. *Gibbs Measures and Phase Transitions*. De Gruyter Studies in Mathematics. Walter de Gruyter Inc, 1988.
- 28 A. Gerschenfeld and A. Montanari. Reconstruction for models on random graphs. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 194–204, 2007.
- 29 R. Gheissari, E. Lubetzky, and Y. Peres. Exponentially slow mixing in the mean-field Swendsen-Wang dynamics. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1981–1988, 2018.
- 30 D. Gillman. A Chernoff bound for random walks on expander graphs. *SIAM Journal on Computing*, 27(4):1203–1220, 1998.

- 31 O. Häggström. The random-cluster model on a homogeneous tree. *Probability Theory and Related Fields*, 104:231–253, 1996.
- 32 T. P. Hayes and A. Sinclair. A general lower bound for mixing of single-site dynamics on graphs. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 511–520, 2005.
- 33 M. Huber. A bounding chain for Swendsen-Wang. *Random Structures & Algorithms*, 22(1):43–59, 2003.
- 34 M. Jerrum. *Counting, sampling and integrating: algorithms and complexity*. Lectures in Mathematics, Birkhäuser Verlag, 2003.
- 35 M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries. *Journal of the ACM*, 51(4):671–697, 2004.
- 36 J. 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.
- 37 R. Kannan, L. Lovász, and M. Simonovits. Random walks and an  $O^*(n^5)$  volume algorithm for convex bodies. *Random structures and algorithms*, 11(1):1–50, 1997.
- 38 D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, 2008.
- 39 F. Martinelli and E. Olivieri. Approach to equilibrium of Glauber dynamics in the one phase region. *Communications in Mathematical Physics*, 161(3):447–486, 1994.
- 40 F. Martinelli, A. Sinclair, and D. Weitz. The Ising model on trees: Boundary conditions and mixing time. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 628–639, 2003.
- 41 F. Martinelli, A. Sinclair, and D. Weitz. Fast mixing for independent sets. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 449–458. colorings and other model on trees. In, 2004.
- 42 F. Martinelli and F. L. Toninelli. On the mixing time of the 2D stochastic Ising model with “plus” boundary conditions at low temperature. *Communications in Mathematical Physics*, 296(1):175–213, 2010.
- 43 E. Mossel and Y. Peres. Information flow on trees. *Annals of Applied Probability*, 13(3):817–844, 2003.
- 44 E. Mossel and A. Sly. Exact thresholds for Ising–Gibbs samplers on general graphs. *The Annals of Probability*, 41(1):294–328, 2013.
- 45 R. Restrepo, D. Štefankovič, C. Vera, E. Vigoda, and L. Yang. Phase transition for Glauber dynamics for independent sets on regular trees. *SIAM Journal on Discrete Mathematics*, 28(2):835–861, 2014.
- 46 L. Saloff-Coste. Lectures on finite Markov chains. In *Lectures on probability theory and statistics*, pages 301–413. 1997.
- 47 A. Sly. Reconstruction for the Potts model. *The Annals of Probability*, 39(4):1365–1406, 2011.
- 48 A. Sly and Y. Zhang. The Glauber dynamics of colorings on trees is rapidly mixing throughout the nonreconstruction regime. *The Annals of Applied Probability*, 27(5):2646–2674, 2017.
- 49 R. H. Swendsen and J. S. Wang. Nonuniversal critical dynamics in Monte Carlo simulations. *Physical Review Letters*, 58:86–88, 1987.
- 50 M. Ullrich. Rapid mixing of Swendsen-Wang and single-bond dynamics in two dimensions. *Dissertationes Mathematicae*, 502:64, 2014.



# Distance Estimation Between Unknown Matrices Using Sublinear Projections on Hamming Cube

Arijit Bishnu ✉🏠

Indian Statistical Institute, Kolkata, India

Arijit Ghosh ✉🏠

Indian Statistical Institute, Kolkata, India

Gopinath Mishra ✉🏠

Indian Statistical Institute, Kolkata, India

---

## Abstract

Using geometric techniques like projection and dimensionality reduction, we show that there exists a randomized sub-linear time algorithm that can estimate the Hamming distance between two matrices. Consider two matrices  $\mathbf{A}$  and  $\mathbf{B}$  of size  $n \times n$  whose dimensions are known to the algorithm but the entries are not. The entries of the matrix are real numbers. The access to any matrix is through an oracle that computes the projection of a row (or a column) of the matrix on a vector in  $\{0, 1\}^n$ . We call this query oracle to be an INNER PRODUCT oracle (shortened as IP). We show that our algorithm returns a  $(1 \pm \epsilon)$  approximation to  $\mathbf{D}_M(\mathbf{A}, \mathbf{B})$  with high probability by making  $\mathcal{O}\left(\frac{n}{\sqrt{\mathbf{D}_M(\mathbf{A}, \mathbf{B})}} \text{poly}\left(\log n, \frac{1}{\epsilon}\right)\right)$  oracle queries, where  $\mathbf{D}_M(\mathbf{A}, \mathbf{B})$  denotes the Hamming distance (the number of corresponding entries in which  $\mathbf{A}$  and  $\mathbf{B}$  differ) between two matrices  $\mathbf{A}$  and  $\mathbf{B}$  of size  $n \times n$ . We also show a matching lower bound on the number of such IP queries needed. Though our main result is on estimating  $\mathbf{D}_M(\mathbf{A}, \mathbf{B})$  using IP, we also compare our results with other query models.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Streaming, sublinear and near linear time algorithms

**Keywords and phrases** Distance estimation, Property testing, Dimensionality reduction, Sub-linear algorithms

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.44

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2107.02666>

**Acknowledgements** The authors wish to thank their colleague Ansuman Banerjee for helpful discussions on GPU architecture and CUDA.

## 1 Introduction

Measuring similarity between entities using a distance function has been a major area of focus in computer science in general and computational geometry in particular [9, 8, 20, 19, 7]. Distance computations require access to the entire data and thus can not escape computations that are linear in time complexity. In this era of big data, seeing the entire data may be too much of an ask and trading precision for a time efficient algorithm is a vibrant area of study in property testing [22]. Testing properties of binary images with sub-linear time algorithms has been a focus of property testing algorithms [27, 26, 11, 24, 10]. Matrices are ubiquitous in the sense that they represent or abstract a whole gamut of structures like adjacency matrices of geometric graphs and visibility graphs, images, experimental data involving 0-1 outcomes, etc. Pairwise distance computations between such matrices is a much-needed programming primitive in image processing and computer vision applications



© Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 44; pp. 44:1–44:22



Leibniz International Proceedings in Informatics

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

so much so that the widely used commercial toolbox MATLAB of MathWorks® [1] has an inbuilt function call named `pdist2(·, ·, ·)` [2] for it. Other open source based software packages also have similar primitives [3]. For all these primitives, the matrices need to be known. But in all situations where access to the matrices are restricted (say, because of security, privacy or communication issues) except for an oracle access, we want to know how much the two matrices differ in their entries. Keeping in line with the above, we focus on distance estimation problem between two matrices whose dimensions are known to the algorithm but the entries are unknown; the access to the matrices will be through an oracle. This oracle, though linear algebraic in flavor, has a geometric connotation to it. We hold back the discussion on the motivation of the oracle till Section 1.1.

### Notations

In this paper, we denote the set  $\{1, \dots, t\}$  by  $[t]$  and  $\{0, \dots, t\}$  by  $[[t]]$ . For a matrix  $\mathbf{A}$ ,  $\mathbf{A}(i, j)$  denotes the element in the  $i$ -th row and  $j$ -th column of  $\mathbf{A}$ . Unless stated otherwise,  $\mathbf{A}$  will be a matrix with real entries.  $\mathbf{A}(i, *)$  and  $\mathbf{A}(*, j)$  denote the  $i$ -th row vector and  $j$ -th column vector of the matrix  $\mathbf{A}$ , respectively. Throughout this paper, the number of rows or columns of a square matrix  $\mathbf{A}$  will be  $n$ . Vectors are matrices of order  $n \times 1$  and will be represented using bold face letters. Without loss of generality, we consider  $n$  to be a power of 2. The  $i$ -th coordinate of a vector  $\mathbf{x}$  will be denoted by  $x_i$ . We will denote by  $\mathbf{1}$  the vector with all coordinates 1. Let  $\{0, 1\}^n$  denote the set of  $n$ -dimensional vectors with entries either 0 or 1. By  $\langle \mathbf{x}, \mathbf{y} \rangle$ , we denote the standard inner product of  $\mathbf{x}$  and  $\mathbf{y}$ , that is,  $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^n x_i y_i$ .  $P$  is a  $(1 \pm \varepsilon)$ -approximation to  $Q$  means  $|P - Q| \leq \varepsilon \cdot Q$ . The statement *with high probability* means that the probability of success is at least  $1 - \frac{1}{n^c}$ , where  $c$  is a positive constant.  $\tilde{\Theta}(\cdot)$  and  $\tilde{O}(\cdot)$  hides a poly  $(\log n, \frac{1}{\varepsilon})$  term in the upper bound.  $\|\cdot\|_p$  denotes the usual  $\ell_p$  distance.

## 1.1 Query oracle definition and motivation, problem statements and our results

► **Definition 1.1** (Matrix distance). The *matrix-distance* between two matrices  $\mathbf{A}$  and  $\mathbf{B}$  of size  $n \times n$  is the number of pairwise mismatches and is denoted and defined as

$$D_M(\mathbf{A}, \mathbf{B}) = |\{(i, j) : i, j \in [n], \mathbf{A}(i, j) \neq \mathbf{B}(i, j)\}|.$$

As alluded to earlier, the matrices cannot be accessed directly, the sizes of the matrices are known but the entries are unknown. We will refer to the problem as the *matrix distance* problem. We consider the following query models to solve the matrix distance problem in this paper.

### Query oracles for unknown matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$

The main query oracle access used in this work is based on the inner product of two vectors and is defined as follows:

**INNER PRODUCT (IP):** Given a row index  $i \in [n]$  (or, a column index  $j \in [n]$ ) and a vector  $\mathbf{v} \in \{0, 1\}^n$ , the IP query to  $\mathbf{A}$  reports the value of  $\langle \mathbf{A}(i, *), \mathbf{v} \rangle$  ( $\langle \mathbf{A}(*, j), \mathbf{v} \rangle$ ). If the input index is for row (column), we refer the corresponding query as row (column) IP query.

This linear algebraic oracle access has a geometric connotation to it in terms of *projection* onto Hamming vectors – we exploit this understanding in our work. This oracle access is also motivated from a practical angle. A dot product operation is a fit case for parallelization

using a SINGLE INSTRUCTION MULTIPLE DATA (SIMD) architecture [23]. Modern day GPU processors provide instruction level parallelism. In effect, NVIDIA GPUs that are built on CUDA architecture, provide dot product between two vectors as a single API call [28, 4]. Thus, there exists practical implementation of the query oracle access that we use. There are also examples of programming languages supporting SIMD intrinsics that can compute dot product [5]. There is a caveat though – in terms of resource, more processors are used. For us, in this work, the query complexity is the number of calls to the IP. As mentioned, modern day architectures allow us to convert each IP query to a one cycle computation with more processors used in parallel.

In the power hierarchy of matrix based query oracles, IP surely wields some power vis-a-vis solving certain problems [14]<sup>1</sup>. An obvious question that confronts an algorithm designer is whether a weaker oracle can do the same job at hand (here, computing matrix distance). With that in mind, we define the following two oracles and show that their query complexity lower bounds on the matrix distance problem match the trivial upper bounds. That shows the justification for use of IP.

**MATRIX ELEMENT (ME):** Given two indices  $i, j \in [n]$ , the ME query to  $\mathbf{A}$  returns the value of  $\mathbf{A}(i, j)$ .

**DECISION INNER PRODUCT (DEC-IP):** Given a row index  $i \in [n]$  (or, a column index  $j \in [n]$ ) and a vector  $\mathbf{v} \in \{0, 1\}^n$ , the DEC-IP query to  $\mathbf{A}$  reports whether  $\langle \mathbf{A}(i, *), \mathbf{v} \rangle$  ( $\langle \mathbf{A}(*, j), \mathbf{v} \rangle$ ) = 0. If the input index is for row (column), we refer the corresponding query as row (column) DEC-IP query.

The following remark highlights the relative power of the query oracles.

► **Remark 1.** Each ME query can be simulated by using one DEC-IP oracle, and each DEC-IP oracle can be simulated by using one IP query.

## Our results

Our main result is an algorithm for estimating the distances between two unknown matrices using IP, and the result is formally stated as follows. Unless otherwise mentioned, all our algorithms are randomized.

► **Theorem 1.2** (Main result: Estimating the distance between two arbitrary matrices). *There exists an algorithm that has IP query oracle access to unknown matrices  $\mathbf{A}$  and  $\mathbf{B}$ , takes an  $\varepsilon \in (0, 1)$  as an input, and returns a  $(1 \pm \varepsilon)$  approximation to  $\mathbf{D}_M(\mathbf{A}, \mathbf{B})$  with high probability, and makes  $\mathcal{O}\left(\left(n/\sqrt{\mathbf{D}_M(\mathbf{A}, \mathbf{B})}\right) \text{poly}\left(\log n, \frac{1}{\varepsilon}\right)\right)$  IP queries.*

We also show that our algorithm (corresponding to the above theorem) is optimal, if we ignore the  $\text{poly}\left(\log n, \frac{1}{\varepsilon}\right)$  term, by showing (in Theorem 4.1) that any algorithm that estimates  $\mathbf{D}_M(\mathbf{A}, \mathbf{B})$  requires  $\Omega\left(n/\sqrt{\mathbf{D}_M(\mathbf{A}, \mathbf{B})}\right)$  IP queries. For the sake of completeness in understanding the power of IP, we study the matrix distance problem also using two weaker oracle access – ME and DEC-IP. Our results are summarized in Table 1 and they involve both upper and almost matching lower bounds in terms of the number of queries needed. Note that all of our lower bounds hold even if one matrix (say  $\mathbf{A}$ ) is known and both matrices ( $\mathbf{A}$  and  $\mathbf{B}$ ) are symmetric binary matrices.

<sup>1</sup> But the IP defined in this paper is weaker than that defined in [14] – in their case, one is allowed to query for inner product of rows/columns of matrices with vectors in  $\mathbb{R}^n$ .

■ **Table 1** Our results. In this table,  $D = \mathbf{D}_M(\mathbf{A}, \mathbf{B})$ .

Query Oracle	ME	DEC-IP	IP
Upper Bound	$\tilde{O}\left(\frac{n^2}{D}\right)$ (Trivial)	$\tilde{O}\left(\frac{n^2}{D}\right)$ (Trivial)	$\tilde{O}\left(\frac{n}{\sqrt{D}}\right)$ (Theorem 1.2)
Lower Bound	$\Omega\left(\frac{n^2}{D}\right)$ (Corollary 4.3)	$\Omega\left(\frac{n^2}{D}\right)$ (Theorem 4.1)	$\Omega\left(\frac{n}{\sqrt{D}}\right)$ (Theorem 4.2)

► **Remark 2.** Note that an IP query to a matrix  $\mathbf{A}$  answers inner product of a specified row (column) with a given binary vector. However, we will describe subroutines (of the algorithm for estimating the distance between two matrices) that ask for inner product of a specified row (column) with a given vector  $\mathbf{r} \in \{-1, 1\}^n$ . This is not a problem as  $\langle \mathbf{A}(i, *), \mathbf{r} \rangle$  ( $\langle \mathbf{A}(*, j), \mathbf{r} \rangle$ ) can be computed by using two IP queries (with binary vectors)<sup>2</sup>. For simplicity, we refer  $\langle \mathbf{A}(i, *), \mathbf{r} \rangle$  ( $\langle \mathbf{A}(*, j), \mathbf{r} \rangle$ ) also as IP query in our algorithm.

## 1.2 Related work

There are works in property testing and sub-linear geometric algorithms [15, 18, 17, 16]. In the specific problem that we deal with in this paper, to the best of our knowledge, Raskhodnikova [26] started the study of property testing of binary images in the *dense image model*, where the number of 1-pixels is  $\Omega(n^2)$ . The notion of distance between matrices of the same size is defined as the number of pixels (matrix entries) on which they differ. The relative distance is the ratio of the distance and the number of pixels in the image. In this model, Raskhodnikova studies three properties of binary images – connectivity, convexity, and being a half-plane – in the property testing framework. Ron and Tsur [27] studied property testing algorithms in the sparse binary image model (the number of 1-pixels is  $O(n)$ ) for connectivity, convexity, monotonicity, and being a line. The distance measure in this model is defined by the fraction of differing entries taken with respect to the actual number of 1’s in the matrix. As opposed to treating binary images as discrete images represented using pixels as in [27, 26], Berman et al. in [10] and [12] treated them as continuous images and studied the problem of property testing for convexity of 2-dimensional figures with only uniform and independent samples from the input. To the best of our knowledge, computing distances between binary images has not been dealt with in the sub-linear time framework.

### Organization of the paper

We prove Theorem 1.2 (in Section 1.1) through a sequence of results. To prove Theorem 1.2 that estimates the distance between two arbitrary matrices, we need a result that estimates the distance between two symmetric matrices (Lemma 2.1 in Section 2) that in turn needs a result on the estimation of the distance between two symmetric matrices with respect to a parameter  $T$  (Lemma 2.2 in Section 2). Lemma 2.2 is the main technical lemma that uses dimensionality reduction via Johnson Lindenstrauss lemma crucially. The technical overview including the proof idea of Lemma 2.2 is in Section 2.1. The detailed proof idea is in Section 2.2. Using communication complexity, we prove our lower bound results in Section 4. The proof of lemma marked with  $\star$  can be found in the full version of the paper [13].

<sup>2</sup> For  $\mathbf{r} \in \{-1, 1\}^n$ , consider  $\mathbf{v}_1, \mathbf{v}_{-1} \in \{0, 1\}^n$  indicator vectors for +1 and -1 coordinates in  $\mathbf{r} \in \{-1, 1\}^n$ , respectively. Then  $\langle \mathbf{A}(i, *), \mathbf{r} \rangle = \langle \mathbf{A}(i, *), \mathbf{v}_1 \rangle - \langle \mathbf{A}(i, *), \mathbf{v}_{-1} \rangle$ . So,  $\langle \mathbf{A}(i, *), \mathbf{r} \rangle$  can be computed with two IP queries  $\langle \mathbf{A}(i, *), \mathbf{v}_1 \rangle$  and  $\langle \mathbf{A}(i, *), \mathbf{v}_{-1} \rangle$ . Similar argument also holds for  $\langle \mathbf{A}(*, j), \mathbf{r} \rangle$ .



## 2 Matrix-Distance between two symmetric matrices

This section builds up towards a proof of Theorem 1.2 by first giving an algorithm that estimates the matrix-distance between two unknown symmetric matrices (instead of arbitrary matrices as in Theorem 1.2) with high probability. The result is formally stated in Lemma 2.1. In Section 3, we will discuss how this result (stated in Lemma 2.1) can be used to prove Theorem 1.2.

► **Lemma 2.1** (Estimating the distance between two symmetric matrices). *There exists an algorithm  $\text{DIST-SYMM-MATRIX}(\mathbf{A}, \mathbf{B}, \varepsilon)$ , that has IP query access to unknown symmetric matrices  $\mathbf{A}$  and  $\mathbf{B}$ , takes an  $\varepsilon \in (0, \frac{1}{2})$  as an input, and returns a  $(1 \pm \varepsilon)$ -approximation to  $\mathbf{D}_M(\mathbf{A}, \mathbf{B})$  with high probability, making  $\tilde{O}\left(n/\sqrt{\mathbf{D}_M(\mathbf{A}, \mathbf{B})}\right)$  IP queries.*

First, we prove a parameterized version of the above lemma in Lemma 2.2 where we are given a parameter  $T$  along with an  $\varepsilon \in (0, \frac{1}{2})$  and we can obtain an approximation guarantee on  $\mathbf{D}_M(\mathbf{A}, \mathbf{B})$  as a function of both  $T$  and  $\varepsilon$ . One can think of  $T$  as a guess for  $\mathbf{D}_M(\mathbf{A}, \mathbf{B})$ .

► **Lemma 2.2** ( $(*)$  Estimating the distance between two symmetric matrices w.r.t. a parameter  $T$ ). *There exists an algorithm  $\text{DIST-SYMM-MATRIX-GUESS}(\mathbf{A}, \mathbf{B}, \varepsilon, T)$ , that has IP query access to unknown symmetric matrices  $\mathbf{A}$  and  $\mathbf{B}$ , takes parameters  $T$  and  $\varepsilon \in (0, \frac{1}{2})$  as inputs, and returns  $\hat{d}$  satisfying  $(1 - \frac{\varepsilon}{10}) \mathbf{D}_M(\mathbf{A}, \mathbf{B}) - \frac{\varepsilon}{1600} T \leq \hat{d} \leq (1 + \frac{\varepsilon}{10}) \mathbf{D}_M(\mathbf{A}, \mathbf{B})$  with high probability, and makes  $\tilde{O}\left(n/\sqrt{T}\right)$  queries. Note that here  $T$  is at least a suitable polynomial in  $\log n$  and  $1/\varepsilon$ .*

In Section 2.1, we discuss some preliminary results to prove Lemma 2.2. The proof of Lemma 2.2 is given in Section 2.2. If the guess  $T \leq \mathbf{D}_M(\mathbf{A}, \mathbf{B})$ ,  $\text{DIST-SYMM-MATRIX-GUESS}(\mathbf{A}, \mathbf{B}, \varepsilon, T)$  (as stated in Lemma 2.2) returns a  $(1 \pm \varepsilon)$ -approximation to  $\mathbf{D}_M(\mathbf{A}, \mathbf{B})$  with high probability. However, this dependence on  $T$  can be overcome to prove Lemma 2.1 by using a standard technique in property testing.

### 2.1 Technical preliminaries to prove Lemma 2.2

The matrix distance  $\mathbf{D}_M(\mathbf{A}, \mathbf{B})$  can be expressed in terms of the notion of a distance between a row (column) of matrix  $\mathbf{A}$  and a row (column) of matrix  $\mathbf{B}$  as follows:

► **Definition 2.3** (Distance between two rows (columns)). Let  $\mathbf{A}$  and  $\mathbf{B}$  be two matrices of order  $n \times n$ . The distance between the  $i$ -th row of  $\mathbf{A}$  and the  $j$ -th row of  $\mathbf{B}$  is denoted and defined as

$$\mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(j, *)) = |\{k \in [n] : \mathbf{A}(i, k) \neq \mathbf{B}(j, k)\}|,$$

Similarly,  $\mathbf{d}_H(\mathbf{A}(*, i), \mathbf{B}(*, j))$  is the distance between the  $i$ -th column of  $\mathbf{A}$  and the  $j$ -th column of  $\mathbf{B}$ .

► **Observation 2.4** (Expressing  $\mathbf{D}_M(\mathbf{A}, \mathbf{B})$  as the sum of distance between rows (columns)). Let  $\mathbf{A}$  and  $\mathbf{B}$  be two  $n \times n$  matrices. The matrix distance between  $\mathbf{A}$  and  $\mathbf{B}$  is given by

$$\mathbf{D}_M(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^n \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *)) = \sum_{i=1}^n \mathbf{d}_H(\mathbf{A}(*, i), \mathbf{B}(*, i)).$$

For a given  $i \in [n]$ , we can approximate  $\mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))$  and  $\mathbf{d}_H(\mathbf{A}(*, i), \mathbf{B}(*, i))$  using IP queries as stated in Lemma 2.5. This can be shown by an application of the well-known Johnson-Lindenstrauss Lemma [6].

► **Lemma 2.5** (Estimating the distance between rows of  $\mathbf{A}$  and  $\mathbf{B}$ ). *Consider IP access to two  $n \times n$  (unknown) matrices  $\mathbf{A}$  and  $\mathbf{B}$ . There is an algorithm  $\text{DIST-BET-ROWS}(i, \alpha, \delta)$ , that takes  $i \in [n]$  and  $\alpha, \delta \in (0, 1)$  as inputs, and reports a  $(1 \pm \alpha)$ -approximation to  $\mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))$  with probability at least  $1 - \delta$ , and makes  $\mathcal{O}\left(\frac{\log n}{\alpha^2} \log \frac{1}{\delta}\right)$  IP queries to both  $\mathbf{A}$  and  $\mathbf{B}$ .*

As it is sufficient for our purpose, in the above lemma, we discussed about estimating the distance between rows of  $\mathbf{A}$  and  $\mathbf{B}$  with the same index. However, we note that, a simple modification to the algorithm corresponding to Lemma 2.5 also works for estimating the distance between any row and/or column pair.

► **Proposition 2.6** (Johnson-Lindenstrauss Lemma). *Let us consider any pair of points  $\mathbf{u}, \mathbf{v} \in \mathbb{R}^N$ . For a given  $\varepsilon \in (0, 1)$  and  $\delta \in (0, 1)$ , there is a map  $f : \mathbb{R}^N \rightarrow \mathbb{R}^d$  such that  $d = \Theta\left(\frac{1}{\varepsilon^2} \log \frac{1}{\delta}\right)$  satisfying the following bound with probability at least  $1 - \delta$ .*

$$(1 - \varepsilon)\|\mathbf{u} - \mathbf{v}\|_2^2 \leq \|f(\mathbf{u}) - f(\mathbf{v})\|_2^2 \leq (1 + \varepsilon)\|\mathbf{u} - \mathbf{v}\|_2^2. \quad (1)$$

► **Remark 3** (An explicit mapping in Johnson-Lindenstrauss Lemma). An explicit mapping  $f : \mathbb{R}^n \rightarrow \mathbb{R}^d$  satisfying Equation 1 is as follows. Consider  $\mathbf{r}_1, \dots, \mathbf{r}_d \in \{-1, 1\}^n$  such that each coordinate of every  $\mathbf{r}_i$  is taken from  $\{-1, 1\}$  uniformly at random. Then for each  $\mathbf{u} \in \{0, 1\}^n$ ,

$$f(\mathbf{u}) = \frac{1}{\sqrt{d}} (\langle \mathbf{u}, \mathbf{r}_1 \rangle, \langle \mathbf{u}, \mathbf{r}_2 \rangle, \dots, \langle \mathbf{u}, \mathbf{r}_d \rangle).$$

### Identity testing between two rows

Now, let us discuss an algorithm where the objective is to decide whether the  $i$ -th row vectors of matrices  $\mathbf{A}$  and  $\mathbf{B}$  are identical. Observe that  $\|\mathbf{A}(i, *) - \mathbf{B}(j, *)\|_2 = 0$  if and only if  $\mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(j, *)) = 0$ . Also notice that, for a function  $f : \mathbb{R}^n \rightarrow \mathbb{R}^d$  satisfying Equation 1,  $\|\mathbf{u} - \mathbf{v}\|_2 = 0$  if and only if  $\|f(\mathbf{u}) - f(\mathbf{v})\|_2 = 0$ . This discussion along with Proposition 2.6 and Remark 3 imply an algorithm (described inside Observation 2.9) that can decide whether corresponding rows of  $\mathbf{A}$  and  $\mathbf{B}$  are identical. Observation 2.9 is stated in a more general form than discussed here. Note that the general form will be needed to show Lemma 2.5. For this purpose, we define the notion of *projecting a vector in  $\{-1, 1\}^n$  onto a set  $S \subseteq [n]$*  as defined below and an observation (Observation 2.8) about evaluating the projection using an IP query.

► **Definition 2.7** (Vector projected onto a set). Let  $\mathbf{A}$  be an  $n \times n$  matrix and  $i \in [n]$ . For a subset  $S \subseteq [n]$ ,  $\mathbf{A}(i, *)|_S \in \mathbb{R}^n$  is defined as the vector having  $\ell$ -th coordinate equals to  $\mathbf{A}(i, \ell)$  if  $\ell \in S$ , and 0, otherwise. Also consider  $\mathbf{r} \in \{-1, 1\}^n$  and a set  $S \subseteq [n]$ . Then the vector  $\mathbf{r}$  projected onto  $S$  is denoted by  $\mathbf{r}|_S \in \{-1, 0, 1\}^n$  and defined as follows: For  $\ell \in [n]$ , the  $\ell$ -th coordinate of  $\mathbf{r}|_S$  is same as that of  $\mathbf{r}$  if  $\ell \in S$ , and 0, otherwise.

► **Observation 2.8.** Let  $\mathbf{A}$  be a  $n \times n$  matrix,  $i \in [n]$ ,  $\mathbf{r} \in \{-1, 1\}^n$  and  $S \subseteq [n]$ . Then  $\langle \mathbf{A}(i, *)|_S, \mathbf{r} \rangle = \langle \mathbf{A}(i, *), \mathbf{r}|_S \rangle$ . That is,  $\langle \mathbf{A}(i, *)|_S, \mathbf{r} \rangle$  can be evaluated by using a IP query  $\langle \mathbf{A}(i, *), \mathbf{r}|_S \rangle$  to matrix  $\mathbf{A}$ .

► **Observation 2.9** (Identity testing between rows of  $\mathbf{A}$  and  $\mathbf{B}$ ). Consider IP access to two  $n \times n$  (unknown) matrices  $\mathbf{A}$  and  $\mathbf{B}$ . There is an algorithm  $\text{IDENTITY}(S, i, \delta)$  that takes  $i \in [n]$ ,  $S \subseteq [n]$  and  $\delta \in (0, 1)$  as inputs, and decides whether  $\mathbf{d}_H(\mathbf{A}(i, *)|_S, \mathbf{B}(i, *)|_S) = 0$  with probability at least  $1 - \delta$ , and makes  $\mathcal{O}\left(\log \frac{1}{\delta}\right)$  IP queries to both  $\mathbf{A}$  and  $\mathbf{B}$ .

**Proof.** Let the vectors  $\mathbf{r}_1, \dots, \mathbf{r}_d \in \{-1, 1\}^n$  be such that each coordinate of every  $\mathbf{r}_j$ ,  $j = 1, \dots, d$ , is taken from  $\{-1, 1\}$  uniformly at random where  $d = \Theta(\log \frac{1}{\delta})$ . Then the algorithm finds  $a_j = \langle \mathbf{A}(i, *)|_S, \mathbf{r}_j \rangle$  and  $b_j = \langle \mathbf{B}(i, *)|_S, \mathbf{r}_j \rangle$  by making one IP query to each of  $\mathbf{A}$  and  $\mathbf{B}$ . This is possible by Observation 2.8. The algorithm makes  $d$  IP queries to each of the matrices  $\mathbf{A}$  and  $\mathbf{B}$ . Take  $\mathbf{a} = \frac{1}{\sqrt{d}}(a_1, \dots, a_d) \in \mathbb{R}^d$  and  $\mathbf{b} = \frac{1}{\sqrt{d}}(b_1, \dots, b_d) \in \mathbb{R}^d$ . By Proposition 2.6 and Remark 3,  $\|\mathbf{a} - \mathbf{b}\|_2 = 0$  if and only if  $\|\mathbf{A}(i, *)|_S, \mathbf{B}(i, *)|_S\|_2 = 0$ . By the definition of distance between a row of one matrix and a row of another matrix (Definition 2.3), note that,  $\|\mathbf{A}(i, *)|_S - \mathbf{B}(j, *)|_S\|_2 = 0$  if and only if  $\mathbf{d}_H(\mathbf{A}(i, *)|_S, \mathbf{B}(j, *)|_S) = 0$ . So, the algorithm finds  $\|\mathbf{a} - \mathbf{b}\|_2$  and, reports  $\|\mathbf{a} - \mathbf{b}\|_2 = 0$  if and only if  $\mathbf{d}_H(\mathbf{A}(i, *)|_S, \mathbf{B}(i, *)|_S) = 0$ . The correctness and query complexity of the algorithm follows from the description itself.  $\blacktriangleleft$

### Estimating the distance between rows induced by a set

Now, consider the algorithm corresponding to Lemma 2.5 (DIST-BET-ROWS( $\cdot, \cdot, \cdot$ )) that can estimate the distance between a row of  $\mathbf{A}$  and a row of  $\mathbf{B}$ . It makes repeated calls to IDENTITY( $\cdot, \cdot, \cdot$ ) in a non-trivial way. Also, algorithm DIST-BET-ROWS( $\cdot, \cdot, \cdot$ ) can be generalized to estimate the distance between a row of  $\mathbf{A}$  and the corresponding row of  $\mathbf{B}$  projected onto the same set  $S \subseteq [n]$ , as stated in the following Lemma.

► **Lemma 2.10** ( $(\star)$  Estimating the distance between rows of  $\mathbf{A}$  and  $\mathbf{B}$  induced by a set  $S \subseteq [n]$ ). Consider IP access to two  $n \times n$  (unknown) matrices  $\mathbf{A}$  and  $\mathbf{B}$ . RESTRICT-DIST-BET-ROWS( $S, i, \alpha, \delta$ ) algorithm, takes  $S \subseteq [n]$ ,  $i \in [n]$  and  $\alpha, \delta \in (0, 1)$  as inputs, and reports a  $(1 \pm \alpha)$ -approximation to  $\mathbf{d}_H(\mathbf{A}(i, *)|_S, \mathbf{B}(i, *)|_S)$  with probability at least  $1 - \delta$ , and makes  $\mathcal{O}\left(\frac{\log n}{\alpha^2} \log \frac{1}{\delta}\right)$  IP queries to both  $\mathbf{A}$  and  $\mathbf{B}$ .

Observe that Lemma 2.5 is a special case of Lemma 2.10 when  $S = [n]$ . Algorithm DIST-BET-ROWS( $i, \alpha, \delta$ ) (corresponding to Lemma 2.5) is directly called as a subroutine from DIST-SYMM-MATRIX-GUESS( $\mathbf{A}, \mathbf{B}, \varepsilon, T$ ). RESTRICT-DIST-BET-ROWS( $S, i, \alpha, \delta$ ) is indirectly called from a subroutine to *sample mismatched element almost uniformly* as explained below.

### Sampling a mismatched element almost uniformly

For a row  $i \in [n]$ , let  $\text{NEQ}(\mathbf{A}, \mathbf{B}, i) = \{j : \mathbf{A}(i, j) \neq \mathbf{B}(i, j)\}$  denote the set of mismatches. Apart from estimating the distance between a row (column) of  $\mathbf{A}$  and the corresponding row (column) of  $\mathbf{B}$ , we can also sample element from  $\text{NEQ}(\mathbf{A}, \mathbf{B}, i)$  *almost uniformly* for any given  $i \in [n]$ .

► **Definition 2.11** (Almost uniform sample). Let  $X$  be a set and  $\alpha \in (0, 1)$ . A  $(1 \pm \alpha)$ -uniform sample from  $X$  is defined as the sample obtained from a distribution  $p$  satisfying  $(1 - \alpha)\frac{1}{|X|} \leq p(x) \leq (1 + \alpha)\frac{1}{|X|}$  for each  $x \in X$ , where  $p(x)$  denotes the probability of getting  $x$  as a sample.

► **Lemma 2.12** ( $(\star)$  Sampling a mismatched element almost uniformly). Consider IP access to two  $n \times n$  (unknown) matrices  $\mathbf{A}$  and  $\mathbf{B}$ . There exists an algorithm APPROX-SAMPLE( $i, \alpha, \delta$ ), that takes  $i \in [n]$  and  $\alpha, \delta \in (0, 1)$  as input, and reports a  $(1 \pm \alpha)$ -uniform sample from the set  $\text{NEQ}(\mathbf{A}, \mathbf{B}, i)$  with probability at least  $1 - \delta$ , and makes  $\mathcal{O}\left(\frac{\log^5 n}{\alpha^2} \log \frac{1}{\delta}\right)$  IP queries to both  $\mathbf{A}$  and  $\mathbf{B}$ .

Note that algorithm APPROX-SAMPLE( $\cdot, \cdot, \cdot$ ) calls repeatedly RESTRICT-DIST-BET-ROWS( $\cdot, \cdot, \cdot$ ). We now have all the ingredients – DIST-BET-ROWS( $i, \alpha, \delta$ ), DIST-SYMM-MATRIX-GUESS( $\mathbf{A}, \mathbf{B}, \varepsilon, T$ ), APPROX-SAMPLE( $i, \alpha, \delta$ ) – to design the final algorithm DIST-SYMM-MATRIX( $\mathbf{A}, \mathbf{B}, \varepsilon$ ).

### Overview of the algorithm

Algorithm  $\text{DIST-SYMM-MATRIX}(\cdot, \cdot, \cdot)$  calls  $\text{DIST-SYMM-MATRIX-GUESS}(\cdot, \cdot, \cdot, \cdot)$  with reduced value of guesses  $O(\log n)$  times to bring down the approximation error of matrix distance within limits. Algorithm  $\text{DIST-SYMM-MATRIX-GUESS}(\mathbf{A}, \mathbf{B}, \varepsilon, T)$  discussed in Lemma 2.2 mainly uses subroutines  $\text{DIST-BET-ROWS}(\cdot, \cdot, \cdot)$  and  $\text{APPROX-SAMPLE}(\cdot, \cdot, \cdot)$  in a nontrivial way. Both of these subroutines use Johnson-Lindenstrauss lemma.

Observe that  $\text{DIST-SYMM-MATRIX-GUESS}(\mathbf{A}, \mathbf{B}, \varepsilon, T)$  estimates  $\mathbf{D}_M(\mathbf{A}, \mathbf{B})$  where the approximation guarantee is parameterized by  $T$ . By Observation 2.4, we have  $\mathbf{D}_M(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^n \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))$ , the sum of the distances among corresponding rows. To estimate  $\sum_{i=1}^n \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))$ , our algorithm  $\text{DIST-SYMM-MATRIX-GUESS}(\mathbf{A}, \mathbf{B}, \varepsilon, T)$  considers a partition of the row indices  $[n]$  into buckets such that the row indices  $i$ 's in the same bucket have roughly the same  $\mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))$  values. Now the problem boils down to estimating the sizes of the buckets. To do so,  $\text{DIST-SYMM-MATRIX-GUESS}(\mathbf{A}, \mathbf{B}, \varepsilon, T)$  finds a random sample  $\Gamma$  having  $\tilde{O}\left(n/\sqrt{T}\right)$  indices from  $[n]$ , calls  $\text{DIST-BET-ROWS}(i, \cdot, \cdot)$  for each of the sample in  $\Gamma$  and partitions  $\Gamma$  into buckets such that  $i$ 's in the same bucket have roughly the same  $\mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))$  values. A *large* bucket is one that contains more than a fixed number of row indices. These steps ensure that the sizes of the *large* buckets are approximated well. Recall that  $\text{APPROX-SAMPLE}(i, \alpha, \delta)$  takes  $i \in [n]$  and  $\alpha, \delta \in (0, 1)$  as input, and reports a  $(1 \pm \alpha)$ -uniform sample from the set  $\text{NEQ}(\mathbf{A}, \mathbf{B}, i)$  with probability at least  $1 - \delta$ . To take care of the *small* buckets,  $\text{DIST-SYMM-MATRIX-GUESS}(\mathbf{A}, \mathbf{B}, \varepsilon, T)$  calls  $\text{APPROX-SAMPLE}(i, \cdot, \cdot)$  for *suitable* number of  $i$ 's chosen uniformly from each large bucket and decides whether the output indices of  $\text{APPROX-SAMPLE}(i, \cdot, \cdot)$  belong to large or small buckets. See the the following section for the technical description of our algorithm.

## 2.2 Proof of Lemma 2.2

Let us consider the following oracle that gives a probabilistic approximate estimate to the distance between the two corresponding rows of  $\mathbf{A}$  and  $\mathbf{B}$ ;  $\mathbf{A}$  and  $\mathbf{B}$  are two unknown  $n \times n$  matrices.

► **Definition 2.13** (Oracle function on the approximate distance between rows). Let  $\beta, \eta \in (0, 1)$ . Oracle  $\mathbb{O}_{\beta, \eta}$  is a function  $\mathbb{O}_{\beta, \eta} : [n] \rightarrow \mathbb{N}$ , which when queried with an  $i \in [n]$ , reports  $\mathbb{O}_{\beta, \eta}(i)$ . Moreover,

$$\mathbb{P}(\text{for every } i \in [n], \mathbb{O}_{\beta, \eta}(i) \text{ is a } (1 \pm \beta)\text{-approximation to } \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))) \geq 1 - \eta.$$

Take  $\beta = \varepsilon/50$  and  $\eta = 1/\text{poly}(n)$  and consider an oracle  $\mathbb{O}_{\beta, \eta}$  as defined above. Also, consider a partitioning of the indices in  $[n]$  into  $t = \Theta\left(\log_{\varepsilon/50} n\right)$  many buckets with respect to  $\mathbb{O}_{\beta, \eta}$  such that the  $i$ 's in the same bucket have *roughly* the same  $\mathbb{O}_{\beta, \eta}(i)$  values. Let  $Y_1, \dots, Y_t \subseteq [n]$  be the resulting buckets with respect to  $\mathbb{O}_{\beta, \eta}$ . Formally, for  $k \in [t]$ ,  $Y_k = \{i \in [n] : (1 + \frac{\varepsilon}{50})^{k-1} \leq \mathbb{O}_{\beta, \eta}(i) < (1 + \frac{\varepsilon}{50})^k\}$ . From the definition of  $\mathbb{O}_{\beta, \eta}$  and the way we are bucketing the elements of  $[n]$ , the following observation follows.

► **Observation 2.14** (Bucketing according to an oracle function). Let  $\beta = \varepsilon/50$  and  $\eta \in (0, 1)$ . Consider any oracle  $\mathbb{O}_{\beta, \eta} : [n] \rightarrow \mathbb{R}$  as defined in Definition 2.13. Let  $Y_1, \dots, Y_t$  be the buckets with respect to  $\mathbb{O}_{\beta, \eta}$ . Then

$$\left(1 - \frac{\varepsilon}{50}\right) \mathbf{D}_M(\mathbf{A}, \mathbf{B}) \leq \sum_{k=1}^t |Y_k| \left(1 + \frac{\varepsilon}{50}\right)^k \leq \left(1 + \frac{\varepsilon}{50}\right)^2 \mathbf{D}_M(\mathbf{A}, \mathbf{B})$$

holds with probability at least  $1 - \eta$ .

**Proof.** From Observation 2.4,  $\mathbf{D}_M(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^n \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))$ . So, by the definition of  $\mathbb{O}_{\beta, \eta}$  along with  $\beta = \varepsilon/50$ , we have

$$\mathbb{P} \left( \left(1 - \frac{\varepsilon}{50}\right) \mathbf{D}_M(\mathbf{A}, \mathbf{B}) \leq \sum_{i=1}^n \mathbb{O}_{\beta, \eta}(i) \leq \left(1 + \frac{\varepsilon}{50}\right) \mathbf{D}_M(\mathbf{A}, \mathbf{B}) \right) \geq 1 - \eta. \quad (2)$$

As  $Y_1, \dots, Y_t$  are the buckets with respect to  $\mathbb{O}_{\beta, \eta}$ , for  $k \in [t]$ ,  $Y_k = \{i \in [n] : (1 + \frac{\varepsilon}{50})^{k-1} \leq \mathbb{O}_{\beta, \eta}(i) < (1 + \frac{\varepsilon}{50})^k\}$ . So,

$$\sum_{i=1}^n \mathbb{O}_{\beta, \eta}(i) \leq \sum_{k=1}^t |Y_k| \left(1 + \frac{\varepsilon}{50}\right)^k \leq \left(1 + \frac{\varepsilon}{50}\right)^2 \sum_{i=1}^n \mathbb{O}_{\beta, \eta}(i) \quad (3)$$

From Equations 2 and 3, the following holds with probability at least  $1 - \eta$ .

$$\left(1 - \frac{\varepsilon}{50}\right) \mathbf{D}_M(\mathbf{A}, \mathbf{B}) \leq \sum_{k=1}^t |Y_k| \left(1 + \frac{\varepsilon}{50}\right)^k \leq \left(1 + \frac{\varepsilon}{50}\right)^2 \mathbf{D}_M(\mathbf{A}, \mathbf{B}). \quad \blacktriangleleft$$

The above observation roughly says that  $\mathbf{D}_M(\mathbf{A}, \mathbf{B})$  can be estimated if we can approximate  $|Y_k|$ 's.

### The existence of the oracle

Before the description of the algorithm, we note that our algorithm does not need to know the specific oracle  $\mathbb{O}_{\beta, \eta} : [n] \rightarrow \mathbb{R}$ . The existence of some oracle function  $\mathbb{O}_{\beta, \eta} : [n] \rightarrow \mathbb{R}$  with respect to which  $[n]$  can be partitioned into buckets  $Y_1, \dots, Y_t$  suffices. Our algorithm calls  $\text{DIST-BET-ROWS}(i, \beta, \eta)$  for some  $i$ 's but at most once for each  $i \in [n]$ . Note that  $\text{DIST-BET-ROWS}(i, \beta, \eta)$  is the algorithm (as stated in Lemma 2.5) that returns a  $(1 \pm \beta)$ -approximation to  $\mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))$  with probability at least  $1 - \eta$ . So, we can think of  $\mathbb{O}_{\beta, \eta} : [n] \rightarrow \mathbb{R}$  such that  $\mathbb{O}_{\beta, \eta}(i) = \hat{a}_i$ , where  $\hat{a}_i$  is the value returned by  $\text{DIST-BET-ROWS}(i, \beta, \eta)$ , if the algorithm  $\text{DIST-BET-ROWS}(i, \beta, \eta)$  is called (once). Otherwise,  $\mathbb{O}_{\beta, \eta}(i)$  is set to some  $(1 \pm \beta)$ -approximation to  $\mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))$ <sup>3</sup>.

### Random sample and bucketing

As has been mentioned in the overview of algorithm in Section 2.1, the problem of estimating matrix distance boils down to estimating the sizes of the buckets  $Y_k$ ,  $k = 1, \dots, t$  and our subsequent action depends on whether the bucket is of *large* or *small* size. But as  $|Y_k|$ 's are unknown, we define a bucket  $Y_k$  to be *large* or *small* depending on the estimate  $|\widehat{Y}_k|$  obtained from a random sample. So, our algorithm starts by taking a random sample  $\Gamma \subseteq [n]$  with replacement, where  $|\Gamma| = \tilde{\mathcal{O}}(n/\sqrt{T})$ , where  $T$  is a guess for  $\mathbf{D}_M(\mathbf{A}, \mathbf{B})$ . Now,  $\widehat{Y}_k = Y_k \cap \Gamma$ , the projection of  $Y_k$  on  $\Gamma$ .

For each  $i$  in the random sample  $\Gamma$ , we call  $\text{DIST-BET-ROWS}(i, \beta, \eta)$  (as stated in Lemma 2.5) and let  $\hat{a}_i$  be the output. By Lemma 2.5, for each  $i \in \Gamma$ ,  $\hat{a}_i$  is a  $(1 \pm \frac{\varepsilon}{50})$ -approximation to  $\mathbf{D}_M(\mathbf{A}(i, *), \mathbf{B}(i, *))$  with high probability. Based on the values of  $\hat{a}_i$ 's, we partition the indices in  $\Gamma$  into  $t$  many buckets  $\widehat{Y}_1, \dots, \widehat{Y}_t$  such that  $i \in \Gamma$  is put into  $\widehat{Y}_k$  if and only if  $(1 + \frac{\varepsilon}{50})^{k-1} \leq \hat{a}_i < (1 + \frac{\varepsilon}{50})^k$ . We define a bucket  $Y_k$  to be large or small depending on  $|\widehat{Y}_k| \geq \tau$  or not, where  $\tau = \frac{|\Gamma| \sqrt{\varepsilon T}}{n \cdot 50t}$ .

<sup>3</sup> This instantiation, for  $\mathbb{O}_{\beta, \eta}(i)$ 's for which  $\text{DIST-BET-ROWS}(i, \beta, \eta)$ 's are never called is to complete the description of function  $\mathbb{O}_{\beta, \eta}$ . This has no bearing on our algorithm as well as its analysis.

## 44:10 Distance Between Matrices Using Sublinear Projections on Hamming Cube

So, if  $|Y_k|$  is *large* (roughly say at least  $\sqrt{\varepsilon T}/t$ ), then it can be well approximated from  $|\widehat{Y}_k|$ . However, it will not be possible to estimate  $|Y_k|$  from  $|\widehat{Y}_k|$  if  $|Y_k|$  is *small*. We explain how to take care of  $Y_k$ 's with *small*  $|Y_k|$ .

Let  $L \subseteq [t]$  and  $S \subseteq [t]$  denote the set of indices for large and small buckets, that is,  $L = \{k : Y_k \text{ is large}\}$  and  $S = [t] \setminus L$ . Also, let  $I_L \subseteq [n]$  and  $I_S \subseteq [n]$  denote the set of indices of rows present in large and small buckets, respectively. From Observation 2.4,  $\mathbf{D}_M(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^n \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))$ . Let us divide the sum  $\sum_{i=1}^n \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))$  into two parts, based on  $I_L$  and  $I_S$ ,  $d_L$

$$d_L = \sum_{i \in I_L} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *)) = \sum_{k \in L} \sum_{i \in Y_k} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))$$

and  $d_S = \sum_{i \in I_S} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *)) = \sum_{k \in S} \sum_{i \in Y_k} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))$ .

That is,  $\mathbf{D}_M(\mathbf{A}, \mathbf{B}) = d_L + d_S$ . In what follows, we describe how our algorithm approximates  $d_L$  and  $d_S$  separately. A pseudocode for algorithm `DIST-SYMM-MATRIX-GUESS`( $\mathbf{A}, \mathbf{B}, \varepsilon, T$ ) can be found in the full version of this paper [13].

### Approximating $d_L$ , the contribution from *large* buckets

We can show in Lemma A.1 (i) and (ii), for each  $k \in L$ ,  $\frac{n}{|I|} |\widehat{Y}_k|$  is a  $(1 \pm \frac{\varepsilon}{50})$ -approximation to  $|Y_k|$  with high probability. Recall that  $d_L = \sum_{k \in L} \sum_{i \in Y_k} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))$ , where  $L$  denotes the set of indices present in large buckets. Our algorithm `DIST-SYMM-MATRIX-GUESS`( $\mathbf{A}, \mathbf{B}, \varepsilon, T$ ) sets  $\widehat{d}_L = \frac{n}{|I|} \sum_{k \in L} |\widehat{Y}_k| (1 + \frac{\varepsilon}{50})^k$  as an estimate for  $d_L$ . Putting everything together, we show in Lemma A.2 that the following holds with high probability.

$$\left(1 - \frac{\varepsilon}{50}\right) d_L \leq \widehat{d}_L \leq \left(1 + \frac{\varepsilon}{50}\right) d_L. \quad (4)$$

### Approximating $d_S$ , the contribution from *small* buckets

$d_S = \sum_{i \in I_S} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))$  can not be approximated directly as in the case of  $d_L$ . To get around the problem of estimating the contribution of small buckets, we partition  $\sum_{i \in I_S} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))$  into two parts by projecting row vectors  $\mathbf{A}(i, *)$ 's and  $\mathbf{B}(i, *)$ 's onto  $I_L$  and  $I_S$ :

$$d_{SL} = \sum_{i \in I_S} \mathbf{d}_H(\mathbf{A}(i, *)|_{I_L}, \mathbf{B}(i, *)|_{I_L}) \text{ and } d_{SS} = \sum_{i \in I_S} \mathbf{d}_H(\mathbf{A}(i, *)|_{I_S}, \mathbf{B}(i, *)|_{I_S}).$$

So,  $d_S = d_{SL} + d_{SS}$ .

As  $\mathbf{A}$  and  $\mathbf{B}$  are symmetric,  $d_{SL} = d_{LS} = \sum_{i \in I_L} \mathbf{d}_H(\mathbf{A}(i, *)|_{I_S}, \mathbf{B}(i, *)|_{I_S})$ . Hence,  $d_S = d_{LS} + d_{SS}$ . We approximate  $d_S$  by arguing that (i)  $d_{SS}$  is *small*, and (ii)  $d_{LS}$  can be approximated well. Informally speaking, the quantity  $d_{SL}$  is all about looking at the large buckets from the small buckets. But as handling small buckets is problematic as opposed to large buckets, we look at the small buckets from the large buckets. Now, as the matrix is symmetric, these two quantities are the same.

**(i)  $d_{SS}$  is small:** Observe that  $d_{SS}$  can be upper bounded, in terms of  $|I_S|$ , as follows:

$$d_{SS} = \sum_{i \in I_S} \mathbf{d}_H(\mathbf{A}(i, *)|_{I_S}, \mathbf{B}(i, *)|_{I_S}) = |\{(i, j) \in I_S \times I_S : \mathbf{A}(i, j) \neq \mathbf{B}(i, j)\}| \leq |I_S|^2.$$

By the definition of  $I_S$ , it is the set of indices present in small buckets ( $Y_k$ 's with  $|\widehat{Y}_k| \leq \tau$ ). With high probability, for any small bucket  $Y_k$ , we can show that  $|Y_k| \leq \sqrt{\varepsilon T}/40t$ . As there are  $t$  many buckets, with high probability,  $|I_S| = \sum_{k \in S} |Y_k| \leq \frac{n}{|\Gamma|} \tau t \leq \frac{\sqrt{\varepsilon T}}{40}$ . So, with high probability,

$$d_{SS} \leq \frac{\varepsilon T}{1600}. \quad (5)$$

The formal proof of the above equation will be given in Claim A.5.

(ii) **Approximating  $d_{LS}$ :** For  $k \in L$ , the set of indices corresponding to large buckets, let  $d_{LS}^k$  be the contribution of bucket  $Y_k$  to  $d_{LS}$ , that is,  $d_{LS}^k = \sum_{i \in Y_k} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *)|_{I_S})$ . So,  $d_{LS} = \sum_{k \in L} d_{LS}^k$ , and  $d_{LS}$  can be approximated by approximating  $d_{LS}^k$  for each  $k \in L$ . To approximate  $d_{LS}^k$ , for each  $k \in L$ , we define  $\zeta_k = d_{LS}^k / (\sum_{i \in Y_k} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *)))$ . We have already argued that  $d_{LS} = \sum_{k \in L} d_{LS}^k$  and recall that  $\sum_{k \in L} \sum_{i \in Y_k} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *)) = d_L$ . So, intuitively,  $\zeta_k$  denotes the ratio of the contribution of bucket  $Y_k$  to  $d_{LS}$  and the contribution of bucket  $Y_k$  to  $d_L$ . By our bucketing scheme, for each  $i \in Y_k$ ,  $(1 - \frac{\varepsilon}{50})^{k-1} \leq \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *)) \leq (1 + \frac{\varepsilon}{50})^k$ , that is,  $\mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))$ 's are roughly the same for each  $i \in Y_k$ . So, any  $d_{LS}^k$  can be approximated by approximating its corresponding  $\zeta_k$ . To do so, we express  $d_{LS}^k$  combinatorially as follows:

$$d_{LS}^k = |\{(i, j) : \mathbf{A}(i, j) \neq \mathbf{B}(i, j) \text{ such that } i \in Y_k \text{ and } j \in I_S\}|.$$

For each  $k \in L$ , our algorithm finds a sample  $Z_k$  of size  $|\widehat{Y}_k|$  many indices from  $\widehat{Y}_k$  with replacement. Then for each  $i \in Z_k$ , our algorithm calls  $\text{APPROX-SAMPLE}(i, \beta, \eta)$ . Recall that  $\text{APPROX-SAMPLE}(i, \beta, \eta)$  (as stated in Lemma 2.12) takes  $i \in [n]$  and  $\beta, \eta \in (0, 1)$  as inputs and returns a  $(1 \pm \beta)$ -uniform sample from the set  $\text{NEQ}(\mathbf{A}, \mathbf{B}, i)$  with probability at least  $1 - \eta$ . Let  $j \in [n]$  be the output of  $\text{NEQ}(\mathbf{A}, \mathbf{B}, i)$ . Then we check whether  $j \in I_S^4$ . Let  $C_k$  be the number of elements  $i \in Z_k$  whose corresponding call to  $\text{APPROX-SAMPLE}(i, \beta, \eta)$  returns a  $j$  with  $j \in I_S$ . Our algorithm takes  $\widehat{\zeta}_k = \frac{C_k}{|\widehat{Y}_k|}$  as an estimate for  $\zeta_k$ . We can show that, (in Lemma A.1 (i) and (ii)), for each  $k \in L$ ,  $\frac{n}{|\Gamma|} |\widehat{Y}_k|$  is a  $(1 \pm \frac{\varepsilon}{50})$ -approximation to  $Y_k$ . Also, when  $T$  is at least a suitable polynomial in  $\log n$  and  $\frac{1}{\varepsilon}$ , we show in Lemma A.1 (iii) and (iv) the followings, respectively:

- $\zeta_k \geq \frac{\varepsilon}{50}$ , then  $\widehat{\zeta}_k$  is a  $(1 \pm \frac{\varepsilon}{40})$ -approximation to  $\zeta_k$  with high probability,
- we show that if  $\zeta_k \leq \frac{\varepsilon}{50}$ , then  $\widehat{\zeta}_k \leq \frac{\varepsilon}{30}$  holds with high probability.

Hence,  $\widehat{d}_{LS} = \frac{n}{|\Gamma|} \sum_{k \in L} \widehat{\zeta}_k |\widehat{Y}_k| (1 + \frac{\varepsilon}{50})^k$  satisfies  $(1 - \frac{\varepsilon}{15}) d_{LS} - \frac{\varepsilon}{25} d_L \leq \widehat{d}_{LS} \leq (1 + \frac{\varepsilon}{15}) d_{LS} + \frac{\varepsilon}{25} d_L$  with high probability. Note that the additive factor in terms of  $d_L$  is due to the way  $\zeta_k$ 's are defined.

In fact our algorithm  $\text{DIST-SYMM-MATRIX-GUESS}(\mathbf{A}, \mathbf{B}, \varepsilon, T)$  sets  $\widehat{d}_S = \widehat{d}_{LS}$ . The intuition behind setting  $\widehat{d}_S = \widehat{d}_{LS}$  is that  $d_S = d_{LS} + d_{SS}$  and  $d_{SS}$  is small. So, with high probability,

$$\left(1 - \frac{\varepsilon}{15}\right) d_{LS} - \frac{\varepsilon}{25} d_L \leq \widehat{d}_S \leq \left(1 + \frac{\varepsilon}{15}\right) d_{LS} + \frac{\varepsilon}{25} d_L. \quad (6)$$

<sup>4</sup> The reason for checking  $j \in I_S$  can be observed from the definition of  $d_{LS}^k = |\{(i, j) : \mathbf{A}(i, j) \neq \mathbf{B}(i, j) \text{ such that } i \in Y_k \text{ and } j \in I_S\}|$ .

## 44:12 Distance Between Matrices Using Sublinear Projections on Hamming Cube

The above will be formally proved in Claim A.6. By Equations 6 and 5, we get  $\widehat{d}_S$  is an estimate for  $d_S$  that satisfies the following with high probability.

$$\left(1 - \frac{\varepsilon}{15}\right) d_S - \frac{\varepsilon T}{1600} - \frac{\varepsilon}{25} d_L \leq \widehat{d}_S \leq \left(1 + \frac{\varepsilon}{15}\right) d_S + \frac{\varepsilon}{25} d_L \quad (7)$$

We will formally show the above equation in Lemma A.3.

### Final output returned by our algorithm $\text{DIST-SYMM-MATRIX-GUESS}(\mathbf{A}, \mathbf{B}, \varepsilon, T)$

Finally, our algorithm returns  $\widehat{d} = \widehat{d}_L + \widehat{d}_S$  as an estimation for  $\mathbf{D}_M(\mathbf{A}, \mathbf{B})$ . Recall that  $\mathbf{D}_M(\mathbf{A}, \mathbf{B}) = d_S + d_L$ . From Equations 4 and 7,  $\widehat{d}$  satisfies, with high probability,

$$\left(1 - \frac{\varepsilon}{10}\right) \mathbf{D}_M(\mathbf{A}, \mathbf{B}) - \frac{\varepsilon}{1600} T \leq \widehat{d} \leq \left(1 + \frac{\varepsilon}{10}\right) \mathbf{D}_M(\mathbf{A}, \mathbf{B}).$$

### The query complexity analysis of algorithm $\text{DIST-SYMM-MATRIX-GUESS}(\mathbf{A}, \mathbf{B}, \varepsilon, T)$

Note that the discussed algorithm works when  $T$  is at least a suitable polynomial in  $\log n$  and  $1/\varepsilon$ . Moreover, the algorithm calls each of  $\text{DIST-BET-ROWS}(i, \beta, \eta)$  and  $\text{APPROX-SAMPLE}(i, \beta, \eta)$  for  $\widetilde{\mathcal{O}}\left(n/\sqrt{T}\right)$  times. Note that  $\beta = \varepsilon/50$  and  $\eta = 1/\text{poly}(n)$ . So, the number of IP queries, made by each call to  $\text{DIST-BET-ROWS}(i, \beta, \eta)$  as well as  $\text{APPROX-SAMPLE}(i, \beta, \eta)$ , is  $\widetilde{\mathcal{O}}(1)$  by Lemma 2.5 and 2.12. Hence, the number of IP queries made by our algorithm is  $\widetilde{\mathcal{O}}\left(n/\sqrt{T}\right)$ . The formal proof of the correctness of  $\text{DIST-SYMM-MATRIX-GUESS}(\mathbf{A}, \mathbf{B}, \varepsilon, T)$  is in Appendix A.

## 3 Distance between two arbitrary matrices

In this Section, we prove our main result (stated as Theorem 1.2 in Section 1).

► **Theorem 3.1** (Theorem 1.2 restated). *There exists an algorithm that has IP query access to unknown matrices  $\mathbf{A}$  and  $\mathbf{B}$ , takes an  $\varepsilon \in (0, 1)$  as an input, and returns a  $(1 \pm \varepsilon)$  approximation to  $\mathbf{D}_M(\mathbf{A}, \mathbf{B})$  with high probability, and makes  $\mathcal{O}\left(\frac{n}{\sqrt{\mathbf{D}_M(\mathbf{A}, \mathbf{B})}} \text{poly}\left(\log n, \frac{1}{\varepsilon}\right)\right)$  queries.*

To prove the above theorem, we use Lemma 2.1 for estimating  $\mathbf{D}_M(\mathbf{A}, \mathbf{B})$  when both  $\mathbf{A}$  and  $\mathbf{B}$  are symmetric. Let  $\Delta^{\mathbf{A}}$  be a matrix defined as  $\Delta^{\mathbf{A}}(i, j) = \mathbf{A}(i, j)$  if  $i \leq j$ , and  $\Delta^{\mathbf{A}}(i, j) = \mathbf{A}(j, i)$ , otherwise. Also, let  $\Delta_{\mathbf{A}}$  be a matrix defined as  $\Delta_{\mathbf{A}}(i, j) = \mathbf{A}(i, j)$  if  $i \geq j$ , and  $\Delta_{\mathbf{A}}(i, j) = \mathbf{A}(j, i)$ , otherwise. Similarly, we can also define  $\Delta^{\mathbf{B}}$  and  $\Delta_{\mathbf{B}}$  similarly. Observe that  $\Delta^{\mathbf{A}}, \Delta_{\mathbf{A}}, \Delta^{\mathbf{B}}$  and  $\Delta_{\mathbf{B}}$  are symmetric matrices, and

$$\mathbf{D}_M(\mathbf{A}, \mathbf{B}) = \frac{1}{2} [\mathbf{D}_M(\Delta_{\mathbf{A}}, \Delta_{\mathbf{B}}) + \mathbf{D}_M(\Delta^{\mathbf{A}}, \Delta^{\mathbf{B}})].$$

So, we can report a  $(1 \pm \varepsilon)$ -approximation to  $\mathbf{D}_M(\mathbf{A}, \mathbf{B})$  by finding a  $(1 \pm \frac{\varepsilon}{2})$ -approximation to both  $\mathbf{D}_M(\Delta_{\mathbf{A}}, \Delta_{\mathbf{B}})$  and  $\mathbf{D}_M(\Delta^{\mathbf{A}}, \Delta^{\mathbf{B}})$  with high probability. This is possible, by Lemma 2.1, if we have IP query access to matrices  $\Delta_{\mathbf{A}}, \Delta_{\mathbf{B}}, \Delta^{\mathbf{A}}$  and  $\Delta^{\mathbf{B}}$ . But we do not have IP query access to  $\Delta_{\mathbf{A}}, \Delta_{\mathbf{B}}, \Delta^{\mathbf{A}}$  and  $\Delta^{\mathbf{B}}$  explicitly. However, we can simulate IP query access to matrices  $\Delta_{\mathbf{A}}$  and  $\Delta^{\mathbf{A}}$  ( $\Delta_{\mathbf{B}}$  and  $\Delta^{\mathbf{B}}$ ) with IP query access to matrix  $\mathbf{A}$  ( $\mathbf{B}$ ), respectively as stated and proved in the observation below. Hence, we are done with the proof of Theorem 3.1



► **Observation 3.2.** An IP query to matrix  $\Delta_{\mathbf{A}}$  ( $\Delta^{\mathbf{A}}$ ) can be answered by using two IP queries to matrix  $\mathbf{A}$ . Also, an IP query to  $\Delta_{\mathbf{B}}$  ( $\Delta^{\mathbf{B}}$ ) can be answered by using two IP queries to matrix  $\mathbf{B}$ .

**Proof.** We prove how an IP query to matrix  $\Delta_{\mathbf{A}}$  can be answered by using two IP queries to matrix  $\mathbf{A}$ . Other parts of the statement can be proved similarly.

Consider an IP query  $\langle \Delta_{\mathbf{A}}(i, *), \mathbf{r} \rangle$  to  $\Delta_{\mathbf{A}}$ , where  $i \in [n]$  and  $\mathbf{r} = (r_1, \dots, r_n) \in \mathbb{R}^n$ . Let  $\mathbf{r}^{\leq i}$  and  $\mathbf{r}^{> i}$  in  $\mathbb{R}^n$  be two vectors defined as follows:  $r_j^{\leq i} = r_j$  if  $j \leq i$ , and  $r_j^{\leq i} = 0$ , otherwise.  $r_j^{> i} = r_j$  if  $j > i$ , and  $r_j^{> i} = 0$ , otherwise. Now, we can deduce that

$$\begin{aligned} \langle \Delta_{\mathbf{A}}(i, *), \mathbf{r} \rangle &= \sum_{j=1}^i \Delta_{\mathbf{A}}(i, j) r_j + \sum_{j=i+1}^n \Delta_{\mathbf{A}}(i, j) r_j \\ &= \sum_{j=1}^i \mathbf{A}(i, j) r_j + \sum_{j=i+1}^n \mathbf{A}(j, i) r_j \\ &= \sum_{j=1}^n \mathbf{A}(i, j) r_j^{\leq i} + \sum_{j=1}^n \mathbf{A}(j, i) r_j^{> i} \\ &= \langle \mathbf{A}(i, *), \mathbf{r}^{\leq i} \rangle + \langle \mathbf{A}(*, i), \mathbf{r}^{> i} \rangle \end{aligned}$$

From the above expression, it is clear that an IP query of the form  $\langle \Delta_{\mathbf{A}}(i, *), \mathbf{r} \rangle$  to matrix  $\Delta_{\mathbf{A}}$  can be answered by making two IP queries of the form  $\langle \mathbf{A}(i, *), \mathbf{r}^{\leq i} \rangle$  and  $\langle \mathbf{A}(*, i), \mathbf{r}^{> i} \rangle$  to matrix  $\mathbf{A}$ . ◀

## 4 Lower bound results

In this Section, if we ignore polylogarithmic term, we show that (in Theorem 4.1) our algorithm to estimate  $\mathbf{D}_{\mathbf{M}}(\mathbf{A}, \mathbf{B})$  using IP query is tight. Apart from Theorem 4.1, we also prove that (in Theorem 4.2) the query complexity of estimating  $\mathbf{D}_{\mathbf{M}}(\mathbf{A}, \mathbf{B})$  using DEC-IP is quadratically larger than that of using IP. The results are formally stated as follows. The lower bounds hold even if the matrices  $\mathbf{A}$  and  $\mathbf{B}$  are symmetric matrices, and one matrix (say  $\mathbf{A}$ ) is known and one matrix (say  $\mathbf{B}$ ) is unknown.

► **Theorem 4.1.** *Let  $\mathbf{A}$  and  $\mathbf{B}$  denote the known and unknown (symmetric) matrices, respectively. Also let  $T \in \mathbb{N}$ . Any algorithm having IP query access to matrix  $\mathbf{B}$ , that distinguishes between  $\mathbf{D}_{\mathbf{M}}(\mathbf{A}, \mathbf{B}) = 0$  or  $\mathbf{D}_{\mathbf{M}}(\mathbf{A}, \mathbf{B}) \geq T$  with probability  $2/3$ , makes  $\Omega\left(\frac{n}{\sqrt{T}}\right)$  queries to  $\mathbf{B}$ .*

► **Theorem 4.2.** *Let  $\mathbf{A}$  and  $\mathbf{B}$  denote the known and unknown matrices, respectively. Also let  $T \in \mathbb{N}$ . Any algorithm having DEC-IP query access to matrix  $\mathbf{B}$ , that distinguishes between  $\mathbf{D}_{\mathbf{M}}(\mathbf{A}, \mathbf{B}) = 0$  or  $\mathbf{D}_{\mathbf{M}}(\mathbf{A}, \mathbf{B}) \geq T$  with probability  $2/3$ , makes  $\Omega\left(\frac{n^2}{T}\right)$  queries to  $\mathbf{B}$ .*

Recall that every ME query to a matrix can be simulated by using a DEC-IP. Hence, the following corollary follows.

► **Corollary 4.3.** *Let  $\mathbf{A}$  and  $\mathbf{B}$  denote the known and unknown matrices, respectively. Also let  $T \in \mathbb{N}$ . Any algorithm having ME query access to matrix  $\mathbf{B}$ , that distinguishes between  $\mathbf{D}_{\mathbf{M}}(\mathbf{A}, \mathbf{B}) = 0$  or  $\mathbf{D}_{\mathbf{M}}(\mathbf{A}, \mathbf{B}) \geq T$  with probability  $2/3$ , makes  $\Omega\left(\frac{n^2}{T}\right)$  queries to  $\mathbf{B}$ .*

We prove Theorems 4.1 and 4.2 by using a reduction from a problem known as DISJOINTNESS in two party communication complexity (See Appendix B).

#### 4.1 Proof of Theorem 4.1

Without loss of generality, assume that  $\sqrt{T}$  is an integer that divides  $N$ . We prove the (stated lower bound) by a reduction from  $\text{DISJOINTNESS}_N$  where  $N = n/\sqrt{T}$ . Let  $\mathbf{x}$  and  $\mathbf{y}$  in  $\{0, 1\}^N$  be the inputs of Alice and Bob, respectively. Now consider matrix  $\mathbf{B}$ , that depends on both  $\mathbf{x}$  and  $\mathbf{y}$ , described as follows.

$\mathbf{B}_{11}$	$\mathbf{B}_{12}$	$\mathbf{B}_{13}$
$\mathbf{B}_{21}$	$\mathbf{B}_{22}$	$\mathbf{B}_{23}$
$\mathbf{B}_{31}$	$\mathbf{B}_{32}$	$\mathbf{B}_{33}$

■ **Figure 1** A pictorial illustration of a block matrix  $\mathbf{B}$  considered in the proof of Theorem 4.2, where  $N = 3$ .

#### Description of matrices $\mathbf{A}$ and $\mathbf{B}$

- (i) matrix  $\mathbf{A}$  is the null matrix;
- (ii) matrix  $\mathbf{B}$  is a block diagonal matrix where  $\mathbf{B}_1, \dots, \mathbf{B}_N$  are diagonal blocks of order  $\sqrt{T} \times \sqrt{T}$  (See Figure 2 for an illustration);
- (iii) Consider  $k \in [N]$ . If  $x_k = y_k = 1$ , then  $\mathbf{B}_k(i, j) = 1$  for each  $i, j \in [\sqrt{T}]$ , that is,  $\mathbf{B}_k$  is an all-one matrix. Otherwise,  $\mathbf{B}_k$  is a null matrix.

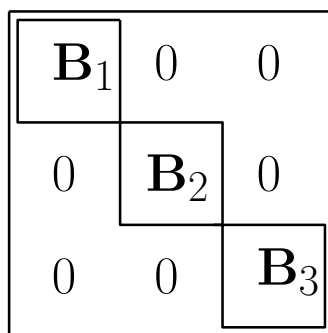
From the description, matrices  $\mathbf{A}$  and  $\mathbf{B}$  are symmetric matrices. Moreover, if  $\mathbf{x}$  and  $\mathbf{y}$  are disjoint, then all of the  $N$  block matrices are null matrices, that is,  $\mathbf{B}$  is also a null matrix. If  $\mathbf{x}$  and  $\mathbf{y}$  are not disjoint, then there is a  $k \in [N]$  such that  $\mathbf{B}_k$  is an all-one matrix, that is, matrix  $\mathbf{B}$  has at least  $T$  many 1s. Recall that here  $\mathbf{A}$  is a null matrix. Hence,  $\mathbf{D}_M(\mathbf{A}, \mathbf{B}) = 0$  if  $\mathbf{x}$  and  $\mathbf{y}$  are disjoint, and  $\mathbf{D}_M(\mathbf{A}, \mathbf{B}) \geq T$  if  $\mathbf{x}$  and  $\mathbf{y}$  are not disjoint.

Observe that we will be done with the proof for the stated lower bound by arguing that Alice and Bob can generate the answer to any IP query, to matrix  $\mathbf{B}$ , with 2 bits of communication. Consider a row IP query  $\langle \mathbf{B}(i, *), \mathbf{r} \rangle$  to  $\mathbf{B}$  for some  $i \in [n]$  and  $\mathbf{r} \in \{0, 1\}^n$ <sup>5</sup>. From the construction of the matrix  $\mathbf{B}$ , there exists a matrix  $\mathbf{B}_j$ , for some  $j \in [N]$ , that completely determines  $\mathbf{B}(i, *)$ . Also, observe that,  $\mathbf{B}_j$  depends on  $x_j$  and  $y_j$  only. So, Alice and Bob can determine  $\mathbf{B}_j$  (hence  $\mathbf{B}(i, *)$ ) with 2 bits of communication. As  $\mathbf{B}$  is a symmetric matrix, there is no need to consider column IP queries as such queries can be answered by using row IP queries.

#### 4.2 Proof of Theorem 4.2

Here also, we assume that  $\sqrt{T}$  is an integer and  $\sqrt{T}$  divides  $N$ , and prove the stated lower bound by a reduction from  $\text{DISJOINTNESS}_N$  where  $N = n^2/T$ . Let  $\mathbf{x}$  and  $\mathbf{y}$  in  $\{0, 1\}^N$  be the inputs of Alice and Bob, respectively. Now consider matrix  $\mathbf{B}$ , that depends on both  $\mathbf{x}$  and  $\mathbf{y}$ , described as follows. In the following description, consider a canonical mapping  $\phi: [N] \rightarrow \left[ \frac{n}{\sqrt{T}} \right] \times \left[ \frac{n}{\sqrt{T}} \right]$ . Note that  $\phi$  is known to both Alice and Bob apriori.

<sup>5</sup> The proof goes through even if  $\mathbf{r} \in \mathbb{R}^n$ .



■ **Figure 2** A pictorial illustration of a block diagonal matrix  $B$  considered in the proof of Theorem 4.1, where  $N = 3$ .

### Description of matrices $\mathbf{A}$ and $\mathbf{B}$

- (i) matrix  $\mathbf{A}$  is an all 1 matrix;
- (ii) matrix  $\mathbf{B}$  is a block matrix where  $\mathbf{B}_{ij}$ 's  $\left(i, j \in \left[\frac{n}{\sqrt{T}}\right]\right)$  are blocks of order  $\sqrt{T} \times \sqrt{T}$  (See Figure 1 for an illustration);
- (iii) Consider  $k \in [N]$ . If  $x_k = y_k = 1$ , then  $\mathbf{B}_{\phi(k)} = 0$ <sup>6</sup> for each  $i, j \in [\sqrt{T}]$ , that is,  $\mathbf{B}_{\phi(k)}$  is an all 0 matrix. Otherwise,  $\mathbf{B}_{\phi(k)}$  is an all 1 matrix.

From the description, matrices  $\mathbf{A}$  and  $\mathbf{B}$  are symmetric matrices. Moreover, if  $\mathbf{x}$  and  $\mathbf{y}$  are disjoint, then all of the  $N$  block matrices are all 1 matrices, that is,  $\mathbf{B}$  is also an all 1 matrix. If  $\mathbf{x}$  and  $\mathbf{y}$  are not disjoint, then there is (exactly) one  $k \in [N]$  such that  $\mathbf{B}_k$  is a null matrix, that is, matrix  $\mathbf{B}$  has exactly  $T$  many 0s. Recall that here  $\mathbf{A}$  is a null matrix. Hence,  $\mathbf{D}_M(\mathbf{A}, \mathbf{B}) = 0$  if  $\mathbf{x}$  and  $\mathbf{y}$  are disjoint, and,  $\mathbf{D}_M(\mathbf{A}, \mathbf{B}) = T$  if  $\mathbf{x}$  and  $\mathbf{y}$  are not disjoint.

Observe that we will be done with the proof for the stated lower bound by arguing that Alice and Bob can generate the answer to any DEC-IP query, to matrix  $\mathbf{B}$ , with 2 bits of communication. Consider a row DEC-IP query  $\langle \mathbf{B}(i, *), \mathbf{r} \rangle$  to  $\mathbf{B}$  for some  $i \in [n]$  and  $\mathbf{r} \in \{0, 1\}^n$ . Without loss of generality, assume that  $\mathbf{r}$  is not a null matrix, as in this case Alice and Bob can decide  $\langle \mathbf{B}(i, *), \mathbf{r} \rangle = 0$  trivially without any communication. Consider a partition of  $\mathbf{r}$  into  $N/\sqrt{T}$  subvectors  $\mathbf{r}_1, \dots, \mathbf{r}_{n/\sqrt{T}} \in \{0, 1\}^{\sqrt{T}}$  in the natural way (See Figure 1 for an illustration). Now we analyze by making two cases. If two of the  $\mathbf{r}_j$ s are not null vectors, from the construction of the matrix  $\mathbf{B}$ , then  $\langle \mathbf{B}(i, *), \mathbf{r} \rangle > 0$ . So, in this case, Alice and Bob report  $\langle \mathbf{B}(i, *), \mathbf{r} \rangle \neq 0$  without any communication between them. Now consider the case when exactly one of the  $\mathbf{r}_j$  is not a null vector. Once again, from the construction of  $\mathbf{B}$ , deciding whether  $\langle \mathbf{B}(i, *), \mathbf{r} \rangle = 0$  is equivalent to deciding whether a particular block (say  $\mathbf{B}_{xy}$ ) is a null matrix. It is because  $\mathbf{r}$  is not a null vector, and each block in  $\mathbf{B}$  is either a null matrix or an all 1 matrix. Let  $k \in \left[\frac{n}{\sqrt{T}}\right]$  be such that  $\phi(k) = (x, y)$ . Note that  $\mathbf{B}_{xy}$  is a null matrix if and only if  $x_k = y_k = 1$ . So, Alice and Bob can determine whether  $\langle \mathbf{B}(i, *), \mathbf{r} \rangle = 0$  with 2 bits of communication. As  $\mathbf{B}$  is a symmetric matrix, there is no need to consider column DEC-IP queries as such queries can be answered by using row IP queries.

<sup>6</sup> Here we abuse the notation slightly. If  $\phi(k) = (i, j)$ , we denote  $\mathbf{B}_{ij}$  by  $\mathbf{B}_{\phi(k)}$ .

## 5 Conclusion

Recall that in an IP as well as in DEC-IP queries, a vector  $\mathbf{v} \in \{0, 1\}^n$  is given as input along with an index for row or column of an unknown matrix. Let  $\text{IP}_{\mathbb{R}}$  and  $\text{DEC-IP}_{\mathbb{R}}$  be the extension of IP and DEC-IP when  $\mathbf{v}$  is a vector in  $\mathbb{R}^n$ . Now let us have a look into Table 1. The lower bound of  $\Omega(n/\sqrt{D})$ , on the number of IP queries to estimate  $D = \mathbf{D}_M(\mathbf{A}, \mathbf{B})$ , also holds even when we have an access to  $\text{IP}_{\mathbb{R}}$  oracle. This also implies a lower bound of  $\Omega(n/\sqrt{D})$  on the number of DEC-IP queries to solve the problem at hand. But our lower bound proof of  $\Omega(n^2/D)$  on the number of DEC-IP queries does not work when we have access to  $\text{IP}_{\mathbb{R}}$  oracle. So, we leave the following problem as open.

### Open problem

What is the query complexity of estimating  $\mathbf{D}_M(\mathbf{A}, \mathbf{B})$  when we have  $\text{DEC-IP}_{\mathbb{R}}$  access to matrices  $\mathbf{A}$  and  $\mathbf{B}$ ?

---

### References

- 1 <https://www.mathworks.com/products/matlab.html>.
- 2 <https://www.mathworks.com/help/stats/pdist.html>.
- 3 <https://docs.scipy.org/doc/scipy-0.7.x/scipy-ref.pdf>.
- 4 <https://developer.download.nvidia.com/cg/dot.html>.
- 5 <https://software.intel.com/content/www/us/en/develop/documentation/cpp-compiler-developer-guide-and-reference/top/compiler-reference/intrinsics/intrinsics-for-intel-streaming-simd-extensions-4-intel-sse4/vectorizing-compiler-and-media-accelerators/floating-point-dot-product-intrinsics.html>.
- 6 Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, 2003. doi:10.1016/S0022-0000(03)00025-4.
- 7 Pankaj K. Agarwal, Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. Computing the discrete fréchet distance in subquadratic time. *SIAM J. Comput.*, 43(2):429–449, 2014. doi:10.1137/130920526.
- 8 Pankaj K. Agarwal, Kyle Fox, Abhinandan Nath, Anastasios Sidiropoulos, and Yusu Wang. Computing the gromov-hausdorff distance for metric trees. *ACM Trans. Algorithms*, 14(2), 2018. doi:10.1145/3185466.
- 9 Pankaj K. Agarwal, Sariel Har-Peled, Micha Sharir, and Yusu Wang. Hausdorff distance under translation for points and balls. 6(4), 2010. doi:10.1145/1824777.1824791.
- 10 Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. Testing convexity of figures under the uniform distribution. In Sándor P. Fekete and Anna Lubiw, editors, *32nd International Symposium on Computational Geometry, SoCG 2016, June 14-18, 2016, Boston, MA, USA*, volume 51 of *LIPICs*, pages 17:1–17:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.SoCG.2016.17.
- 11 Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. Tolerant testers of image properties. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPICs*, pages 90:1–90:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.ICALP.2016.90.
- 12 Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. Testing convexity of figures under the uniform distribution. *Random Struct. Algorithms*, 54(3):413–443, 2019. doi:10.1002/rsa.20797.
- 13 Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Distance estimation between unknown matrices using sublinear projections on hamming cube. *CoRR*, abs/2107.02666, 2021. arXiv:2107.02666.

- 14 Arijit Bishnu, Arijit Ghosh, Gopinath Mishra, and Manaswi Paraashar. Inner product oracle can estimate and sample. *CoRR*, abs/1906.07398, 2019. [arXiv:1906.07398](https://arxiv.org/abs/1906.07398).
- 15 Bernard Chazelle, Ding Liu, and Avner Magen. Sublinear geometric algorithms. *SIAM J. Comput.*, 35(3):627–646, 2005. doi:10.1137/S009753970444572X.
- 16 Artur Czumaj, Funda Ergün, Lance Fortnow, Avner Magen, Ilan Newman, Ronitt Rubinfeld, and Christian Sohler. Sublinear-time approximation of euclidean minimum spanning tree. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA*, pages 813–822. ACM/SIAM, 2003. URL: <http://dl.acm.org/citation.cfm?id=644108.644242>.
- 17 Artur Czumaj and Christian Sohler. Property testing with geometric queries. In Friedhelm Meyer auf der Heide, editor, *Algorithms - ESA 2001, 9th Annual European Symposium, Aarhus, Denmark, August 28-31, 2001, Proceedings*, volume 2161 of *Lecture Notes in Computer Science*, pages 266–277. Springer, 2001. doi:10.1007/3-540-44676-1\_22.
- 18 Artur Czumaj, Christian Sohler, and Martin Ziegler. Property testing in computational geometry. In Mike Paterson, editor, *Algorithms - ESA 2000, 8th Annual European Symposium, Saarbrücken, Germany, September 5-8, 2000, Proceedings*, volume 1879 of *Lecture Notes in Computer Science*, pages 155–166. Springer, 2000. doi:10.1007/3-540-45253-2\_15.
- 19 Anne Driemel and Sariel Har-Peled. Jaywalking your dog: Computing the fréchet distance with shortcuts. *SIAM J. Comput.*, 42(5):1830–1866, 2013. doi:10.1137/120865112.
- 20 Anne Driemel, Sariel Har-Peled, and Carola Wenk. Approximating the fréchet distance for realistic curves in near linear time. *Discret. Comput. Geom.*, 48(1):94–127, 2012. doi:10.1007/s00454-012-9402-z.
- 21 D. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 1st edition, 2009.
- 22 Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. doi:10.1017/9781108135252.
- 23 John L. Hennessy and David A. Patterson. *Computer Architecture - A Quantitative Approach (5. ed.)*. Morgan Kaufmann, 2012.
- 24 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.
- 25 Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, 1997.
- 26 Sofya Raskhodnikova. Approximate testing of visual properties. In Sanjeev Arora, Klaus Jansen, José D. P. Rolim, and Amit Sahai, editors, *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques, 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2003 and 7th International Workshop on Randomization and Approximation Techniques in Computer Science, RANDOM 2003, Princeton, NJ, USA, August 24-26, 2003, Proceedings*, volume 2764 of *Lecture Notes in Computer Science*, pages 370–381. Springer, 2003. doi:10.1007/978-3-540-45198-3\_31.
- 27 Dana Ron and Gilad Tsur. Testing properties of sparse images. *ACM Trans. Algorithms*, 10(4):17:1–17:52, 2014. doi:10.1145/2635806.
- 28 Jason Sanders and Edward Kandrot. *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Addison-Wesley, Upper Saddle River, NJ, 2010.

**A Formal correctness proof of  $\text{DIST-SYMM-MATRIX-GUESS}(\mathbf{A}, \mathbf{B}, \varepsilon, T)$** 

From the above discussion, we need to only prove for the case when  $T$  is at least a suitable polynomial in  $\log n$  and  $\frac{1}{\varepsilon}$ . The proof (of correctness) is based on the following lemma that can be proved by mainly using Chernoff bound (See Appendix C) and some specific details of the algorithm.

► **Lemma A.1** (Intermediate Lemma needed to prove the correctness). *Let  $\varepsilon \in (0, \frac{1}{2})$ ,  $\beta = \frac{\varepsilon}{50}$  and  $\eta = \frac{1}{\text{poly}(n)}$ . Consider an oracle  $\mathbb{O}_{\beta, \eta} : [n] \rightarrow \mathbb{R}$ , as defined in Definition 2.13, with respect to which algorithm  $\text{DIST-SYMM-MATRIX-GUESS}(\mathbf{A}, \mathbf{B}, \varepsilon, T)$  found  $\widehat{Y}_1, \dots, \widehat{Y}_t \subseteq \Gamma$ . Let  $Y_1, \dots, Y_t$  be the buckets into which  $[n]$  is partitioned w.r.t.  $\mathbb{O}_{\beta, \eta}$ . Then, for  $k \in [t]$ ,*

- (i) *if  $|Y_k| \geq \frac{\sqrt{\varepsilon T}}{50t}$ , then  $\mathbb{P}\left(\left|\frac{n}{|\Gamma|}|\widehat{Y}_k| - |Y_k|\right| \geq \frac{\varepsilon}{50}|Y_k|\right) \leq \frac{1}{\text{poly}(n)}$ ;*
- (ii) *if  $|Y_k| \leq \frac{\sqrt{\varepsilon T}}{50t}$ , then  $\mathbb{P}\left(\frac{n}{|\Gamma|}|\widehat{Y}_k| \geq \frac{\sqrt{\varepsilon T}}{40t}\right) \leq \frac{1}{\text{poly}(n)}$ ;*
- (iii) *if  $\zeta_k \geq \frac{\varepsilon}{50}$ , then  $\mathbb{P}\left(\left|\widehat{\zeta}_k - \zeta_k\right| \geq \frac{\varepsilon}{40}\zeta_k\right) \leq \frac{1}{\text{poly}(n)}$ ;*
- (iv) *if  $\zeta_k \leq \frac{\varepsilon}{50}$ , then  $\mathbb{P}\left(\widehat{\zeta}_k \geq \frac{\varepsilon}{30}\right) \leq \frac{1}{\text{poly}(n)}$ .*

The proof of the above lemma is presented in the full version of this paper [13]. Here, we prove the correctness of algorithm  $\text{DIST-SYMM-MATRIX}(\mathbf{A}, \mathbf{B}, \varepsilon, T)$  via two claims stated below.

► **Lemma A.2** (Approximating  $d_L$ ).  $(1 - \frac{\varepsilon}{50})d_L \leq \widehat{d}_L \leq (1 + \frac{\varepsilon}{50})d_L$  with high probability.

► **Lemma A.3** (Approximating  $d_S$ ).  $(1 - \frac{\varepsilon}{15})d_S - \frac{\varepsilon T}{1600} - \frac{\varepsilon}{25}d_L \leq \widehat{d}_S \leq (1 + \frac{\varepsilon}{15})d_S + \frac{\varepsilon}{25}d_L$ . holds with high probability.

Recall that  $\mathbf{D}_M(\mathbf{A}, \mathbf{B}) = d_L + d_S$ . Assuming that the above two claims hold,  $\widehat{d} = \widehat{d}_L + \widehat{d}_S$  satisfies  $(1 - \frac{\varepsilon}{10})\mathbf{D}_M(\mathbf{A}, \mathbf{B}) - \frac{\varepsilon}{1600}T \leq \widehat{d} \leq (1 + \frac{\varepsilon}{10})\mathbf{D}_M(\mathbf{A}, \mathbf{B})$  with high probability. Note that  $\widehat{d}$  satisfies the requirement for an estimate of  $\mathbf{D}_M(\mathbf{A}, \mathbf{B})$  as stated in Lemma 2.2. Now, it remains to show Lemma A.2 and A.3. We first prove the following claim that follows from our bucketing scheme and will be used in the proofs of Lemma A.2 and A.3. The following claim establishes the connection between the size of a bucket with the sum of the distances between rows (with indices in the same bucket) of matrices  $\mathbf{A}$  and  $\mathbf{B}$ .

► **Claim A.4.**  $\forall k \in [t]$ ,

$$\sum_{i \in Y_k} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *)) \leq |Y_k| \left(1 + \frac{\varepsilon}{50}\right)^k \leq \left(1 + \frac{\varepsilon}{50}\right) \sum_{i \in Y_k} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, )),$$

holds with probability at least  $1 - \frac{1}{\text{poly}(n)}$ .

Proof.  $Y_1, \dots, Y_t \subseteq [n]$  be the buckets into which  $[n]$  is partitioned, where

$$Y_k = \{i \in [n] : \left(1 + \frac{\varepsilon}{50}\right)^{k-1} \leq \mathbb{O}_{\beta, \eta}(i) < \left(1 + \frac{\varepsilon}{50}\right)^k\}.$$

So,

$$\text{For each } k \in [t], \sum_{i \in Y_k} \mathbb{O}_{\beta, \eta}(i) \leq |Y_k| \left(1 + \frac{\varepsilon}{50}\right)^k \leq \left(1 + \frac{\varepsilon}{50}\right) \sum_{i \in Y_k} \mathbb{O}_{\beta, \eta}(i) \quad (8)$$

Here  $\beta = \frac{\varepsilon}{50}$  and  $\eta = \frac{1}{\text{poly}(n)}$ .

Oracle  $\mathbb{O}_{\beta, \eta} : [n] \rightarrow \mathbb{R}$  is a function, as defined in Definition 2.13, such that  $\mathbb{O}_{\beta, \eta}(i)$  equals to  $\widehat{a}_i$ , the value returned by  $\text{DIST-BET-ROWS}(i, \beta, \eta)$ , if the algorithm

$\text{DIST-BET-ROWS}(i, \beta, \eta)$  is called (once). Otherwise,  $\mathbb{O}_{\beta, \eta}(i)$  is set to some  $(1 \pm \beta)$ -approximation to  $\mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))$ . So, Equation 8 implies that the following holds with probability at least  $1 - \frac{1}{\text{poly}(n)}$ .

$$\forall k \in [t], \sum_{i \in Y_k} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *)) \leq |Y_k| \left(1 + \frac{\varepsilon}{50}\right)^k \leq \left(1 + \frac{\varepsilon}{50}\right) \sum_{i \in Y_k} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *)). \blacktriangleleft$$

Now we will show Lemma A.2.

**Proof of Lemma A.2.** Note that  $d_L = \sum_{i \in I_L} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))$  and  $\widehat{d}_L = \frac{n}{|\Gamma|} \sum_{k \in L} |\widehat{Y}_k| \left(1 + \frac{\varepsilon}{50}\right)^k$ .

By the definition of  $d_L$  as well as Claim A.4, we get

$$\mathbb{P}\left(d_L \leq \sum_{k \in L} |Y_k| \left(1 + \frac{\varepsilon}{50}\right)^k \leq \left(1 + \frac{\varepsilon}{50}\right) d_L\right) \geq 1 - \frac{1}{\text{poly}(n)}. \quad (9)$$

Recall that, for  $k \in [t]$  in the set  $L$  of large buckets,  $\widehat{Y}_k \geq \tau$ . Here  $\tau = \frac{|\Gamma|}{n} \frac{\sqrt{\varepsilon T}}{40t}$ . By Lemma A.1 (ii) and (i), for each  $k \in L$ ,  $\frac{n}{|\Gamma|} |\widehat{Y}_k|$  is an  $(1 \pm \frac{\varepsilon}{50})$ -approximation to  $|Y_k|$  with probability at least  $1 - \frac{1}{\text{poly}(n)}$ . So, the following holds with probability at least  $1 - \frac{1}{\text{poly}(n)}$ .

$$\left(1 - \frac{\varepsilon}{50}\right) d_L \leq \frac{n}{|\Gamma|} \sum_{k \in L} |\widehat{Y}_k| \left(1 + \frac{\varepsilon}{50}\right)^k \leq \left(1 + \frac{\varepsilon}{50}\right)^2 d_L \quad (10)$$

By the definition of  $\widehat{d}_L$  along with taking  $\varepsilon \in (0, \frac{1}{2})$ , we conclude that

$$\mathbb{P}\left(\left(1 - \frac{\varepsilon}{20}\right) d_L \leq \widehat{d}_L \leq \left(1 + \frac{\varepsilon}{20}\right) d_L\right) \geq 1 - \frac{1}{\text{poly}(n)}. \quad \blacktriangleleft$$

**Proof of Lemma A.3.** Recall that  $d_S = \sum_{i \in I_S} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))$  and  $\widehat{d}_S = \frac{n}{|\Gamma|} \sum_{k \in L} \widehat{\zeta}_k \left(1 + \frac{\varepsilon}{50}\right)^k$ . Moreover,  $d_S = d_{SL} + d_{SS}$ , where

$$d_{SL} = \sum_{i \in I_S} \mathbf{d}_H(\mathbf{A}(i, *)|_{I_L}, \mathbf{B}(i, *)|_{I_L}) \text{ and } d_{SS} = \sum_{i \in I_S} \mathbf{d}_H(\mathbf{A}(i, *)|_{I_S}, \mathbf{B}(i, *)|_{I_S}).$$

Also, as  $\mathbf{A}$  and  $\mathbf{B}$  are symmetric matrices,  $d_{SL} = d_{LS} = \sum_{i \in I_L} \mathbf{d}_H(\mathbf{A}(i, *)|_{I_S}, \mathbf{B}(i, *)|_{I_S})$ .

First we show that

► **Claim A.5.**  $\mathbb{P}(d_{SS} \leq \frac{\varepsilon T}{1600}) \geq 1 - \frac{1}{\text{poly}(n)}$ .

Then we show that

► **Claim A.6.**  $\mathbb{P}\left(\left(1 - \frac{\varepsilon}{15}\right) d_{LS} - \frac{\varepsilon}{25} d_L \leq \widehat{d}_S \leq \left(1 + \frac{\varepsilon}{15}\right) d_{LS} + \frac{\varepsilon}{25} d_L\right) \geq 1 - \frac{1}{\text{poly}(n)}$ .

As  $d_S = d_{SS} + d_{LS}$ , the above two claims imply the Lemma, that is, the following holds with probability at least  $1 - \frac{1}{\text{poly}(n)}$ .

$$\left(1 - \frac{\varepsilon}{15}\right) d_S - \frac{\varepsilon T}{1600} - \frac{\varepsilon}{25} d_L \leq \widehat{d}_S \leq \left(1 + \frac{\varepsilon}{15}\right) d_S + \frac{\varepsilon}{25} d_L.$$

So, it remains to show Claims A.5 and A.6.

## 44:20 Distance Between Matrices Using Sublinear Projections on Hamming Cube

Proof of Claim A.5. Note that

$$d_{SS} = \sum_{i \in I_S} \mathbf{d}_H(\mathbf{A}(i, *)|_{I_S}, \mathbf{B}(i, *)|_{I_S}) = |\{(i, j) \in I_S \times I_S : \mathbf{A}(i, j) \neq \mathbf{B}(i, j)\}|,$$

where  $I_S$  denotes the set of indices in the  $Y_k$ 's with  $k \in S$  and  $S$  is the set of small buckets. So,  $|I_S| = \sum_{k \in S} |Y_k|$ . By the definition of  $S$ , for every  $k \in S$ ,  $|\widehat{Y}_k| < \tau = \frac{|\Gamma|}{n} \frac{\sqrt{\varepsilon T}}{50t}$ . By Lemma A.1

(i), we have  $|Y_k| \leq (1 + \frac{\varepsilon}{50}) \frac{n}{|\Gamma|} |\widehat{Y}_k| \leq \frac{\sqrt{\varepsilon T}}{40t}$  with probability at least  $1 - \frac{1}{\text{poly}(n)}$ . This implies that  $|I_S| = \sum_{k \in S} |Y_k| \leq \frac{\sqrt{\varepsilon T}}{40}$  with probability at least  $1 - \frac{1}{\text{poly}(n)}$ . Hence, by the definition of  $d_{SS}$ , we have the following with probability at least  $1 - \frac{1}{\text{poly}(n)}$ .

$$d_{SS} = \sum_{i \in I_S} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *)) \leq |I_S|^2 \leq \frac{\varepsilon T}{1600}. \quad \blacktriangleleft$$

Proof of Claim A.6. Note that  $d_{LS} = \sum_{i \in I_L} \mathbf{d}_H(\mathbf{A}(i, *)|_{I_S}, \mathbf{B}(i, *)|_{I_S})$ . Recall that  $d_{LS} = \sum_{k \in L} d_{LS}^k$ , where  $d_{LS}^k = \sum_{i \in Y_k} \mathbf{d}_H(\mathbf{A}(i, *)|_{I_S}, \mathbf{B}(i, *)|_{I_S})$ . Also, recall that  $\zeta_k = \frac{d_{LS}^k}{\sum_{i \in Y_k} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))}$ . So,  $d_{LS}^k = \zeta_k \sum_{i \in Y_k} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *))$ . Hence,

$$d_{LS} = \sum_{k \in L} \zeta_k \sum_{i \in Y_k} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *)). \quad (11)$$

By Claim A.4, the following holds with probability at least  $1 - \frac{1}{\text{poly}(n)}$ .

$$\forall k \in [t], \zeta_k \sum_{i \in Y_k} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *)) \leq \zeta_k |Y_k| \left(1 + \frac{\varepsilon}{50}\right)^k \leq \zeta_k \left(1 + \frac{\varepsilon}{50}\right) \sum_{i \in Y_k} \mathbf{d}_H(\mathbf{A}(i, *), \mathbf{B}(i, *)).$$

Taking sum over all  $k \in L$  and then applying Equation 11, the following holds with probability at least  $1 - \frac{1}{\text{poly}(n)}$ .

$$\mathbb{P}\left(d_{LS} \leq \sum_{k \in L} \zeta_k |Y_k| \left(1 + \frac{\varepsilon}{50}\right)^k \leq \left(1 + \frac{\varepsilon}{50}\right) d_{LS}\right) \geq 1 - \frac{1}{\text{poly}(n)}.$$

Recall that, for  $k \in [t]$  in the set  $L$  of large buckets,  $|\widehat{Y}_k| \geq \tau$ . Here  $\tau = \frac{|\Gamma|}{n} \frac{\sqrt{\varepsilon T}}{40t}$ . By Lemma A.1 (ii) and (i), for each  $k \in L$ ,  $\frac{n}{|\Gamma|} |\widehat{Y}_k|$  is a  $(1 \pm \frac{\varepsilon}{50})$ -approximation to  $|Y_k|$  with probability at least  $1 - \frac{1}{\text{poly}(n)}$ . So,

$$\mathbb{P}\left(\left(1 - \frac{\varepsilon}{50}\right) d_{LS} \leq \frac{n}{|\Gamma|} \sum_{k \in L} \zeta_k |\widehat{Y}_k| \left(1 + \frac{\varepsilon}{50}\right)^k \leq \left(1 + \frac{\varepsilon}{50}\right)^2 d_{LS}\right) \geq 1 - \frac{1}{\text{poly}(n)}. \quad (12)$$

Having the above equation, consider  $\widehat{d}_S = \frac{n}{|\Gamma|} \sum_{k \in L} \widehat{\zeta}_k |\widehat{Y}_k| \left(1 + \frac{\varepsilon}{50}\right)^k$  whose upper and lower bound is to be proved as stated in Claim A.6. Breaking the sum into two parts depending the values of  $\widehat{\zeta}_k$ 's, we have

$$\widehat{d}_S = \frac{n}{|\Gamma|} \sum_{k \in L: \widehat{\zeta}_k \geq \frac{\varepsilon}{50}} \widehat{\zeta}_k |\widehat{Y}_k| \left(1 + \frac{\varepsilon}{50}\right)^k + \frac{n}{|\Gamma|} \sum_{k \in L: \widehat{\zeta}_k < \frac{\varepsilon}{50}} \widehat{\zeta}_k |\widehat{Y}_k| \left(1 + \frac{\varepsilon}{50}\right)^k. \quad (13)$$

We prove the desired upper and lower bound on  $\widehat{d}_S$  separately by using the following observation about upper and lower bounds of the two terms in Equation 13.



► **Observation A.7.**

- (i)  $\frac{n}{|\Gamma|} \sum_{k \in L: \zeta_k \geq \frac{\varepsilon}{50}} \widehat{\zeta}_k \left| \widehat{Y}_k \right| \left(1 + \frac{\varepsilon}{50}\right)^k \leq \left(1 + \frac{\varepsilon}{15}\right) d_{LS}$  with probability  $1 - \frac{1}{\text{poly}(n)}$ .
- (ii)  $\frac{n}{|\Gamma|} \sum_{k \in L: \zeta_k < \frac{\varepsilon}{50}} \widehat{\zeta}_k \left| \widehat{Y}_k \right| \left(1 + \frac{\varepsilon}{25}\right)^k \leq \frac{\varepsilon}{35} d_L$  with probability at least  $1 - \frac{1}{\text{poly}(n)}$ .
- (iii)  $\frac{n}{|\Gamma|} \sum_{k \in L: \zeta_k \geq \frac{\varepsilon}{50}} \widehat{\zeta}_k \left| \widehat{Y}_k \right| \left(1 + \frac{\varepsilon}{50}\right)^k \geq \left(1 - \frac{\varepsilon}{15}\right) d_{LS} - \frac{\varepsilon}{25} d_L$  with probability  $1 - \frac{1}{\text{poly}(n)}$ .

**Proof.**

- (i) By Lemma A.1 (iii), for each  $k \in [t]$  with  $\zeta_k \geq \frac{\varepsilon}{50}$ ,  $\widehat{\zeta}_k$  is a  $(1 \pm \frac{\varepsilon}{40})$ -approximation to  $\zeta_k$  with probability at least  $1 - \frac{1}{\text{poly}(n)}$ . So, with probability at least  $1 - \frac{1}{\text{poly}(n)}$ , we can derive the following.

$$\begin{aligned} \frac{n}{|\Gamma|} \sum_{k \in L: \zeta_k \geq \frac{\varepsilon}{50}} \widehat{\zeta}_k \left| \widehat{Y}_k \right| \left(1 + \frac{\varepsilon}{40}\right)^k &\leq \left(1 + \frac{\varepsilon}{40}\right) \frac{n}{|\Gamma|} \sum_{k \in L: \zeta_k \geq \frac{\varepsilon}{50}} \zeta_k \left| \widehat{Y}_k \right| \left(1 + \frac{\varepsilon}{50}\right)^k \\ &\leq \left(1 + \frac{\varepsilon}{40}\right) \left(1 + \frac{\varepsilon}{50}\right)^2 d_{LS} \quad (\because \text{By Equation 12}) \\ &\leq \left(1 + \frac{\varepsilon}{15}\right) d_{LS}. \end{aligned}$$

- (ii) By Lemma A.1 (iv), for each  $k \in [t]$  with  $\zeta_k < \frac{\varepsilon}{50}$ ,  $\widehat{\zeta}_k$  is at most  $\frac{\varepsilon}{30}$  with probability at least  $1 - \frac{1}{\text{poly}(n)}$ . Hence, the following derivations hold with probability  $1 - \frac{1}{\text{poly}(n)}$ .

$$\begin{aligned} \frac{n}{|\Gamma|} \sum_{k \in L: \zeta_k \geq \frac{\varepsilon}{50}} \zeta_k \left| \widehat{Y}_k \right| \left(1 + \frac{\varepsilon}{50}\right)^k &\leq \frac{\varepsilon}{30} \cdot \frac{n}{|\Gamma|} \sum_{k \in L: \zeta_k < \frac{\varepsilon}{50}} \left| \widehat{Y}_k \right| \left(1 + \frac{\varepsilon}{50}\right)^k \\ &\leq \frac{\varepsilon}{30} \cdot \frac{n}{|\Gamma|} \sum_{k \in L} \left| \widehat{Y}_k \right| \left(1 + \frac{\varepsilon}{50}\right)^k \\ &\leq \frac{\varepsilon}{30} \left(1 + \frac{\varepsilon}{50}\right)^2 d_L \quad (\text{By Equation 10}) \\ &\leq \frac{\varepsilon}{25} d_L \end{aligned}$$

- (iii) Note that

$$\frac{n}{|\Gamma|} \sum_{k \in L: \zeta_k \geq \frac{\varepsilon}{50}} \widehat{\zeta}_k \left| \widehat{Y}_k \right| \left(1 + \frac{\varepsilon}{50}\right)^k \geq \frac{n}{|\Gamma|} \sum_{k \in L} \widehat{\zeta}_k \left| \widehat{Y}_k \right| \left(1 + \frac{\varepsilon}{50}\right)^k - \frac{n}{|\Gamma|} \sum_{k \in L: \zeta_k < \frac{\varepsilon}{50}} \widehat{\zeta}_k \left| \widehat{Y}_k \right| \left(1 + \frac{\varepsilon}{50}\right)^k.$$

By Lemma A.1 (iii), for each  $k \in [t]$  with  $\zeta_k \geq \frac{\varepsilon}{50}$ ,  $\widehat{\zeta}_k$  is a  $(1 \pm \frac{\varepsilon}{40})$ -approximation to  $\zeta_k$  with probability at least  $1 - \frac{1}{\text{poly}(n)}$ . Also, by Observation A.7 (iii),

$\frac{n}{|\Gamma|} \sum_{k \in L: \zeta_k \leq \frac{\varepsilon}{50}} \widehat{\zeta}_k \left| \widehat{Y}_k \right| \left(1 + \frac{\varepsilon}{50}\right)^k \leq \frac{\varepsilon}{25} d_L$  with probability at least  $1 - \frac{1}{\text{poly}(n)}$ . So, we can derive the following with probability at least  $1 - \frac{1}{\text{poly}(n)}$ .

$$\begin{aligned} \frac{n}{|\Gamma|} \sum_{k \in L: \zeta_k \geq \frac{\varepsilon}{50}} \widehat{\zeta}_k \left| \widehat{Y}_k \right| \left(1 + \frac{\varepsilon}{50}\right)^k &\geq \left(1 - \frac{\varepsilon}{40}\right) \frac{n}{|\Gamma|} \sum_{k \in L} \zeta_k \left| \widehat{Y}_k \right| \left(1 + \frac{\varepsilon}{50}\right)^k - \frac{\varepsilon}{25} d_L \\ &\geq \left(1 - \frac{\varepsilon}{40}\right)^2 d_{LS} - \frac{\varepsilon}{25} d_L \quad (\text{By Equation 12}) \\ &\geq \left(1 - \frac{\varepsilon}{15}\right) d_{LS} - \frac{\varepsilon}{25} d_L. \quad \blacktriangleleft \end{aligned}$$

Considering the expression for  $\widehat{d}_S$  in Equation 13 along with Observation A.7 (i) and (ii), we can derive the desired upper bound on  $\widehat{d}_S$  (as follows) that holds with probability at least  $1 - \frac{1}{\text{poly}(n)}$ .

$$\widehat{d}_S \leq \left(1 + \frac{\varepsilon}{15}\right) d_{LS} + \frac{\varepsilon}{25} d_L.$$

## 44:22 Distance Between Matrices Using Sublinear Projections on Hamming Cube

For the lower bound part of  $\widehat{d}_S$ , again consider the expression for  $\widehat{d}_S$  in Equation 13 along with Observation A.7 (iii). We have the following with probability at least  $1 - \frac{1}{\text{poly}(n)}$ .

$$\widehat{d}_S \geq \left(1 - \frac{\varepsilon}{15}\right) d_{LS} - \frac{\varepsilon}{25} d_L. \quad \triangleleft$$

### B Communication complexity

In two-party communication complexity there are two parties, Alice and Bob, that wish to compute a function  $\Pi : \{0, 1\}^N \times \{0, 1\}^N \rightarrow \{0, 1\}$ . Alice is given  $\mathbf{x} \in \{0, 1\}^N$  and Bob is given  $\mathbf{y} \in \{0, 1\}^N$ . Let  $x_i$  ( $y_i$ ) denote the  $i$ -th bit of  $\mathbf{x}$  ( $\mathbf{y}$ ). While the parties know the function  $\Pi$ , Alice does not know  $\mathbf{y}$ , and similarly, Bob does not know  $\mathbf{x}$ . Thus they communicate bits following a pre-decided protocol  $\mathcal{P}$  in order to compute  $\Pi(\mathbf{x}, \mathbf{y})$ . We say a randomized protocol  $\mathcal{P}$  computes  $\Pi$  if for all  $(\mathbf{x}, \mathbf{y}) \in \{0, 1\}^N \times \{0, 1\}^N$  we have  $\mathbb{P}[\mathcal{P}(\mathbf{x}, \mathbf{y}) = \Pi(\mathbf{x}, \mathbf{y})] \geq 2/3$ . The model provides the parties access to common random string of arbitrary length. The cost of the protocol  $\mathcal{P}$  is the maximum number of bits communicated, where maximum is over all inputs  $(\mathbf{x}, \mathbf{y}) \in \{0, 1\}^N \times \{0, 1\}^N$ . The communication complexity of the function is the cost of the most efficient protocol computing  $\Pi$ . For more details on communication complexity, see [25]. We now define DISJOINTNESS function on  $N$  bits and state its two-way randomized communication complexity.

► **Definition B.1.** Let  $N \in \mathbb{N}$ . The DISJOINTNESS $_N$  on  $N$  bits is a function DISJOINTNESS $_N : \{0, 1\}^N \times \{0, 1\}^N \rightarrow \{0, 1\}$  such that DISJOINTNESS $_N(\mathbf{x}, \mathbf{y}) = 0$  if there exists an  $i \in [N]$  such that  $x_i = y_i = 1$ , and 1, otherwise.

► **Proposition B.2.** [25] *The randomized communication complexity of DISJOINTNESS $_N$  is  $\Omega(N)$  even if it is promised that there exists at most one  $i \in [n]$  such that  $x_i = y_i = 1$ .*

### C Probability Results

► **Lemma C.1** (See [21]). *Let  $X = \sum_{i \in [n]} X_i$  where  $X_i, i \in [n]$ , are independent random variables,  $X_i \in [0, 1]$  and  $\mathbb{E}[X]$  is the expected value of  $X$ . Then for  $\epsilon \in (0, 1)$ ,  $\Pr[|X - \mathbb{E}[X]| > \epsilon \mathbb{E}[X]] \leq \exp\left(-\frac{\epsilon^2}{3} \mathbb{E}[X]\right)$ .*

► **Lemma C.2** (See [21]). *Let  $X = \sum_{i \in [n]} X_i$  where  $X_i, i \in [n]$ , are independent random variables,  $X_i \in [0, 1]$  and  $\mathbb{E}[X]$  is the expected value of  $X$ . Suppose  $\mu_L \leq \mathbb{E}[X] \leq \mu_H$ , then for  $0 < \epsilon < 1$ ,*

- (i)  $\Pr[X > (1 + \epsilon)\mu_H] \leq \exp\left(-\frac{\epsilon^2}{3} \mu_H\right)$ .
- (ii)  $\Pr[X < (1 - \epsilon)\mu_L] \leq \exp\left(-\frac{\epsilon^2}{2} \mu_L\right)$ .

# Decision Tree Heuristics Can Fail, Even in the Smoothed Setting

Guy Blanc ✉

Stanford University, CA, USA

Jane Lange ✉

Massachusetts Institute of Technology, Cambridge, MA, USA

Mingda Qiao ✉

Stanford University, CA, USA

Li-Yang Tan ✉

Stanford University, CA, USA

---

## Abstract

Greedy decision tree learning heuristics are mainstays of machine learning practice, but theoretical justification for their empirical success remains elusive. In fact, it has long been known that there are simple target functions for which they fail badly (Kearns and Mansour, STOC 1996).

Recent work of Brutzkus, Daniely, and Malach (COLT 2020) considered the smoothed analysis model as a possible avenue towards resolving this disconnect. Within the smoothed setting and for targets  $f$  that are  $k$ -juntas, they showed that these heuristics successfully learn  $f$  with depth- $k$  decision tree hypotheses. They conjectured that the same guarantee holds more generally for targets that are depth- $k$  decision trees.

We provide a counterexample to this conjecture: we construct targets that are depth- $k$  decision trees and show that even in the smoothed setting, these heuristics build trees of depth  $2^{\Omega(k)}$  before achieving high accuracy. We also show that the guarantees of Brutzkus et al. cannot extend to the agnostic setting: there are targets that are very close to  $k$ -juntas, for which these heuristics build trees of depth  $2^{\Omega(k)}$  before achieving high accuracy.

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

**Keywords and phrases** decision trees, learning theory, smoothed analysis

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.45

**Category** RANDOM

**Funding** *Jane Lange*: NSF Award #CCF-2006664

**Acknowledgements** We thank the anonymous reviewers, whose suggestions have helped improved this paper.

## 1 Introduction

Greedy decision tree learning heuristics are among the earliest and most basic algorithms in machine learning. Well-known examples include ID3 [28], its successor C4.5 [29], and CART [6], all of which continue to be widely employed in everyday ML applications. These simple heuristics build a decision tree for labeled dataset  $S$  in a greedy, top-down fashion. They first identify a “good” attribute to query as the root of the tree. This induces a partition of  $S$  into  $S_0$  and  $S_1$ , and the left and right subtrees are built recursively using  $S_0$  and  $S_1$  respectively.

In more detail, each heuristic is associated with an *impurity function*  $\mathcal{G} : [0, 1] \rightarrow [0, 1]$  that is concave, symmetric around  $\frac{1}{2}$ , and satisfies  $\mathcal{G}(0) = \mathcal{G}(1) = 0$  and  $\mathcal{G}(\frac{1}{2}) = 1$ . Examples include the binary entropy function  $\mathcal{G}(p) = H(p)$  that is used by ID3 and C4.5, and the Gini



© Guy Blanc, Jane Lange, Mingda Qiao, and Li-Yang Tan;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 45; pp. 45:1–45:16



Leibniz International Proceedings in Informatics

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

## 45:2 Decision Tree Heuristics Can Fail, Even in the Smoothed Setting

impurity function  $\mathcal{G}(p) = 4p(1-p)$  that is used by CART; Kearns and Mansour [21] proposed and analyzed the function  $\mathcal{G}(p) = 2\sqrt{p(1-p)}$ . For a target function  $f : \mathbb{R}^n \rightarrow \{0, 1\}$  and a distribution  $\mathcal{D}$  over  $\mathbb{R}^n$ , these heuristics build a decision tree hypothesis for  $f$  as follows:

1. *Split*: Query  $\mathbb{1}[x_i \geq \theta]$  as the root of the tree, where  $x_i$  and  $\theta$  are chosen to (approximately) maximize the *purity gain with respect to  $\mathcal{G}$* :

$$\mathcal{G}\text{-purity-gain}_{\mathcal{D}}(f, x_i) := \mathcal{G}(\mathbb{E}[f]) - (\Pr[x_i \geq \theta] \cdot \mathcal{G}(\mathbb{E}[f_{x_i \geq \theta}]) + \Pr[x_i < \theta] \cdot \mathcal{G}(\mathbb{E}[f_{x_i < \theta}])),$$

where the expectations and probabilities above are with respect to randomly drawn labeled examples  $(x, f(x))$  where  $x \sim \mathcal{D}$ , and  $f_{x_i \geq \theta}$  denotes the restriction of  $f$  to inputs satisfying  $x_i \geq \theta$  (and similarly for  $f_{x_i < \theta}$ ).

2. *Recurse*: Build the left and right subtrees by recursing on  $f_{x_i \geq \theta}$  and  $f_{x_i < \theta}$  respectively.
3. *Terminate*: The recursion terminates when the depth of the tree reaches a user-specified depth parameter. Each leaf  $\ell$  of the tree is labeled by  $\text{round}(\mathbb{E}[f_{\ell}])$ , where we associate  $\ell$  with the restriction corresponding to the root-to- $\ell$  path within the tree and  $\text{round}(p) := \mathbb{1}[p \geq \frac{1}{2}]$ .

Given the popularity and empirical success of these heuristics<sup>1</sup>, it is natural to seek theoretical guarantees on their performance:

*Let  $f : \mathbb{R}^n \rightarrow \{0, 1\}$  be a target function and  $\mathcal{D}$  be a distribution over  $\mathbb{R}^n$ . Can we obtain a high-accuracy hypothesis for  $f$  by growing a depth- $k'$  tree using these heuristics, where  $k'$  is not too much larger than  $k$ , the optimal decision tree depth for  $f$ ?* ( $\diamond$ )

### 1.1 Background and prior work

#### A simple and well-known impossibility result

Unfortunately, it has long been known [21, 20] that no such guarantee is possible even under favorable feature and distributional assumptions. Consider the setting of binary features (i.e.  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ ) and the uniform distribution  $\mathcal{U}$  over  $\{0, 1\}^n$ , and suppose  $f$  is the parity of two unknown features  $x_i \oplus x_j$  for  $i, j \in [n]$ . It can be easily verified that for all impurity functions  $\mathcal{G}$ , all features have the same purity gain:  $\mathcal{G}\text{-purity-gain}_{\mathcal{U}}(f, x_{\ell}) = 0$  for all  $\ell \in [n]$ , regardless of whether  $\ell \in \{i, j\}$ . Therefore, these heuristics may build a tree of depth  $\Omega(n)$ , querying irrelevant variables  $x_{\ell}$  where  $\ell \notin \{i, j\}$ , before achieving any nontrivial accuracy. This is therefore an example where the target  $f$  is computable by a decision tree of depth  $k = 2$ , and yet these heuristics may build a tree of depth  $k' = \Omega(n)$  before achieving any nontrivial accuracy.

#### Smoothed analysis

In light of such impossibility results, a line of work has focused on establishing provable guarantees for restricted classes of target functions [13, 25, 7, 3, 2]; we give an overview of these results in Section 1.3.

---

<sup>1</sup> CART and C4.5 were named as two of the “Top 10 algorithms in data mining” by the International Conference on Data Mining (ICDM) community [33]; other algorithms on this list include  $k$ -means,  $k$ -nearest neighbors, Adaboost, and PageRank, all of whose theoretical properties are the subjects of intensive study. C4.5 has also been described as “probably the machine learning workhorse most widely used in practice to date” [32].

The focus of our work is instead on *smoothed analysis* as an alternative route towards evading these impossibility results, an approach that was recently considered by Brutzkus, Daniely, and Malach [8]. Smoothed analysis is by now a standard paradigm for going beyond worst-case analysis. Roughly speaking, positive results in this model show that “hard instances are pathological.” Smoothed analysis has been especially influential in accounting for the empirical effectiveness of algorithms widely used in practice, a notable example being the simplex algorithm for linear programming [31]. The idea of analyzing greedy decision tree learning heuristics through the lens of smoothed analysis is therefore very natural.

A *smoothed product distribution* over  $\{0, 1\}^n$ , a notion introduced by Kalai, Samrodnitsky, and Teng [19], is obtained by randomly and independently perturbing the bias of each marginal of a product distribution. For smoothed product distributions, Brutzkus et al. proved strong guarantees on the performance of greedy decision tree heuristics when run on targets that are *juntas*, functions that depend only on a small number of its features. For a given impurity function  $\mathcal{G}$ , let us write  $\mathcal{A}_{\mathcal{G}}$  to denote the corresponding decision tree learning heuristic.

► **Theorem 1** (Performance guarantee for targets that are  $k$ -juntas [8]). *For all impurity functions  $\mathcal{G}$  and for all target functions  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  that are  $k$ -juntas, if  $\mathcal{A}_{\mathcal{G}}$  is trained on examples drawn from a smoothed product distribution, it learns a decision tree hypothesis of depth  $k$  that achieves perfect accuracy.*

(Therefore Theorem 1 shows that the smoothed setting enables one to circumvent the impossibility result discussed above, which was based on targets that are 2-juntas.)

Every  $k$ -junta is computable by a depth- $k$  decision tree, but a depth- $k$  decision tree can depend on as many as  $2^k$  variables. Brutzkus et al. left as an open problem of their paper a conjecture that the guarantees of Theorem 1 hold more generally for targets that are depth- $k$  decision trees:

► **Conjecture 2** (Performance guarantee for targets that are depth- $k$  decision trees). *For all impurity functions  $\mathcal{G}$  and for all target functions  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  that are depth- $k$  decision trees, if  $\mathcal{A}_{\mathcal{G}}$  is trained on examples drawn from a smoothed product distribution, it learns a decision tree hypothesis of depth  $O(k)$  that achieves high accuracy.*

In other words, Conjecture 2 states that for all targets  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ , the sought-for guarantee ( $\diamond$ ) holds if the heuristics are trained on examples drawn from a smoothed product distribution.

## 1.2 This work: Lower bounds in the smoothed setting

Our main result is a counterexample to Conjecture 2. We construct targets that are depth- $k$  decision trees for which all greedy impurity-based heuristics, even in the smoothed setting, may grow a tree of depth  $2^{\Omega(k)}$  before achieving high accuracy. This lower bound is close to being maximally large since Theorem 1 implies an upper bound of  $O(2^k)$ . Our result is actually stronger than just a lower bound in the smoothed setting: our lower bound holds with respect to any product distribution that is balanced in the sense that its marginals are not too skewed.

► **Theorem 3** (Our main result: a counterexample to Conjecture 2; informal). *Conjecture 2 is false: For all  $k = k(n)$ , there are target functions  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  that are depth- $k$  decision trees such that for all impurity functions  $\mathcal{G}$ , if  $\mathcal{A}_{\mathcal{G}}$  is trained on examples drawn from any balanced product distribution, its decision tree hypothesis does not achieve high accuracy unless it has depth  $2^{\Omega(k)}$ .*

By building on our proof of Theorem 3, we also show that the guarantees of Brutzkus et al. for  $k$ -juntas cannot extend to the agnostic setting:

► **Theorem 4** (Theorem 1 does not extend to the agnostic setting; informal). *For all  $\varepsilon$  and  $k = k(n)$ , there are target functions  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  that are  $\varepsilon$ -close to a  $k$ -junta such that for all impurity functions  $\mathcal{G}$ , if  $\mathcal{A}_{\mathcal{G}}$  is trained on examples drawn from any balanced product distribution, its decision tree hypothesis does not achieve high accuracy unless it has depth  $\varepsilon \cdot 2^{\Omega(k)}$ .*

In particular, there are targets that are  $2^{-\Omega(k)}$ -close to  $k$ -juntas, for which these heuristics have to construct a decision tree hypothesis of depth  $2^{\Omega(k)}$  before achieving high accuracy. Taken together with the positive result of Brutzkus et al., Theorems 3 and 4 add to our understanding of the strength and limitations of greedy decision tree learning heuristics.

Our lower bounds are based on new generalizations of the *addressing function*. Since the addressing function is often a useful extremal example in a variety of settings, we are hopeful that these generalizations and our analysis of them will see further utility beyond the applications of this paper.

### 1.3 Related Work

As mentioned above, there has been a substantial line of work on establishing provable guarantees for greedy decision tree heuristics when run in restricted classes of target functions. Fiat and Pechyony [13] considered the class of read-once DNF formulas and halfspaces; the Ph.D. thesis of Lee [25] considered the class of monotone functions; Brutzkus, Daniely, and Malach [7] considered conjunctions and read-once DNF formulas; recent works of [3, 2] build on the work of Lee and further studied monotone target functions. (All these works focus on the case of binary features and product distributions over examples.)

Kearns and Mansour [21], in one of the first papers to study these heuristics from a theoretical perspective, showed that they can be viewed as boosting algorithms, with internal nodes of the decision tree hypothesis playing the role of weak learners. Their subsequent work with Dietterich [11] provide experimental results that complement the theoretical results of [21]; see also the survey of Kearns [20].

Finally, we mention that decision trees are one of the most intensively studied concept classes in learning theory. The literature on this problem is rich and vast (see e.g. [12, 30, 5, 15, 9, 24, 4, 16, 22, 26, 18, 27, 14, 23, 19, 19, 17, 10, 1]), studying it from a variety of perspectives and providing both positive and negative results. However, the algorithms developed in these works do not resemble the greedy heuristics used in practice, and indeed, most of them are not proper (in the sense of returning a hypothesis that is itself a decision tree).<sup>2</sup>

## 2 Preliminaries

Recall that an impurity function  $\mathcal{G} : [0, 1] \rightarrow [0, 1]$  is concave, symmetric with respect to  $\frac{1}{2}$ , and satisfies  $\mathcal{G}(0) = \mathcal{G}(1) = 0$  and  $\mathcal{G}(\frac{1}{2}) = 1$ . We further quantify the concavity and smoothness of  $\mathcal{G}$  as follows:

<sup>2</sup> Quoting [21], “it seems fair to say that despite their other successes, the models of computational learning theory have not yet provided significant insight into the apparent empirical success of programs like C4.5 and CART.”

► **Definition 5** (Impurity functions).  $\mathcal{G}$  is an  $(\alpha, L)$ -impurity function if  $\mathcal{G}$  is  $\alpha$ -strongly concave and  $L$ -smooth, i.e.,  $\mathcal{G}$  is twice-differentiable and  $\mathcal{G}''(x) \in [-L, -\alpha]$  for every  $x \in [0, 1]$ .

For a boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  and index  $i \in [n]$ , we write  $f_{x_i=0}$  and  $f_{x_i=1}$  to denote the restricted functions obtained by fixing the  $i$ -th input bit of  $f$  to either 0 or 1. Formally, each  $f_{x_i=b}$  is a function over  $\{0, 1\}^n$  defined as  $f_{x_i=b}(x) = f(x^{i \rightarrow b})$ , where  $x^{i \rightarrow b}$  denotes the string obtained by setting the  $i$ -th bit of  $x$  to  $b$ . More generally, a *restriction*  $\pi$  is a list of constraints of form “ $x_i = b$ ” in which every index  $i$  appears at most once. For restriction  $\pi = (x_{i_1} = b_1, x_{i_2} = b_2, \dots)$ , the restricted function  $f_\pi : \{0, 1\}^n \rightarrow \{0, 1\}$  is similarly defined as  $f_\pi(x) = f(x^{i_1 \rightarrow b_1, i_2 \rightarrow b_2, \dots})$ .

► **Definition 6** (Purity gain). Let  $\mathcal{D}$  be a distribution over  $\{0, 1\}^n$  and  $p_i = \Pr_{x \sim \mathcal{D}} [x_i = 1]$ . The  $\mathcal{G}$ -purity gain of querying variable  $x_i$  on boolean function  $f$  is defined as

$$\mathcal{G}\text{-purity-gain}_{\mathcal{D}}(f, x_i) := \mathcal{G} \left( \mathbb{E}_{x \sim \mathcal{D}} [f(x)] \right) - p_i \mathcal{G} \left( \mathbb{E}_{x \sim \mathcal{D}} [f_{x_i=1}(x)] \right) - (1-p_i) \mathcal{G} \left( \mathbb{E}_{x \sim \mathcal{D}} [f_{x_i=0}(x)] \right).$$

In a decision tree, each node  $v$  naturally corresponds to a restriction  $\pi_v$  formed by the variables queried by the ancestors of  $v$  (excluding  $v$  itself). We use  $f_v$  as a shorthand for  $f_{\pi_v}$ . We say that a decision tree learning algorithm is *impurity-based* if, in the tree returned by the algorithm, every internal node  $v$  queries a variable that maximizes the purity gain with respect to  $f_v$ .

► **Definition 7** (Impurity-based algorithms). A decision tree learning algorithm is  $\mathcal{G}$ -impurity-based if the following holds for every  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  and distribution  $\mathcal{D}$  over  $\{0, 1\}^n$ : When learning  $f$  on  $\mathcal{D}$ , the algorithm outputs a decision tree such that for every internal node  $v$ , the variable  $x_i$  that is queried at  $v$  satisfies  $\mathcal{G}\text{-purity-gain}_{\mathcal{D}}(f_v, x_i) \geq \mathcal{G}\text{-purity-gain}_{\mathcal{D}}(f_v, x_j)$  for every  $j \in [n]$ .

The above definition assumes that the algorithm exactly maximizes the  $\mathcal{G}$ -purity gain at every split, while in reality, the purity gains can only be estimated from a finite dataset. We therefore consider an idealized setting that grants the learning algorithm with infinitely many training examples, which, intuitively, strengthens our lower bounds. (Our lower bounds show that in order for an algorithm to recover a good tree – a high-accuracy hypothesis whose depth is close to that of the target – it would need to query a variable that has *exponentially smaller* purity gain than that of the variable with the largest purity gain. Hence, if purity gains are estimated using finitely many random samples as is done in reality, the strength of our lower bounds imply that with extremely high probability, impurity-based heuristics will fail to build a good tree; see Remark 15 for a detailed discussion.)

When a decision tree queries variable  $x_i$  on function  $f$ , it naturally induces two restricted functions  $f_{x_i=0}$  and  $f_{x_i=1}$ . The following lemma states that the purity gain of querying  $x_i$  is roughly the squared difference between the averages of the two functions, up to a factor that depends on the impurity function  $\mathcal{G}$  and the data distribution  $\mathcal{D}$ . We say that a product distribution over  $\{0, 1\}^n$  is  $\delta$ -balanced if the expectation of each of the  $n$  coordinates is in  $[\delta, 1 - \delta]$ .

► **Lemma 8.** For any  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ ,  $\delta$ -balanced product distribution  $\mathcal{D}$  over  $\{0, 1\}^n$  and  $(\alpha, L)$ -impurity function  $\mathcal{G}$ , it holds for  $\kappa = \max \left( \frac{2}{\alpha\delta(1-\delta)}, \frac{L}{8} \right)$  and every  $i \in [n]$  that

$$\frac{1}{\kappa} \leq \frac{\mathcal{G}\text{-purity-gain}_{\mathcal{D}}(f, x_i)}{\left[ \mathbb{E}_{x \sim \mathcal{D}} [f_{x_i=0}(x)] - \mathbb{E}_{x \sim \mathcal{D}} [f_{x_i=1}(x)] \right]^2} \leq \kappa.$$

## 45:6 Decision Tree Heuristics Can Fail, Even in the Smoothed Setting

**Proof of Lemma 8.** Let  $p_i = \Pr_{x \sim \mathcal{D}} [x_i = 1]$  and  $\mu_b = \mathbb{E}_{x \sim \mathcal{D}} [f_{x_i=b}(x)]$  respectively. Then, we have  $\mathbb{E}_{x \sim \mathcal{D}} [f(x)] = p_i \mu_1 + (1 - p_i) \mu_0$ , and the purity gain can be written as

$$\mathcal{G}\text{-purity-gain}_{\mathcal{D}}(f, x_i) = \mathcal{G}(p_i \mu_1 + (1 - p_i) \mu_0) - p_i \mathcal{G}(\mu_1) - (1 - p_i) \mathcal{G}(\mu_0).$$

Since  $\mathcal{G}$  is  $\alpha$ -strongly concave and  $L$ -smooth, the above is bounded between  $\frac{\alpha}{2} \cdot p_i(1 - p_i) \cdot (\mu_0 - \mu_1)^2$  and  $\frac{L}{2} \cdot p_i(1 - p_i) \cdot (\mu_0 - \mu_1)^2$ . Since  $\mathcal{D}$  is  $\delta$ -balanced, we have  $\delta(1 - \delta) \leq p_i(1 - p_i) \leq \frac{1}{4}$ . It follows that

$$\frac{\alpha}{2} \cdot \delta(1 - \delta) \leq \frac{\alpha}{2} \cdot p_i(1 - p_i) \leq \frac{\mathcal{G}\text{-purity-gain}_{\mathcal{D}}(f, x_i)}{(\mu_0 - \mu_1)^2} \leq \frac{L}{2} \cdot p_i(1 - p_i) \leq \frac{L}{8}.$$

Thus, the ratio is bounded between  $1/\kappa$  and  $\kappa$ .  $\blacktriangleleft$

Our lower bounds hold with respect to all  $\delta$ -balanced product distributions. We compare this to the definition of a *c-smoothened*  $\delta$ -balanced product distribution from [8].

**► Definition 9 (Smooth distributions).** A *c-smoothened*  $\delta$ -balanced product distribution is a random product distribution over  $\{0, 1\}^n$  where the marginal for the  $i^{\text{th}}$  bit is 1 with probability  $\hat{p}_i + \Delta_i$  for fixed  $\hat{p}_i \in (\delta + c, 1 - \delta - c)$  and  $\Delta_i$  drawn i.i.d. from  $\text{Uniform}([-c, c])$ .

Since our lower bounds hold against all  $\delta$ -balanced product distributions, it also holds against all *c-smoothened*  $\delta$ -balanced product distributions.

### 3 Proof overview and formal statements of our results

Our goal is to construct a target function that can be computed by a depth- $k$  decision tree, but on which impurity-based algorithms must build to depth  $2^{\Omega(k)}$  or have large error. To do so, we construct a decision tree target  $T$  where the variables with *largest* purity gain are at the *bottom* layer of  $T$  (adjacent to its leaves). Intuitively, impurity-based algorithms will build their decision tree hypothesis for  $T$  by querying all the variables in the bottom layer of  $T$  before querying any of the variables higher up in  $T$ . Our construction will be such that until the higher up variables are queried, it is impossible to approximate the target with any nontrivial error. Summarizing informally, we show that impurity-based algorithms build its decision tree hypothesis for our target by querying variables in exactly the “wrong order”.

The starting point of our construction is the well known *addressing function*. For  $k \in \mathbb{N}$ , the addressing function  $f : \{0, 1\}^{k+2^k} \rightarrow \{0, 1\}$  is defined as follows: Given “addressing bits”  $z \in \{0, 1\}^k$  and “memory bits”  $y \in \{0, 1\}^{2^k}$ , the output  $f(y, z)$  is the  $z^{\text{th}}$  bit of  $y$ , where “ $z^{\text{th}}$  bit” is computed by interpreting  $z$  as a base-2 integer. Note that the addressing function is computable by a decision tree of depth  $k + 1$  that first queries the  $k$  addressing bits, followed by the appropriate memory bit.

For our lower bound, we would like the variables with the highest purity gain to be the memory bits. However, for smoothed product distributions, the addressing bits might have higher purity gain than the memory bits, and impurity-based algorithms might succeed in learning the addressing function. We therefore modify the addressing function by making each addressing bit the parity of multiple new bits. We show that by making each addressing bit the parity of sufficiently many new bits, we can drive the purity gain of these new bits down to the point where the memory bits have the highest purity gain as desired – in fact, larger than the addressing bits by a multiplicative factor of  $e^{\Omega(k)}$ . (Making each addressing bit the parity of multiple new bits increases the depth of the target, so this introduces technical challenges we have to overcome in order to achieve the strongest parameters.)

Our main theorem is formally restated as follows.



► **Theorem 10** (Formal version of Theorem 3). *Fix  $L \geq \alpha > 0$  and  $\delta \in (0, \frac{1}{2}]$ . There are boolean functions  $f_1, f_2, \dots$  such that: (1)  $f_k$  is computable by a decision tree of depth  $O(k/\delta)$ ; (2) For every  $\delta$ -balanced product distribution  $\mathcal{D}$  over the domain of  $f_k$  and every  $(\alpha, L)$ -impurity function  $\mathcal{G}$ , any  $\mathcal{G}$ -impurity based decision tree heuristic, when learning  $f_k$  on  $\mathcal{D}$ , returns a tree that has either depth  $\geq 2^k$  or an  $\Omega(\delta)$  error.*

An extension of our construction and its analysis shows that the guarantees of Brutzkus et al. for targets that are  $k$ -juntas cannot extend to the agnostic setting. Roughly speaking, while our variant of the addressing function from Theorem 10 is far from all  $k$ -juntas, it can be made close to one by fixing most of the memory bits. We obtain our result by showing that our analysis continues to hold under such a restriction.

► **Theorem 11** (Formal version of Theorem 4). *Fix  $L \geq \alpha > 0$ ,  $\delta \in (0, \frac{1}{2}]$  and  $\varepsilon \in (0, 1]$ . There are boolean functions  $f_1, f_2, \dots$  such that for every  $\delta$ -balanced product distribution  $\mathcal{D}$  over the domain of  $f_k$ : (1)  $f_k$  is  $\varepsilon$ -close to an  $O(k/\delta)$ -junta with respect to  $\mathcal{D}$ ; (2) For every  $(\alpha, L)$ -impurity function  $\mathcal{G}$ , any  $\mathcal{G}$ -impurity based decision tree heuristic, when learning  $f_k$  on  $\mathcal{D}$ , returns a tree that has either a depth of  $\Omega(\varepsilon \cdot 2^k)$  or an  $\Omega(1)$  error.*

## 4 Warm-Up: A Weaker Lower Bound

We start by giving a simplified construction that proves a weaker version of Theorem 10, in which the  $O(k/\delta)$  depth in condition (1) is relaxed to  $O(k^2/\delta)$ . For integers  $c, k \geq 1$ , we define a boolean function  $f_{c,k} : \{0, 1\}^{ck^2+2^k} \rightarrow \{0, 1\}$  as follows. The input of  $f_{c,k}$  is viewed as two parts:  $ck^2$  addressing bits  $x_{i,j}$  indexed by  $i \in [k]$  and  $j \in [ck]$ , and  $2^k$  memory bits  $y_a$  indexed by  $a \in \{0, 1\}^k$ . The function value  $f_{c,k}(x, y)$  is defined by first computing  $z_i(x) = \bigoplus_{j=1}^{ck} x_{i,j}$  for every  $i \in [k]$ , and then assigning  $f_{c,k}(x, y) = y_{z(x)}$ .

In other words,  $f_{c,k}$  is a disjoint composition of the  $k$ -bit addressing function and the parity function over  $ck$  bits. Given addressing bits  $x$  and memory bits  $y$ , the function first computes a  $k$ -bit address by taking the XOR of the addressing bits in each group of size  $ck$ , and then retrieves the memory bit with the corresponding address. Clearly,  $f_{c,k}$  can be computed by a decision tree of depth  $ck^2 + 1$  that first queries all the  $ck^2$  addressing bits and then queries the relevant memory bit in the last layer.

### 4.1 Address is Almost Uniform

Drawing input  $(x, y)$  from a distribution  $\mathcal{D}$  naturally defines a distribution over  $\{0, 1\}^k$  of the  $k$ -bit address  $z(x) = (z_1(x), z_2(x), \dots, z_k(x))$ . The following lemma states that when  $\mathcal{D}$  is a  $\delta$ -balanced product distribution, the distribution of  $z(x)$  is almost uniform in the  $\ell_\infty$  sense. Furthermore, this almost uniformity holds even if one of the addressing bits  $x_{i,j}$  is fixed.

► **Lemma 12.** *Suppose that  $c \geq \frac{\ln 5}{\delta}$  and  $\mathcal{D}$  is a  $\delta$ -balanced product distribution over the domain of  $f_{c,k}$ . Then,*

$$\left| \Pr_{(x,y) \sim \mathcal{D}} [z(x) = a] - 2^{-k} \right| \leq 5^{-k}, \forall a \in \{0, 1\}^k.$$

Furthermore, for every  $i \in [k]$ ,  $j \in [ck]$  and  $b \in \{0, 1\}$ ,

$$\left| \Pr_{(x,y) \sim \mathcal{D}} [z(x) = a | x_{i,j} = b] - 2^{-k} \right| \leq 5^{-k}, \forall a \in \{0, 1\}^k.$$

The proof of Lemma 12 uses the following simple fact, which states that the XOR of independent biased random bits is exponentially close to an unbiased coin flip.

► **Lemma 13.** *Suppose that  $x_1, x_2, \dots, x_n$  are independent Bernoulli random variables, each with an expectation between  $\delta$  and  $1 - \delta$ . Then,  $|\Pr[x_1 \oplus x_2 \oplus \dots \oplus x_n = 1] - \frac{1}{2}| \leq \frac{1}{2}(1 - 2\delta)^n \leq \frac{1}{2} \exp(-2\delta n)$ .*

**Proof of Lemma 12.** Since  $z_i(x) = \bigoplus_{j=1}^{ck} x_{i,j}$  and  $\mathcal{D}$  is  $\delta$ -balanced, Lemma 13 gives

$$\left| \Pr_{(x,y) \sim \mathcal{D}} [z_i(x) = 1] - \frac{1}{2} \right| \leq \frac{1}{2} \exp(-2\delta ck) \leq \frac{1}{2} \cdot 5^{-k}.$$

Note that the bits of  $z(x)$  are independent, so  $\Pr_{(x,y) \sim \mathcal{D}} [z(x) = a]$  is given by

$$\prod_{i=1}^k \Pr_{(x,y) \sim \mathcal{D}} [z_i(x) = a_i] \leq \left( \frac{1}{2} + \frac{1}{2} \cdot 5^{-k} \right)^k = 2^{-k} \cdot (1 + 5^{-k})^k \leq 2^{-k} \cdot (1 + (2/5)^k) = 2^{-k} + 5^{-k},$$

where the third step applies  $(1 + x)^k \leq 1 + 2^k x$  for  $x \in [0, 1]$  and integers  $k \geq 1$ . Similarly,

$$\Pr_{(x,y) \sim \mathcal{D}} [z(x) = a] \geq \left( \frac{1}{2} - \frac{1}{2} \cdot 5^{-k} \right)^k \geq 2^{-k} \cdot (1 - 5^{-k})^k \geq 2^{-k} - 5^{-k},$$

where the last two steps apply  $(1 - x)^k \geq 1 - kx$  and  $k \cdot 2^{-k} \leq 1$ . This proves the first part.

The proof of the “furthermore” part is essentially the same, except that conditioning on  $x_{i,j} = b$ ,  $z_i(x)$  becomes the XOR of  $ck - 1$  independent bits and  $b$ . By Lemma 13, we have

$$\left| \Pr_{(x,y) \sim \mathcal{D}} [z_i(x) = 1 | x_{i,j} = b] - \frac{1}{2} \right| \leq \frac{1}{2} \exp(-2\delta(ck - 1)) \leq \frac{1}{2} \exp(-\delta ck) \leq \frac{1}{2} \cdot 5^{-k},$$

and the rest of the proof is the same. ◀

## 4.2 Memory Bits are Queried First

The following technical lemma states that the purity gain of  $f_{c,k}$  is maximized by a memory bit, regardless of the impurity function and the data distribution. Therefore, when an impurity-based algorithm (in the sense of Definition 7) learns  $f_{c,k}$ , the root of the decision tree will always query a memory bit. Furthermore, this property also holds for restrictions of  $f_{c,k}$  as long as the restriction only involves the memory bits.

► **Lemma 14.** *Fix  $L \geq \alpha > 0$  and  $\delta \in (0, \frac{1}{2}]$ . Let  $c_0 = \frac{\ln 5}{\delta}$  and  $k_0 = \frac{\ln(2\kappa)}{\ln(5/4)} + 1$ , where  $\kappa$  is chosen as in Lemma 8. The following holds for every function  $f_{c,k}$  with  $c \geq c_0$  and  $k \geq k_0$ : For any  $(\alpha, L)$ -impurity function  $\mathcal{G}$ ,  $\delta$ -balanced product distribution  $\mathcal{D}$  and restriction  $\pi$  of size  $< 2^k$  that only contains the memory bits of  $f_{c,k}$ , the purity gain  $\mathcal{G}$ -purity-gain $_{\mathcal{D}}((f_{c,k})_{\pi}, \cdot)$  is maximized by a memory bit.*

**Proof of Lemma 14.** Fix  $c \geq c_0$  and  $k \geq k_0$  and shorthand  $f$  for  $f_{c,k}$ . We will prove a stronger claim: with respect to  $f_{\pi}$ , every memory bit (that is not in  $\pi$ ) gives a much higher purity gain than every addressing bit does.

### Purity gain of the memory bits

Fix a memory bit  $y_a$  ( $a \in \{0, 1\}^k$ ) that does not appear in restriction  $\pi$ . Let  $\mu_b = \mathbb{E}_{(x,y) \sim \mathcal{D}} [f_{\pi, y_a=b}(x, y)]$  for  $b \in \{0, 1\}$ . By the law of total expectation,

$$\begin{aligned} \mu_b &= \Pr_{(x,y) \sim \mathcal{D}} [z(x) = a] \cdot \mathbb{E}_{(x,y) \sim \mathcal{D}} [f_{\pi, y_a=b}(x, y) | z(x) = a] \\ &\quad + \Pr_{(x,y) \sim \mathcal{D}} [z(x) \neq a] \cdot \mathbb{E}_{(x,y) \sim \mathcal{D}} [f_{\pi, y_a=b}(x, y) | z(x) \neq a] \\ &= \Pr_{(x,y) \sim \mathcal{D}} [z(x) = a] \cdot b + \Pr_{(x,y) \sim \mathcal{D}} [z(x) \neq a] \cdot \mathbb{E}_{(x,y) \sim \mathcal{D}} [f_{\pi}(x, y) | z(x) \neq a]. \end{aligned}$$

Here the second step holds since  $f_{\pi, y_a=b}(x, y)$  evaluates to  $b$  when the address  $z(x)$  equals  $a$ , and  $f_{\pi, y_a=b}$  agrees with  $f_\pi$  when  $z(x) \neq a$ . Since only the first term above depends on  $b$ , we have

$$|\mu_0 - \mu_1| = \Pr_{(x,y) \sim \mathcal{D}} [z(x) = a] \geq 2^{-k} - 5^{-k} \geq \frac{1}{2} \cdot 2^{-k},$$

where the second step follows from  $c \geq c_0$  and Lemma 12. Finally, by Lemma 8,  $\mathfrak{G}\text{-purity-gain}_{\mathcal{D}}(f_\pi, y_a) \geq \frac{1}{\kappa}(\mu_0 - \mu_1)^2 \geq \frac{1}{4\kappa} \cdot 2^{-2k}$ .

### Purity gain of the addressing bits

Similarly, we fix an addressing bit  $x_{i,j}$  and define the average  $\mu_b = \mathbb{E}_{(x,y) \sim \mathcal{D}} [f_{\pi, x_{i,j}=b}(x, y)]$ . Since  $\mathcal{D}$  is a product distribution,  $\mu_b$  is equal to the conditional expectation  $\mathbb{E}_{(x,y) \sim \mathcal{D}} [f_\pi(x, y) | x_{i,j} = b]$ . Then, by the law of total expectation, we can write  $\mu_b$  as

$$\begin{aligned} \mu_b &= \sum_{a \in \{0,1\}^k} \Pr_{(x,y) \sim \mathcal{D}} [z(x) = a | x_{i,j} = b] \cdot \mathbb{E}_{(x,y) \sim \mathcal{D}} [f_\pi(x, y) | z(x) = a, x_{i,j} = b] \\ &= \sum_{a \in \{0,1\}^k} \Pr_{(x,y) \sim \mathcal{D}} [z(x) = a | x_{i,j} = b] \cdot \mathbb{E}_{(x,y) \sim \mathcal{D}} [f_\pi(x, y) | z(x) = a]. \end{aligned}$$

Here the second step holds since  $f_\pi(x, y)$  and  $x_{i,j}$  are independent conditioning on the address  $z(x)$ ; in other words, once we know the value of  $z(x)$ , it doesn't matter how  $x$  is set in determining the output of  $f$ .

Let  $c_a$  denote  $\mathbb{E}_{(x,y) \sim \mathcal{D}} [f_\pi(x, y) | z(x) = a]$ , and let  $\mathcal{P}_b$  be the distribution of  $z(x)$  conditioning on  $x_{i,j} = b$ . Then,  $\mu_b$  is exactly given by  $\mathbb{E}_{a \sim \mathcal{P}_b} [c_a]$ . Since each  $c_a$  is in  $[0, 1]$ ,  $|\mu_0 - \mu_1|$  is upper bounded by the total variation distance between  $\mathcal{P}_0$  and  $\mathcal{P}_1$ :

$$\begin{aligned} |\mu_0 - \mu_1| &\leq \frac{1}{2} \sum_{a \in \{0,1\}^k} |\mathcal{P}_0(a) - \mathcal{P}_1(a)| \\ &\leq \frac{1}{2} \sum_{a \in \{0,1\}^k} (|\mathcal{P}_0(a) - 2^{-k}| + |\mathcal{P}_1(a) - 2^{-k}|) \\ &\leq \frac{1}{2} \cdot 2^k \cdot 2 \cdot 5^{-k} = (2/5)^k. \end{aligned} \tag{Lemma 12}$$

Finally, applying Lemma 8 shows that  $\mathfrak{G}\text{-purity-gain}_{\mathcal{D}}(f_\pi, x_{i,j}) \leq \kappa(\mu_0 - \mu_1)^2 \leq \kappa \cdot (2/5)^{2k}$ .

Recall that  $k \geq k_0 > \frac{\ln(2\kappa)}{\ln(5/4)}$ , so we have  $\kappa \cdot (2/5)^{2k} < \frac{1}{4\kappa} \cdot 2^{-2k}$ . Therefore, the purity gain of every memory bit outside the restriction is strictly larger than that of any addressing bit, and the lemma follows immediately.  $\blacktriangleleft$

► **Remark 15.** The proof above bounds the purity gain of each memory bit and each addressing bit by  $\Omega((1/2)^{2k})$  and  $O((2/5)^{2k})$  respectively. For Lemma 14 to hold when the purity gains are estimated from a finite dataset, it suffices to argue that each estimate is accurate up to an  $O((2/5)^{2k})$  additive error. By a standard concentration argument, to estimate the purity gains for all restriction  $\pi$  of size  $\leq h$ ,  $2^{O(h+k)}$  training examples are sufficient. When applied later in the proof of Theorem 10, this finite-sample version of Lemma 14 would imply that impurity-based algorithms need to build a tree of depth  $h$  as soon as the sample size reaches  $2^{\Omega(h+k)}$ .

### 4.3 Proof of the Weaker Version

Now we are ready to prove the weaker version of Theorem 10. We will apply Lemma 14 to argue that the tree returned by an impurity-based algorithm never queries an addressing bit (unless all the  $2^k$  memory bits have been queried), and then show that every such decision tree must have an error of  $\Omega(\delta)$ .

**Proof of Theorem 10 (weaker version).** Fix integer  $c \geq \frac{\ln 5}{\delta}$  and consider the functions  $f_{c,1}, f_{c,2}, \dots$ . Since each  $f_{c,k}$  is represented by a decision tree of depth  $ck^2 + 1 = O(k^2/\delta)$ , it remains to show that impurity-based algorithms fail to learn  $f_{c,k}$ . Fix integer  $k \geq k_0$  (where  $k_0$  is chosen as in Lemma 14) and  $\delta$ -balanced product distribution  $\mathcal{D}$  over the domain of  $f_{c,k}$ . In the following, we use shorthand  $f$  for  $f_{c,k}$ .

#### Small trees never query addressing bits

Let  $T$  be the decision tree returned by a  $\mathcal{G}$ -impurity-based algorithm when learning  $f$  on  $\mathcal{D}$ . If  $T$  has depth  $> 2^k$ , we are done, so we assume that  $T$  has depth at most  $2^k$ . We claim that  $T$  never queries the addressing bits of  $f$ . Suppose otherwise, that an addressing bit is queried at node  $v$  in  $T$ , and no addressing bits are queried by the ancestors of  $v$ . Then, the restriction  $\pi_v$  associated with node  $v$  only contains the memory bits of  $f$ . Since  $T$  has depth  $\leq 2^k$ , the size of  $\pi_v$  is strictly less than  $2^k$ . Then, by Lemma 14, the  $\mathcal{G}$ -purity gain with respect to  $f_v$  is maximized by a memory bit. This contradicts the assumption that the algorithm is  $\mathcal{G}$ -impurity-based.

#### Trivial accuracy if no addressing bits are queried

We have shown that  $T$  only queries the memory bits of  $f$ . We may further assume that  $T$  queries *all* the  $2^k$  memory bits before reaching any of its leaves, i.e.,  $T$  is a full binary tree of depth  $2^k$ . This assumption is without loss of generality because we can add dummy queries on the memory bits to the leaves of depth  $< 2^k$ , and label all the resulting leaves with the same bit. This change does not modify the function represented by  $T$ .

Assuming that  $T$  is full, every leaf  $\ell$  of  $T$  is labeled by  $2^k$  bits  $(c_a)_{a \in \{0,1\}^k}$ , meaning that each memory bit  $y_a$  is fixed to  $c_a$  on the root-to- $\ell$  path. The expectation of the restricted function  $f_\ell$  is then given by  $\mu_\ell := \mathbb{E}_{(x,y) \sim \mathcal{D}} [c_{z(x)}]$ . Clearly, the error of  $T$  is minimized when each leaf  $\ell$  is labeled with  $\mathbf{1} [\mu_\ell \geq \frac{1}{2}]$ , and the conditional error when reaching leaf  $\ell$  is  $\min(\mu_\ell, 1 - \mu_\ell)$ .

It remains to show that for a large fraction of leaves  $\ell$ ,  $\mu_\ell$  is bounded away from 0 and 1, so that  $\min(\mu_\ell, 1 - \mu_\ell)$  is large. When leaf  $\ell$  is randomly chosen according to distribution  $\mathcal{D}$ , the corresponding  $\mu_\ell$  is given by

$$\mu_\ell = \sum_{a \in \{0,1\}^k} \Pr_{(x,y) \sim \mathcal{D}} [z(x) = a] \cdot c_a, \quad (1)$$

where  $(c_a)_{a \in \{0,1\}^k}$  are  $2^k$  independent Bernoulli random variables with means in  $[\delta, 1 - \delta]$ .

By Lemma 12 and our choice of  $c \geq c_0$ ,  $\Pr_{(x,y) \sim \mathcal{D}} [z(x) = a] \leq 2 \cdot 2^{-k}$  holds for every  $a \in \{0,1\}^k$ . Thus, each term in (1) is bounded between 0 and  $2 \cdot 2^{-k}$ . Furthermore, since each  $c_a$  has expectation at least  $\delta$ ,  $\mathbb{E}[\mu_\ell] \geq \delta$ . Then, Hoeffding's inequality guarantees that over the random choice of  $(c_a)_{a \in \{0,1\}^k}$ ,  $\mu_\ell \geq \delta/2$  holds with probability at least  $1 - \exp\left(-\frac{2 \cdot (\delta/2)^2}{2^k \cdot (2 \cdot 2^{-k})^2}\right) = 1 - \exp(-2^k \delta^2/8)$ , which is lower bounded by  $2/3$  for all sufficiently large  $k$ . By a symmetric argument,  $\mu_\ell \leq 1 - \delta/2$  also holds with probability  $\geq 2/3$ . Therefore,

with probability  $\geq 1/3$  over the choice of leaf  $\ell$ ,  $\mu_\ell \in [\delta/2, 1 - \delta/2]$  holds and thus the conditional error on leaf  $\ell$  is at least  $\delta/2$ . This shows that the error of  $T$  over distribution  $\mathcal{D}$  is lower bounded by  $\delta/6$ , which completes the proof.  $\blacktriangleleft$

## 5 Proof of Theorem 10

When proving the weaker version of Theorem 10, each hard instance  $f_{c,k}$  has  $\Theta(k^2)$  addressing bits grouped into  $k$  disjoint subsets, and the  $k$ -bit address is defined by the XOR of bits in each subset. We will prove Theorem 10 using a slightly different construction that computes address from  $k$  overlapping subsets of only  $O(k)$  addressing bits.

For integers  $c, k \geq 1$  and a list of  $k$  sets  $S = (S_1, S_2, \dots, S_k)$  where each  $S_i \subseteq [ck]$ , we define a boolean function  $f_{c,k,S} : \{0, 1\}^{ck+2^k} \rightarrow \{0, 1\}$  as follows. The input of  $f_{c,k,S}$  is again divided into two parts:  $ck$  addressing bits  $x_1, x_2, \dots, x_{ck}$  and  $2^k$  memory bits  $y_a$  indexed by a  $k$ -bit address  $a$ . The function value  $f(x, y)$  is computed by taking  $z_i(x) = \bigoplus_{j \in S_i} x_j$  and then  $f(x, y) = y_{z(x)}$ . Clearly,  $f_{c,k,S}$  can be computed by a decision tree of depth  $ck + 1$  that first queries all the  $ck$  addressing bits  $x_1, x_2, \dots, x_{ck}$ , and then queries the relevant memory bit  $y_{z(x)}$ .

Let  $\Delta_{i \in I}^k S_i$  denote the  $k$ -ary symmetric difference of sets  $S_1$  through  $S_k$ , i.e., the set of elements that appear in an odd number of sets. We say that a list of sets  $S = (S_1, S_2, \dots, S_k)$  has *distance*  $d$ , if any non-empty collection of sets has a symmetric difference of size at least  $d$ , i.e.,  $|\Delta_{i \in I}^k S_i| \geq d$  for every non-empty  $I \subseteq [k]$ . In the following, we prove analogs of Lemmas 12 and 14 for function  $f_{c,k,S}$  assuming that  $S$  has a large distance; Theorem 10 would then follow immediately.

**► Lemma 16.** *Suppose that  $\mathcal{D}$  is a  $\delta$ -balanced product distribution over the domain of  $f_{c,k,S}$  and  $S$  has distance  $d \geq \frac{\ln 5}{\delta} \cdot k$ . Then,*

$$\left| \Pr_{(x,y) \sim \mathcal{D}} [z(x) = a] - 2^{-k} \right| \leq 5^{-k}, \forall a \in \{0, 1\}^k.$$

Furthermore, for every  $i \in [ck]$  and  $b \in \{0, 1\}$ ,

$$\left| \Pr_{(x,y) \sim \mathcal{D}} [z(x) = a | x_i = b] - 2^{-k} \right| \leq 5^{-k}, \forall a \in \{0, 1\}^k.$$

We prove Lemma 16 by noting that the distribution of  $z(x)$  has exponentially small Fourier coefficients (except the degree-0 one) under the assumptions, and is thus close to the uniform distribution over  $\{0, 1\}^k$ . More concretely, our goal is to show that, for every  $I \subseteq [k]$  the quantity  $\bigoplus_{i \in I} z_i(x)$  is 1 with probability nearly exactly  $\frac{1}{2}$ . Afterwards, we will show this is sufficient to guarantee that the distribution of  $z(x)$  is close to the uniform distribution.

**Proof of Lemma 16.** Since  $z_i(x) = \bigoplus_{j \in S_i} x_j$ , we have  $\bigoplus_{i \in I} z_i(x) = \bigoplus_{j \in S_I} x_j$  for every  $I \subseteq [k]$ , where  $S_I = \Delta_{i \in I}^k S_i$  is the symmetric difference of the corresponding sets. Since  $S$  has distance  $d$ ,  $|S_I| \geq d$  for every non-empty  $I \subseteq [k]$  and thus  $\bigoplus_{i \in I} z_i(x)$  is the XOR of at least  $d$  independent bits. Note that  $1 - 2 \bigoplus_{i \in I} z_i(x) = \prod_{i \in I} (1 - 2z_i(x))$ . By Lemma 13 and  $d \geq \frac{\ln 5}{\delta} \cdot k$ ,

$$\left| \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[ \prod_{i \in I} (1 - 2z_i(x)) \right] \right| = 2 \cdot \left| \Pr_{(x,y) \sim \mathcal{D}} \left[ \bigoplus_{i \in I} z_i(x) = 1 \right] - \frac{1}{2} \right| \leq \exp(-2\delta d) \leq 5^{-k}. \quad (2)$$

## 45:12 Decision Tree Heuristics Can Fail, Even in the Smoothed Setting

Note that for  $b_1, b_2 \in \{0, 1\}$ , we have  $\mathbb{1}[b_1 = b_2] = \frac{(1-2b_1)(1-2b_2)+1}{2}$ . Therefore, for every  $a \in \{0, 1\}^k$ ,

$$\begin{aligned}
\left| \Pr_{(x,y) \sim \mathcal{D}} [z(x) = a] - 2^{-k} \right| &= \left| \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[ \prod_{i=1}^k \frac{(1-2a_i)(1-2z_i(x)) + 1}{2} \right] - 2^{-k} \right| \\
&= 2^{-k} \left| \sum_{I \subseteq [k]} \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[ \prod_{i \in I} (1-2a_i)(1-2z_i(x)) \right] - 1 \right| \\
&\hspace{15em} \text{(expansion of product and linearity)} \\
&= 2^{-k} \left| \sum_{I \subseteq [k]: I \neq \emptyset} \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[ \prod_{i \in I} (1-2a_i)(1-2z_i(x)) \right] \right| \\
&\hspace{15em} \text{(empty product equals 1)} \\
&\leq 2^{-k} \sum_{I \subseteq [k]: I \neq \emptyset} \left| \prod_{i \in I} (1-2a_i) \right| \cdot \left| \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[ \prod_{i \in I} (1-2z_i(x)) \right] \right| \\
&\hspace{15em} \text{(triangle inequality and linearity)} \\
&= 2^{-k} \sum_{I \subseteq [k]: I \neq \emptyset} \left| \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[ \prod_{i \in I} (1-2z_i(x)) \right] \right| \quad (|1-2a_i| = 1) \\
&\leq 2^{-k} \cdot (2^k - 1) \cdot 5^{-k} < 5^{-k}. \quad \text{(Inequality (2))}
\end{aligned}$$

The proof of the “furthermore” part is the same, except that after conditioning on  $x_i = b$ , each  $\bigoplus_{j \in I} z_j(x)$  is now the XOR of at least  $d - 1$  independent bits, and the remaining proof goes through.  $\blacktriangleleft$

We note that the proof of Lemma 14 depends on the definition of  $z(x)$  only through the application of Lemma 12. Thus, Lemma 16 directly implies the following analog of Lemma 14:

► **Lemma 17.** *Fix  $L \geq \alpha > 0$  and  $\delta \in (0, \frac{1}{2}]$ . Let  $c_0 = \frac{\ln 5}{\delta}$  and  $k_0 = \frac{\ln(2\kappa)}{\ln(5/4)} + 1$ , where  $\kappa$  is chosen as in Lemma 8. The following holds for every function  $f_{c,k,S}$  such that  $k \geq k_0$  and  $S$  has distance  $c_0 k$ : For any  $(\alpha, L)$ -impurity function  $\mathcal{G}$ ,  $\delta$ -balanced product distribution  $\mathcal{D}$  and restriction  $\pi$  of size  $< 2^k$  that only contains the memory bits of  $f_{c,k,S}$ , the purity gain  $\mathcal{G}$ -purity-gain $_{\mathcal{D}}((f_{c,k,S})_{\pi}, \cdot)$  is maximized by a memory bit.*

Finally, we prove Theorem 10 by showing the existence of a set family  $S$  with a good distance.

**Proof of Theorem 10.** Fix  $\delta \in (0, \frac{1}{2}]$ . The Gilbert–Varshamov bound for binary linear codes implies that for some  $c = \Theta(1/\delta)$ , there exists a binary linear code with rate  $\frac{1}{c}$  and relative distance  $\frac{\ln 5}{\delta c}$ . It follows that for every sufficiently large  $k$ , there exists  $S^{(k)} = (S_1^{(k)}, S_2^{(k)}, \dots, S_k^{(k)})$  such that each  $S_i^{(k)} \subseteq [ck]$  and  $S^{(k)}$  has distance  $\frac{\ln 5}{\delta} \cdot k$ . This can be done by using the  $i$ -th basis of the linear code as the indicator vector of subset  $S_i^{(k)}$  for each  $i \in [k]$ .

We prove Theorem 10 using functions  $f_{c,1,S^{(1)}}, f_{c,2,S^{(2)}}, \dots$ . Since each  $f_{c,k,S^{(k)}}$  can be represented by a decision tree of depth  $ck + 1 = O(k/\delta)$ , it remains to prove that impurity-based algorithms fail to learn  $f_{c,k,S^{(k)}}$ . Lemma 17 guarantees that the tree returned by such algorithms either has depth  $> 2^k$ , or never queries any addressing bits. In the latter case, by the same calculation as in the proof of the weaker version, the decision tree must have an  $\Omega(\delta)$  error on distribution  $\mathcal{D}$ .  $\blacktriangleleft$

## 6 Proof of Theorem 11

We prove Theorem 11 using the construction of  $f_{c,k,S}$  in Section 5, where  $S = (S_1, S_2, \dots, S_k)$  is a list of  $k$  subsets of  $[ck]$  and each  $S_i$  specifies how the  $i$ -th bit of the address,  $z_i(x)$ , is computed from the addressing bits  $x_1$  through  $x_{ck}$ . Note that  $f_{c,k,S}$  itself depends on  $\Omega(2^k)$  input bits and is thus not an  $O(k)$ -junta. Nevertheless, we will show that, after we fix most of the memory bits of  $f_{c,k,S}$ , the function is indeed close to a  $(ck)$ -junta with relevant inputs being the  $ck$  addressing bits. Then, as in the proof of Theorem 10, we will argue that impurity-based heuristics still query the (unfixed) memory bits before querying any of the addressing bits, resulting in a tree that is either exponentially deep or far from the target function.

**Proof of Theorem 11.** As in the proof of Theorem 10, we can find functions  $f_{c,1,S^{(1)}}, f_{c,2,S^{(2)}}, \dots$  for some  $c = \Theta(1/\delta)$  such that each  $S^{(k)}$  has distance  $\geq \frac{\ln 5}{\delta} \cdot k$ . We fix a sufficiently large integer  $k$  and shorthand  $f$  for  $f_{c,k,S^{(k)}}$  in the following.

Partition  $\{0, 1\}^k$  into three sets  $A^0$ ,  $A^1$  and  $A^{\text{free}}$  such that  $|A^0| = |A^1|$  and  $\varepsilon \cdot 2^{k-2} \leq |A^{\text{free}}| \leq \varepsilon \cdot 2^{k-1}$ . Consider the restriction  $\pi$  of function  $f$  such that the memory bit  $y_a$  is fixed to be 0 for every  $a \in A^0$  and fixed to be 1 for every  $a \in A^1$ ; the memory bits with addresses in  $A^{\text{free}}$  are left as “free” variables. We will prove the theorem using  $f_\pi$  as the  $k$ -th function in the family.

### $f_\pi$ is close to a junta

Consider the function  $g : \{0, 1\}^{ck+2^k} \rightarrow \{0, 1\}$  defined as  $g(x, y) = \mathbf{1}[z(x) \in A^1]$ , where  $z(x)$  denotes  $(z_1(x), z_2(x), \dots, z_k(x))$  and each  $z_i(x) = \bigoplus_{j \in S_i^{(k)}} x_j$ . Clearly,  $g(x, y)$  only depends on  $x \in \{0, 1\}^{ck}$  and is thus a  $(ck)$ -junta. Furthermore, for every input  $(x, y)$  such that  $z(x) \in A^0$  (resp.  $z(x) \in A^1$ ), both  $f_\pi$  and  $g$  evaluate to 0 (resp. 1). Thus,  $f_\pi$  and  $g$  may disagree only if  $z(x) \in A^{\text{free}}$ . It follows that for every  $\delta$ -balanced product distribution  $\mathcal{D}$ ,

$$\begin{aligned} \Pr_{(x,y) \sim \mathcal{D}} [f_\pi(x, y) \neq g(x, y)] &\leq \Pr_{(x,y) \sim \mathcal{D}} [z(x) \in A^{\text{free}}] \\ &\leq |A^{\text{free}}| \cdot (2^{-k} + 5^{-k}) && \text{(Lemma 16)} \\ &\leq \varepsilon \cdot 2^{k-1} \cdot (2^{-k} + 5^{-k}) < \varepsilon. && (|A^{\text{free}}| \leq \varepsilon \cdot 2^{k-1}) \end{aligned}$$

Therefore,  $f_\pi$  is  $\varepsilon$ -close to an  $O(k/\delta)$ -junta (namely,  $g$ ) with respect to distribution  $\mathcal{D}$ .

### Impurity-based algorithms fail to learn $f_\pi$

Let  $T$  be the decision tree returned by an  $\mathfrak{G}$ -impurity based algorithm when learning  $f_\pi$  on distribution  $\mathcal{D}$ . By Lemma 17,  $T$  must query all the free memory bits with addresses in  $A^{\text{free}}$  before querying any of the addressing bits. Thus, either  $T$  has depth  $> |A^{\text{free}}| = \Omega(\varepsilon \cdot 2^k)$ , or  $T$  only queries the free memory bits of  $f_\pi$ .

In the latter case, we may again assume without loss of generality that  $T$  queries all the free memory bits  $(y_a)_{a \in A^{\text{free}}}$  before reaching any of its leaves, i.e.,  $T$  is a full binary tree of depth  $|A^{\text{free}}|$ . Then, every leaf  $\ell$  naturally specifies  $2^k$  bits  $(c_a)_{a \in \{0,1\}^k}$  defined as

$$c_a = \begin{cases} 0, & a \in A^0, \\ 1, & a \in A^1, \\ b, & a \in A^{\text{free}}, y_a \text{ is fixed to } b \text{ on the root-to-}\ell \text{ path.} \end{cases}$$

## 45:14 Decision Tree Heuristics Can Fail, Even in the Smoothed Setting

Let  $\mu_\ell := \mathbb{E}_{(x,y) \sim \mathcal{D}} [c_{z(x)}]$ . Again, the minimum possible error conditioning on reaching leaf  $\ell$  is  $\min(\mu_\ell, 1 - \mu_\ell)$ , achieved by labeling  $\ell$  with  $\mathbb{1}[\mu_\ell \geq \frac{1}{2}]$ . On the other hand, we have

$$\begin{aligned} \mu_\ell &\geq \Pr_{(x,y) \sim \mathcal{D}} [z(x) \in A^1] \\ &\geq |A^1| \cdot (2^{-k} - 5^{-k}) && \text{(Lemma 16)} \\ &\geq \frac{2^k - |A^{\text{free}}|}{2} \cdot 2^{-(k+1)} && (2|A^1| + |A^{\text{free}}| = 2^k) \\ &\geq \frac{2^k - 2^{k-1}}{2} \cdot 2^{-(k+1)} = \frac{1}{8}, && (|A^{\text{free}}| \leq \varepsilon \cdot 2^{k-1} \leq 2^{k-1}) \end{aligned}$$

and a similar calculation shows  $\mu_\ell \leq \frac{7}{8}$ . We conclude that the error of the decision tree  $T$  over distribution  $\mathcal{D}$  is at least  $\frac{1}{8} = \Omega(1)$ .  $\blacktriangleleft$

## 7 Conclusion

We have constructed target functions for which greedy decision tree learning heuristics fail badly, even in the smoothed setting. Our lower bounds complement and strengthen the parity-of-two-features example discussed in the introduction, which showed that these heuristics fail badly in the non-smoothed setting.

It can be reasonably argued that real-world data sets do not resemble the target functions considered in this paper or the parity-of-two-features example. Perhaps the sought-for guarantee ( $\diamond$ ), while false for certain target functions even in the smoothed setting, is nonetheless true for broad and natural classes of targets? It would be interesting to reexamine, through the lens of smoothed analysis, provable guarantees for restricted classes of functions that have been established. For example, can the guarantees of [3, 2] for monotone target functions and product distributions be further strengthened in the smoothed setting? The target functions considered in this paper, as well as the parity-of-two-features example, are non-monotone.

---

## References

- 1 Guy Blanc, Neha Gupta, Jane Lange, and Li-Yang Tan. Universal guarantees for decision tree induction via a higher-order splitting criterion. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)*, 2020.
- 2 Guy Blanc, Jane Lange, and Li-Yang Tan. Provable guarantees for decision tree induction: the agnostic setting. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020. Available at <https://arxiv.org/abs/2006.00743>.
- 3 Guy Blanc, Jane Lange, and Li-Yang Tan. Top-down induction of decision trees: rigorous guarantees and inherent limitations. In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151, pages 1–44, 2020.
- 4 Avrim Blum, Merrick Furst, Jeffrey Jackson, Michael Kearns, Yishay Mansour, and Steven Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 253–262, 1994.
- 5 Avrim Blum. Rank- $r$  decision trees are a subclass of  $r$ -decision lists. *Inform. Process. Lett.*, 42(4):183–185, 1992. doi:10.1016/0020-0190(92)90237-P.
- 6 Leo Breiman, Jerome Friedman, Charles Stone, and Richard Olshen. *Classification and regression trees*. Wadsworth International Group, 1984.
- 7 Alon Brutzkus, Amit Daniely, and Eran Malach. On the Optimality of Trees Generated by ID3. *ArXiv*, abs/1907.05444, 2019.



- 8 Alon Brutzkus, Amit Daniely, and Eran Malach. ID3 learns juntas for smoothed product distributions. In *Proceedings of the 33rd Annual Conference on Learning Theory (COLT)*, pages 902–915, 2020.
- 9 Nader Bshouty. Exact learning via the monotone theory. In *Proceedings of 34th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 302–311, 1993.
- 10 Sitan Chen and Ankur Moitra. Beyond the low-degree algorithm: mixtures of subcubes and their applications. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*, pages 869–880, 2019.
- 11 Tom Dietterich, Michael Kearns, and Yishay Mansour. Applying the weak learning framework to understand and improve C4.5. In *Proceedings of the 13th International Conference on Machine Learning (ICML)*, pages 96–104, 1996.
- 12 Andrzej Ehrenfeucht and David Haussler. Learning decision trees from random examples. *Information and Computation*, 82(3):231–246, 1989.
- 13 Amos Fiat and Dmitry Pechyony. Decision trees: More theoretical justification for practical algorithms. In *Proceedings of the 15th International Conference on Algorithmic Learning Theory (ALT)*, pages 156–170, 2004.
- 14 Parikshit Gopalan, Adam Kalai, and Adam Klivans. Agnostically learning decision trees. In *Proceedings of the 40th ACM Symposium on Theory of Computing (STOC)*, pages 527–536, 2008.
- 15 Thomas Hancock. Learning  $k\mu$  decision trees on the uniform distribution. In *Proceedings of the 6th Annual Conference on Computational Learning Theory (COT)*, pages 352–360, 1993.
- 16 Thomas Hancock, Tao Jiang, Ming Li, and John Tromp. Lower bounds on learning decision lists and trees. *Information and Computation*, 126(2):114–122, 1996.
- 17 Elad Hazan, Adam Klivans, and Yang Yuan. Hyperparameter optimization: A spectral approach. *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.
- 18 Jeffrey C. Jackson and Rocco A. Servedio. On learning random dnf formulas under the uniform distribution. *Theory of Computing*, 2(8):147–172, 2006. doi:10.4086/toc.2006.v002a008.
- 19 Adam Kalai, Alex Samorodnitsky, and Shang-Hua Teng. Learning and smoothed analysis. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 395–404, 2009.
- 20 Michael Kearns. Boosting theory towards practice: recent developments in decision tree induction and the weak learning framework (invited talk). In *Proceedings of the 13th National Conference on Artificial intelligence (AAAI)*, pages 1337–1339, 1996.
- 21 Michael Kearns and Yishay Mansour. On the boosting ability of top-down decision tree learning algorithms. In *Proceedings of the 28th Annual Symposium on the Theory of Computing (STOC)*, pages 459–468, 1996.
- 22 Michael Kearns and Yishay Mansour. On the boosting ability of top-down decision tree learning algorithms. *Journal of Computer and System Sciences*, 58(1):109–128, 1999.
- 23 Adam Klivans and Rocco Servedio. Toward attribute efficient learning of decision lists and parities. *Journal of Machine Learning Research*, 7(Apr):587–602, 2006.
- 24 Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.
- 25 Homin Lee. *On the learnability of monotone functions*. PhD thesis, Columbia University, 2009.
- 26 Dinesh Mehta and Vijay Raghavan. Decision tree approximations of boolean functions. *Theoretical Computer Science*, 270(1-2):609–623, 2002.
- 27 Ryan O’Donnell and Rocco Servedio. Learning monotone decision trees in polynomial time. *SIAM Journal on Computing*, 37(3):827–844, 2007.
- 28 Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.
- 29 Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
- 30 Ronald Rivest. Learning decision lists. *Machine learning*, 2(3):229–246, 1987.

## 45:16 Decision Tree Heuristics Can Fail, Even in the Smoothed Setting

- 31 Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, 2004.
- 32 Ian Witten, Eibe Frank, Mark Hall, and Christopher Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.
- 33 Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008.

# On the Structure of Learnability Beyond P/Poly

Ninad Rajgopal ✉ 

University of Warwick, Coventry, UK

Rahul Santhanam ✉

University of Oxford, UK

---

## Abstract

---

Motivated by the goal of showing stronger structural results about the complexity of learning, we study the learnability of strong concept classes beyond P/poly, such as PSPACE/poly and EXP/poly. We show the following:

1. (Unconditional Lower Bounds for Learning) Building on [31], we prove unconditionally that BPE/poly cannot be weakly learned in polynomial time over the uniform distribution, even with membership and equivalence queries.
2. (Robustness of Learning) For the concept classes EXP/poly and PSPACE/poly, we show unconditionally that worst-case and average-case learning are equivalent, that PAC-learnability and learnability over the uniform distribution are equivalent, and that membership queries do not help in either case.
3. (Reducing Succinct Search to Decision for Learning) For the decision problems  $R_{K_t}$  and  $R_{K_S}$  capturing the complexity of learning EXP/poly and PSPACE/poly respectively, we show a *succinct search to decision* reduction: for each of these problems, the problem is in BPP iff there is a probabilistic polynomial-time algorithm computing circuits encoding proofs for positive instances of the problem. This is shown via a more general result giving succinct search to decision results for PSPACE, EXP and NEXP, which might be of independent interest.
4. (Implausibility of Oblivious Strongly Black-Box Reductions showing NP-hardness of learning NP/poly) We define a natural notion of hardness of learning with respect to oblivious strongly black-box reductions. We show that learning PSPACE/poly is PSPACE-hard with respect to oblivious strongly black-box reductions. On the other hand, if learning NP/poly is NP-hard with respect to oblivious strongly black-box reductions, the Polynomial Hierarchy collapses.

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

**Keywords and phrases** Hardness of Learning, Oracle Circuit Classes, Succinct Search, Black-Box Reductions

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.46

**Category** RANDOM

**Funding** This work was supported in part by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2014)/ERC Grant Agreement no. 615075.

*Ninad Rajgopal*: Supported in part by Tom Gur's UKRI Future Leaders Fellowship MR/S031545/1.<sup>1</sup>

**Acknowledgements** Ninad is grateful to Igor Carboni Oliveira for many inspiring discussions, one of which led to the results in Section 2.

## 1 Introduction

What is the complexity of learning polynomial-size circuits? Despite extensive research on this question, our knowledge is still fairly sparse. For weak concept classes such as decision trees [34, 32], DNFs [34, 27] or even constant-depth circuits with parity gates [12],

---

<sup>1</sup> Most of this work was done when Ninad Rajgopal was affiliated with the University of Oxford.



© Ninad Rajgopal and Rahul Santhanam;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 46; pp. 46:1–46:23



Leibniz International Proceedings in Informatics

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

reasonably efficient learning algorithms under the uniform distribution are known for various models of learning. For stronger concept classes, learning is believed to be hard, but the evidence for this is not as strong as one might hope. Cryptographic assumptions such as the existence of one-way functions are known to imply that learning polynomial-size circuits is hard [29, 18]. However, we still seem far from showing that PAC-learning polynomial-size circuits is NP-hard - indeed [5] give negative results for certain kinds of black-box reductions to learning.

In this paper, we adopt a fresh perspective of approaching the learnability question from above, i.e. via circuit classes which are more powerful than P/poly. We consider commonly held beliefs about the complexity of learning, and establish these beliefs unconditionally for strong concept classes such as PSPACE/poly and EXP/poly. Of course the very learnability of these concept classes has some unlikely implications, eg., that these classes are approximable by efficient Boolean circuits. The point is that this is still consistent with our complexity-theoretic understanding, and we would like to know what current techniques are capable of proving *unconditionally* about learning. Partly this is to understand the limitations of current techniques, and partly this is to understand what structural properties of the stronger concept classes enable us to show unconditional results about them.

We begin by outlining our main results and comparing them with previous work.

## 1.1 Unconditional Results for Hardness of Learning

Our first set of results deals with unconditional hardness of learning circuit classes. Most complexity theorists believe that learning polynomial-size circuits is unconditionally hard, but of course proving this is at least as hard as the P vs NP problem. We ask: what is the smallest concept class  $\mathcal{C}/\text{poly}^2$  for which we can *prove* learning to be hard? Clearly, if we can prove that  $\mathcal{C}/\text{poly}$  cannot be approximated by efficient circuits, i.e., there does not even *exist* a good hypothesis for all concepts in the class, then hardness of learning follows. This observation implies for example that learning MAEXP is hard, by using known circuit lower bounds for this class [11].

But can we show hardness of learning unconditionally for some concept class where it is consistent with our current understanding of complexity theory that a good hypothesis exists for every concept in the class? We give an affirmative answer by ruling out PAC-learning with membership and equivalence queries unconditionally for the class BPE/poly.

The notion of PAC-learning  $\mathcal{C}/\text{poly}$ , for a uniform class  $\mathcal{C}$  above P such as EXP or BPE, can have different interpretations. Standard definitions for PAC-learning (cf. [30]) consider the task of learning to be efficient if it is polynomial in the size of the target concept over  $n$  inputs (assume that the accuracy  $\varepsilon$  and confidence  $\delta$  are both  $1/\text{poly}(n)$ ) and the hypothesis class is P/poly.<sup>3</sup> The standard definition of PAC-learning in  $\text{poly}(n)$  time using P/poly as its hypothesis class naturally extends to the concept class  $\mathcal{C}/\text{poly}$  as the size of the target concept is *still* polynomial in the input size  $n$ . For the classes  $\mathcal{C}$  we consider, PAC-learnability of  $\mathcal{C}/\text{poly}$  in  $\text{poly}(n)$  time using polynomial-sized hypothesis circuits is still consistent with our current understanding of complexity theory (as we do not have any unconditional average-case lower bounds for  $\mathcal{C}$  against P/poly), and therefore worth studying.

<sup>2</sup> For any uniform complexity class  $\mathcal{C}$ , define the class  $\mathcal{C}/\text{poly}$  as the set of languages  $L$  for which there is a language  $\mathcal{C}$ -machine  $M$  and a family of strings  $\{a_n\}$ , where  $a_n \in \{0, 1\}^{\text{poly}(n)}$ , such that for every  $x \in \{0, 1\}^n$ ,  $x \in L \iff M$  accepts  $(x, a_n)$

<sup>3</sup> In general, the definition requires the hypothesis class  $H$  to be *polynomially evaluable*, which means that there exists an algorithm that on input any instance  $x \in \{0, 1\}^n$  and an encoding of the hypothesis  $h \in H_n$ , outputs the value  $h(x)$  in time polynomial in  $n$  and the size of the hypothesis encoding. It is well known that P/poly is polynomially evaluable.

We say that a class  $\mathcal{C}$  is  $(\varepsilon, \delta)$ -learnable using membership queries over distribution  $\mathcal{D}$  in polynomial time, if there exists a probabilistic polynomial time learning algorithm which given oracle access to any  $f \in \mathcal{C}$ , with probability at least  $1 - \delta$ , outputs a polynomial-sized hypothesis circuit that approximates  $f$  up to an error  $\varepsilon$  over the target distribution  $\mathcal{D}$ . This definition also extends to the case of  $(\varepsilon, \delta)$ -learning using random examples.

BPE/poly can equivalently be defined as the class of languages computable by polynomial-sized circuit families with oracle gates to some function in BPE, with the oracle query size restricted to  $O(n)$ . We prove the unconditional hardness of learning BPE/poly in polynomial time using membership queries even over the uniform distribution using P/poly as the hypothesis class. Hardness of exactly learning BPE/poly with membership and equivalence queries, even using randomized algorithms follows directly from this via [4].

► **Theorem 1.1.** *For every constant  $k \in \mathbb{N}$ , BPE/poly cannot be  $(1/2 - 1/n^k, 1/n)$ -learnt over the uniform distribution using membership queries by randomized learning algorithms running in polynomial time.*

To prove this, we adapt techniques used by [31, 36] to show that randomized PAC-learning algorithms imply circuit lower bounds. [38] show the existence of a PSPACE-Complete function  $f^*$  which is in  $\text{DSPACE}[n]$ , such that  $f^*$  is downward self-reducible and self-correctible (see Appendix A for definitions). Using the techniques of [31], along with the fact that  $f^*$  belongs to BPE, we see that PSPACE collapses to BPP. Using a padding argument and diagonalizing  $\text{DSPACE}[2^{O(n)}]$  against functions which can be approximated by polynomial-sized circuits, we obtain a contradiction to the fact that for every function in BPE/poly, the learner gives a hypothesis circuit which approximates it well.

## 1.2 Robustness for Hardness of Learning

We believe that polynomial-size circuits are hard to learn in a *robust* sense, i.e., that the precise details of the learning model do not matter. Hardness should hold irrespective of whether we consider PAC-learning or learning over the uniform distribution, worst-case learning or average-case learning over some samplable distribution on concepts, and whether or not the learning model is allowed to use membership queries. We do not know how to show that this robustness holds for P/poly, but we are able to show it unconditionally for EXP/poly and PSPACE/poly.

We now consider the class EXP/poly, which can be equivalently defined as the circuit class  $\text{P}^{\text{EXP}}/\text{poly}$  i.e. the class of languages that can be computed by a polynomial sized circuit family with EXP oracle gates.

Showing non-trivial derandomization of BPP, i.e.  $\text{EXP} \neq \text{BPP}$ , is one of the most fundamental questions in complexity theory.<sup>4</sup> We prove that the problem of non-trivial derandomization of BPP is equivalent to the hardness of learning EXP/poly efficiently in most standard models of PAC-learning. In addition, these results extend to not just showing that EXP/poly is hard to learn in the worst-case, but also on average with respect to polynomially samplable distributions over EXP/poly.<sup>5</sup> This also gives us an intriguing situation, where hardness of learning EXP/poly using random examples also implies the hardness of learning EXP/poly using membership queries.

<sup>4</sup> It is worth mentioning that [26] show that  $\text{EXP} \neq \text{BPP}$  is equivalent to the fact that BPP can be derandomized on average in deterministic sub-exponential time (over infinitely many input lengths).

<sup>5</sup> In particular, the results hold for polynomially samplable distribution families over EXP/poly, where for each  $n$ , there exists a distribution in the family over circuit encodings of  $n$ -variate functions in EXP/poly, implicitly defining a distribution on  $n$ -variate functions in EXP/poly (see Remark A.3 for more details.)

The following results are stated for hardness of strong learning. However, they also hold for the setting of weak learnability, by standard equivalences between weak learning and strong learning for PAC-learners [17].

► **Theorem 1.2** (Equivalences for hardness of learning EXP/poly). *The following statements are equivalent.*

1. **Non-trivial derandomization of BPP:**  $\text{EXP} \neq \text{BPP}$ .
2. **Hardness of PAC-learning EXP/poly in the worst-case using random examples:** *There exists  $c \geq 0$ , such that EXP/poly is not  $(1/n^c, 1/20n)$ -PAC-learnable in polynomial time using random examples.*
3. **Hardness of PAC-learning EXP/poly in the worst-case using membership queries:** *There exists  $c \geq 0$ , such that EXP/poly is not  $(1/n^c, 1/20n)$ -PAC-learnable in polynomial time using membership queries.*
4. **Hardness of PAC-learning EXP/poly on average using random examples:** *There exists  $c \geq 0$ , such that EXP/poly is not  $(1/n^c, 1/20n)$ -PAC-learnable in polynomial time on average using random examples, with respect to polynomially samplable distributions over EXP/poly.*
5. **Hardness of PAC-learning EXP/poly on average using membership queries:** *There exists  $c \geq 0$ , such that EXP/poly is not  $(1/n^c, 1/20n)$ -PAC-learnable in polynomial time on average using membership queries, with respect to polynomially samplable distributions over EXP/poly.*

A contrasting result to this is the equivalence between the existence of one-way functions (OWFs) and the hardness of learning P/poly in polynomial time on average with respect to polynomially samplable distributions over P/poly using random examples [25, 8]. Theorem 1.2 not only lends an analogous equivalence between a complexity theoretic assumption that BPP has a non-trivial derandomization and the hardness of learning EXP/poly in polynomial time on average using random examples, but also extends this equivalence to hardness of learning EXP/poly efficiently in the worst-case. Note that showing such an equivalence between the existence of OWFs and hardness of learning P/poly efficiently in the worst-case has been open for decades.<sup>6</sup>

Furthermore, our proof techniques also let us extend all these equivalences to the case where  $\mathcal{C} = \text{PSPACE}$ .

► **Corollary 1.3.** *The following statements are equivalent.*

1.  $\text{PSPACE} \neq \text{BPP}$ .
2. *There exists  $c \geq 0$ , such that PSPACE/poly is not  $(1/n^c, 1/20n)$ -PAC-learnable in polynomial time using random examples (also using membership queries).*
3. *There exists  $c \geq 0$ , such that PSPACE/poly is not  $(1/n^c, 1/20n)$ -PAC-learnable in polynomial time on average using random examples (also using membership queries) with respect to polynomially samplable distributions over PSPACE/poly.*

Essentially, the proof of showing conditional hardness of PAC-learning EXP/poly uses the fact that strongly learning EXP/poly using random examples over the *uniform distribution* implies that  $\text{EXP} = \text{BPP}$ . This also means that the hardest distribution to learn EXP/poly is over the uniform distribution. The same ideas hold for PAC-learning EXP/poly using membership queries too.

---

<sup>6</sup> In particular, we do not know if hardness of learning P/poly efficiently using random examples in the worst-case implies OWFs.

Our techniques used to show these equivalences are inspired from results on uniform derandomization by [26, 38], which were further used by [15, 31] to show circuit lower bounds based on the existence of learning algorithms. We use special properties of functions in EXP and PSPACE like downward self-reducibility and self-correctibility to show that learning these functions would imply a collapse for EXP and PSPACE to BPP.

### 1.3 Reducing Succinct Search to Decision for Learning

Recently, [12] established an important connection between *natural proofs* and *learning*. They showed that natural proofs of strong lower bounds against a circuit class  $\mathcal{C}/\text{poly}$  imply efficient learning algorithms for  $\mathcal{C}/\text{poly}$  over the uniform distribution with membership queries, as long as the class  $\mathcal{C}/\text{poly}$  satisfies some mild closure properties. One way to interpret their result is as an *approximate search to decision* reduction for learning. The decision version of learning polynomial-size circuits is the language MCSP consisting of truth tables of functions that have small circuits, i.e., for which a good hypothesis exists. The search version is to find a small circuit for a positive instance of MCSP. [12] show that if MCSP is polynomial-time decidable (which is implied by the existence of natural proofs against P/poly), then the search version of MCSP can be solved approximately, in the sense that we can efficiently compute a polynomially larger sized circuit that approximates the truth table well.

The language  $R_{Kt}$  (resp.  $R_{KS}$ ) of strings with high Kt complexity (resp. high KS complexity) plays an analogous role to MCSP in the theory of learning EXP/poly (resp. PSPACE/poly). We ask if search to decision reductions can be established for these languages as well. However, it is unclear a priori what it would mean to solve search efficiently for a problem that does not have polynomial-size proofs or witnesses. We introduce the notion of *succinct search*. To efficiently solve a search problem succinctly is to efficiently compute for any YES instance of the problem a circuit that encodes a possibly exponential-size proof for the instance. We use the PCP theorem for NEXP [6] and the Easy Witness Lemma [24] to show that for the classes PSPACE, EXP and NEXP, efficient decidability of the class is equivalent to efficiently solving succinct search for every language in the class. We then use results from [3] to argue that for  $R_{Kt}$  and  $R_{KS}$ , efficient solvability is equivalent to solving succinct search efficiently. Note that this connection is for succinctly solving the search problem *exactly* rather than just for approximate search as in [12].

► **Theorem 1.4** (Equivalence of Succinct Search and Decision for Learning EXP/poly and PSPACE/poly). *Let  $L$  be  $R_{Kt}$  or  $R_{KS}$ .  $L \in \text{BPP}$  iff for each polynomial-time verifier  $V$  for  $L$ , succinct search is efficiently solvable for  $L$  with respect to  $V$ .*

### 1.4 Barriers for Establishing NP-Hardness of Learning

We next look at questions pertaining to hardness of learning classes of the form  $\mathcal{C}/\text{poly}$ , where  $\mathcal{C} \subseteq \text{PH}$ . We only focus on the hardness of PAC-learning  $\mathcal{C}/\text{poly}$  with random examples. In this section, we consider the limitations of proving the NP-hardness of PAC-learning NP/poly, i.e. the class of polynomial size non-deterministic circuits, using random examples, via a black-box reduction from deciding SAT.

Informally, a black-box reduction from problem  $A$  to  $B$ , solves  $A$  given access to any oracle solving  $B$ . Black-box reductions have been ubiquitously used in complexity theory to prove conditional lower bounds. However, for many fundamental questions in complexity theory, there have been results showing why such reductions are limited in power. Various results have conditionally ruled out special-cases of black-box reductions for showing average-case hardness of NP [14, 10], existence of one-way functions [1, 5, 9] and the existence of hitting set generators [22], from hardness of SAT.

For the case of showing hardness of learnability, a  $B$ -adaptive black-box reduction  $R$  from some language  $L$  to PAC-learning a class  $\mathcal{C}$  using random examples is defined by two phases

- The first phase consists of  $B$  adaptive rounds of probabilistic polynomial time algorithms, each of which generates queries to the learner oracle. In more detail, each round uses the input  $z$  to the reduction, fresh randomness and the hypotheses returned by the  $\mathcal{C}$ -learner oracle in the previous rounds, and constructs joint distributions (that serve as example oracles for the learner). It then samples a set of independent labeled examples from each of these distributions as queries to the learner oracle.
- In the second phase, a probabilistic polynomial time algorithm takes all the hypotheses from the first phase and decides whether  $z \in L$ , with high probability.

[5] study the question of the existence of black-box Turing reductions from any language in NP to PAC-learning P/poly using random examples. They consider a strongly black-box reduction, where a reduction is strongly black-box if it runs correctly given any oracle for the learner, as well as the hypotheses output by the learner. For a special case of such a reduction, where the access to the learner and the hypothesis oracles is additionally non-adaptive, they show that such a reduction from SAT to PAC-learning P/poly using random examples collapses NP to CoAM (which implies a collapse of PH to the second level). Additionally, they show that if any language  $L$  reduces to PAC-learning P/poly using random examples via an  $O(1)$ -adaptive black-box reduction, then the hardness of  $L$  implies the existence of an auxiliary-input one-way function (which is a major breakthrough in cryptography).<sup>7</sup>

We define a natural special-case of such a reduction, called an *oblivious* strongly black-box reduction, where the obliviousness of a reduction implies that the queries made to the learner do not depend on the input  $z$  to the reduction, and try to understand its limitations for showing NP-hardness of PAC-learning NP/poly. At a first glance, ruling out oblivious reductions may seem very restrictive, since ideally, one would like to allow reductions whose queries to the learner can depend on the input to the reduction. However, we observe the proof of Corollary 1.3 which shows hardness of PAC-learning PSPACE/poly assuming PSPACE  $\neq$  BPP and reformulate it as an oblivious black-box reduction of the form defined above. In particular, for  $f^*$  being the PSPACE-Complete function given by [38] which is downward self-reducible and self-correctible, we observe that

► **Lemma 1.5.** *There exists an oblivious,  $n$ -adaptive, strongly black-box reduction from deciding  $f^*$  to PAC-learning PSPACE/poly using random examples over the uniform distribution.*

On the other hand, for the case of learning NP/poly using random examples, we show that oblivious strongly black-box reductions from SAT imply a collapse of the polynomial hierarchy. Our main result for the section is

► **Theorem 1.6 (Informal).** *If there exists an oblivious,  $\text{poly}(n)$ -adaptive, strongly black-box reduction from deciding SAT to learning NP/poly using random examples over polynomially samplable distributions, then PH collapses to the third level.*<sup>8</sup>

<sup>7</sup> They also show the impossibility of Karp reductions from SAT to PAC-learning P/poly using random examples, unless NP collapses to SZKA.

<sup>8</sup> We actually show a stronger result that the existence of such a reduction implies that  $\text{NP} \subseteq \text{CoAM}^{\text{poly}}$ , where  $\text{CoAM}^{\text{poly}}$  is the class of languages recognized by constant-round CoAM protocols with advice, where we require proper acceptance/rejection probabilities only when the advice is correct.



Theorem 1.6 implies that standard techniques used for worst-case to average-case reductions, pseudo-random generator constructions from uniform hardness assumptions and in particular, hardness of efficiently PAC-learning classes like PSPACE/poly, cannot be used to show the NP-hardness of PAC-learning NP/poly using random examples.

Theorem 1.6 compares to some previous results in the following way:

- It shows a conditional impossibility result by ruling out a restricted version of adaptive, strongly black-box reductions to learning P/poly using random examples, in contrast to [5], who only rule out fully non-adaptive, strongly black-box reductions, from a slightly weaker assumption ( $\text{NP} \not\subseteq \text{CoAM}$ ).
- Furthermore, the result by [22] which conditionally rules out a non-adaptive black box reduction from deciding SAT to breaking a Hitting Set Generator (HSG), in turn rules out fully non-adaptive, strongly black-box reductions from SAT to learning NP/poly using membership queries over the uniform distribution (by suitably changing the definition of the reduction to the learner).

Indeed, the ideas of [26] can be used to show that hardness of learning NP/poly using membership queries over the uniform distribution, implies the existence of a hitting set generator which hits sufficiently dense circuits. We strengthen this observation by not only extending the reduction to a restricted version of the adaptive case, but also by ruling out a weaker reduction to learning NP/poly with random examples.

- In a similar way, [20] conditionally rule out the existence of mildly adaptive (each query length up to  $n$ , where  $n$  is the length of the input instance, appears in very few levels of adaptivity), strongly black-box reductions from an EXP-Complete problem to learning NP/poly using membership queries (and in fact, learning EXP/poly).

Our result rules out the restricted cases of mildly adaptive, strongly black-box reductions which show the NP-hardness of learning NP/poly using random examples and hence, is a conceptual strengthening of [20], as we rule out a hardness result from a stronger assumption.

- On the other hand, Schapire [37] shows that a *non-uniform* hardness assumption like  $\text{NP/poly} \neq \text{P/poly}$  actually implies the hardness of PAC-learning NP/poly in polynomial time using random examples. They show that if NP/poly is learnable in polynomial time, then there exists an algorithm which takes any  $m$  labeled samples of a target  $f_n \in \text{NP/poly}$ , runs in time  $\text{poly}(m, n)$ , and with high probability, outputs a hypothesis of size  $\text{poly}(n)$  (independent of  $m$ ) that is consistent with all the labeled samples.

In particular, if  $m = 2^n$ , then for every  $f_n \in \text{NP/poly}$ , the algorithm outputs a polynomial-sized hypothesis circuit which computes it correctly on all inputs, thus contradicting the assumption  $\text{NP/poly} \neq \text{P/poly}$ . Note that the result uses that for any  $f_n$ , we get a polynomial-sized circuit that computes it, and in fact, the algorithm runs in  $\text{poly}(m, n) = 2^{O(n)}$  time and is not useful in terms of contradicting a uniform assumption.

It is worth noting that our result has no implications for showing the impossibility of adaptive, black-box NP-hardness reductions which imply the average-case hardness for NP [14, 10], existence of one-way functions [1, 5] or the existence of HSGs [20, 22].

**Overview of the techniques.** The proof of Theorem 1.6 builds on the Feigenbaum-Fortnow [14] protocol, which simulates a type of non-adaptive randomized reduction  $A$  from SAT to an NP problem  $\mathcal{Q}$ , by an AM protocol with polynomial-sized advice, and shows that  $\text{coNP} \subseteq \text{NP/poly}$ .<sup>9</sup>

<sup>9</sup> Their motivation (and [10]) was to rule out certain kinds of non-adaptive, worst-case to average-case black-box reductions for NP.

Suppose that on input  $x$ ,  $A$  makes  $q$  non-adaptive queries to  $\mathcal{Q}$ , sampled independently from certain distribution  $X$ . Very briefly, their AM protocol does the following. For  $K$  large enough, the verifier first generates  $K$  tuples of  $q$  non-adaptive queries by running  $A(x)$  independently  $K$  times. The verifier asks the prover to send a witness to each query which is a YES instance (which it can verify easily). This ensures that the prover cannot cheat if the query is a NO instance and the only way it can cheat is by claiming a YES instance to be a NO instance. Now, if the verifier has the proportion  $p$  of YES instances of  $\mathcal{Q}$  over the distribution  $X$ , then with high probability it knows that the number of YES instances among the  $Kq$  queries is concentrated around  $q \cdot (pK \pm O(\sqrt{K}))$ . The verifier answers with a reject if the number of YES instances is much lesser than  $pqK$ .

The honest prover answers each query correctly (with correct witnesses if necessary) and with high probability, the number of YES instances are close to the expectation. Hence, the verifier can pick any of  $K$  runs of  $A(x)$  using the prover's answers to its queries and the output will be correct with high probability. On the other hand, the cheating prover cannot cheat on more than  $O(q\sqrt{K})$  YES instances, with high probability. If we choose  $K \gg O(q\sqrt{K})$ , then on most of the  $K$  independent runs of  $A$ , all its queries are answered correctly and the reduction gives the correct answer. Thus, if we pick one of the runs at random and get  $A(x)$  by using the prover's answers to its queries, the verifier answers wrongly with low probability.

Consider an oblivious,  $B$ -adaptive, strongly black-box reduction  $R$  from  $L$  to an oracle which learns NP/poly. Suppose we are able to fix  $S_1, \dots, S_t$ , which are sets of labeled examples drawn independently from the joint distributions  $(X_1, f_1(X_1)), \dots, (X_t, f_t(X_t))$  where  $f_1, \dots, f_t \in \text{NP/poly}$ , as the queries made to the learner. Furthermore, let  $h_1, \dots, h_t$  be a set of fixed hypotheses circuits, some of which are used to generate  $S_1, \dots, S_t$ , such that each  $h_i$   $(1 - \varepsilon_0)$ -approximates  $f_i$  over  $X_i$ , for some  $\varepsilon_0 > 0$ . Because  $R$  is strongly black-box, each hypothesis is also accessed as an oracle and we see that  $L$  is decided by the algorithm  $M$  in the second phase, which has access to  $h_1, \dots, h_t$ . Now, the  $t$  oracles to  $M$  can be replaced by a single oracle  $\mathcal{O}$  which takes as input  $i \in [t]$  and  $y \in \{0, 1\}^n$ , and outputs  $h_i(y)$  ( $\mathcal{O}$  can be thought of as a table with  $t$  rows and  $2^n$  columns). We then adapt the techniques of [14] to design an AM protocol for  $L$  with polynomial sized advice, where the verifier expects that the prover answers according to  $\mathcal{O}$ .

The obliviousness of the reduction helps us in fixing the queries made by  $R$ , and implicitly, the corresponding hypotheses output by the oracle. In other words, this helps us fix the proportions of YES instances for each  $f_i$  non-uniformly, as the queries generated to the learner do not depend on the input to the reduction. We do this by inductively fixing the queries made by the reduction starting from the first round of adaptivity. Fixing a "good" polynomial-sized random string  $r^*$  used by the first phase non-uniformly (using Adleman's trick), we first get the queries to the learner made in the first round.

For any other round  $b \geq 2$ , assume that the queries to the learner up to round  $(b - 1)$  and the functionality of the hypothesis oracles used to generate them up to round  $(b - 2)$  are fixed. Using the fact that  $r^*$  is also fixed, we consider the set of all tuples of joint distributions that can be generated in the  $b^{\text{th}}$  round depending on the answers to the oracle queries of the hypotheses seen so far, and arbitrarily choose one of them. Note that, this implicitly fixes the functionality of the hypothesis oracles for the queries generated in round  $b - 1$ . We continue this process and fix all the queries made to the learner by all the rounds from the first phase.

The details of the results from this section have been delegated to Appendix B because of space constraints.

## 1.5 Further Discussion

**Connections to Karp-Lipton Style Theorems.** There is an analogy between our results on implications of learnability and Karp-Lipton style theorems. A Karp-Lipton style theorem for a uniform class  $\mathcal{C}$  gives an unlikely uniform implication of the assumption that  $\mathcal{C}$  has polynomial-size circuits. The original theorem of Karp and Lipton [28] shows such an implication for  $\mathcal{C} = \text{NP}$ : if  $\text{NP} \subseteq \text{P/poly}$ , then  $\Sigma_2 = \Pi_2$ . Karp-Lipton style theorems are now known for many other classes, including  $\mathcal{C} = \text{P}^{\#\text{P}}$  [35],  $\mathcal{C} = \text{PSPACE}$  [6] and  $\mathcal{C} = \text{EXP}$  [6]. In each of these cases,  $\mathcal{C} \subseteq \text{P/poly}$  implies  $\mathcal{C} = \text{MA}$ , applying techniques from the theory of interactive proofs [35, 6].

Similarly, in some of our results (i.e., Theorem 1.1, Theorem 1.2 and Corollary 1.3), we study implications of learnability for classes  $\mathcal{C}/\text{poly}$ , where  $\mathcal{C} = \text{BPE}, \text{EXP}$  or  $\text{PSPACE}$ . Since the learner is required to output a polynomial-size Boolean circuit, the learnability assumption already implies that  $\mathcal{C}$  is approximated by polynomial-size circuits, where the approximation is over the distribution on the examples. We are interested in establishing strong uniform implications of these assumptions, showing that the assumption is actually false in the case  $\mathcal{C} = \text{BPE}$ , and that the assumption implies a simulation of  $\mathcal{C}$  in  $\text{BPP}$  in the other cases. What enables us to show stronger implications than in corresponding Karp-Lipton style theorems is that the learner uniformly produces a good hypothesis by our assumption. However, the learner is assumed to have access to random examples or membership queries which cannot be efficiently simulated - this makes our simulation task more challenging, and we therefore exploit various structural properties of complete languages. We also need to deal with the issue of approximation, while standard Karp-Lipton style theorems have as their antecedent an exact simulation by efficient circuits.

**Open Questions.** One question which stems from our work is to explore the possibility of showing the hardness of PAC-learning  $\text{NP}/\text{poly}$  efficiently using random examples assuming that  $\text{NP} \neq \text{BPP}$ . A potential direction is to consider non black-box reductions for the  $\text{NP}$ -hardness of PAC-learning  $\text{NP}/\text{poly}$ . This viewpoint has lent itself some success in the case of worst-case to average-case reductions [19, 21, 22] and in our case, hardness of efficiently PAC-learning  $\text{EXP}/\text{poly}$ . Indeed, the reduction for  $\text{EXP}/\text{poly}$  only works if the learning algorithm runs in polynomial time, although the reduction still uses the learning algorithm as an oracle.<sup>10</sup> Moreover, [13] show a non black-box reduction from an approximate version of  $\text{MCSP}$  to learning  $\text{P}/\text{poly}$  by sub-exponential-sized circuits (and thus, learning  $\text{NP}/\text{poly}$ ). Note that, it is unclear if approximate  $\text{MCSP}$  is  $\text{NP}$ -hard and this reduction does not imply the  $\text{NP}$ -hardness of PAC-learning  $\text{NP}/\text{poly}$  efficiently.

Another important question is to explore an analogue of the  $\text{PH}$  collapse for learnability. In other words, does polynomial time learnability of  $\text{NP}/\text{poly}$  imply polynomial time learnability of  $\text{PH}/\text{poly}$ ? Note that, under a strong assumption of the existence of a (possibly adaptive and non-relativizable) worst-case to average-case reduction for  $\text{NP}$ , we can use the techniques in Lemma 3.2 along with the downward-self-reducibility of  $\text{SAT}$  to show such a collapse. On the other hand, [23] also shows that there exists an oracle  $O$  with respect to which  $\text{DistNP}^O \subseteq \text{AvgP}^O$  and  $\Sigma_2^O \not\subseteq \text{HeurSIZE}^O[2^{n^\alpha}]$ . Essentially, this result negates the existence of any relativizable reductions which show a statement analogous to the  $\text{PH}$  collapse for average-case algorithms i.e. if  $\text{NP}$  is easy on average, then  $\Sigma_2$  is easy on average too. In a similar spirit, can we prove that no relativizable technique can show that if  $\text{NP}/\text{poly}$  is learnable in polynomial time, then  $\Sigma_2/\text{poly}$  is learnable in polynomial time as well?

<sup>10</sup>For  $\text{EXP}$  to collapse to  $\text{PSPACE}$ , we need the  $\text{EXP}/\text{poly}$  learner to be efficient so that it outputs polynomial-sized hypothesis circuits for any language in  $\text{EXP}$ , and this further implies  $\text{EXP} \subseteq \text{P/poly}$ .

## 2 Unconditional Results for Hardness of Learning

Firstly, we show the hardness of learning BPE/poly over the uniform distribution using membership queries by randomized polynomial time algorithms. The proof of this result uses the following lemma from [31]. Preliminaries and definitions can be found in Appendix A.

► **Lemma 2.1.** *Let  $\mathcal{C}$  be any circuit class,  $s : \mathbb{N} \rightarrow \mathbb{N}$  be a size function and  $f^*$  be the PSPACE-Complete problem from Theorem A.8. There exists constant  $c \in \mathbb{N}$  such that if  $\mathcal{C}[s(n)]$  is learnable up to error  $n^{-c}$  in time  $T(n)$ , then at least one of the following holds :*

- $f^* \notin \mathcal{C}[s(n)]$ .
- $f^* \in \text{BPTIME}[\text{poly}(T(n))]$ .

We also need Lemma A.11 (Appendix A) which proves the existence of functions which cannot be approximated by  $n^{\log n}$ -sized circuits.

**Proof of Theorem 1.1.** Towards a contradiction, assume that there exists constants  $k, d \geq 1$  and a randomized learning algorithm  $A$  which learns BPE/poly in  $O(n^d)$  time over the uniform distribution using membership queries, up to error  $1/2 - 1/n^k$  and confidence  $1/n$ , for every large enough input length  $n$ . By non-uniformly fixing a good random string, we ensure that for every function  $g \in \text{BPE/poly}$ , there exists  $c$  such that  $A$  always outputs a hypothesis circuit of size  $O(n^c)$  which computes  $g$  on at least  $(1/2 + 1/n^k)$ -fraction of  $n$ -length inputs. Thus, for every function in BPE/poly, there exists a family of polynomial-sized circuits  $\{h_n\}_{n \in \mathbb{N}}$  which  $(1/2 + 1/n^k)$ -approximates it, where  $h_i$  is the hypothesis output by the learner on input length  $i$ .

We next show that the existence of such a learner implies the existence of a function in BPE which cannot be  $(1/2 + 1/n^k)$ -approximated by polynomial sized circuits. Consider the PSPACE-Complete function  $f^*$  from Theorem A.8 which is computable in time  $\text{DSPACE}[n]$ .  $f^*$  is in BPE/poly (since  $f^*$  can be computed in E) and we use the learning algorithm for BPE/poly in Lemma 2.1 to see that  $\text{PSPACE} \subseteq \text{BPP}$ . Using a padding argument we observe that  $\text{DSPACE}[2^{O(n)}] \subseteq \text{BPE}$ . From Lemma A.11, we see that there exists a function which cannot be  $(1/2 + 1/n^k)$ -computed by circuits of size  $n^{\log n}$ . We can easily construct a Turing Machine which lexicographically searches for the truth table of a function on  $n$  inputs which cannot be  $(1/2 + 1/n^k)$ -approximated by  $n^{\log n}$  sized circuits in  $2^{O(n)}$  space and answers according to the first one it finds. From this we have that  $\text{DSPACE}[2^{O(n)}]$ , and thus BPE/poly cannot be  $(1/2 + 1/n^k)$ -approximated by  $n^{\log n}$  sized circuits, which leads to a contradiction. ◀

► **Remark 2.2.** [36] show that if for each  $c$ , a circuit class  $\mathcal{C}[n^c]$  is  $(1/2 - 1/n^c, 1/n)$ -learnable using membership queries over the uniform distribution in  $2^n/n^{\omega(1)}$  time, then for each  $c$ , there exists  $L_c \in \text{BPE}$  such that  $L_c \notin \mathcal{C}[n^c]$  (Theorem 12). For any  $c$ , the idea of picking  $\mathcal{C}[n^c] = \text{SIZE}^{\text{BPE}}[n^c]$ , with linear-sized queries to BPE oracles and using the learning algorithm  $A$  which learns BPE/poly in their result to achieve a contradiction (as any function in BPE can be computed by constant sized  $\text{SIZE}^{\text{BPE}}$ -circuits with linear-sized oracle queries) does not work, as [36] crucially uses that  $\mathcal{C}[n^c]$  has to be a subset of  $\text{SIZE}[n^{c'}]$  for some  $c' = O(c)$ .

On the other hand, Theorem 4 in [15] shows that if  $\mathcal{C}$  is learnable using membership queries over the uniform distribution in polynomial time then  $\text{BPE} \not\subseteq \mathcal{C}[\text{poly}(n)]$ . Proving Theorem 1.1 by setting  $\mathcal{C}$  as BPE/poly again does not really work, as [15]'s result only holds true when  $\mathcal{C} = \text{P/poly}$ , as it depends on the collapse of EXP to P/poly.

We next consider hardness of learning E/poly deterministically over the uniform distribution using membership queries. E/poly can equivalently be defined as the class of languages which can be computed by polynomial-sized circuit families with oracle access to some function in E, with the constraint that the oracle queries are of size  $O(n)$ .

We first rule out deterministic *exact* learners for E/poly running in time  $O(2^n/n)$  in Angluin's model of learning [4], i.e. the learners have access to a membership oracle, as well as an equivalence oracle, where the learner presents a hypothesis circuit to the equivalence oracle, receives yes if the hypothesis exactly computes the target concept and receives a counter-example for the hypothesis, otherwise.

We also extend this result to rule out any deterministic learners for E/poly using membership queries, i.e. even learners which can output an approximate hypothesis. In particular, we can show that, for every constant  $\delta \in [0, 1/2 - 1/n)$ , E/poly is hard to learn up to error  $\delta$  over the uniform distribution using membership queries, even by deterministic learning algorithms which run in time  $2^n/n$ . These ideas can also be extended to show similar results for unconditional hardness of learning PSPACE/poly by deterministic polynomial time learners. The proofs follow from simple diagonalization-like arguments such as that used by [31] and can be found in the full version.

### 3 Robustness of Hardness of Learning

In this section, we establish the equivalences in Theorem 1.2 for hardness of learning EXP/poly. We first state the following results necessary for its proof.

► **Lemma 3.1.** *Let  $\text{EXP} = \text{BPP}$ . Then, for every  $c > 0$ , EXP/poly can be  $(1/n^c, 1/20n)$ -PAC-learned using random examples in time polynomial in  $n$ .*

**Proof Sketch.** The proof uses the collapse of EXP into BPP, firstly to observe that EXP/poly is in P/poly. Next, we construct an exponential time procedure which takes as inputs a size parameter  $s(n)$ , a set of examples of length  $n$  and their labels, and outputs a hypothesis circuit of size at most  $s(n)$  which is consistent with the examples, if there exists one, via an exhaustive search. From our assumption, this runs in probabilistic polynomial time. Using an argument based on Occam's razor [30], we obtain a polynomial time learner for P/poly. ◀

► **Lemma 3.2.** *Let  $\text{EXP} \neq \text{BPP}$ . Then, there exists  $c \geq 0$  such that EXP/poly is not  $(1/n^c, 1/20n)$ -learnable in the worst-case using random examples over the uniform distribution in time polynomial in  $n$ .*

**Proof.** Towards a contradiction assume that there exists a constant  $a > 0$  and an  $O(n^a)$ -time learner  $\mathcal{A}$  that  $(1/n^c, 1/20n)$ -learns EXP/poly using random examples over  $U_n$ , for every  $c \geq 0$ . We first show that the existence of the learner  $\mathcal{A}$  for EXP/poly implies that  $\text{EXP} \subseteq \text{P/poly}$ . Let  $g^*$  be an EXP-Complete problem which is self-correctible, whose existence is given by Theorem A.7, with  $c_1 \geq 0$  being the corresponding constant. Use  $\mathcal{A}$  to  $(1/n^{c_1}, 1/20n)$ -learn  $g^*$  using random examples over the uniform distribution. Let  $\mathcal{A}'$  be the algorithm which takes as input  $y \in \{0, 1\}^n$  in addition to the inputs of  $\mathcal{A}$  and runs the learner  $\mathcal{A}$ , following which it returns the evaluation of the hypothesis circuit output by  $\mathcal{A}$  on the input  $y$ . In other words, for every  $n \in \mathbb{N}$ , we have

$$\Pr_{\substack{w \in \{0,1\}^{r(n)} \\ x_1, \dots, x_m \sim U_n}} \left\{ \Pr_{y \sim U_n} \{ \mathcal{A}'(1^n, w, (x_1, g^*(x_1)), \dots, (x_m, g^*(x_m)), y) = g^*(y) \} \geq 1 - 1/n^{c_1} \right\} \\ \geq 1 - 1/20n$$

where both  $r(n)$  and  $m = m(n) = \text{poly}(n)$ .

## 46:12 On the Structure of Learnability Beyond P/Poly

By amplifying the correctness of  $\mathcal{A}'$  using standard techniques, we can then non-uniformly fix the random strings  $w, x_1, \dots, x_m$  and the values of  $g^*$  on each  $x_i$  to get a polynomial sized circuit  $C$ , which takes input  $y \in \{0, 1\}^n$  and outputs the answer of  $\mathcal{A}'$  on the advice string and  $y$ . Thus  $C_n$  agrees with  $g^*$  on at least  $(1 - 1/n^{c_1})$ -fraction of the inputs. Using  $C_n$  with the self-correctibility of  $g^*$  (and fixing another “good” random string non-uniformly in the resulting algorithm), we get a polynomial-sized circuit which computes  $g^*$  on every input and by the EXP-Completeness of  $g^*$ , we see that  $\text{EXP} \subseteq \text{P/poly}$ .

Since  $\text{EXP} \subseteq \text{P/poly}$ , we use Lemma A.9 to observe that  $f^*$  given by Theorem A.8 is now an EXP-Complete problem that is both downward self-reducible and self-correctible. Let  $c_2$  be the constant associated with the self-corrector for  $f^*$ . For any integer  $k$ , given a procedure  $B_k$  which computes  $f^*$  on every instance of size  $k$  with high probability, we use  $\mathcal{A}$  together with the downward self-reduction for  $f^*$ , followed by the self-corrector for  $f^*$  to obtain a procedure  $B_{k+1}$  that computes  $f^*$  on any input of size  $k + 1$ . We use this inductively, to compute  $f^*$  on  $n$  inputs in probabilistic polynomial time.

More precisely, consider the following algorithm  $B_n$  which computes  $f^*$  on a given input  $x$  and does the following. First, it starts with a procedure  $B_{k_0}$ , for a constant  $k_0$ , which can be computed easily using a look-up table. Assuming that we have the procedure  $B_k$  for some input length  $k \leq n$ , we show how to construct the procedure  $B_{k+1}$  inductively. We use the learner  $\mathcal{A}$  to learn the function  $f_{k+1}^*$  up to error  $1/(k+1)^{c_2}$ . For every input  $f^*(y)$  passed to  $\mathcal{A}$ , where  $y$  is a string randomly picked from  $\{0, 1\}^{k+1}$ , we use  $B_k$  with the downward self-reduction of  $f^*$  to compute  $f^*(y)$ .  $\mathcal{A}$  outputs a hypothesis  $h_{k+1}$  which computes  $f_{k+1}^*$  on at least a  $(1 - 1/(k+1)^c)$ -fraction of the inputs with high probability. We now use the self-corrector for  $f^*$  to obtain from  $h_{k+1}$  a procedure  $B_{k+1}$  which is correct on every input of size  $k + 1$  with probability  $1 - \gamma$  (by using standard error reduction arguments), for some  $\gamma > 0$  which we pick later. Repeating this process at most  $n$  times, we obtain  $B_n$  and output  $B_n(x)$ .

First, we show that  $B_n$  outputs  $f^*(x)$  with probability at least  $2/3$ . Let  $d(n)$  be the number of queries made by the DSR to the oracle  $f_{n-1}^*$  in computing  $f^*(x)$  on any input  $x$  of length  $n$ . The idea is that at each stage  $k$ , the procedure  $B_k$  fails only if at least one of  $m(n) \cdot d(n)$  queries answered by  $B_{k-1}$  is incorrect, with probability at most  $m(n)d(n)\gamma \leq 1/20n$  for  $\gamma = 1/20nm(n)d(n)$ , or if  $\mathcal{A}$  fails to output the right hypothesis, with probability at most  $1/20n$ . Thus, the total failure probability at each stage is at most  $1/10n$  and over the  $n$  stages, using the union bound, the total failure probability is at most  $1/10 + \gamma \leq 1/3$ .

We inductively observe that every stage  $B_k$  runs in time  $\text{poly}(k)$ . It is easily seen that  $B_{k_0}$  runs in constant time. Assume that  $B_{k-1}$  runs in  $\text{poly}(k-1)$  time. At stage  $k$ , the time taken to compute  $f^*$  on  $m(k)$  many inputs of length  $k$  is  $O(m(k) \cdot d(k) \cdot \text{poly}(k-1)) \leq \text{poly}(k)$ . After this,  $\mathcal{A}$  takes  $O(k^d)$  time to output  $h_k$  of size at most  $k^d$ , which is used by the  $\text{poly}(k)$ -time self-corrector to compute  $f^*$  on all inputs of size  $k$  with high probability. Thus,  $B_k$  runs in time  $\text{poly}(k) = \text{poly}(n)$ . Since there at most  $n$  stages, the total running time of  $B_n$  is  $\text{poly}(n)$ . This shows that  $f^* \in \text{BPP}$  and contradicts the original assumption. ◀

Using a very similar proof idea, we obtain an analogous statement to Lemma 3.2, but now for worst-case learning  $\text{EXP/poly}$  using membership queries.

► **Lemma 3.3.** *Suppose that  $\text{EXP/poly}$  is  $(1/n^c, 1/20n)$ -learnable in the worst case for every  $c \geq 0$  over the uniform distribution  $U_n$  using membership queries in time  $\text{poly}(n)$ . Then,  $\text{EXP} = \text{BPP}$ .*

Informally (see Appendix A for a formal definition), the task of an  $(\epsilon, \delta)$ -average-case learner for a class  $\mathcal{C}$  over the uniform distribution using random examples, is to  $(\epsilon, \delta)$ -learn an unknown target function which is generated according to a fixed distribution over

$\mathcal{C}$  (defined as an ensemble of distributions over appropriate representations for  $\mathcal{C}$ ). The ideas from Lemma 3.2 can also be extended to show similar implications for the setting of average-case learning EXP/poly using random examples (and membership queries too).

► **Lemma 3.4.** *Suppose that EXP/poly is  $(1/n^c, 1/20n)$ -learnable on average for every  $c \geq 0$  with respect to polynomially samplable distributions over EXP/poly and the uniform distribution  $U_n$  using random examples in time  $\text{poly}(n)$ . Then,  $\text{EXP} = \text{BPP}$ .*

► **Remark 3.5.** In Lemma 3.4, the samplable distributions we consider are over  $\text{SIZE}^{\text{EXP}}[n^k]$ -circuit encodings in  $R_n \subseteq \{0,1\}^{r(n)}$ , where  $r(n) = O(n^{2k+1})$  (see Remark A.3 for more details). In particular, for the distribution  $\mathcal{P}$  over  $\text{SIZE}^{\text{EXP}}[n^k]$  supported only on the function  $g^*$  (or  $f^*$ ) used in the proof of Lemma 3.2, the sampler  $S_{\mathcal{P}}$  takes  $1^{n^k}$  as input and outputs a  $\text{SIZE}^{\text{EXP}}[n^k]$ -circuit encoding of  $g^*$  (or  $f^*$ ) which is just an EXP-oracle gate on  $n$  inputs and this encoding is of size  $O(n^2)$ . The running time of this sampling algorithm is polynomial in the input size.

We use the same ideas as that of Lemma 3.2 to prove that even learning EXP/poly in polynomial time using random examples from the uniform distribution with respect to just these two distributions over EXP/poly, is enough to collapse EXP to BPP.

The formal details of the intermediate results can be found in the full version. We now prove the equivalences for efficiently learning EXP/poly.

**Proof of Theorem 1.2.** The following implications establish the desired equivalences.

$(b) \implies (a), (c) \implies (a)$ : The contrapositives of each of these implications follow from Lemma 3.1. In particular, PAC-learning EXP/poly with error at most  $1/n^c$  for any  $c > 0$  using random examples, implies PAC-learnability of EXP/poly using membership queries, where the queries are just made on the random examples given to the learner.

$(d) \implies (b), (e) \implies (c)$ : Follows from the definitions, since PAC-learning EXP/poly in the worst case in  $\text{poly}(n)$  time using random examples implies PAC-learnability for EXP/poly on average in  $\text{poly}(n)$  time using random examples, for any distribution over EXP/poly. A similar implication holds for learning with membership queries too.

$(a) \implies (b)$ : For any  $c > 0$ , suppose EXP/poly is  $(1/n^c, 1/20n)$  PAC-learnable in polynomial time using random examples over every arbitrary distribution. In particular, this means that EXP/poly can be  $(1/n^c, 1/20n)$ -learnt in polynomial time using random examples over the uniform distribution. The implication follows from the contrapositive of Lemma 3.2.

$(a) \implies (c)$ : Similar to the previous implication, we see that EXP/poly is  $(1/n^c, 1/20n)$ -learnable in polynomial time using membership queries over the uniform distribution. The implication holds from the contrapositive of Lemma 3.3.

$(a) \implies (d), (a) \implies (e)$ : The implications follow from Lemma 3.4 and its corresponding extension to learning on average with membership queries. ◀

The proof of Corollary 1.3 (equivalences for learning PSPACE/poly) follows from the same ideas as Theorem 1.2. In more detail, Lemma 3.1 extends easily as the procedure which searches for a polynomial-sized consistent hypothesis also runs in polynomial space. Lemmas 3.2, 3.3 and 3.4 can also be extended, by learning the downward-self-reducible and self-correctible PSPACE-Complete function  $f^*$  (from Theorem A.8) directly, and using it to compute  $f^*$  on every input.

## 4 Reducing Succinct Search to Decision

The key concepts in this section are verifiability and succinct search. We define verifiers first.

► **Definition 4.1.** *Given language  $L \subseteq \{0, 1\}^*$  and polynomial-time computable relation  $V(\cdot, \cdot)$ , we say that  $V$  is a verifier for  $L$  if for each  $x \in \{0, 1\}^*$ ,  $x \in L$  iff  $\exists y V(x, y)$ .*

*Given language  $L$ , a verifier  $V$  for  $L$ , and function  $f : \mathbb{N} \rightarrow \mathbb{N}$ , we say that  $L$  has  $f(n)$ -size proofs with respect to  $V$ , such that for each  $x \in \{0, 1\}^*$ ,  $x \in L$  implies  $\exists y, |y| \leq f(|x|) : V(x, y)$ . We say that  $L$  has  $f(n)$ -size proofs if there is a verifier  $V$  for  $L$  such that  $L$  has  $f(n)$ -size proofs with respect to  $V$ .*

*Given language  $L$ , a verifier  $V$  for  $L$  and a machine class  $\mathcal{D}$ , we say that  $L$  has  $\mathcal{D}$ -computable proofs with respect to  $V$  if there is a machine  $M \in \mathcal{D}$  such that for each  $x \in \{0, 1\}^*$ ,  $x \in L$  implies  $V(x, M(x))$ . We say that  $L$  has  $\mathcal{D}$ -computable proofs if there is a verifier  $V$  for  $L$  such that  $L$  has  $\mathcal{D}$ -computable proofs with respect to  $V$ .*

Note that NP is the class of languages with polynomial-sized proofs, NEXP is the class of languages with exponential-sized proofs, and for  $\mathcal{D} \in \{\text{EXP}, \text{PSPACE}\}$ ,  $\mathcal{D}$  is the class of languages with  $\mathcal{D}$ -computable proofs (where we abuse notation and use  $\mathcal{D}$  to refer both to a machine class and to the class of languages computable by such machines).

Next we define succinct search. We will assume w.l.o.g. that the proof size for any verifier is a power of 2 - this can be ensured by padding the proof if necessary.

► **Definition 4.2.** *Given language  $L$  and verifier  $V$  for  $L$ , we say that **succinct search is easy** for  $L$  with respect to  $V$  if there is a probabilistic polynomial-time machine  $N$  such that for each  $x \in L$ , there is a  $V$ -proof  $y$  such that with probability  $1 - o(1)$ ,  $\text{tt}(N(x)) = y$ , where for Boolean circuit  $C$ ,  $\text{tt}(C)$  denotes the truth table of the function computed by  $C$ .*

Thus succinct search is easy for  $L$  with respect to a verifier  $V$  if there is a probabilistic polynomial-time machine outputting compressed descriptions of  $V$ -proofs with high probability for any positive instance of  $L$ .

Using the downward self-reducibility of SAT, it is straightforward to see that  $\text{NP} \subseteq \text{BPP}$  iff for each  $L \in \text{NP}$  and for every verifier  $V$  such that  $L$  has poly-size proofs with respect to  $V$ , succinct search is easy for  $L$  with respect to  $V$ . We now show analogous results for PSPACE, EXP and NEXP. First we show for each of these classes that easiness of the class implies easiness of succinct search.

We need the Easy Witness Lemma of Impagliazzo, Kabanets and Wigderson [24].

► **Lemma 4.3** ([24]). *If  $\text{NEXP} \subseteq \text{P/poly}$ , then for each  $L \in \text{NEXP}$  and for each verifier  $V$  for  $L$  such that  $L$  has exponential-size proofs with respect to  $V$ , for each  $x \in L$ , there is a polynomial-size circuit  $C_x$  such that  $V(x, \text{tt}(C_x))$  holds.*

► **Lemma 4.4.** *The following implications hold:*

1. *Let  $\mathcal{D} \in \{\text{PSPACE}, \text{EXP}\}$ . If  $\mathcal{D} = \text{BPP}$ , then for each  $L \in \mathcal{D}$  and for each verifier  $V$  such that  $L$  has  $\mathcal{D}$ -computable proofs with respect to  $V$ , succinct search is easy for  $L$  with respect to  $V$ .*
2. *If  $\text{NEXP} = \text{BPP}$ , then for each  $L \in \text{NEXP}$  and for each verifier  $V$  such that  $L$  has exponential-size proofs with respect to  $V$ , succinct search is easy for  $L$  with respect to  $V$ .*

**Proof.** We establish the first item. Let  $\mathcal{D} \in \{\text{PSPACE}, \text{EXP}\}$ , and assume  $\mathcal{D} = \text{BPP}$ . Let  $L \in \mathcal{D}$  and  $V$  be a verifier for  $L$  such that  $L$  has  $\mathcal{D}$ -computable proofs with respect to  $V$ . We construct a probabilistic poly-time machine  $N$  such that for each input  $x \in L$ , there is a  $V$ -proof  $y$  such that with high probability  $\text{tt}(N(x)) = y$ . Let  $M$  be a  $\mathcal{D}$ -machine outputting  $V$ -proofs for positive instances of  $L$ .



Consider the language  $L' = \{\langle x, i \rangle \mid i^{\text{th}} \text{ bit of } M(x) \text{ is } 1\}$ . Since  $M$  is a  $\mathcal{D}$  machine, we have that  $L' \in \mathcal{D}$ . By assumption,  $\mathcal{D} = \text{BPP}$ , therefore there is a probabilistic poly-time machine  $N'$  deciding  $L'$ . Assume w.l.o.g. that  $N'$  has error at most  $2^{-|y|^2}$  on any input  $y$ . Given input  $x$ ,  $N$  operates as follows. It first computes a probabilistic poly-size circuit  $C'$  simulating  $N'$ . This can be done using the standard efficient conversion of efficient algorithms into small circuits. It then hardwires  $x$  into the first part of the input for  $C'$ , obtaining a circuit  $C'_x$ . It then fixes the random input of the circuit  $C'_x$  to a uniformly generated random string  $r$  to obtain a circuit  $D'_{x,r}$ , which it outputs.

Since the error of  $N'$  is smaller than  $2^{-|y|^2}$  on any input  $y$ , by a simple union bound, with probability  $1 - o(1)$  over the choice of the random string  $r$ ,  $D'_{x,r}$  correctly computes the  $i^{\text{th}}$  bit of  $M(x)$  for each  $i \in [m]$ . For  $x \in L$ ,  $V(x, M(x))$  holds, and therefore  $N$  efficiently solves succinct search for  $L$  with respect to  $V$ .

We establish the second item. Assume  $\text{NEXP} = \text{BPP}$  and let  $L \in \text{NEXP}$  and  $V$  be a verifier for  $L$  such that  $L$  has exponential-size proofs with respect to  $V$ . Since  $\text{NEXP} = \text{BPP}$ , we have that  $\text{NEXP} \subseteq \text{P/poly}$ . By Lemma 4.3, there is a polynomial  $p$  such that for each  $x \in L$ , there is a circuit  $C_x$  of size at most  $p(|x|)$  such that  $V(x, \text{tt}(C_x))$  holds.

Consider the language  $L' = \{\langle x, i \rangle \mid \text{There is a circuit } C \text{ of size } p(|x|) \text{ such that } V(x, \text{tt}(C)) \text{ is } 1, \text{ and the } i^{\text{th}} \text{ bit of the lexicographically first such circuit is } 1\}$ . Clearly  $L' \in \text{EXP}$ , just by enumerating circuits of size  $p(|x|)$  in lexicographic order and finding the first one encoding a  $V$ -proof for  $x$ , if one exists. Since  $\text{EXP} = \text{BPP}$ , there is a probabilistic poly-time machine  $N'$  deciding  $L'$  with error exponentially small. We construct a probabilistic poly-time machine  $N$  as follows: on input  $x$ ,  $N$  runs  $N'$  on  $\{\langle x, i \rangle\}$  for each  $i$  at most the description length of a circuit of size  $p(|x|)$ . It outputs the circuit  $C$  whose description has bit  $i$  set to 1 iff  $N'$  accepts on  $\{\langle x, i \rangle\}$ . Since  $N'$  has error exponentially small, we have that with error exponentially small,  $N$  outputs a circuit  $C$  encoding a  $V$ -proof of  $x$ , and therefore  $N$  efficiently solves succinct search for  $L$  with respect to  $V$ . ◀

For the reverse directions, we use the PCP characterization of  $\text{NEXP}$  [6, 16], where we only require polynomial upper bound on query complexity of the verifier.

► **Theorem 4.5** ([6, 16]). *Let  $L \in \text{NEXP}$ . There is a probabilistic poly-time oracle machine  $V'$  such that:*

1. *For each  $x \in L$ , there is  $y$  of length exponential in  $|x|$  such that  $V'(x)$  accepts with probability at least  $2/3$  when given oracle access to  $y$ .*
2. *For each  $x \notin L$  and for all  $y$ ,  $V'(x)$  accepts with probability at most  $1/3$  when given oracle access to  $y$ .*

We now show that easiness of succinct search implies easiness of decision for any  $L \in \text{NEXP}$ .

► **Lemma 4.6.** *Let  $L \in \text{NEXP}$  and  $V$  be a verifier such that  $L$  has exponential-size proofs with respect to  $V$ . If succinct search is easy for  $L$  with respect to  $V$ , then  $L \in \text{BPP}$ .*

**Proof.** Let  $L \in \text{NEXP}$ . We show that  $L \in \text{BPP}$ . By Theorem 4.5, there is a probabilistic poly-time oracle machine  $V'$  such that if  $x \in L$ , there is  $y$  of length exponential in  $|x|$  for which  $V'$  accepts with high probability on  $x$  when given oracle access to  $y$ , and if  $x \notin L$  rejects with high probability irrespective of the oracle.

Now consider a verifier  $V$  for  $L$  which given input  $x$  and proof  $y$ , accepts iff  $V'(x)$  accepts with oracle  $y$  on a majority of its computation paths. Since succinct search is easy for  $L$  with respect to  $V$ , there is a probabilistic poly-time machine  $N$  such that for input  $x \in L$ , there is a  $V$ -proof  $y$  for  $x$  such that with high probability  $\text{tt}(N(x)) = y$ . We define a probabilistic poly-time machine  $W$  that on input  $x$  simulates  $V'(x)$  as follows. It first runs  $N(x)$  to find a circuit  $C$ . It then runs  $V(x)$ , answering all oracle calls to  $y$  by simulating  $C$  on input corresponding to the bit of  $y$  that is queried. It accepts iff  $V(x)$  accepts.

If  $x \in L$ , by using the assumption that  $N$  solves succinct search,  $W(x)$  accepts with probability close to  $2/3$ . If  $x \notin L$ ,  $W(x)$  rejects with probability close to  $2/3$  since the circuit  $C$  output by  $N(x)$  corresponds to some purported  $V'$ -proof, and every such  $V'$ -proof is rejected with high probability by  $V$  when given oracle access to the proof. ◀

► **Theorem 4.7.** *Let  $\mathcal{D} \in \{\text{PSPACE}, \text{EXP}\}$ .  $\mathcal{D} = \text{BPP}$  iff for each  $L \in \mathcal{D}$  and for each verifier  $V$  for  $L$  such that  $L$  has  $\mathcal{D}$ -computable proofs with respect to  $V$ , succinct search is easy for  $L$  with respect to  $V$ .*

$\text{NEXP} = \text{BPP}$  iff for each  $L \in \text{NEXP}$  and for each verifier  $V$  such that  $L$  has exponential-size proofs with respect to  $V$ , succinct search is easy for  $L$  with respect to  $V$ .

**Proof.** The forward directions of both items follow from Lemma 4.4. The backward direction of the second item follows Lemma 4.6. The backward direction of the first item follows from Lemma 4.6 and the fact that for  $\mathcal{D} \in \{\text{PSPACE}, \text{EXP}\}$ , if  $L \in \mathcal{D}$  and  $V$  is a verifier for  $L$  such that  $L$  has  $\mathcal{D}$ -computable proofs with respect to  $V$ , then  $L \in \text{NEXP}$  and  $L$  has exponential-size proofs with respect to  $V$ . ◀

We now prove Theorem 1.4. Define  $R_{Kt}$  as the language consisting of strings  $x$  such that  $Kt(x) \geq |x|/2$ . Similarly,  $R_{KS}$  is the language consisting of strings  $x$  such that  $KS(x) \geq |x|/2$  [3] (see Appendix A.3 for formal definitions of  $Kt$  and  $KS$  complexity).

► **Theorem 4.8** (Theorem 1.4 stated formally).  $R_{Kt} \in \text{BPP}$  iff for each verifier  $V$  for  $R_{Kt}$  such that  $R_{Kt}$  has  $\text{EXP}$ -computable proofs with respect to  $V$ , succinct search is easy for  $R_{Kt}$  with respect to  $V$ .

$R_{KS} \in \text{BPP}$  iff for each verifier  $V$  for  $R_{KS}$  such that  $R_{KS}$  has  $\text{PSPACE}$ -computable proofs with respect to  $V$ , succinct search is easy for  $R_{KS}$  with respect to  $V$ .

**Proof.** The backward directions of both items follow from Lemma 4.6 and the facts that  $R_{Kt}$  and  $R_{KS}$  are in  $\text{NEXP}$ .

For the forward direction of the first item, we use the result shown in [3] that  $R_{Kt} \in \text{BPP}$  implies  $\text{EXP} = \text{BPP}$ . Combining this with the first item of Lemma 4.4 completes the proof.

For the forward direction of the second item, we use the theorem shown in [3] that  $R_{KS} \in \text{BPP}$  implies  $\text{PSPACE} = \text{BPP}$ . Combining this with the first item of Lemma 4.4 completes the proof. ◀

---

## References

- 1 Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on np-hardness. In Jon M. Kleinberg, editor, *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 701–710. ACM, 2006. doi:10.1145/1132516.1132614.
- 2 Eric Allender. When worlds collide: Derandomization, lower bounds, and kolmogorov complexity. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 1–15. Springer, 2001.
- 3 Eric Allender, Harry Buhrman, Michal Koucký, Dieter Van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM Journal on Computing*, 35(6):1467–1493, 2006.
- 4 Dana Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988.
- 5 Benny Applebaum, Boaz Barak, and David Xiao. On basing lower-bounds for learning on worst-case assumptions. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 211–220. IEEE Computer Society, 2008. doi:10.1109/FOCS.2008.35.

- 6 László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational complexity*, 1(1):3–40, 1991.
- 7 László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993. doi:10.1007/BF01275486.
- 8 Avrim Blum, Merrick Furst, Michael Kearns, and Richard J Lipton. Cryptographic primitives based on hard learning problems. In *Annual International Cryptology Conference*, pages 278–291. Springer, 1993.
- 9 Andrej Bogdanov and Christina Brzuska. On basing size-verifiable one-way functions on np-hardness. 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 I*, volume 9014 of *Lecture Notes in Computer Science*, pages 1–6. Springer, 2015. doi:10.1007/978-3-662-46494-6\_1.
- 10 Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for np problems. *SIAM Journal on Computing*, 36(4):1119–1159, 2006.
- 11 Harry Buhrman, Lance Fortnow, and Thomas Thierauf. Nonrelativizing separations. In *Proceedings. Thirteenth Annual IEEE Conference on Computational Complexity (Formerly: Structure in Complexity Theory Conference)(Cat. No. 98CB36247)*, pages 8–12. IEEE, 1998.
- 12 Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In Ran Raz, editor, *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, volume 50 of *LIPICs*, pages 10:1–10:24. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPICs.CCC.2016.10.
- 13 Lijie Chen, Shuichi Hirahara, Igor Carboni Oliveira, Ján Pich, Ninad Rajgopal, and Rahul Santhanam. Beyond natural proofs: Hardness magnification and locality. 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 70:1–70:48. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ITCS.2020.70.
- 14 Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM Journal on Computing*, 22(5):994–1005, 1993.
- 15 Lance Fortnow and Adam R Klivans. Efficient learning algorithms yield circuit lower bounds. *Journal of Computer and System Sciences*, 75(1):27–36, 2009.
- 16 Lance Fortnow, John Rompel, and Michael Sipser. On the power of multi-prover interactive protocols. *Theoretical Computer Science*, 134(2):545–557, 1994.
- 17 Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.
- 18 Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct randolli functions. In *25th Annual Symposium on Foundations of Computer Science, 1984.*, pages 464–479. IEEE, 1984.
- 19 Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. If NP languages are hard on the worst-case, then it is easy to find their hard instances. *Comput. Complex.*, 16(4):412–441, 2007. doi:10.1007/s00037-007-0235-8.
- 20 Dan Gutfreund and Salil P. Vadhan. Limitations of hardness vs. randomness under uniform reductions. In Ashish Goel, Klaus Jansen, José D. P. Rolim, and Ronitt Rubinfeld, editors, *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, August 25-27, 2008. Proceedings*, volume 5171 of *Lecture Notes in Computer Science*, pages 469–482. Springer, 2008. doi:10.1007/978-3-540-85363-3\_37.
- 21 Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 247–258. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00032.

- 22 Shuichi Hirahara and Osamu Watanabe. On nonadaptive reductions to the set of random strings and its dense subsets. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:25, 2019. URL: <https://eccc.weizmann.ac.il/report/2019/025>.
- 23 Russell Impagliazzo. Relativized separations of worst-case and average-case complexities for NP. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, USA, June 8-10, 2011*, pages 104–114. IEEE Computer Society, 2011. doi:10.1109/CCC.2011.34.
- 24 Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.
- 25 Russell Impagliazzo and Levin LA. No better ways to generate hard np instances than picking uniformly at random. In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, pages 812–821. IEEE, 1990.
- 26 Russell Impagliazzo and Avi Wigderson. Randomness vs time: Derandomization under a uniform assumption. *J. Comput. Syst. Sci.*, 63(4):672–688, 2001. doi:10.1006/jcss.2001.1780.
- 27 Jeffrey C Jackson. An efficient membership-query algorithm for learning dnf with respect to the uniform distribution. *Journal of Computer and System Sciences*, 55(3):414–440, 1997.
- 28 Richard M Karp and Richard J Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the twelfth annual ACM symposium on Theory of computing*, pages 302–309. ACM, 1980.
- 29 Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95, 1994.
- 30 Michael J Kearns, Umesh Virkumar Vazirani, and Umesh Vazirani. *An introduction to computational learning theory*. MIT press, 1994.
- 31 Adam R. Klivans, Pravesh Kothari, and Igor Carboni Oliveira. Constructing hard functions using learning algorithms. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 86–97. IEEE Computer Society, 2013. doi:10.1109/CCC.2013.18.
- 32 Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.
- 33 Leonid A Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61(1):15–37, 1984.
- 34 Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. *Journal of the ACM (JACM)*, 40(3):607–620, 1993.
- 35 Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, pages 2–10. IEEE, 1990.
- 36 Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In Ryan O’Donnell, editor, *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, volume 79 of *LIPICs*, pages 18:1–18:49. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPICs.CCC.2017.18.
- 37 Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.
- 38 Luca Trevisan and Salil Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007.
- 39 Chee K Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical computer science*, 26(3):287–300, 1983.

## A Preliminaries

Let  $\mathcal{F} = \{\mathcal{F}_n\}$ , where  $\mathcal{F}_n$  is set of all Boolean functions over  $\{0, 1\}^n$ , where each  $f_n \in \mathcal{F}_n$  is a function  $f_n : \{0, 1\}^n \rightarrow \{0, 1\}$ . Define  $\text{tt}(f)$  as the truth table of a function  $f_n$  of length  $2^n$ . On the other hand, given a string  $x \in \{0, 1\}^{2^n}$ , define  $\text{fn}(x)$  as the function on  $n$  inputs whose truth table is  $x$ . For every  $n \in \mathbb{N}$ , define  $U_n$  as the uniform distribution over  $\{0, 1\}^n$ .

### A.1 Samplability and Learnability

Let  $\mathcal{C} = \{\mathcal{C}_n\}$ , where  $\mathcal{C}_n \subseteq \mathcal{F}_n$  be a class of functions over  $\{0, 1\}^n$  and  $\mathcal{D} = \{\mathcal{D}_n\}$  be a distribution family over  $\{0, 1\}^*$ , where  $\mathcal{D}_n$  is a distribution over  $\{0, 1\}^n$ .

► **Definition A.1** (Worst-case PAC-learning using random examples). *For any  $0 \leq \epsilon, \delta < 1/2$ , a class  $\mathcal{C}$  is  $(\epsilon, \delta)$ -PAC-learnable in the worst-case using random examples in time  $T(n)$ , if there exists a randomized algorithm  $\mathcal{A}$  such that*

- *For every  $n \in \mathbb{N}$ , for every  $f \in \mathcal{C}_n$ , for every  $\mathcal{D}_n$  over  $\{0, 1\}^n$ ,  $\mathcal{A}$  takes inputs  $1^n, \epsilon, \delta$ , a set of  $m = m(n)$  labeled samples  $(x_1, f(x_1)), \dots, (x_m, f(x_m))$  where each  $x_i \sim \mathcal{D}_n$ , and  $w \in \{0, 1\}^{r(n)}$  as internal randomness.  $\mathcal{A}$  then outputs the description of a circuit  $h$  such that*

$$\Pr_{w \in \{0, 1\}^{r(n)}, x_1, \dots, x_m \sim \mathcal{D}_n} \left\{ \Pr_{y \in \mathcal{D}_n} \{h(y) = f(y)\} \geq 1 - \epsilon \right\} \geq 1 - \delta$$

- *$\mathcal{A}$  runs in time at most  $T(n)$ .*<sup>11</sup>

We can also restrict the learnability to a fixed distribution like the uniform distribution  $U_n$ , where the learner takes random examples chosen over the uniform distribution and hypothesis error is also measured over the uniform distribution. Unless specified otherwise, we use the class of polynomial-sized Boolean circuits P/poly, as the hypothesis class for our learning algorithms.

Furthermore, we can extend this definition to PAC-learning over membership queries by giving the learner  $\mathcal{A}$  oracle access to the function  $f \in \mathcal{C}_n$ , in addition to the random examples drawn from some fixed distribution  $\mathcal{D}_n$  over  $\{0, 1\}^n$ .

To define learnability on average, let  $\mathcal{P} = \{\mathcal{P}_n\}$  be a distribution ensemble over  $\mathcal{C}$ , where  $\mathcal{P}_n$  is a fixed distribution over  $\mathcal{C}_n$ .

► **Definition A.2** (Samplable distributions). *Let  $\mathcal{P}$  be a distribution ensemble over  $\mathcal{C}$ , where for every  $n \in \mathbb{N}$ ,  $\mathcal{P}_n$  is a distribution over the truth tables of  $\mathcal{C}_n$ . Let  $N = 2^n$ . For any non-decreasing function  $S(N) \geq N$ , we say that  $\mathcal{P}$  is samplable in time  $S(N)$ , if there exists a randomized algorithm  $A$  such that for every  $N = 2^n$ , using  $m(N)$  bits of randomness (where  $m(N) \leq S(N)$ ),  $A(1^N, y)$  is distributed identically to  $\mathcal{P}_n$ , where the distribution is over the string  $y$  picked uniformly at random from  $\{0, 1\}^{m(N)}$  and  $A$  runs in time  $S(N)$ .*

*In other words, if  $y$  is picked uniformly at random from  $\{0, 1\}^{m(N)}$  then  $A(1^N, y)$  outputs a truth table from  $\mathcal{C}_n$  which is distributed according to  $\mathcal{P}_n$ . Furthermore, we say that  $\mathcal{P}$  is polynomially samplable if  $S(N) = \text{poly}(N)$ .*

► **Remark A.3.** For the special case where  $\mathcal{C}$  is a class of fixed polynomial sized circuits like  $\text{SIZE}[n^k]$  (or  $\text{SIZE}^{\text{EXP}}[n^k]$ ) for any arbitrary fixed  $k$ , we define a circuit representation scheme for  $\mathcal{C}_n$  given by the set  $R_n \subset \{0, 1\}^{r(n)}$ , where  $r(n) = O(n^k \log n)$ , such that every  $\sigma \in R_n$  is

<sup>11</sup>Note that this immediately implies that  $m(n) \leq T(n)$

a  $\mathcal{C}$ -circuit encoding of a function in  $\mathcal{C}_n$ . Note that this mapping is onto and each function in  $\mathcal{C}_n$  has many representations in  $R_n$ . We also assume that there exists a uniform circuit sequence in  $\mathcal{C}$ , which interprets this encoding as a  $\mathcal{C}$ -circuit and evaluates computations given this encoding.

Now, we can define a distribution ensemble  $\mathcal{P}$  over  $\mathcal{C}$ , where each  $\mathcal{P}_n$  is a distribution over the  $\mathcal{C}$ -circuit encodings, which implicitly defines a distribution over  $\mathcal{C}_n$ . We also define  $S(r(n))$ -samplability of  $\mathcal{P}$ , if there exists a randomized algorithm  $A$  running in time  $S(r(n))$  such that for every  $n \in \mathbb{N}$ ,  $A(1^{r(n)}, y)$  is distributed identically to  $\mathcal{P}_n$ , where the distribution is over the random strings  $y \in \{0, 1\}^{m(n)}$ .

► **Definition A.4** (Average-case learnability [8]). *Let  $\mathcal{C}$  be a class of Boolean functions and  $\mathcal{P} = \{\mathcal{P}_n\}$  be a distribution ensemble over  $\mathcal{C}$ . For any  $0 < \epsilon, \delta < 1/2$ , we say that  $\mathcal{C}$  is  $(\epsilon, \delta)$ -PAC-learnable on average using random examples with respect to  $\mathcal{P}$  in time  $T(n)$ , if there exists a randomized algorithm  $\mathcal{A}$  running in time at most  $T(n)$  such that*

- *For every large enough  $n$ , for any fixed  $f$  drawn according to  $\mathcal{P}_n$ , for every  $\mathcal{D}_n$  over  $\{0, 1\}^n$ ,  $\mathcal{A}$  takes inputs  $1^n, \epsilon, \delta$ , a set of  $m = m(n)$  labeled samples  $(x_1, f(x_1)), \dots, (x_m, f(x_m))$  where each  $x_i \sim \mathcal{D}_n$  and  $w \in \{0, 1\}^*$  (the internal randomness of  $\mathcal{A}$ ) and outputs the description of a circuit  $h$  such that*

$$\Pr_{\substack{f \sim \mathcal{P}_n \\ w \in \{0, 1\}^* \\ x_1, \dots, x_m, y \sim \mathcal{D}_n}} \left\{ \Pr_{y \in \mathcal{D}_n} \{h(y) = f(y)\} \geq 1 - \epsilon \right\} \geq 1 - \delta$$

- *$\mathcal{A}$  runs in time at most  $T(n)$ .*

Furthermore, for any  $0 < \epsilon, \delta < 1/2$ , we say that  $\mathcal{C}$  is  $(\epsilon, \delta)$ -PAC-learnable on average with respect to polynomially samplable distributions over  $\mathcal{C}$  using random examples in time  $T(n)$  if there exists a learning algorithm  $\mathcal{A}$  that runs in time  $T(n)$  such that for every polynomially samplable distribution ensemble  $\mathcal{P}$  over  $\mathcal{C}$ , we have that for every large enough  $n$ ,  $\mathcal{A}$   $(\epsilon, \delta)$ -PAC-learns  $\mathcal{C}_n$  on average using random examples with respect to  $\mathcal{P}_n$ .

We can naturally extend this definition to average-case learning  $\mathcal{C}$  with respect to  $\mathcal{P}$  and a fixed distribution over the examples like  $U_n$ , as well as average-case PAC-learning  $\mathcal{C}$  with membership queries with respect to  $\mathcal{P}$ .

## A.2 Self-Reducibility

In our reductions, we use the following special properties of a function.

► **Definition A.5** (Downward self-reducibility). *A function  $f_n : \{0, 1\}^n \rightarrow \{0, 1\}$  is downward-self-reducible if there is a deterministic polynomial time algorithm  $A$  such that for all  $x \in \{0, 1\}^n$ ,  $A^{f_{n-1}}(x) = f_n(x)$ .*

► **Definition A.6** (Self-Correctibility). *A function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  is said to be self-correctible if there exists a constant  $c \geq 0$  and a probabilistic polynomial-time algorithm  $A$  such that, for every large enough  $n$ , for any function  $\mathcal{O} : \{0, 1\}^n \rightarrow \{0, 1\}$  that agrees with  $f_n$  with probability  $(1 - 1/n^c)$  over the uniform distribution on inputs of length  $n$ , we have that  $\Pr\{A^{\mathcal{O}}(x) = f_n(x)\} \geq 2/3$  for any  $x \in \{0, 1\}^n$ .*

[7] show that any function  $f$  on  $n$  Boolean inputs can be transformed into a function  $f^*$  on  $n$  inputs from a large enough finite field, such that  $f^*$  coincides with  $f$  on the subset  $\{0, 1\}^n$ .

► **Theorem A.7** ([7]). *There exists an EXP-Complete problem  $g^*$  which is self-correctible.*

Furthermore, Trevisan and Vadhan [38] construct a PSPACE-Complete problem which is based on a careful arithmetization and padding of TQBF (using the interactive proof system for PSPACE), which has both these properties.

► **Theorem A.8** ([38]). *There exists a PSPACE-Complete language  $f^* \in \text{DSPACE}[n]$  that is both self-correctible and downward self-reducible (DSR).*

We also use the following results.

► **Lemma A.9.** *If  $\text{EXP} \subseteq \text{P/poly}$ , then  $\text{EXP} = \text{PSPACE}$ . In particular, the function  $f^*$  (from Theorem A.8) is complete for EXP.*

► **Lemma A.10** (Hoeffding's inequality). *Let  $X_1, \dots, X_n$  be independent random variables such that  $0 \leq X_i \leq 1$  for every  $i \in [n]$ . Let  $X = \sum_{i=1}^n X_i$ . Then, for any  $t > 0$ , we have*

$$\Pr\{|X - \mathbf{E}[X]| \geq t\} \leq 2 \exp(-2t^2/n)$$

The following Lemma proves that a random function cannot even be approximated by small-sized circuits and follows from an application of Lemma A.10.

► **Lemma A.11** (Lemma 4 [36]). *For any  $s(n) \geq n$  and  $\delta \in [0, 1/2]$ , we have*

$$\Pr_{f \sim \mathcal{F}_n} \{\exists \text{ circuit of size } \leq s(n) \text{ computing } f \text{ on } \geq (1/2 + \delta)\text{-fraction of the inputs}\} \leq \exp(-\delta^2 2^n + 10s \log s)$$

### A.3 Kolmogorov Complexity

Fix a universal machine  $U$ . Levin [33] defined the following notion of time-bounded Kolmogorov complexity: The  $\text{Kt}$  complexity of a string  $x$  is the minimum  $\text{Kt}(x)$  over  $|p| + \log(t)$  such that  $U(p) = x$  in at most  $t$  steps. It is known [2] that  $\text{Kt}(x)$  is polynomially related to the size of the smallest EXP-oracle circuit computing the function with truth table  $x$  (truncating  $x$  to its longest initial segment with length a power of two).

Similarly,  $\text{KS}(x)$  is the minimum over  $|p| + s$  such that  $U(p) = x$  in at most space  $s$ . It is known [2] that  $\text{KS}(x)$  is polynomially related to the size of the smallest PSPACE-oracle circuit computing the function with truth table  $x$  (truncating  $x$  to its longest initial segment with length a power of two).

Let  $\text{R}_{\text{Kt}}$  be the language consisting of strings  $x$  such that  $\text{Kt}(x) \geq |x|/2$  [3]. Similarly, let  $\text{R}_{\text{KS}}$  be the language consisting of strings  $x$  such that  $\text{KS}(x) \geq |x|/2$  [3].

## B Barriers for Conditional Hardness of Learning

Firstly, we formally define what it means to have a Black-Box Turing reduction from a language  $L$  to a PAC-learning algorithm for a class  $\mathcal{C}$ . Fix the error of the learner to be  $\epsilon = 1/\text{poly}(n)$  (we ignore the confidence parameter, but this only makes our hardness results stronger).

► **Definition B.1** (Turing Reduction to Learning  $\mathcal{C}$ ). *A  $B$ -adaptive black-box reduction from deciding  $L$  to PAC-learning  $\mathcal{C}$  using random examples up to error  $\epsilon$ , is a tuple of probabilistic polynomial time algorithms  $R = (T_1, \dots, T_B, M)$  where  $R$  is given an input  $z \in \{0, 1\}^n$  and randomness  $w \in \{0, 1\}^*$ . For each query,  $R$  constructs a joint distribution  $(X, f(X))$*

## 46:22 On the Structure of Learnability Beyond P/Poly

over  $\{0,1\}^r \times \{0,1\}$  for some  $r \leq n$  and  $f \in \mathcal{C}$ , samples a set  $S = \{(x_i, y_i)\}_{i \leq \text{poly}(n)}$  of independent labeled examples according to  $(X, Y)$  and passes it to the learner. Let  $t(n)$  be the query complexity of each round of adaptivity.  $R$  decides  $z$  by doing the following -

- For each  $1 \leq j \leq B$ ,  $T_j$  gets input  $z$ , fresh random bits from  $w$  and all the  $(j-1) \cdot t(n)$  hypothesis circuits answered for the queries from the previous rounds ( $T_1$  only has  $z$  and randomness  $w$  as input), and outputs  $t(n)$  new queries  $S_{j1}, \dots, S_{jt}$  for the learner, each of which are sets of labeled examples sampled from joint distributions  $(X_{j1}, Y_{j1}), \dots, (X_{jt}, Y_{jt})$ .
- $R$  only has oracle access to the learner.
- $M$  takes as input  $z$ , fresh random bits from  $w$  and the  $B \cdot t(n)$  hypothesis circuits which are the answers made by the learner for all the queries asked by  $T_1, \dots, T_B$ , and outputs the answer.
- The reduction guarantees that if for every oracle  $A$  that is a  $\mathcal{C}$ -circuit learner, if every hypothesis circuit returned by the learner is  $(1 - \varepsilon)$ -close with respect to its corresponding query given to the learner by  $T_1, \dots, T_B$ , then  $M(z) = L(z)$  with high probability over the internal randomness of the reduction  $R$ .

► **Definition B.2.** For any  $B$ -adaptive black-box reduction  $R = (T_1, \dots, T_B)$  from deciding  $L$  to PAC-learning  $\mathcal{C}$  using random examples up to error  $\varepsilon$ , we have

- $R$  is called **strongly black-box**, if  $T_1, \dots, T_B, M$  only have oracle access to the hypothesis circuits and  $M$  decides  $L$  given access to any  $(1 - \varepsilon)$ -close hypothesis circuit answered to each query made by  $T_1, \dots, T_B$ .
- If  $B = 1$ , we call the reduction as **non-adaptive**, and if  $R$  is strongly black-box and  $M$  also makes only non-adaptive queries to the hypotheses circuits, we call the reduction as **fully non-adaptive**.
- $R$  is **oblivious**, if  $T_1, \dots, T_B$  output new queries using only fresh randomness from  $w$  as input and access to the hypotheses generated during the previous rounds. Furthermore,  $M$  accesses each hypothesis using non-adaptively generated, identically distributed queries made from the corresponding distribution over which each hypothesis is guaranteed to be a good approximation. In particular, the obliviousness of the reduction implies the fact that the queries to the learner do not depend on the input  $z$ .

Unless mentioned we think of the query complexity  $t(n) = \text{poly}(n)$ . It is worth to note that since the algorithms  $T_1, \dots, T_B$  are polynomial time algorithms, each joint distribution  $(X, Y)$  must be efficiently samplable.

We first prove Lemma 1.5. This is a reformulation of the proof of Lemma 3.2 used to show hardness of learning PSPACE/poly from a PSPACE-Complete language, into the framework of a black-box reduction.

**Proof of Lemma 1.5.** This is a readaptation of the proof of Corollary 1.3 (via Lemma 3.2). Consider  $R = (T_1, \dots, T_n, M)$  as an  $n$ -adaptive reduction from deciding  $f^*$  to learning PSPACE/poly using random examples over the uniform distribution, where  $T_1, \dots, T_n, M$  are probabilistic polynomial time algorithms which are defined as follows.

For every  $k \leq n$ ,  $T_k$  makes exactly one query to the learner which is the set of examples  $S_k = \{(x_i, y_i)\}_{i \leq \text{poly}(n)}$  drawn from the joint distribution  $(U_k, f^*(U_k))$ , where  $U_k$  is the uniform distribution over  $\{0,1\}^k$ . In the  $k^{\text{th}}$  round of adaptivity,  $T_k$  only makes oracle queries to the hypothesis  $h_{k-1}$  output in the last round. Indeed, let  $h'_{k-1}$  be the oracle circuit which uses  $h_{k-1}$  as an oracle in the self-corrector algorithm for  $f^*$ , and computes  $f^*$  on all  $k-1$  length inputs with high probability. It then outputs a set  $S_k$  of independent labeled samples  $(x_i, y_i)$ , where each  $x_i$  is sampled uniformly at random from  $U_k$  and  $y_i = f^*(x_i)$



computed by using the downward self-reducibility of  $f^*$  with  $h'_{k-1}$ .  $M$  takes the final hypothesis  $h_n$  output by the learner over  $n$  inputs and outputs the value of the self-corrector of  $f^*$  with the oracle  $h_n$ . The correctness of  $R$  and the run-time analyses of  $T_1, \dots, T_n, M$  follow from the proof techniques of Lemma 3.2.

We next show that  $R$  has the required properties. As the self-corrector and the downward self-reduction for  $f^*$  work for any oracle which satisfy the appropriate constraints,  $R$  is correct for any oracle which outputs *any* correct hypothesis for  $f^*$  with respect to the uniform distribution (over different input lengths). Further, it makes only oracle queries to the learner, as well as to all the hypothesis circuits  $h_1, \dots, h_n$ . This makes the reduction strongly black-box. By the property of the self-corrector,  $M$  only makes queries sampled from  $U_n$  to  $h_n$ , which is the same as the query made to the learner. The obliviousness now follows, since only  $f^*$  is learnt in each query, irrespective of the choice of  $z$ . ◀

The main result of the section is the following.

► **Theorem B.3.** *There exists a universal constant  $c > 0$  such that the following holds. For any language  $L$ ,  $\varepsilon_0 = 1/n^c$  and any  $B = \text{poly}(n)$ , if there exists an oblivious,  $B$ -adaptive, strongly black-box reduction from  $L$  to PAC-learning  $\text{NP}/\text{poly}$  using random examples over polynomially samplable distributions up to error  $\varepsilon_0$ , then  $\overline{L} \in \text{AM}^{\text{poly}}$ .*

Recall that the class  $\text{AM}^{\text{poly}}$  refers to the class of languages recognized by constant-round interactive protocols with advice, where we require proper acceptance/rejection probabilities only when the advice is correct. [14] show that  $\text{AM}^{\text{poly}} = \text{NP}/\text{poly}$ . Using Theorem B.3 with  $L = \text{SAT}$ , we get

► **Corollary B.4.** *There exists a universal constant  $c > 0$  such that the following holds. For  $\varepsilon_0 = 1/n^c$  and any  $B = \text{poly}(n)$ , if there exists an oblivious,  $B$ -adaptive, strongly black-box reduction from deciding SAT to learning  $\text{NP}/\text{poly}$  using random examples from polynomially samplable distributions up to error  $\varepsilon_0$ , then  $\text{coNP} \subseteq \text{NP}/\text{poly}$ .*

Corollary B.4 easily implies Theorem 1.6, since  $\text{coNP} \subseteq \text{NP}/\text{poly}$  implies that  $\Sigma_3^P = \Pi_3^P$  [39].

► **Remark B.5.** In addition, we can also extend the proof to the case where  $M$  still makes non-adaptive queries but is not constrained distributionally in its access to all the hypotheses, by directly applying the techniques of [10] for the simulation of  $R$  in  $\text{AM}^{\text{poly}}$ .

The details of the proof of Theorem B.3 can be found in the full version (see Section 1.4 for a sketch).



# The Critical Mean-Field Chayes-Machta Dynamics

**Antonio Blanca** ✉

Pennsylvania State University, University Park, PA, USA

**Alistair Sinclair** ✉

University of California at Berkeley, CA, USA

**Xusheng Zhang** ✉

Pennsylvania State University, University Park, PA, USA

---

## Abstract

The random-cluster model is a unifying framework for studying random graphs, spin systems and electrical networks that plays a fundamental role in designing efficient Markov Chain Monte Carlo (MCMC) sampling algorithms for the classical ferromagnetic Ising and Potts models. In this paper, we study a natural non-local Markov chain known as the *Chayes-Machta dynamics* for the mean-field case of the random-cluster model, where the underlying graph is the complete graph on  $n$  vertices. The random-cluster model is parametrized by an *edge probability*  $p$  and a *cluster weight*  $q$ . Our focus is on the critical regime:  $p = p_c(q)$  and  $q \in (1, 2)$ , where  $p_c(q)$  is the threshold corresponding to the order-disorder phase transition of the model. We show that the mixing time of the Chayes-Machta dynamics is  $O(\log n \cdot \log \log n)$  in this parameter regime, which reveals that the dynamics does not undergo an exponential slowdown at criticality, a surprising fact that had been predicted (but not proved) by statistical physicists. This also provides a nearly optimal bound (up to the  $\log \log n$  factor) for the mixing time of the mean-field Chayes-Machta dynamics in the only regime of parameters where no non-trivial bound was previously known. Our proof consists of a multi-phased coupling argument that combines several key ingredients, including a new local limit theorem, a precise bound on the maximum of symmetric random walks with varying step sizes, and tailored estimates for critical random graphs. In addition, we derive an improved comparison inequality between the mixing time of the Chayes-Machta dynamics and that of the local Glauber dynamics on general graphs; this results in better mixing time bounds for the local dynamics in the mean-field setting.

**2012 ACM Subject Classification** Mathematics of computing  $\rightarrow$  Markov processes; Theory of computation  $\rightarrow$  Design and analysis of algorithms; Theory of computation  $\rightarrow$  Random walks and Markov chains; Theory of computation  $\rightarrow$  Random network models; Mathematics of computing  $\rightarrow$  Random graphs

**Keywords and phrases** Markov Chains, Mixing Times, Random-cluster Model, Ising and Potts Models, Mean-field, Chayes-Machta Dynamics, Random Graphs

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.47

**Category** RANDOM

**Related Version** *Full Version:* <https://arxiv.org/abs/2102.03004v2>

**Funding** *Antonio Blanca:* Research supported in part by NSF grant CCF-1850443.

*Alistair Sinclair:* Research supported in part by NSF grant CCF-1815328.

*Xusheng Zhang:* Research supported in part by NSF grant CCF-1850443.

## 1 Introduction

The *random-cluster model* generalizes classical random graph and spin system models, providing a unifying framework for their study [12]. It plays an indispensable role in the design of efficient Markov Chain Monte Carlo (MCMC) sampling algorithms for the ferromagnetic Ising/Potts model [23, 6, 17] and has become a fundamental tool in the study of phase transitions [1, 11, 10].



© Antonio Blanca, Alistair Sinclair, and Xusheng Zhang;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 47; pp. 47:1–47:15



Leibniz International Proceedings in Informatics

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

The random-cluster model is defined on a finite graph  $G = (V, E)$  with an edge probability parameter  $p \in (0, 1)$  and a cluster weight  $q > 0$ . The set of *configurations* of the model is the set of all subsets of edges  $A \subseteq E$ . The probability of each configuration  $A$  is given by the Gibbs distribution:

$$\mu_{G,p,q}(A) = \frac{1}{Z} \cdot p^{|A|} (1-p)^{|E|-|A|} q^{c(A)}, \quad (1)$$

where  $c(A)$  is the number of connected components in  $(V, A)$  and  $Z := Z(G, p, q)$  is the normalizing factor called the *partition function*.

The special case when  $q = 1$  corresponds to the independent bond percolation model, where each edge of the graph  $G$  appears independently with probability  $p$ . Independent bond percolation is also known as the Erdős-Rényi random graph model when  $G$  is the complete graph.

For integer  $q \geq 2$ , the random-cluster model is closely related to the ferromagnetic  $q$ -state Potts model. Configurations in the  $q$ -state Potts model are the assignments of spin values  $\{1, \dots, q\}$  to the vertices of  $G$ ; the  $q = 2$  case corresponds to the Ising model. A sample  $A \subseteq E$  from the random-cluster distribution can be easily transformed into one for the Ising/Potts model by independently assigning a random spin from  $\{1, \dots, q\}$  to each connected component of  $(V, A)$ . Random-cluster based sampling algorithms, which include the widely-studied Swendsen-Wang dynamics [22], are an attractive alternative to Ising/Potts Markov chains since they are often efficient at “low-temperatures” (large  $p$ ). In this parameter regime, several standard Ising/Potts Markov chains are known to converge slowly.

In this paper we investigate the *Chayes-Machta (CM) dynamics* [9], a natural Markov chain on random-cluster configurations that converges to the random-cluster measure. The CM dynamics is a generalization to non-integer values of  $q$  of the widely studied Swendsen-Wang dynamics [22]. As with all applications of the MCMC method, the primary object of study is the *mixing time*, i.e., the number of steps until the dynamics is close to its stationary distribution, starting from the worst possible initial configuration. We are interested in understanding how the mixing time of the CM dynamics grows as the size of the graph  $G$  increases, and in particular how it relates to the phase transition of the model.

Given a random-cluster configuration  $(V, A)$ , one step of the CM dynamics is defined as follows:

- (i) activate each connected component of  $(V, A)$  independently with probability  $1/q$ ;
- (ii) remove all edges connecting active vertices;
- (iii) add each edge between active vertices independently with probability  $p$ , leaving the rest of the configuration unchanged.

We call (i) the *activation* sub-step, and (ii) and (iii) combined the *percolation* sub-step. It is easy to check that this dynamics is reversible with respect to the Gibbs distribution (1) and thus converges to it [9]. For integer  $q$ , the CM dynamics may be viewed as a variant of the Swendsen-Wang dynamics. In the Swendsen-Wang dynamics, each connected component of  $(V, A)$  receives a random color from  $\{1, \dots, q\}$ , and the edges are updated within each color class as in (ii) and (iii) above; in contrast, the CM dynamics updates the edges of exactly *one* color class. However, note that the Swendsen-Wang dynamics is only well-defined for integer  $q$ , while the CM dynamics is feasible for any real  $q > 1$ . Indeed, the CM dynamics was introduced precisely to allow this generalization.

The study of the interplay between phase transitions and the mixing time of Markov chains goes back to pioneering work in mathematical physics in the late 1980s. This connection for the specific case of the CM dynamics on the complete  $n$ -vertex graph, known as the *mean-field model*, has received some attention in recent years (see [5, 13, 16]) and is the focus of this

paper. As we shall see, the mean-field case is already quite non-trivial, and has historically proven to be a useful starting point in understanding various types of dynamics on more general graphs. We note that, so far, the mean-field is the only setting in which there are tight mixing time bounds for the CM dynamics; all other known bounds are deduced indirectly via comparison with other Markov chains, thus incurring significant overhead [6, 4, 15, 3, 23, 5].

The phase transition for the mean-field random-cluster model is fairly well-understood [8, 20]. In this setting, it is natural to re-parameterize by setting  $p = \zeta/n$ ; the phase transition then occurs at the *critical* value  $\zeta = \zeta_{\text{CR}}(q)$ , where  $\zeta_{\text{CR}}(q) = q$  when  $q \in (0, 2]$  and  $\zeta_{\text{CR}}(q) = 2\left(\frac{q-1}{q-2}\right) \log(q-1)$  for  $q > 2$ . For  $\zeta < \zeta_{\text{CR}}(q)$  all components are of size  $O(\log n)$  with high probability (w.h.p.); that is, with probability tending to 1 as  $n \rightarrow \infty$ . On the other hand, for  $\zeta > \zeta_{\text{CR}}(q)$  there is a unique giant component of size  $\approx \theta n$ , where  $\theta = \theta(\zeta, q)$ . The phase transition is thus analogous to that in  $G(n, p)$  corresponding to the emergence of a giant component.

The phase structure of the mean-field random-cluster model, however, is more subtle and depends crucially on the second parameter  $q$ . In particular, when  $q > 2$  the model exhibits *phase coexistence* at the critical threshold  $\zeta = \zeta_{\text{CR}}(q)$ . Roughly speaking, this means that when  $\zeta = \zeta_{\text{CR}}(q)$ , the set of configurations with all connected components of size  $O(\log n)$ , and set of configurations with a unique giant component, contribute each a constant fraction of the probability mass. For  $q \leq 2$ , on the other hand, there is no phase coexistence.

Phase coexistence at  $\zeta = \zeta_{\text{CR}}(q)$  when  $q > 2$  has significant implications for the speed of convergence of Markov chains, including the CM dynamics. The following detailed connection between the phase structure of the model and the mixing time  $\tau_{\text{mix}}^{\text{CM}}$  of the CM dynamics was recently established in [5, 2, 16]. When  $q > 2$ , we have:

$$\tau_{\text{mix}}^{\text{CM}} = \begin{cases} \Theta(\log n) & \text{if } \zeta \notin [\zeta_{\text{L}}, \zeta_{\text{R}}]; \\ \Theta(n^{1/3}) & \text{if } \zeta = \zeta_{\text{L}}; \\ e^{\Omega(n)} & \text{if } \zeta \in (\zeta_{\text{L}}, \zeta_{\text{R}}), \end{cases} \quad (2)$$

where  $(\zeta_{\text{L}}, \zeta_{\text{R}})$  is the so-called *metastability window*. It is known that  $\zeta_{\text{R}} = q$ , but  $\zeta_{\text{L}}$  does not have a closed form; see [5, 20]; we note that  $\zeta_{\text{CR}}(q) \in (\zeta_{\text{L}}, \zeta_{\text{R}})$  for  $q > 2$ .

When  $q \in (1, 2]$ , there is no metastability window, and the mixing time of the mean-field CM dynamics is  $\Theta(\log n)$  for all  $\zeta \neq \zeta_{\text{CR}}(q)$ . In view of these results, the only case remaining open is when  $q \in (1, 2]$  and  $\zeta = \zeta_{\text{CR}}(q)$ . Our main result shown below concerns precisely this regime, which is particularly delicate and had resisted analysis until now for reasons we explain in our proof overview.

► **Theorem 1.** *The mixing time of the CM dynamics on the complete  $n$ -vertex graph when  $\zeta = \zeta_{\text{CR}}(q) = q$  and  $q \in (1, 2)$  is  $O(\log n \cdot \log \log n)$ .*

A  $\Omega(\log n)$  lower bound is known for the mixing time of the mean-field CM dynamics that holds for all  $p \in (0, 1)$  and  $q > 1$  [5]. Therefore, our result is tight up to the lower order  $O(\log \log n)$  factor, and in fact even better as we explain in Remark 16. The conjectured tight bound when  $\zeta = \zeta_{\text{CR}}(q)$  and  $q \in (1, 2)$  is  $\Theta(\log n)$ . We mention that the  $\zeta = \zeta_{\text{CR}}(q)$  and  $q = 2$  case, which is quite different and not covered by Theorem 1, was considered earlier in [19] for the closely related Swendsen-Wang dynamics, and a tight  $\Theta(n^{1/4})$  bound was established for its mixing time. The same mixing time bound is expected for the CM dynamics in this regime; see Remark 7 for further comments about the  $\zeta = \zeta_{\text{CR}}(q)$ ,  $q = 2$  case.

Our result establishes a striking behavior for random-cluster dynamics when  $q \in (1, 2)$ . Namely, there is no slowdown (exponential or power law) in this regime at the critical threshold  $\zeta = \zeta_{\text{CR}}(q)$ . Note that for  $q > 2$ , as described in (2) above, the mixing time of the

dynamics undergoes an exponential slowdown, transitioning from  $\Theta(\log n)$  when  $\zeta < \zeta_L$ , to a power law at  $\zeta = \zeta_L$ , and to exponential in  $n$  when  $\zeta \in (\zeta_L, \zeta_R)$ . The absence of a critical slowdown for  $q \in (1, 2)$  was in fact predicted by the statistical physics community [14], and our result provides the first rigorous proof of this phenomenon.

Our second result concerns the local *Glauber dynamics* for the random-cluster model. In each step, the Glauber dynamics updates a single edge of the current configuration chosen uniformly at random; a precise definition of this Markov chain is given in the full version of the present paper [7]. In [5], it was established that any upper bound on the mixing time  $\tau_{\text{mix}}^{\text{CM}}$  of the CM dynamics can be translated to one for the mixing time  $\tau_{\text{mix}}^{\text{GD}}$  of the Glauber dynamics, at the expense of a  $\tilde{O}(n^4)$  factor; the  $\tilde{O}$  notation hides polylogarithmic factors. In particular, it was proved in [5] that  $\tau_{\text{mix}}^{\text{GD}} \leq \tau_{\text{mix}}^{\text{CM}} \cdot \tilde{O}(n^4)$ . We provide here an improvement of this comparison inequality.

► **Theorem 2.** *For all  $q > 1$  and all  $\zeta = O(1)$ ,  $\tau_{\text{mix}}^{\text{GD}} \leq \tau_{\text{mix}}^{\text{CM}} \cdot O(n^3(\log n)^2)$ .*

To prove this theorem, we establish a general comparison inequality that holds for any graph, any  $q \geq 1$  and any  $p \in (0, 1)$ ; see the full version of this paper [7] for a precise statement. When combined with the known mixing time bounds for the CM dynamics on the complete graph, Theorem 2 yields that the random-cluster Glauber dynamics mixes in  $\tilde{O}(n^3)$  steps when  $q > 2$  and  $\zeta \notin (\zeta_L, \zeta_R)$ , or when  $q \in (1, 2)$  and  $\zeta = O(1)$ . In these regimes, the mixing time of the Glauber dynamics was previously known to be  $\tilde{O}(n^4)$  and is conjectured to be  $\tilde{O}(n^2)$ ; the improved comparison inequality in Theorem 2 gets us closer to this conjectured tight bound. We note, however, that even if one showed the conjectured optimal bound for the mixing time of the Glauber dynamics, the CM is faster, even if we take into account the computational cost associated to implementing its steps.

We conclude this introduction with some brief remarks about our analysis techniques, which combine several key ingredients in a non-trivial way. Our bound on the mixing time uses the well-known technique of *coupling*: in order to show that the mixing time is  $O(\log n \cdot \log \log n)$ , it suffices to couple the evolutions of two copies of the dynamics, starting from two arbitrary configurations, in such a way that they arrive at the *same* configuration after  $O(\log n)$  steps with probability  $\Omega(1/\log \log n)$ . (The moves of the two copies can be correlated any way we choose, provided that each copy, viewed in isolation, is a valid realization of the dynamics.) Because of the delicate nature of the phase transition in the random-cluster model, combined with the fact that the percolation sub-step of the CM dynamics is critical when  $\zeta = q$ , our coupling is somewhat elaborate and proceeds in multiple phases. The first phase consists of a *burn-in period*, where the two copies of the chain are run independently and the evolution of their largest components is observed until they have shrunk to their “typical” sizes. This part of the analysis is inspired by similar arguments in earlier work [5, 19, 13].

In the second phase, we design a coupling of the activation of the connected components of the two copies which uses: (i) a local limit theorem, which can be thought of as a stronger version of a central limit theorem; (ii) a precise understanding of the distribution of the maximum of symmetric random walks on  $\mathbb{Z}$  with varying step sizes; and (iii) precise estimates for the component structure of random graphs. We develop tailored versions of these probabilistic tools for our setting and combine them to guarantee that the same number of vertices from each copy are activated in each step w.h.p. for sufficiently many steps. This phase of the coupling is the main novelty in our analysis, and allows us to quickly converge to the same configuration.

The rest of the paper is organized as follows. In Section 2, we give a detailed overview of our proof. The proof of the key step in our coupling construction is provided in Section 3; all others proofs are deferred to the full version of this paper [7].

## 2 Proof sketch and techniques

We now give a detailed sketch of the multi-phased coupling argument for proving Theorem 1. We start by formally defining the notions of mixing and coupling times. Let  $\Omega_{\text{RC}}$  be the set of random-cluster configurations of a graph  $G$ ; let  $\mathcal{M}$  be the transition matrix of a random-cluster Markov chain with stationary distribution  $\mu = \mu_{G,p,q}$ , and let  $\mathcal{M}^t(X_0, \cdot)$  be the distribution of the chain after  $t$  steps starting from  $X_0 \in \Omega_{\text{RC}}$ . The  $\varepsilon$ -mixing time of  $\mathcal{M}$  is given by

$$\tau_{\text{mix}}^{\mathcal{M}}(\varepsilon) := \max_{X_0 \in \Omega_{\text{RC}}} \min_{t \geq 0} \{ \|\mathcal{M}^t(X_0, \cdot) - \mu(\cdot)\|_{\text{TV}} \leq \varepsilon \},$$

where  $\|\cdot\|_{\text{TV}}$  denotes total variation distance. In particular, the *mixing time* of  $\mathcal{M}$  is  $\tau_{\text{mix}}^{\mathcal{M}} := \tau_{\text{mix}}^{\mathcal{M}}(1/4)$ .

A *(one step) coupling* of the Markov chain  $\mathcal{M}$  specifies, for every pair of states  $(X_t, Y_t) \in \Omega_{\text{RC}} \times \Omega_{\text{RC}}$ , a probability distribution over  $(X_{t+1}, Y_{t+1})$  such that the processes  $\{X_t\}$  and  $\{Y_t\}$  are valid realizations of  $\mathcal{M}$ , and if  $X_t = Y_t$  then  $X_{t+1} = Y_{t+1}$ . The *coupling time*, denoted  $T_{\text{coup}}$ , is the minimum  $T$  such that  $\Pr[X_T \neq Y_T] \leq 1/4$ , starting from the worst possible pair of configurations in  $\Omega_{\text{RC}}$ . It is a standard fact that  $\tau_{\text{mix}}^{\mathcal{M}} \leq T_{\text{coup}}$ ; moreover, when  $\Pr[X_T = Y_T] \geq \delta$  for some coupling, then  $\tau_{\text{mix}}^{\mathcal{M}} = O(T\delta^{-1})$  (see, e.g., [18]).

We provide first a high level description of our coupling for the CM dynamics. For this, we require the following notation. For a random cluster configuration  $X$ , let  $L_i(X)$  denote the size of the  $i$ -th largest connected component in  $(V, X)$ , and let  $\mathcal{R}_i(X) := \sum_{j \geq i} L_j(X)^2$ ; in particular,  $\mathcal{R}_1(X)$  is the sum of the squares of the sizes of all the components of  $(V, X)$ . Our coupling has three main phases:

1. *Burn-in period*: run two copies  $\{X_t\}, \{Y_t\}$  independently, starting from a pair of arbitrary initial configurations, until  $\mathcal{R}_1(X_T) = O(n^{4/3})$  and  $\mathcal{R}_1(Y_T) = O(n^{4/3})$ .
2. *Coupling to the same component structure*: starting from  $X_T$  and  $Y_T$  such that  $\mathcal{R}_1(X_T) = O(n^{4/3})$  and  $\mathcal{R}_1(Y_T) = O(n^{4/3})$ , we design a two-phased coupling that reaches two configurations with the same component structure as follows:
  - 2a. A two-step coupling after which the two configurations agree on all “large components”;
  - 2b. A coupling that after  $O(\log n)$  additional steps reaches two configurations that will also have the same “small component” structure.
3. *Coupling to the same configuration*: starting from two configurations with the same component structure, there is a straightforward coupling that couples the two configurations in  $O(\log n)$  steps w.h.p.

We proceed to describe each of these phases in detail.

### 2.1 The burn-in period

During the initial phase, two copies of the dynamics evolve independently. This is called a *burn-in period* and in our case consists of three sub-phases.

In the first sub-phase of the burn-in period the goal is to reach a configuration  $X$  such that  $\mathcal{R}_2(X) = O(n^{4/3})$ . For this, we use a lemma from [2], which shows that after  $T = O(\log n)$  steps of the CM dynamics  $\mathcal{R}_2(X_T) = O(n^{4/3})$  with at least constant probability; this holds when  $\zeta = q$  for any initial configuration  $X_0$  and any  $q > 1$ .

► **Lemma 3** ([2], Lemma 3.42). *Let  $q > 1$  and  $\zeta = q$ , and let  $X_0$  be an arbitrary random-cluster configuration. Then, for any constant  $C \geq 0$ , after  $T = O(\log n)$  steps  $\mathcal{R}_2(X_T) = O(n^{4/3})$  and  $L_1(X_T) > Cn^{2/3}$  with probability  $\Omega(1)$ .*

In the second and third sub-phases of the burn-in period, we use the fact that when  $\mathcal{R}_2(X_t) = O(n^{4/3})$ , the number of activated vertices is well concentrated around  $n/q$  (its expectation). This is used to show that the size of the largest component contracts at a constant rate for  $T = O(\log n)$  steps until a configuration  $X_T$  is reached such that  $\mathcal{R}_1(X_T) = O(n^{4/3})$ . This part of the analysis is split into two sub-phases because the contraction for  $L_1(X_t)$  requires a more delicate analysis when  $L_1(X_t) = o(n)$ ; this is captured in the following two lemmas.

► **Lemma 4.** *Let  $\zeta = q$  and  $q \in (1, 2)$ . Suppose  $\mathcal{R}_2(X_0) = O(n^{4/3})$ . Then, for any constant  $\delta > 0$ , there exists  $T = T(\delta) = O(1)$  such that  $\mathcal{R}_2(X_T) = O(n^{4/3})$  and  $L_1(X_T) \leq \delta n$  with probability  $\Omega(1)$ .*

► **Lemma 5.** *Let  $\zeta = q$  and  $q \in (1, 2)$ . Suppose  $\mathcal{R}_2(X_0) = O(n^{4/3})$  and that  $L_1(X_0) \leq \delta n$  for a sufficiently small constant  $\delta$ . Then, with probability  $\Omega(1)$ , after  $T = O(\log n)$  steps  $\mathcal{R}_1(X_T) = O(n^{4/3})$ .*

Lemmas 4 and 5 are proved in the full paper [7]. Combining them with Lemma 3 immediately yields the following theorem.

► **Theorem 6.** *Let  $\zeta = q$ ,  $q \in (1, 2)$  and let  $X_0$  be an arbitrary random-cluster configuration of the complete  $n$ -vertex graph. Then, with probability  $\Omega(1)$ , after  $T = O(\log n)$  steps  $\mathcal{R}_1(X_T) = O(n^{4/3})$ .*

► **Remark 7.** The contraction of  $L_1(X_t)$  established by Lemmas 4 and 5 only occurs when  $q \in (1, 2)$ ; when  $q > 2$  the quantity  $L_1(X_t)$  may increase in expectation, whereas for  $q = 2$  we have  $\mathbb{E}[L_1(X_{t+1}) \mid X_t] \approx L_1(X_t)$ , and the contraction of the size of the largest component is due instead to fluctuations caused by a large second moment. (This is what causes the power law slowdown when  $\zeta = q = 2$ .)

► **Remark 8.** Sub-steps (ii) and (iii) of the CM dynamics are equivalent to replacing the active portion of the configuration by a  $G(m, q/n)$  random graph, where  $m$  is the number of active vertices. Since  $\mathbb{E}[m] = n/q$ , one key challenge in the proofs of Lemmas 4 and 5, and in fact in the entirety of our analysis, is that the random graph  $G(m, q/n)$  is critical or almost critical w.h.p. since  $m \cdot q/n \approx 1$ ; consequently its structural properties are not well concentrated and cannot be maintained for the required  $O(\log n)$  steps of the coupling. This is one of the key reasons why the  $\zeta = \zeta_{\text{CR}}(q) = q$  regime is quite delicate.

## 2.2 Coupling to the same component structure

For the second phase of the coupling, we assume that we start from a pair of configurations  $X_0, Y_0$  such that  $\mathcal{R}_1(X_0) = O(n^{4/3})$ ,  $\mathcal{R}_1(Y_0) = O(n^{4/3})$ . The goal is to show that after  $T = O(\log n)$  steps, with probability  $\Omega(1/\log \log n)$ , we reach two configurations  $X_T$  and  $Y_T$  with the same component structure; i.e.,  $L_j(X_T) = L_j(Y_T)$  for all  $j \geq 1$ . In particular, we prove the following.

► **Theorem 9.** *Let  $\zeta = q$ ,  $q \in (1, 2)$  and suppose  $X_0, Y_0$  are random-cluster configurations such that  $\mathcal{R}_1(X_0) = O(n^{4/3})$  and  $\mathcal{R}_1(Y_0) = O(n^{4/3})$ . Then, there exists a coupling of the CM steps such that after  $T = O(\log n)$  steps  $X_T$  and  $Y_T$  have the same component structure with probability  $\Omega((\log \log n)^{-1})$ .*

Our coupling construction for proving Theorem 9 has two main sub-phases. The first is a two-step coupling after which the two configurations agree on all the components of size above a certain threshold  $B_\omega = n^{2/3}/\omega(n)$ , where  $\omega(n)$  is a slowly increasing function. For convenience and definiteness we set  $\omega(n) = \log \log \log \log n$ . In the second sub-phase we take care of matching the small component structures.



We note that when the same number of vertices are activated from each copy of the chain, we can easily couple the percolation sub-step (with an arbitrary bijection between the activated vertices) and replace the configuration on the active vertices in both chains with the same random sub-graph; consequently, the component structure in the updated sub-graph would be identical. Our goal is thus to design a coupling of the activation of the components that activates the same number of vertices in both copies in every step.

In order for the initial two-step coupling to succeed, certain (additional) properties of the configurations are required. These properties are achieved with a continuation of the initial burn-in phase for a small number of  $O(\log \omega(n))$  steps. For a random-cluster configuration  $X$ , let  $\tilde{\mathcal{R}}_\omega(X) = \sum_{j: L_j(X) \leq B_\omega} L_j(X)^2$  and let  $I(X)$  denote the number of isolated vertices of  $X$ . Our extension of the burn-in period is captured by the following lemma.

► **Lemma 10.** *Let  $\zeta = q$ ,  $q \in (1, 2)$  and suppose  $X_0$  is such that  $\mathcal{R}_1(X_0) = O(n^{4/3})$ . Then, there exists  $T = O(\log \omega(n))$  and a constant  $\beta > 0$  such that  $\tilde{\mathcal{R}}_\omega(X_T) = O(n^{4/3}\omega(n)^{-1/2})$ ,  $\mathcal{R}_1(X_T) = O(n^{4/3})$  and  $I(X_T) = \Omega(n)$  with probability  $\Omega(\omega(n)^{-\beta})$ .*

With these bounds on  $\tilde{\mathcal{R}}_\omega(X_T)$ ,  $\tilde{\mathcal{R}}_\omega(Y_T)$ ,  $I(X_T)$  and  $I(Y_T)$ , we construct the two-step coupling for matching the *large* component structure. The construction crucially relies on a new local limit theorem (Theorem 17). In particular, under our assumptions, when  $\omega(n)$  is small enough, there are few components with sizes above  $B_\omega$ . Hence, we can condition on the event that all of them are activated simultaneously. The difference in the number of active vertices generated by the activation of these large components can then be “corrected” by a coupling of the activation of the smaller components; for this we use our new local limit theorem.

Specifically, our local limit theorem applies to the random variables corresponding to the number of activated vertices from the small components of each copy. We prove it using a result of Mukhin [21] and the fact that, among the small components, there are (roughly speaking) many components of many different sizes. To establish the latter we require a refinement of known random graph estimates (see Lemma 23).

To formally state our result we introduce some additional notation. Let  $\mathcal{S}_\omega(X)$  be the set of connected components of  $X$  with sizes greater than  $B_\omega$ . At step  $t$ , the activation of the components of two random-cluster configurations  $X_t$  and  $Y_t$  is done using a maximal matching  $W_t$  between the components of  $X_t$  and  $Y_t$ , with the restriction that only components of equal size are matched to each other. For an increasing positive function  $g$  and each integer  $k \geq 0$ , define  $\hat{N}_k(t, g) := \hat{N}_k(X_t, Y_t, g)$  as the number of matched pairs in  $W_t$  whose component sizes are in the interval

$$\mathcal{I}_k(g) = \left[ \frac{\vartheta n^{2/3}}{2g(n)^{2k}}, \frac{\vartheta n^{2/3}}{g(n)^{2k}} \right],$$

where  $\vartheta > 0$  is a fixed large constant (independent of  $n$ ).

► **Lemma 11.** *Let  $\zeta = q$ ,  $q \in (1, 2)$  and suppose  $X_0, Y_0$  are random-cluster configurations such that  $\mathcal{R}_1(X_0) = O(n^{4/3})$ ,  $\tilde{\mathcal{R}}_\omega(X_0) = O(n^{4/3}\omega(n)^{-1/2})$ ,  $I(X_0) = \Omega(n)$  and similarly for  $Y_0$ . Then, there exists a two-step coupling of the CM dynamics such that  $\mathcal{S}_\omega(X_2) = \mathcal{S}_\omega(Y_2)$  with probability  $\exp(-O(\omega(n)^9))$ .*

*Moreover,  $L_1(X_2) = O(n^{2/3}\omega(n))$ ,  $\mathcal{R}_2(X_2) = O(n^{4/3})$ ,  $\tilde{\mathcal{R}}_\omega(X_2) = O(n^{4/3}\omega(n)^{-1/2})$ ,  $I(X_2) = \Omega(n)$ ,  $\hat{N}_k(2, \omega(n)) = \Omega(\omega(n)^{3 \cdot 2^{k-1}})$  for all  $k \geq 1$  such that  $n^{2/3}\omega(n)^{-2^{k-1}} \rightarrow \infty$ , and similarly for  $Y_2$ .*

From the first part of the lemma we obtain two configurations that agree on all of their large components, as desired, while the second part guarantees additional structural properties for the resulting configurations so that the next sub-phase of the coupling can also succeed with the required probability.

In the second sub-phase, after the large component are matched, we can design a coupling that activates exactly the same number of vertices from each copy of the chain. To analyze this coupling we use a precise estimate on the distribution of the maximum of symmetric random walks over integers (with steps of different sizes). We are first required to run the chains coupled for  $T = O(\log \omega(n))$  steps, so that certain additional structural properties appear. Let  $M(X_t)$  and  $M(Y_t)$  be the components in the matching  $W_t$  that belong to  $X_t$  and  $Y_t$ , respectively, and let  $D(X_t)$  and  $D(Y_t)$  be the complements of  $M(X_t)$  and  $M(Y_t)$ . Let  $Z_t = \sum_{C \in D(X_t) \cup D(Y_t)} |C|^2$ .

► **Lemma 12.** *Let  $\zeta = q$ ,  $q \in (1, 2)$ . Suppose  $X_0$  and  $Y_0$  are random-cluster configurations such that  $\mathcal{S}_\omega(X_0) = \mathcal{S}_\omega(Y_0)$ , and  $\hat{N}_k(0, \omega(n)) = \Omega(\omega(n)^{3 \cdot 2^{k-1}})$  for all  $k \geq 1$  such that  $n^{2/3} \omega(n)^{-2^{k-1}} \rightarrow \infty$ . Suppose also that  $L_1(X_0) = O(n^{2/3} \omega(n))$ ,  $\mathcal{R}_2(X_0) = O(n^{4/3})$ ,  $\hat{\mathcal{R}}_\omega(X_0) = O(n^{4/3} \omega(n)^{-1/2})$ ,  $I(X_0) = \Omega(n)$ , and similarly for  $Y_0$ .*

*Then, there exists a coupling of the CM steps such that with probability  $e^{-O((\log \omega(n))^2)}$  after  $T = O(\log \omega(n))$  steps:  $\mathcal{S}_\omega(X_T) = \mathcal{S}_\omega(Y_T)$ ,  $Z_T = O(n^{4/3} \omega(n)^{-1/2})$ ,  $\hat{N}_k(T, \omega(n)^{1/2}) = \Omega(\omega(n)^{3 \cdot 2^{k-2}})$  for all  $k \geq 1$  such that  $n^{2/3} \omega(n)^{-2^{k-1}} \rightarrow \infty$ ,  $\mathcal{R}_1(X_T) = O(n^{4/3})$ ,  $I(X_T) = \Omega(n)$ , and similarly for  $Y_T$ .*

The proof of Lemma 12 also uses our local limit theorem (Theorem 17).

The final step of our construction is a coupling of the activation of the components of size less than  $B_\omega$ , so that exactly the same number of vertices are activated from each copy in each step w.h.p.

► **Lemma 13.** *Let  $\zeta = q$ ,  $q \in (1, 2)$  and suppose  $X_0$  and  $Y_0$  are random-cluster configurations such that  $\mathcal{S}_\omega(X_0) = \mathcal{S}_\omega(Y_0)$ ,  $Z_0 = O(n^{4/3} \omega(n)^{-1/2})$ , and  $\hat{N}_k(0, \omega(n)^{1/2}) = \Omega(\omega(n)^{3 \cdot 2^{k-2}})$  for all  $k \geq 1$  such that  $n^{2/3} \omega(n)^{-2^{k-1}} \rightarrow \infty$ . Suppose also that  $\mathcal{R}_1(X_0) = O(n^{4/3})$ ,  $I(X_0) = \Omega(n)$  and similarly for  $Y_0$ . Then, there exist a coupling of the CM steps and a constant  $\beta > 0$  such that after  $T = O(\log n)$  steps,  $X_T$  and  $Y_T$  have the same component structure with probability  $\Omega((\log \log n)^{-\beta})$ .*

We comment briefly on how we prove this lemma. Our starting point is two configurations with the same “large” component structure; i.e.,  $\mathcal{S}_\omega(X_0) = \mathcal{S}_\omega(Y_0)$ . We use the maximal matching  $W_0$  to couple the activation of the large components in  $X_0$  and  $Y_0$ . The small components *not matched* by  $W_0$ , i.e., those counted in  $Z_0$ , are then activated independently. This creates a discrepancy  $\mathcal{D}_0$  between the number of active vertices from each copy. Since  $E[\mathcal{D}_0] = 0$  and  $\text{Var}(\mathcal{D}_0) = \Theta(Z_0) = \Theta(n^{4/3} \omega(n)^{-1/2})$ , it follows from Hoeffding’s inequality that  $\mathcal{D}_0 \leq n^{2/3} \omega(n)^{-1/4}$  w.h.p. To fix this discrepancy, we use the small components *matched* by  $W_0$ . Specifically, under the assumptions in Lemma 13, we can construct a coupling of the activation of the small components so that the difference in the number of activated vertices from the small components from each copy is exactly  $\mathcal{D}_0$  with probability  $\Omega(1)$ . This part of the construction utilizes random walks over the integers; in particular, we use a lower bound for the maximum of such a random walk.

We need to repeat this process until  $Z_t = 0$ ; this takes  $O(\log n)$  steps since  $Z_t \approx (1 - 1/q)^t Z_0$ . However, there are a few complications. First, the initial assumptions on the component structure of the configurations are not preserved for this many steps w.h.p., so we need to relax the requirements as the process evolves. This is in turn possible because the discrepancy  $\mathcal{D}_t$  decreases with each step, which implies that the probability of success of the coupling increases at each step.

Proof of Lemma 10, 12 and 13 is provided in the full version of the present paper [7]. We now indicate how these lemmas lead to a proof of Theorem 9 stated earlier.

**Proof of Theorem 9.** Suppose  $\mathcal{R}_1(X_0) = O(n^{4/3})$  and  $\mathcal{R}_1(Y_0) = O(n^{4/3})$ . It follows from Lemma 10, 11, 12 and 13 that there exists a coupling of the CM steps such that after  $T = O(\log n)$  steps,  $X_T$  and  $Y_T$  could have the same component structure. This coupling succeeds with probability at least

$$\rho = \Omega(\omega(n)^{-\beta_1}) \cdot \exp(-O(\omega(n)^9)) \cdot \exp(-O((\log \omega(n))^2)) \cdot \Omega((\log \log \log n)^{-\beta_2}),$$

where  $\beta_1, \beta_2 > 0$  are constants. Thus,  $\rho = \Omega((\log \log n)^{-1})$ , since  $\omega(n) = \log \log \log \log n$ . ◀

► **Remark 14.** We pause to mention that this delicate coupling for the activation of the components is not required when  $\zeta = q$  and  $q > 2$ . In that regime, the random-cluster model is super-critical, so after the first  $O(\log n)$  steps, the component structure is much simpler, with exactly one large component. On the other hand, when  $\zeta = q$  and  $q \in (1, 2]$  the model is critical, which, combined with the fact mentioned earlier that the percolation sub-step of the dynamics is also critical when  $\zeta = q$ , makes the analysis of the CM dynamics in this regime quite subtle.

### 2.3 Coupling to the same configuration

In the last phase of the coupling, suppose we start with two configurations  $X_0, Y_0$  with the same component structure. We are still required to bound the number of steps until the same configuration is reached. The following lemma from [5] supplies the desired bound.

► **Lemma 15** ([5], Lemma 24). *Let  $q > 1$ ,  $\zeta > 0$  and let  $X_0, Y_0$  be two random-cluster configurations with the same component structure. Then, there exists a coupling of the CM steps such that after  $T = O(\log n)$  steps,  $X_T = Y_T$  w.h.p.*

Combining the results for each of the phases of the coupling, we now prove Theorem 1.

**Proof of Theorem 1.** By Theorem 6, after  $t_0 = O(\log n)$  steps, with probability  $\Omega(1)$ , we have  $\mathcal{R}_1(X_{t_0}) = O(n^{4/3})$  and  $\mathcal{R}_1(Y_{t_0}) = O(n^{4/3})$ . If this is the case, Theorem 9 and Lemma 15 imply that there exists a coupling of the CM steps such that with probability  $\Omega((\log \log n)^{-1})$  after an additional  $t_1 = O(\log n)$  steps,  $X_{t_0+t_1} = Y_{t_0+t_1}$ . Consequently, we obtain that  $\tau_{\text{mix}}^{\text{CM}} = O(\log n \cdot \log \log n)$  as claimed. ◀

► **Remark 16.** The probability of success in Theorem 9, which governs the lower order term  $O(\log \log n)$  in our mixing time bound, is controlled by our choice of the function  $\omega(n)$  for the definition of “large components”. By choosing  $\omega(n)$  that goes to  $\infty$  more slowly, we could improve our mixing time bound to  $O(\log n \cdot g(n))$  where  $g(n)$  is any function that tends to infinity arbitrarily slowly. However, it seems that new ideas are required to obtain a bound of  $O(\log n)$  (matching the known lower bound). In particular, the fact that  $\omega(n) \rightarrow \infty$  is crucially used in some of our proofs. Our specific choice of  $\omega(n)$  yields the  $O(\log n \cdot \log \log n)$  bound and makes our analysis cleaner.

## 3 Coupling to the same component structure: proof of Lemma 11

To prove Lemma 11, we use a local limit theorem to construct a two-step coupling of the CM dynamics that reaches two configurations with the same large component structure. The construction of Markov chain couplings using local limit theorems is not common (see [19] for another example), but it appears to be a powerful technique that may have other interesting applications. We provide next a brief introduction to local limit theorems.

### 3.1 Local limit theorem

Let  $c_1 \leq \dots \leq c_m$  be integers and for  $i = 1, \dots, m$  let  $X_i$  be the random variable that is equal to  $c_i$  with probability  $r \in (0, 1)$ , and it is zero otherwise. Let us assume that  $X_1, \dots, X_m$  are independent random variables. Let  $S_m = \sum_{i=1}^m X_i$ ,  $\mu_m = \mathbb{E}[S_m]$  and  $\sigma_m^2 = \text{Var}(S_m)$ . We say that a *local limit theorem* holds for  $S_m$  if for every integer  $a \in \mathbb{Z}$ :

$$\Pr[S_m = a] = \frac{1}{\sqrt{2\pi}\sigma_m} \exp\left(-\frac{(a - \mu_m)^2}{2\sigma_m^2}\right) + o(\sigma_m^{-1}). \quad (3)$$

We prove, under some conditions, a local limit theorem that applies to the random variables corresponding to the number of active vertices from small components. Recall that for an increasing positive function  $g$  and each integer  $k \geq 0$ , we defined the intervals

$$\mathcal{I}_k(g) = \left[ \frac{\vartheta n^{2/3}}{2g(n)^{2^k}}, \frac{\vartheta n^{2/3}}{g(n)^{2^k}} \right],$$

where  $\vartheta > 0$  is a fixed large constant.

► **Theorem 17.** *Let  $c_1 \leq \dots \leq c_m$  be integers, and suppose  $X_1, \dots, X_m$  are independent random variables such that  $X_i$  is equal to  $c_i$  with probability  $r \in (0, 1)$ , and  $X_i$  is zero otherwise. Let  $g : \mathbb{N} \rightarrow \mathbb{R}$  be an increasing positive function such that  $g(m) \rightarrow \infty$  and  $g(m) = o(\log m)$ . Suppose  $c_m = O(m^{2/3}g(m)^{-1})$ ,  $\sum_{i=1}^m c_i^2 = O(m^{4/3}g(m)^{-1/2})$  and  $c_i = 1$  for all  $i \leq \rho m$ , where  $\rho \in (0, 1)$  is independent of  $m$ . Let  $\ell > 0$  be the smallest integer such that  $m^{2/3}g(m)^{-2^\ell} = o(m^{1/4})$ . If for all  $1 \leq k \leq \ell$ , we have  $|\{i : c_i \in \mathcal{I}_k(g)\}| = \Omega(g(m)^{3 \cdot 2^{k-1}})$ , then a local limit theorem holds for  $S_m = \sum_{i=1}^m X_i$ .*

Theorem 17 follows from a general local limit theorem proved in [21]; a proof is given in the full paper [7]. We next compile a number of (mostly standard) facts about the  $G(n, p)$  random graph model which will be used in our proof of Lemma 11.

### 3.2 Random graphs estimates

We use  $G \sim G(n, p)$  to denote a random graph  $G$  sampled from the standard  $G(n, p)$  model, in which every edge appears independently with probability  $p$ . For a graph  $G$ , with a slight abuse of notation, let  $L_i(G)$  denote the size of the  $i$ -th largest connected component in  $G$ , and let  $\mathcal{R}_i(G) := \sum_{j \geq i} L_j(G)^2$ ; note that the same notation is used for the components of a random-cluster configuration, but it will always be clear from context which case is meant.

► **Lemma 18** ([19], Lemma 5.7). *Let  $I(G)$  denote the number of isolated vertices in  $G$ . If  $np = O(1)$ , then there exists a constant  $C > 0$  such that  $\Pr[I(G) > Cn] = 1 - O(n^{-1})$ .*

► **Lemma 19** ([2], Lemma 2.16). *If  $np > 0$ , we have  $\mathbb{E}[\mathcal{R}_2(G)] = O(n^{4/3})$ .*

► **Lemma 20.** *Let  $G \sim G(n, \frac{1+\varepsilon}{n})$  with  $\varepsilon = o(1)$ . For any positive constant  $\rho \leq 1/10$ , there exist constants  $C \geq 1$  and  $c > 0$  such that if  $\varepsilon^3 n \geq C$ , then*

$$\Pr[|L_1(G) - 2\varepsilon n| > \rho\varepsilon n] = O(\exp(-c\varepsilon^3 n)).$$

For the next results, suppose that  $G \sim G(n, \frac{1+\lambda n^{-1/3}}{n})$ , where  $\lambda = \lambda(n)$  may depend on  $n$ .

► **Lemma 21.** *If  $|\lambda| = O(1)$ , then  $\mathbb{E}[\mathcal{R}_1(G)] = O(n^{4/3})$ .*

All the random graph facts stated so far can be either found in the literature, or follow directly from well-known results. The following lemmas are slightly more refined versions of similar results in the literature.

► **Lemma 22.** *Suppose  $|\lambda| = O(h(n))$  and let  $B_h = n^{2/3}h(n)^{-1}$ , where  $h : \mathbb{N} \rightarrow \mathbb{R}$  is a positive increasing function such that  $h(n) = o(\log n)$ . Then, for any  $\alpha \in (0, 1)$  there exists a constant  $C = C(\alpha) > 0$  such that, with probability at least  $\alpha$ ,*

$$\sum_{j: L_j(G) \leq B_h} L_j(G)^2 \leq Cn^{4/3}h(n)^{-1/2}.$$

► **Lemma 23.** *Let  $S_B = \{j : B \leq L_j(G) \leq 2B\}$  and suppose there exists a positive increasing function  $g$  such that  $g(n) \rightarrow \infty$ ,  $g(n) = o(n^{1/3})$ ,  $|\lambda| \leq g(n)$  and  $B \leq n^{2/3}g(n)^{-2}$ . If  $B \rightarrow \infty$ , then there exists constants  $\delta_1, \delta_2 > 0$  independent of  $n$  such that*

$$\Pr \left[ |S_B| \leq \frac{\delta_1 n}{B^{3/2}} \right] \leq \frac{\delta_2 B^{3/2}}{n}.$$

Finally, the following corollary of Lemma 23 will also be useful. For a graph  $H$ , let  $N_k(H, g)$  be the number of components of  $H$  whose sizes are in the interval  $\mathcal{I}_k(g)$ . We note that with a slight abuse of notation, for a random-cluster configuration  $X$ , we also use  $N_k(X, g)$  for the number of connected components of  $X$  in  $\mathcal{I}_k(g)$ .

► **Lemma 24.** *Let  $m \in (n/2q, n]$  and let  $g$  be an increasing positive function such that  $g(n) = o(m^{1/3})$ ,  $g(n) \rightarrow \infty$  and  $|\lambda| \leq g(m)$ . If  $H \sim G(m, \frac{1+\lambda m^{-1/3}}{m})$ , there exists a constant  $b > 0$  such that, with probability at least  $1 - O(g(n)^{-3})$ ,  $N_k(H, g) \geq bg(n)^{3 \cdot 2^{k-1}}$  for all  $k \geq 1$  such that  $n^{2/3}g(n)^{-2^k} \rightarrow \infty$ .*

The proofs of Lemmas 20-24 are given in the full version of the paper [7].

### 3.3 Proof of Lemma 11

For a random-cluster configuration  $X$ , let  $A(X)$  denote the random variable corresponding to the number of vertices activated by step (i) of the CM dynamics from  $X$ . We provide next the proof of Lemma 11.

**Proof of Lemma 11.** First, both  $\{X_t\}, \{Y_t\}$  perform one independent CM step from the initial configurations  $X_0, Y_0$ . We start by establishing that  $X_1$  and  $Y_1$  preserve the structural properties assumed for  $X_0$  and  $Y_0$ .

By assumption  $\mathcal{R}_1(X_0) = O(n^{4/3})$ , so Hoeffding's inequality implies that the number of activated vertices from  $X_0$  is such that

$$A(X_0) \in I := \left[ n/q - O(n^{2/3}), n/q + O(n^{2/3}) \right]$$

with probability  $\Omega(1)$ . Then, the percolation step is distributed as a

$$G \left( A(X_0), \frac{1 + \lambda A(X_0)^{-1/3}}{A(X_0)} \right)$$

random graph, with  $|\lambda| = O(1)$  with probability  $\Omega(1)$ . Conditioning on this event, from Lemma 18 we obtain that  $I(X_1) = \Omega(n)$  w.h.p. Moreover, from Lemma 21 and Markov's inequality we obtain that  $\mathcal{R}_1(X_1) = O(n^{4/3})$  with probability at least 99/100 and from Lemma 22 that  $\tilde{\mathcal{R}}_\omega(X_1) = O(n^{4/3}\omega(n)^{-1/2})$  also with probability at least 99/100.

We show next that  $X_1$  and  $Y_1$ , in addition to preserving the structural properties of  $X_0$  and  $Y_0$ , also have many connected components with sizes in certain carefully chosen intervals. This fact will be crucial in the design of our coupling. When  $A(X_0) \in I$ , by Lemmas 23 and 24 and a union bound, for all integer  $k \geq 0$  such that  $n^{2/3}\omega(n)^{-2^k} \rightarrow \infty$ ,

## 47:12 The Critical Mean-Field Chayes-Machta Dynamics

$N_k(X_1, \omega) = \Omega(\omega(n)^{3 \cdot 2^{k-1}})$  w.h.p. (Recall, that  $N_k(X_1, \omega)$  denotes the number of connected components of  $X_1$  with sizes in the interval  $\mathcal{I}_k(\omega)$ .) We will also require a bound for the number of components with sizes in the interval

$$J = \left[ \frac{cn^{2/3}}{\omega(n)^6}, \frac{2cn^{2/3}}{\omega(n)^6} \right],$$

where  $c > 0$  is a constant such that  $J$  does not intersect any of the  $\mathcal{I}_k(\omega)$ 's intervals. Let  $W_X$  (resp.,  $W_Y$ ) be the set of components of  $X_1$  (resp.,  $Y_1$ ) with sizes in the interval  $J$ . Lemma 23 then implies that for some positive constants  $\delta_1, \delta_2$  independent of  $n$ ,

$$\Pr \left[ |W_X| \geq \delta_1 n \left( \frac{\omega(n)^6}{cn^{2/3}} \right)^{3/2} \right] \geq 1 - \frac{\delta_2}{n} \left( \frac{cn^{2/3}}{\omega(n)^6} \right)^{3/2} = 1 - O(\omega(n)^{-9}).$$

All the bounds above apply also to the analogous quantities for  $Y_1$  with the same respective probabilities. Therefore, by a union bound, all these properties hold simultaneously for both  $X_1$  and  $Y_1$  with probability  $\Omega(1)$ . We assume that this is indeed the case and proceed to describe the second step of the coupling, in which we shall use each of the established properties for  $X_1$  and  $Y_1$ .

Let  $C_X$  and  $C_Y$  be the set of components in  $X_1$  and  $Y_1$ , respectively, with sizes larger than  $B_\omega$ . (Recall that  $B_\omega = n^{2/3}\omega(n)^{-1}$ , where  $\omega(n) = \log \log \log \log n$ .) Since  $\mathcal{R}_1(X_1) = O(n^{4/3})$ , the total number of components in  $C_X$  is  $O(\omega(n)^2)$ ; moreover, it follows from the Cauchy-Schwarz inequality that the total number of vertices in the components in  $C_X$ , denoted  $\|C_X\|$ , is  $O(n^{2/3}\omega(n))$ ; the same holds for  $C_Y$ .

Without loss of generality, let us assume that  $\|C_X\| \geq \|C_Y\|$ . Let

$$\Gamma = \{C \subset W_Y : \|C_Y \cup C\| \geq \|C_X\|\},$$

and let  $C_{\min} = \arg \min_{C \in \Gamma} \|C_Y \cup C\|$ . In words,  $C_{\min}$  is the smallest subset  $C$  of components of  $W_Y$  so that  $\|C_Y \cup C\| \geq \|C_X\|$ . Since every component in  $W_Y$  has size at least  $cn^{2/3}\omega(n)^{-6}$  and  $|W_Y| = \Omega(\omega(n)^9)$ , the number of vertices in  $W_Y$  is  $\Omega(n^{2/3}\omega(n)^3)$  and so  $\Gamma \neq \emptyset$ . In addition, the number components in  $C_{\min}$  is  $O(\omega(n)^9)$ . Let  $C'_Y = C_Y \cup C_{\min}$  and observe that the number of components in  $C'_Y$  is also  $O(\omega(n)^9)$  and that

$$0 \leq \|C'_Y\| - \|C_X\| \leq 2cn^{2/3}\omega(n)^{-6}.$$

Note that  $\|C_X\| - \|C_Y\|$  may be  $\Omega(n^{2/3}\omega(n))$  (i.e., much larger than  $\|C'_Y\| - \|C_X\|$ ). Hence, if all the components from  $C_Y$  and  $C_X$  were activated, the difference in the number of active vertices could be  $\Omega(n^{2/3}\omega(n))$ . This difference cannot be corrected by our coupling for the activation of the small components. We shall require instead that all the components from  $C'_Y$  and  $C_X$  are activated so that the difference is  $O(n^{2/3}\omega(n)^{-6})$  instead.

We now describe a coupling of the activation sub-step for the second step of the CM dynamics. As mentioned, our goal is to design a coupling in which the same number of vertices are activated from each copy. If indeed  $A(X_1) = A(Y_1)$ , then we can choose an arbitrary bijective map  $\varphi$  between the activated vertices of  $X_1$  and the activated vertices of  $Y_1$  and use  $\varphi$  to couple the percolation sub-step. Specifically, if  $u$  and  $v$  were activated in  $X_1$ , the state of the edges  $\{u, v\}$  in  $X_2$  and  $\{\varphi(u), \varphi(v)\}$  in  $Y_2$  would be the same. This yields a coupling of the percolation sub-step such that  $X_2$  and  $Y_2$  agree on the subgraph update at time 1.

Suppose then that in the second CM step all the components in  $C_X$  and  $C'_Y$  are activated simultaneously. If this is the case, then the difference in the number of activated vertices is  $d \leq 2cn^{2/3}\omega(n)^{-6}$ . We will use a local limit theorem (i.e., Theorem 17) to argue that

there is a coupling of the activation of the remaining components in  $X_1$  and  $Y_1$  such that the total number of active vertices in both copies is the same with probability  $\Omega(1)$ . Since all the components in  $C_X$  and  $C'_Y$  are activated with probability  $\exp(-O(\omega(n)^9))$ , the overall success probability of the coupling will be  $\exp(-O(\omega(n)^9))$ .

Now, let  $x_1, x_2, \dots, x_m$  be the sizes of the components of  $X_1$  that are not in  $C_X$  (in increasing order). Let  $\hat{A}(X_1)$  be the random variable corresponding to the number of active vertices from these components. Observe that  $\hat{A}(X_1)$  is the sum of  $m$  independent random variables, where the  $j$ -th variable in the sum is equal to  $x_j$  with probability  $1/q$ , and it is 0 otherwise. We claim that sequence  $x_1, x_2, \dots, x_m$  satisfies all the conditions in Theorem 17.

First, note that since the number of isolated vertices in  $X_1$  is  $\Omega(n)$ ,  $m = \Theta(n)$  and so  $x_m = O(m^{2/3}\omega(m)^{-1})$ ,  $\sum_{i=1}^m x_i^2 = \hat{R}_\omega(X_1) = O(m^{4/3}\omega(m)^{-1/2})$  and  $x_i = 1$  for all  $i \leq \rho m$ , where  $\rho \in (0, 1)$  is independent of  $m$ . Moreover, since  $N_k(X_1, \omega) \geq \Omega(\omega(n)^{3 \cdot 2^{k-1}})$  for all  $k \geq 1$  such that  $n^{2/3}\omega(n)^{-2^k} \rightarrow \infty$ ,

$$|\{i : x_i \in \mathcal{I}_k(\omega)\}| = \Omega(\omega(m)^{3 \cdot 2^{k-1}}).$$

Since  $N_0(X_1, \omega) = \Omega(\omega(n)^{3/2})$ , we also have

$$\sum_{i=1}^m x_i^2 \geq N_0(X_1, \omega) \cdot \frac{\vartheta^2 n^{4/3}}{4\omega(n)^2} = \Omega(m^{4/3}\omega(m)^{-1/2}).$$

Let  $\mu_X = \mathbb{E}[\hat{A}(X_1)] = q^{-1} \sum_{i=1}^m x_i$  and let

$$\sigma_X^2 = \text{Var}(\hat{A}(X_1)) = q^{-1}(1 - q^{-1}) \sum_{i=1}^m x_i^2 = \Theta(m^{4/3}\omega(m)^{-1/2}).$$

Hence, Theorem 17 implies that  $\Pr[\hat{A}(X_1) = a] = \Omega(\sigma_X^{-1})$  for any  $a \in [\mu_X - \sigma_X, \mu_X + \sigma_X]$ . Similarly, we get  $\Pr[\hat{A}(Y_1) = a] = \Omega(\sigma_Y^{-1})$  for any  $a \in [\mu_Y - \sigma_Y, \mu_Y + \sigma_Y]$ , with  $\hat{A}(Y_1)$ ,  $\mu_Y$  and  $\sigma_Y$  defined analogously for  $Y_1 \setminus C'_Y$ . Note that  $\mu_X - \mu_Y = O(n^{2/3}\omega(n)^{-6})$  and  $\sigma_X, \sigma_Y = \Theta(n^{2/3}\omega(n)^{-1/4})$ . Without loss of generality, suppose  $\sigma_X < \sigma_Y$ . Then for any  $a \in [\mu_X - \sigma_X/2, \mu_Y + \sigma_X/2]$  and  $d = O(n^{2/3}\omega(n)^{-6})$ , we have

$$\min \left\{ \Pr[\hat{A}(X_1) = a], \Pr[\hat{A}(Y_1) = a - d] \right\} = \min \left\{ \Omega(\sigma_X^{-1}), \Omega(\sigma_Y^{-1}) \right\} = \Omega(\sigma_Y^{-1}).$$

Hence, there exists a coupling  $\mathbb{P}$  of  $\hat{A}(X_1)$  and  $\hat{A}(Y_1)$  so that  $\mathbb{P}[\hat{A}(X_1) = a, \hat{A}(Y_1) = a - d] = \Omega(\sigma_Y^{-1})$  for all  $a \in [\mu_X - \sigma_X/2, \mu_Y + \sigma_X/2]$ . Therefore, there is a coupling of  $\hat{A}(X_1)$  and  $\hat{A}(Y_1)$  such that

$$\Pr[\hat{A}(X_1) - \hat{A}(Y_1) = d] = \Omega(\sigma_X/\sigma_Y) = \Omega(1).$$

Putting all these together, we deduce that  $A(X_1) = A(Y_1)$  with probability  $e^{-O(\omega(n)^9)}$ . If this is the case, the edge re-sampling step is coupled bijectively (as described above) so that  $\mathcal{S}_\omega(X_2) = \mathcal{S}_\omega(Y_2)$ .

It remains for us to guarantee the additional desired structural properties of  $X_2$  and  $Y_2$ , which follow straightforwardly from the random graph estimates we stated at the beginning of the section. First note that by Hoeffding's inequality, with probability  $\Omega(1)$ ,

$$\left| A(X_1) - \frac{n}{q} - \frac{(q-1)|C_X|}{q} \right| = O(n^{2/3}).$$

Hence, in the percolation sub-step the active subgraph is replaced by

$$F \sim G \left( A(X_1), \frac{1 + \lambda A(X_1)^{-1/3}}{A(X_1)} \right),$$

where  $|\lambda| = O(\omega(n))$  with probability  $\Omega(1)$  since  $|C_X| = O(n^{2/3}\omega(n))$ . Conditioning on this event, since the components of  $F$  contribute to both  $X_2$  and  $Y_2$ , Lemma 24 implies that w.h.p.  $\hat{N}_k(2, \omega(n)) = \Omega(\omega(n)^{3 \cdot 2^{k-1}})$  for all  $k \geq 1$  such that  $n^{2/3}\omega(n)^{-2^k} \rightarrow \infty$ . Moreover, from Lemma 18 we obtain that  $I(X_2) = \Omega(n)$  w.h.p. From Lemma 19 and Markov's inequality, we obtain that  $\mathcal{R}_2(X_2) = O(n^{4/3})$  with probability at least 99/100 and from Lemma 22 that  $\tilde{\mathcal{R}}_\omega(X_2) = O(n^{4/3}\omega(n)^{-1/2})$  also with probability at least 99/100. All these bounds apply also to the analogous quantities for  $Y_2$  with the same respective probabilities.

Finally, we derive the bound for  $L_1(X_2)$  and  $L_1(Y_2)$ . First, notice  $L_1(F)$  is stochastically dominated by  $L_1(F')$ , where

$$F' \sim G\left(A(X_1), \frac{1 + |\lambda|A(X_1)^{-1/3}}{A(X_1)}\right).$$

Under the assumption that  $|\lambda| = O(\omega(n))$ , if  $|\lambda| \rightarrow \infty$ , then Lemma 20 implies that  $L_1(F') = O(|\lambda|A(X_1)^{2/3}) = O(n^{2/3}\omega(n))$  w.h.p.; otherwise,  $|\lambda| = O(1)$  and by Lemma 21 and Markov's inequality,  $L_1(F') = O(n^{2/3})$  with probability at least 99/100. Thus,  $L_1(F) = O(n^{2/3}\omega(n))$  with probability at least 99/100. We also know that the largest inactivated component in  $X_1$  has size less than  $n^{2/3}\omega(n)^{-1}$ , so  $L_1(X_2) = O(n^{2/3}\omega(n))$  with probability at least 99/100. The same holds for  $Y_2$ . Therefore, by a union bound, all these properties hold simultaneously for both  $X_2$  and  $Y_2$  with probability  $\Omega(1)$ , as claimed.  $\blacktriangleleft$

---

## References

- 1 V. Beffara and H. Duminil-Copin. The self-dual point of the two-dimensional random-cluster model is critical for  $q \geq 1$ . *Probability Theory and Related Fields*, 153:511–542, 2012.
- 2 A. Blanca. *Random-cluster dynamics*. PhD thesis, UC Berkeley, 2016.
- 3 A. Blanca and R. Gheissari. Random-cluster dynamics on random regular graphs in tree uniqueness. *Communications in Mathematical Physics*, 2021.
- 4 A. Blanca, R. Gheissari, and E. Vigoda. Random-cluster dynamics in  $\mathbb{Z}^2$ : rapid mixing with general boundary conditions. *Annals of Applied Probability*, 30(1):418–459, 2020.
- 5 A. Blanca and A. Sinclair. Dynamics for the mean-field random-cluster model. *Proceedings of the 19th International Workshop on Randomization and Computation (RANDOM)*, pages 528–543, 2015.
- 6 A. Blanca and A. Sinclair. Random-Cluster Dynamics in  $\mathbb{Z}^2$ . *Probability Theory and Related Fields*, 168:821–847, 2017.
- 7 Antonio Blanca, Alistair Sinclair, and Xusheng Zhang. The critical mean-field chayes-machta dynamics, 2021. [arXiv:2102.03004](https://arxiv.org/abs/2102.03004).
- 8 B. Bollobás, G.R. Grimmett, and S. Janson. The random-cluster model on the complete graph. *Probability Theory and Related Fields*, 104(3):283–317, 1996.
- 9 L. Chayes and J. Machta. Graphical representations and cluster algorithms II. *Physica A*, 254:477–516, 1998.
- 10 H. Duminil-Copin, M. Gagnebin, M. Harel, I. Manolescu, and V. Tassion. Discontinuity of the phase transition for the planar random-cluster and Potts models with  $q > 4$ . *Annales de l'ENS*, 2016. To Appear.
- 11 H. Duminil-Copin, V. Sidoravicius, and V. Tassion. Continuity of the Phase Transition for Planar Random-Cluster and Potts Models with  $1 \leq q \leq 4$ . *Communications in Mathematical Physics*, 349(1):47–107, 2017.
- 12 C.M. Fortuin and P.W. Kasteleyn. On the random-cluster model I. Introduction and relation to other models. *Physica*, 57(4):536–564, 1972.
- 13 A. Galanis, D. Štefankovič, and E. Vigoda. Swendsen-Wang algorithm on the mean-field Potts model. *Proceedings of the 19th International Workshop on Randomization and Computation (RANDOM)*, pages 815–828, 2015.



- 14 T. Garoni. Personal communication, 2015.
- 15 R. Gheissari and E. Lubetzky. Quasi-polynomial mixing of critical two-dimensional random cluster models. *Random Structures & Algorithms*, 56(2):517–556, 2020.
- 16 R. Gheissari, E. Lubetzky, and Y. Peres. Exponentially slow mixing in the mean-field Swendsen-Wang dynamics. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1981–1988. SIAM, 2018.
- 17 H. Guo and M. Jerrum. Random cluster dynamics for the Ising model is rapidly mixing. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1818–1827. SIAM, 2017.
- 18 D.A. Levin and Y. Peres. *Markov Chains and Mixing Times*. MBK. American Mathematical Society, 2017. URL: <https://books.google.com/books?id=f208DwAAQBAJ>.
- 19 Y. Long, A. Nachmias, W. Ning, and Y. Peres. A power law of order  $1/4$  for critical mean-field Swendsen-Wang dynamics. *Memoirs of the American Mathematical Society*, 232(1092), 2011.
- 20 Malwina Luczak and Tomasz Luczak. The phase transition in the cluster-scaled model of a random graph. *Random Structures & Algorithms*, 28(2):215–246, 2006.
- 21 A.B. Mukhin. Local limit theorems for lattice random variables. *Theory of Probability & Its Applications*, 36(4):698–713, 1992.
- 22 R.H. Swendsen and J.S. Wang. Nonuniversal critical dynamics in Monte Carlo simulations. *Physical Review Letters*, 58:86–88, 1987.
- 23 M. Ullrich. Swendsen-Wang is faster than single-bond dynamics. *SIAM Journal on Discrete Mathematics*, 28(1):37–48, 2014.



# On the Robust Communication Complexity of Bipartite Matching

Sepehr Assadi  

Rutgers University, Piscataway, NJ, USA

Soheil Behnezhad  

University of Maryland, College Park, MD, USA

---

## Abstract

---

We study the *robust* – à la Chakrabarti, Cormode, and McGregor [STOC’08] – communication complexity of the maximum bipartite matching problem. The edges of an *adversarially* chosen  $n$ -vertex bipartite graph  $G$  are partitioned *randomly* between Alice and Bob. Alice has to send a single message to Bob, using which Bob has to output an approximate maximum matching of  $G$ . We are particularly interested in understanding the best approximation ratio possible by protocols that use a near-optimal message size of  $n \cdot \text{polylog}(n)$ .

The communication complexity of bipartite matching in this setting under an *adversarial* partitioning is well-understood. In their beautiful paper, Goel, Kapralov, and Khanna [SODA’12] gave a  $2/3$ -approximate protocol with  $O(n)$  communication and showed that this approximation is tight unless we allow more than a near-linear communication. The complexity of the robust version, i.e., with a *random* partitioning of the edges, however remains wide open. The best known protocol, implied by a very recent random-order streaming algorithm of the authors [ICALP’21], uses  $O(n \log n)$  communication to obtain a  $(2/3 + \varepsilon_0)$ -approximation for a constant  $\varepsilon_0 \sim 10^{-14}$ . The best known lower bound, on the other hand, leaves open the possibility of all the way up to even a  $(1 - \varepsilon)$ -approximation using near-linear communication for constant  $\varepsilon > 0$ .

In this work, we give a new protocol with a significantly better approximation. Particularly, our protocol achieves a 0.716 expected approximation using  $O(n)$  communication. This protocol is based on a new notion of *distribution-dependent sparsifiers* which give a natural way of sparsifying graphs sampled from a *known* distribution. We then show how to lift the assumption on knowing the graph’s distribution via minimax theorems. We believe this is a particularly powerful method of designing communication protocols and might find further applications.

**2012 ACM Subject Classification** Theory of computation → Graph algorithms analysis; Theory of computation → Streaming, sublinear and near linear time algorithms

**Keywords and phrases** Maximum Matching, Communication Complexity, Random-Order Streaming

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.48

**Category** RANDOM

**Funding** *Sepehr Assadi*: Research supported in part by the NSF CAREER award CCF-2047061 and a gift from Google Research.

*Soheil Behnezhad*: Research supported by Google PhD Fellowship.

## 1 Introduction

Consider the following communication game. We have an  $n$ -vertex bipartite graph  $G = (L, R, E)$  whose edges are partitioned into  $E^A$  and  $E^B$  given to Alice and Bob, respectively (both players know  $L$  and  $R$ ). The goal is to compute an approximate maximum matching of  $G$  by Alice sending a single message to Bob and Bob outputting the solution. What is the tradeoff between the size of Alice’s message and the approximation ratio of the output matching, or in other words, the *one-way communication complexity* of bipartite matching?



© Sepehr Assadi and Soheil Behnezhad;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 48; pp. 48:1–48:17



Leibniz International Proceedings in Informatics

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

It is known that  $\Omega(n^2)$  communication is necessary for finding a maximum matching [23] and this is clearly sufficient by Alice sending her entire input. But the situation is more interesting for approximate protocols. A  $1/2$ -approximation with  $O(n)$  communication can be obtained by Alice sending a maximum matching of her input to Bob and  $\Omega(n)$  communication is clearly needed for any constant factor approximation. More interestingly, Goel, Kapralov, and Khanna [25] showed that  $O(n)$  communication even suffices to obtain a  $2/3$ -approximation and that this is the “right” answer: any better approximation requires  $n^{1+\Omega(1/\log \log n)} \gg n \cdot \text{polylog}(n)$  communication.

In this paper, we study a *robust* variant of this problem – à la Chakrabarti, Cormode, and McGregor [18] – wherein the graph  $G$  is still chosen adversarially, but now its edges are instead *randomly partitioned* between the two players, i.e., each edge is independently given to one of the players chosen uniformly at random. This model of random partitioning was introduced in [18] to go beyond the “doubly worst case” analysis of communication games, namely, adversarial inputs and adversarial partitions, and sheds more light into the source of hardness: whether it is due to a pathological partitioning of inputs or rather it holds for most input partitions.

Our main result is a substantial improvement over the  $2/3$ -approximations for adversarial partitions [25] under this random partition model.

► **Result 1** (Formalized in Theorem 17). *There is a randomized one-way protocol with  $O(n)$  communication that achieves an expected  $0.716$ -approximation for the bipartite matching problem under a random partitioning of the input edges between Alice and Bob.*

Prior to our work, the best known approximation ratio achievable for this problem was  $(2/3 + \varepsilon_0)$  for some  $\varepsilon_0 \sim 10^{-14}$ , obtained via the very recent random-order streaming algorithm of the same authors of this paper in [2].

We note that our protocol in this result can be considered *non-explicit*: we show the existence of the protocol rather than explicitly designing the protocol itself (see Section 1.1 for details). Alternatively, the protocol can be found also via a brute-force search in doubly exponential time.

## 1.1 Our Techniques

The  $2/3$ -approximation protocol of [25] (and follow-ups in [3] that simplified it or [37] that extended it to the online batch-arrival model) are all based on finding a suitable subgraph of Alice’s input that preserves large matchings approximately, namely, a matching sparsifier (similar-in-spirit to cut sparsifiers [13, 14]). These subgraphs are defined through a series of graph-theoretic constraints: a novel decomposition into expanding sets (matching skeleton) in [25, 37] (see also [33]), and edge-degree bounded subgraphs in [3] (defined first in [16, 17] for dynamic graph algorithms). We take an entirely different approach in this paper.

The first step of our approach is a way of introducing distributional assumptions about the input, while still solving the problem in its full generality. In particular, in this step, we reduce the general problem to the case that the input graph  $G$  is sampled from some arbitrary but known distribution  $\mathcal{G}$  of graphs. We achieve this via combining several relatively standard ideas specific to the matching problem with an application of Yao’s minimax principle [41] (the so-called “hard direction” of this principle; see Section 5). This is the main *conceptual* step of our approach.

The second step is to design a protocol for the problem assuming that it is additionally given an input distribution  $\mathcal{G}$  of the input graph. We achieve this through a new notion of “distribution-dependent sparsifiers” described below. This is our main *technical* step.

**Distribution-dependent sparsifiers.** Distribution-dependent sparsifiers can be used whenever we know a distribution  $\mathcal{G}$  for inputs of Alice and Bob. In particular, the knowledge of  $\mathcal{G}$  allows us to determine the “importance” of each edge in Alice’s input  $E^A$ : this is the probability that this particular edge belongs to a fixed maximum matching (say, the lexicographically-first maximum matching) of a graph sampled from  $\mathcal{G} \mid E^A$ , i.e., input graphs after conditioning on Alice’s input. The main part of our argument is to show that these importances can be used to sparsify the Alice’s graph to  $O(n)$  edges, while allowing Bob to find a large matching of the entire graph in expectation.

For our analysis of these sparsifiers, we need to show that the edges  $T$  communicated by Alice and the edges  $E^B$  given to Bob combined, include a large matching. We do so by constructing a large fractional matching  $\mathbf{x}$  on the edges in  $T \cup E^B$ . Our construction of  $\mathbf{x}$  is online, in the sense that we decide on the value of  $\mathbf{x}$  induced on  $T$  before sampling Bob’s subgraph  $E^B$  from  $\mathcal{G} \mid E^A$ . Thanks to the fact that Alice picks the edges of  $T$  according to their importance, we can construct  $\mathbf{x}$  on  $T$  such that the fractional value around each vertex  $v$  is equal to the probability that  $v$  is matched in the optimum solution via an edge given to Alice. This is particularly useful because it implies that (i) the size of  $\mathbf{x}$  induced on  $T$  equals the expected number of edges of Alice in an optimum matching, and that (ii) if a vertex is unlikely to be matched via an edge of Alice in the optimum solution, then the fractional matching  $\mathbf{x}$  induced on  $T$  does not occupy this vertex by much, leaving room for the rest of the edges in  $E^B$  to use it.

As a warm-up in Section 3, we show how the ideas above lead to a very simple 0.656-approximate protocol under an *adversarial* partitioning of the input. This is only slightly worse than the optimal  $2/3$ -approximation for this problem, but more importantly, this warm-up conveys the key intuitions behind distribution-dependent sparsifiers and how they are extremely useful for matching in the communication setting. The protocol for our 0.716-approximation in Section 4 for the robust communication model is very similar, but its analysis is more involved and in particular is based on a careful examination of edge importance distributions under a random partitioning.

## 1.2 Further Aspects of Our Results

**Random-order streams.** The one-way communication model in general is strongly motivated by applications to *graph streaming algorithms* [23]. The *robust* communication model, in the same vein, is closely related to *random-order* streaming algorithms wherein the edges of the graph arrive in a random order. In particular, lower bounds in the (robust) communication model directly imply space lower bounds in the (random-order) streaming model [18] and upper bounds are sources of inspiration and stepping stones for designing streaming algorithms (see, e.g., [1, 25, 31] for instances of communication protocols that were turned into streaming algorithms in the context of the matching problem).

Maximum matchings have been studied extensively in random-order streams [1, 2, 15, 22, 24, 34, 35], leading to a  $2/3$ -approximation algorithm of [15] that hit a natural barrier for this problem, and the recent algorithm of [2] that improved this approximation to strictly more than  $2/3$  (for a tiny constant improvement). We hope our ideas in this paper can lead to a significantly-better-than- $2/3$  approximation in random-order streams.

We remark that [2] proves the following robust communication lower bound (and thus a random-order streaming lower bound also): any  $(1 - \Theta(1/\log n))$ -approximation to maximum matching in the robust communication model requires  $n^{1+\Omega(1/\log \log n)} \gg n \cdot \text{polylog}(n)$  communication. Closing the gap between our upper bound and the lower bound of [2] remains a fascinating open question. Finally, we note that our improved protocol also has the following

message: either one should be able to achieve a significantly-better-than- $2/3$  approximation (say, a 0.716-approximation) in random-order streams, or any lower bound technique for proving impossibility of such a result should deviate from the standard two-party communication complexity lower bound approach.

**Non-explicit protocols.** As remarked earlier, the protocol in our main result can be considered non-explicit. Alternatively, the players may need to first spend a doubly-exponential time to find the protocol, and only then they can use it to solve the problem (this is due to the arguments in the first step of our approach and in particular using Yao’s minimax principle). From an algorithmic perspective, this is a weakness of our particular method of protocol design. On the other hand, we find our method particularly strong and insightful from a communication complexity point of view as we shall elaborate below.

Firstly, communication complexity is a non-uniform model of computation with players of unbounded computational power, and the only resource of interest is the communication cost of protocols. In this regard, our protocol uses the “full power” of this model to achieve its approximation ratio using the optimal  $O(n)$  communication.

Secondly, and more importantly, there is a general gap in the study of communication complexity of graph problems: almost all protocols designed in the literature are based on algorithmic tools that are tailored to time-efficient protocols, while all known lower bounds are information-theoretic and hold even for protocols with computationally unbounded players. Can this inconsistent treatment be a contributing factor to the substantial gaps between known upper and lower bounds for various problems, including the robust communication complexity of bipartite matching? If so, then our approach in this paper allows us to explore a wider set of natural protocols for the problems at hand and move toward achieving tight(er) bounds on communication complexity. This will in turn suggest that purely information-theoretic complexity lower bounds cannot prove “strong enough” lower bounds for computationally-efficient algorithms as well. We leave the question of proving communication lower bounds for computationally-efficient protocols, which is the dual approach to our work in this paper, as a very interesting research direction for future work.

### 1.3 Further Related Work

The communication complexity of bipartite matching has been extensively studied from various angles including exact protocols [21, 27, 30], non-deterministic protocols [40], protocols with limited rounds of communication [3, 5, 23, 25, 26], or multi-party protocols [6, 26, 29, 32, 33] to name a few (this is by no means a comprehensive summary of previous results).

The one-way communication complexity of matching, in particular, is directly related to streaming algorithms. In fact, a key motivation in the work of Goel, Kapralov, and Khanna [25] was to determine whether there is a better-than- $1/2$ -approximation algorithm for the matching problem in the streaming model that uses  $\tilde{O}(n)$  space, a longstanding open problem in this area. The lower bound in [25] implies that there is no semi-streaming algorithm with approximation ratio better than  $2/3$ ; this lower bound was later improved by Kapralov to a  $1 - 1/e \approx 0.63$  in [31] and to  $\frac{1}{1+\ln 2} \approx 0.59$  in [32]. Additionally, the communication protocols in [25] were also generalized in the same paper to achieve a  $(1 - 1/e)$ -approximation in vertex-arrival streams.

Finally, we should point out that the work of [25] on the one-way communication complexity of bipartite matching has been quite instrumental and paved the path for various follow-ups including optimal algorithms for vertex-arrival streaming model [25, 31], state-of-the-art lower bounds for streaming matching in both insertion-only streams [31, 32] and

dynamic streams [6, 20], online batch-arrival algorithms [37] and fault-tolerant algorithms [3] for maximum matching, stochastic matching problem [3, 4, 10, 11], and using RS graphs for proving communication lower bounds for other problems such as matrix rank [5, 9], independent sets [7, 19], and reachability [8].

## 2 Preliminaries

**Notation.** For any graph  $G$ , we use  $n$  to denote the number of vertices and  $\mu(G)$  to denote the maximum matching size. A fractional matching  $\mathbf{x}$  on a graph  $G$  is an assignment of values  $x_e$  to the edges  $e$  of  $G$  such that  $x_e \geq 0$  for all edges  $e$  and for each vertex  $v$ ,  $x_v := \sum_{e \ni v} x_e \leq 1$ . We use  $|\mathbf{x}|$  as a shorthand for  $\sum_e x_e$  which is the size of fractional matching  $\mathbf{x}$ .

The following standard fact implies that to show a bipartite graph has an integral matching of size  $\mu$ , it suffices to construct a fractional matching of size  $\mu$  on it.

► **Fact 1.** *Let  $\mathbf{x}$  be a fractional matching of a bipartite graph  $G$ . Then  $G$  has an integral matching of size at least  $|\mathbf{x}|$ .*

**Communication model.** We study the standard two-party communication model of Yao [42] and in particular in the one-way model (see the excellent textbook by Kushilevitz and Nisan [36]). The only slight derivation is that we focus on randomly partitioned inputs, wherein the input graph is still chosen adversarially, but every edge in the graph is sent to one of the players chosen independently and uniformly at random. To our knowledge, this model was first introduced by Chakrabarti, Cormode, and McGregor in [18].

Unless specified otherwise, we assume that protocols are randomized and both players have access to the same *shared* source of randomness, referred to as *public coins*; however, one can always use Newman's theorem [39] to turn public coins into private coins with a negligible overhead. The communication cost of any protocol in this model is the worst-case length of the communicated messages; to be consistent with prior work on this problem in [3, 25, 31, 33], we measure the length of messages in  $\Theta(\log n)$ -bit words as opposed to the more standard convention of bits. Finally, we note that the main resource of interest in this model is the communication and the players are assumed to be *computationally unbounded*.

## 3 Warm-up: A 0.656-Approximation Under Adversarial Partitions

In this section, we describe a one-way protocol for the bipartite matching problem and prove that it achieves an approximation factor of  $(4\sqrt{2}-5) \approx 0.656$  under an *adversarial* partitioning of the edges. While this protocol is slightly worse than the optimal  $2/3$ -approximate protocols in [3, 25] and its analysis shares some similarity with [12], we believe it is still instructive as it acts as a gentle introduction to the ideas used in our main protocol of Section 4.

A key technique introduced in this work is the notion of **distribution-dependent sparsifiers**. For now, let us assume that there is a *known* distribution  $\mathcal{G}$  from which the inputs  $E^A$  and  $E^B$  of Alice and Bob are sampled. Now, suppose Alice has received  $E^A$  as input and plans to send a message to Bob. In order to do this, Alice considers the distribution of inputs conditioned on her input, i.e.,  $\mathcal{G} \mid E^A$ . The message sent by Alice is then a subgraph of her input (the sparsifier), wherein each edge is included depending on the probability that this edge belongs to a fixed maximum matching of a graph sampled from  $\mathcal{G} \mid E^A$ .

Finally, we can lift the assumption on the knowledge of  $\mathcal{G}$  using minimax theorems: distribution-dependent sparsifiers give us a deterministic protocol for *each* distribution of

inputs with approximation ratio at least  $\alpha$  for every distribution; thus, there should also exist a *single* randomized protocol that achieves the same  $\alpha$ -approximation for *all* inputs. See Section 5 for this argument<sup>1</sup>.

### 3.1 The Protocol

We now describe our new distribution-dependent protocol. For the rest of this proof, we assume that Alice and Bob are given the distribution of inputs  $\mathcal{G}$ . For each edge  $e \in E^A$ , we define:

$$a_e := \Pr_{G \sim \mathcal{G}}[e \in \text{MM}(G) \mid E^A], \quad (1)$$

where function  $\text{MM}(\cdot)$  deterministically returns a fixed maximum matching of its input (for instance, the lexicographically-first one, or the one returned by the Hopcroft-Karp algorithm [28]). In words,  $a_e$  is the probability that  $e$  belongs to a fixed maximum matching of a graph  $G$  sampled from  $\mathcal{G}$  conditioned on the input  $E^A$  given to Alice. We are going to treat  $a_e$  as the “importance” of edge  $e$  in  $E^A$ . Observe that since Alice is aware of  $\mathcal{G}$ , she can compute  $a_e$  for each edge  $e \in E^A$ .

**Fractional matching interpretation.** Consider the vector  $\mathbf{a} := \{a_e\}_{e \in E^A}$ . We claim that  $\mathbf{a}$  is a feasible *fractional* matching of  $E^A$ : (i) for every edge  $e \in E^A$ , we have  $a_e \geq 0$  as  $a_e$  is a probability, and (ii) for all vertices  $v$ ,  $a_v := \sum_{e \ni v} a_e \leq 1$  as it can be confirmed that:

$$a_v = \Pr_{G \sim \mathcal{G}}[v \text{ matched in MM}(G) \text{ by edges of } E^A \mid E^A]. \quad (2)$$

This view of  $\mathbf{a}$  presents a natural way of sparsifying Alice’s input. Basically, we can sparsify the support of  $\mathbf{a}$  via the standard cycle-canceling method (see Lemma 2 below) so that instead of (possibly up to)  $\Omega(n^2)$  edges, it will only have  $O(n)$  edges while still preserving the fractional matching of each *vertex* (but not necessarily the edges). This allows us to obtain another fractional matching  $\mathbf{a}'$  that preserves key properties of  $\mathbf{a}$  but is much sparser and thus Alice can simply send this fractional matching directly to Bob.

► **Lemma 2** (Cycle-Canceling Lemma – Folklore). *Let  $\mathbf{f}$  be any fractional matching of  $E^A$ . There is another fractional matching  $\mathbf{f}'$  on  $E^A$  such that:*

- **Sparsification property:** *There are at most  $n - 1$  edges  $e$  in  $E^A$  with  $f'_e > 0$ .*
- **Preserving marginals and size:** *For every vertex  $v$ ,  $f'_v = f_v$ , also implying  $|\mathbf{f}'| = |\mathbf{f}|$ .*

**Proof.** Iteratively take a cycle in the support of  $\mathbf{f}$ , then alternately decrease and increase the value of edges in a way that the minimum value edge gets value zero. Since all cycles are even-length, the fractional matching around each vertices remains unchanged throughout the process. Once there are no more cycles, the remaining fractional matching is a forest with at most  $n - 1$  edges. ◀

We can now formalize the protocol as follows.

By Lemma 2 this protocol requires  $O(n)$  communication (in fact, only  $n - 1$  edges). Thus, it only remains to analyze the approximation ratio of Algorithm 1 in the following.

<sup>1</sup> There is an important subtlety here: distribution-dependent sparsifiers approximate the matching *in expectation* over the choice of graphs in the distribution; in other words, the output matching is close to the optimal matching in expectation. To apply Yao’s minimax principle however, one needs an *instance-wise* approximation for the input graph. Thus, the argument in this part is not a black-box application of minimax theorems.



■ **Algorithm 1** A simple distribution-dependent sparsifier protocol.

- 
- (i) Given edges  $E^A$  as input to Alice, she computes the vector  $\mathbf{a} = \{a_e\}_{e \in E^A}$  using Eq (1); as discussed above,  $\mathbf{a}$  is a valid fractional matching of  $E^A$ .
  - (ii) Alice obtains fractional matching  $\mathbf{a}'$  by running cycle canceling on  $\mathbf{a}$  (Lemma 2) and then sends the edges  $T$  in the support of  $\mathbf{a}'$  to Bob.
  - (iii) Bob, given message  $T$  from Alice and input  $E^B$ , returns a maximum matching of  $E^B \cup T$ .
- 

► **Proposition 3.** *For any input distribution  $\mathcal{G}$  on adversarial partitions, Algorithm 1 achieves a  $4\sqrt{2} - 5 \approx 0.6568$  approximation in expectation and uses  $O(n)$  communication.*

We prove this proposition in the next section.

### 3.2 The Analysis: Proof of Proposition 3

Recall that  $T$  is the support of the fractional matching  $\mathbf{a}'$  that Alice sends to Bob. For the analysis, we only need to show that  $T \cup E^B$  includes a large fractional matching (by Fact 1). To do so, we construct a fractional matching  $\mathbf{x}$  supported on  $T \cup E^B$  in the following way:

$$x_e = \begin{cases} a'_e & \text{if } e \in T, \\ 1 - \max\{a'_u, a'_v\} & \text{if } e = (u, v) \in \text{MM}(G) \cap E^B. \end{cases} \quad (3)$$

Intuitively, once the subgraph  $E^A$  is given to Alice, we immediately commit her fractional matching  $\mathbf{a}'$  to the final fractional matching  $\mathbf{x}$ . Then, after the subgraph  $E^B$  of Bob is revealed, on any edge  $e = (u, v) \in \text{MM}(G) \cap E^B$ , we set  $x_e = 1 - \max\{a'_u, a'_v\}$  which is the largest possible fractional value that does not violate its endpoints' fractional matching constraints due to  $\mathbf{a}'$ .

In what follows, for any choice of  $E^A$ , we lower-bound the ratio  $\mathbf{E}[\|\mathbf{x}\| \mid E^A]$  to  $\mathbf{E}[\mu(G) \mid E^A]$  which implies the approximation ratio of our protocol. We emphasize that  $\mathbf{x}$  is only constructed for the analysis and in the protocol, Bob simply returns a maximum matching of  $T \cup E^B$ .

Consider a maximum matching edge  $uv$  which belongs to the input of Bob, i.e.  $uv \in \text{MM}(G) \cap E^B$  and suppose that  $a'_v > a'_u$ . Observe that in  $\mathbf{x}$ , we set  $x_{uv} = 1 - \max\{a'_u, a'_v\} = 1 - a'_v$ . In this case, we say that vertex  $v$  is *responsible* for edge  $uv$ . Based on this, we define<sup>2</sup>:

$$b_v := \Pr[\exists uv \in \text{MM}(G) \cap E^B \text{ such that } a'_v > a'_u \mid E^A], \quad (4)$$

i.e.,  $b_v$  is the probability that  $v$  is responsible for some edge. We first bound the size of  $\text{MM}(G)$  based on the values  $a_v$  and  $b_v$ .

▷ **Claim 4.**  $\mathbf{E}[\mu(G) \mid E^A] = \sum_v \frac{1}{2} a_v + b_v$ .

*Proof.* We claim that,

- (i)  $\mathbf{E}[\|\text{MM}(G) \cap E^A\| \mid E^A] = \frac{1}{2} \sum_v a_v$ : by the definition of  $a_v$  in Eq (2) and the fact that the number of vertices matched in any matching is twice the size of the matching;
- (ii)  $\mathbf{E}[\|\text{MM}(G) \cap E^B\| \mid E^A] = \sum_v b_v$ : since each responsible vertex has an edge in  $\text{MM}(G) \cap E^B$  and for each such edge, exactly one of its neighbors is responsible.

The claim now follows by adding up the two equations above.  $\triangleleft$

<sup>2</sup> In case of ties, we break ties arbitrarily so that only one vertex is responsible for an edge.

We now also bound the size of  $\mathbf{x}$  based on  $a_v$  and  $b_v$  values.

▷ **Claim 5.** For any vertex  $v$ , define  $g_v := \frac{1}{2}a_v + (1 - a_v)b_v$ . Then,  $\mathbf{E}[|\mathbf{x}| \mid E^A] = \sum_v g_v$ .

*Proof.* By definition,

$$\sum_v g_v = \sum_v \left( \frac{1}{2}a_v + (1 - a_v)b_v \right) = |\mathbf{a}| + \sum_v (1 - a_v)b_v.$$

The first term  $|\mathbf{a}|$  in the sum corresponds to the part of fractional matching  $\mathbf{x}$  constructed on the edges  $T$  sent by Alice, using the fractional matching  $\mathbf{a}'$ , where we have  $|\mathbf{a}'| = |\mathbf{a}|$  by Lemma 2.

It thus remains to prove that contribution of  $\mathbf{x}$  on the remaining edges (i.e. those given to Bob in  $\text{MM}(G) \cap E^B$ ), has expected size  $\sum_v (1 - a_v)b_v$ . This follows from the fact that each vertex  $v$  is responsible for some edge  $uv \in \text{MM}(G) \cap E^B$  with probability  $b_v$  by Eq (4), and that when this happens, we set  $x_{uv} = 1 - a'_v = 1 - a_v$  (as  $a'_v = a_v$  for all  $v$  by Lemma 2). Noting that exactly one of the endpoints of each edge  $e \in \text{MM}(G) \cap E^B$  is responsible for it, we get that  $\mathbf{x}$  on the set of edges given to Bob has expected size exactly  $\sum_v (1 - a_v)b_v$ , completing the proof. ◁

Claims 4 and 5 imply that the approximation factor of Algorithm 1 is

$$\frac{\mathbf{E}[|\mathbf{x}| \mid E^A]}{\mathbf{E}[\mu(G) \mid E^A]} = \frac{\sum_v g_v}{\sum_v \frac{1}{2}a_v + b_v}. \quad (5)$$

To lower bound this ratio, we use Fact 6 below.<sup>3</sup>

► **Fact 6.** For all  $a, b \geq 0$  satisfying  $a + b \leq 1$ , it holds that  $\frac{0.5a + (1-a)b}{0.5a + b} \geq 4\sqrt{2} - 5$ .

Now to use Fact 6 to lower bound the approximation factor, first recall that for each vertex  $v$ , by the definition of  $a_v$  and  $b_v$  in Eq (2) and (4), we have,

$$\begin{aligned} a_v + b_v &\leq \Pr[v \text{ is matched in } \text{MM}(G) \cap E^A \mid E^A] + \Pr[v \text{ is matched in } \text{MM}(G) \cap E^B \mid E^A] \\ &= \Pr[v \text{ is matched in } \text{MM}(G) \mid E^A] \leq 1. \end{aligned}$$

Thus, we can apply Fact 6 and get that for each vertex  $v$ ,  $\frac{g_v}{0.5a_v + b_v} = \frac{0.5a_v + (1-a_v)b_v}{0.5a_v + b_v} \geq 4\sqrt{2} - 5$ . This implies that

$$\frac{\mathbf{E}[|\mathbf{x}| \mid E^A]}{\mathbf{E}[\mu(G) \mid E^A]} \stackrel{(5)}{=} \frac{\sum_v g_v}{\sum_v \frac{1}{2}a_v + b_v} \geq \frac{\sum_v (4\sqrt{2} - 5)(\frac{1}{2}a_v + b_v)}{\sum_v \frac{1}{2}a_v + b_v} = 4\sqrt{2} - 5,$$

which proves Proposition 3 that Algorithm 1 achieves a  $(4\sqrt{2} - 5)$ -approximation.

► **Remark 7.** There are distributions for which the inequality above is actually equality. That is, we have  $\mathbf{E}[|\mathbf{x}| \mid E^A] = (4\sqrt{2} - 5)\mathbf{E}[\mu(G) \mid E^A]$ . Therefore, this analysis based on the construction of fractional matching  $\mathbf{x}$  cannot show an approximation factor better than  $(4\sqrt{2} - 5)$  for this protocol.

That being said, by “scaling” the fractional matching  $\mathbf{a}$  of Alice before sparsifying it, one can in fact achieve a  $(2/3)$ -approximation which is optimal for adversarial partitions with  $O(n)$  communication [25]. We use this scaling idea in our protocol in Section 4.

<sup>3</sup> Mathematica can verify Fact 6; see e.g., this page on WolframAlpha.

## 4 A 0.7167-Approximation Under Random Partitions

In this section, we show that a properly “scaled” variant of our distribution-dependent sparsifier of Section 3 – formalized as Algorithm 2 – achieves a significantly better approximation factor of 0.7167 in expectation, under a *random* partitioning of the edges between the players.

► **Theorem 8.** *There is a deterministic one-way protocol that given any arbitrary but known distribution  $\mathcal{G}$  of input graphs, and a graph  $G$  sampled from  $\mathcal{G}$  partitioned randomly between Alice and Bob, outputs a matching  $M(G)$  in  $G$  such that  $\mathbf{E} |M(G)| \geq 0.7167 \cdot \mathbf{E}[\mu(G)]$ . The protocol requires communicating at most  $n - 1$  edges from Alice to Bob.*

### 4.1 The Protocol

Recall from our Algorithm 1 in Section 3 that Alice, given her subgraph  $E^A$ , first defines a fractional matching  $\mathbf{a}$  on  $E^A$  where for each edge  $e \in E^A$ ,  $a_e = \Pr_{G \sim \mathcal{G}}[e \in \text{MM}(G) \mid E^A]$ , and then applies cycle canceling on  $\mathbf{a}$  and sends the support of the resulting fractional matching  $\mathbf{a}'$  to Bob. Our protocol in this section is very similar, except that instead of applying cycle-canceling on  $\mathbf{a}$ , we first “scale”  $\mathbf{a}$  to obtain another fractional matching  $\mathbf{z}$  and then send the support of cycle-canceled version  $\mathbf{z}'$  of  $\mathbf{z}$  to Bob. To be more precise about what we mean by scaling  $\mathbf{a}$ , let us define:

$$h(x, y) := \min \left\{ \frac{3}{2}, \frac{1}{x}, \frac{1}{y} \right\}. \quad (6)$$

Now for each edge  $e = (u, v) \in E^A$  we define

$$z_e := h(a_v, a_u) \cdot a_e. \quad (7)$$

Noting that  $\mathbf{a}$  is a fractional matching, we get that  $a_v \leq 1$ ,  $a_u \leq 1$ , which implies  $h(a_u, a_v) \geq 1$  and thus  $z_e \geq a_e$ . This means that indeed  $\mathbf{z} = \{z_e\}_{e \in E^A}$  is entry-wise larger than  $\mathbf{a}$ . But can this scaling violate fractional matching constraints, i.e., for some  $v$ ,  $z_v := \sum_{e \ni v} z_e > 1$ ? As a simple consequence of our definition of function  $h$ , it turns out that indeed  $\mathbf{z}$  is still a fractional matching.

► **Observation 9.** *Let  $\mathbf{z}$  be obtained as above, then  $\mathbf{z}$  is a fractional matching of  $E^A$ .*

**Proof.** It is clear that  $\mathbf{z} \geq 0$  since  $z_e \geq a_e \geq 0$  for each edge  $e$ . To see why  $z_v \leq 1$  for all  $v$ , observe that for each edge  $e = (u, v)$ ,  $z_e = h(a_u, a_v)a_e \leq \frac{1}{a_v}a_e$ ; hence  $z_v \leq \frac{1}{a_v} \sum_{e \ni v} a_e = a_v/a_v = 1$ . ◀

Note that the proof of Observation 9 only uses  $h(x, y) \leq \min\{\frac{1}{x}, \frac{1}{y}\}$ . The reason that we defined  $h$  to be  $\min\{\frac{3}{2}, \frac{1}{x}, \frac{1}{y}\}$  will be apparent later when analyzing the approximation.

Our scaled protocol can thus be formalized as follows.

Since the support of  $\mathbf{z}'$  has  $n - 1$  edges, Algorithm 2 only requires communicating  $n - 1$  edges. It thus only remains to analyze its approximation ratio.

### 4.2 The Analysis of Algorithm 2

As in Section 3, to analyze the size of matching  $\text{MM}(T \cup E^B)$  reported by Bob, we construct a large fractional matching  $\mathbf{x}$  on  $T \cup E^B$  and then use the fact that the maximum matching of this graph is at least as large as any fractional matching on it. Our construction of this fractional matching  $\mathbf{x}$  is also in fact the same as our construction in Section 3 with the

■ **Algorithm 2** A scaled distribution-dependent sparsifier protocol.

- 
- (i) Given edges  $E^A$  as input to Alice, she computes the vector  $\mathbf{a} = \{a_e\}_{e \in E^A}$  using Eq (1).
  - (ii) Alice then constructs  $\mathbf{z} = \{z_e\}_{e \in E^A}$  using Eq (7); by Observation 9  $\mathbf{z}$  is a valid fractional matching of  $E^A$ .
  - (iii) Alice obtains a fractional matching  $\mathbf{z}'$  by running cycle canceling on  $\mathbf{z}$  (Lemma 2) and then sends the edges  $T$  in the support of  $\mathbf{z}'$  to Bob.
  - (iv) Bob, given message  $T$  from Alice and input  $E^B$ , returns a maximum matching of  $T \cup E^B$ .
- 

difference that we first commit the sparsified version  $\mathbf{z}'$  of the *scaled* fractional matching  $\mathbf{z}$  to  $\mathbf{x}$ . More formally, we have:

$$x_e := \begin{cases} z'_e & \text{if } e \in T, \\ 1 - \max\{z'_u, z'_v\} & \text{if } e = (u, v) \in \text{MM}(G) \cap E^B. \end{cases}$$

To analyze the size of  $\mathbf{x}$ , we need a few definitions. Definition 11 below for  $b_v$  is equivalent to the definition of  $b_v$  in Section 3, but instead of vector  $\mathbf{a}$ , for each edge  $e \in \text{MM}(G) \cap E^B$  the vertex with higher  $\mathbf{z}$  is made responsible. To be more formal and to avoid ties (for pairs of vertices with  $z_u = z_v$ ) we first define an ordering over the vertices in Definition 10 below and then define  $b_v$ .

► **Definition 10.** *Based on fractional matching  $\mathbf{z}$ , we define a total ordering over the vertex set  $V$  as follows. For any pair of vertices  $u$  and  $v$  with  $z_u \neq z_v$ , we say  $v \succ u$  if  $z_v > z_u$ . For pairs  $u, v$  with  $z_u = z_v$  we break the tie arbitrarily; say  $v \succ u$  if the ID of  $v$  is larger than  $u$ .*

► **Definition 11.** *For each vertex  $v$ , define  $b_v := \Pr[\exists u : uv \in \text{MM}(G) \cap E^B \text{ and } v \succ u \mid E^A]$ .*

Based on this definition of  $b_v$  and similar to Claim 4 of Section 3, we get that:

▷ **Claim 12.**  $\mathbf{E}[\mu(G) \mid E^A] = \sum_v \frac{1}{2}a_v + b_v$ .

Proof. Follows from the same argument in the proof of Claim 4. ◁

The next step is where we start to substantially deviate from the analysis of Section 3. We first give an informal explanation of why a different approach might be needed to analyze Algorithm 2 (the reader may choose to skip this informal explanation and jump to the new analysis after). After that, we formally describe our actual analysis which is based on a notion of “contribution sharing”.

**Informal explanation: why a different analysis is needed.** In Claim 5 of Section 3 we showed  $\mathbf{E}[|\mathbf{x}| \mid E^A] = \sum_v \frac{1}{2}a_v + (1 - a_v)b_v$ , implying intuitively that each vertex  $v$  contributes an expected size of  $g_v = \frac{1}{2}a_v + (1 - a_v)b_v$  to  $\mathbf{x}$ . We then proved the claimed approximation ratio by comparing this contribution  $g_v$  of each vertex  $v$  with  $\frac{1}{2}a_v + b_v$ , which can be thought of as the portion of the benchmark  $\mathbf{E}[\mu(G) \mid E^A] = \sum_v \frac{1}{2}a_v + b_v$  charged to vertex  $v$ .

A straightforward generalization of this framework for analyzing Algorithm 2 would be as follows: It is not hard to see that  $\mathbf{E}[|\mathbf{x}| \mid E^A] = \sum_v \frac{1}{2}z_v + (1 - z_v)b_v$  (the proof follows from a similar argument to Claim 5); thus it suffices to show that the contribution  $g_v = \frac{1}{2}z_v + (1 - z_v)b_v$  of each vertex is large compared to the portion  $\frac{1}{2}a_v + b_v$  of the optimum charged to this vertex. The problem with this type of argument, however, is that it is hard

to measure exactly how the scaling part of Algorithm 2 is useful. In particular, take a vertex  $v$  and suppose that for every neighbor  $u$  of  $v$  in  $E^A$ , it holds that  $a_u = 1$ . This way, for each edge  $e = (v, u) \in E^A$  we would have  $h(a_v, a_u) = 1$  and thus  $z_e = h(a_v, a_u)a_e = a_e$ . That is, the edges of vertex  $v$  are in fact not scaled at all. This would mean that  $z_v = a_v$  and thus  $g_v = \frac{1}{2}z_v + (1 - z_v)b_v = \frac{1}{2}a_v + (1 - a_v)b_v$ , which is not any different from the guarantee we would get for vertex  $v$  without any scaling.

The issue discussed above intuitively implies that in defining the contribution  $g_v$  of each vertex, not only we should take into account the values of  $z_v$  and  $b_v$ , but that in fact the values of  $a_u$  for neighbors  $u$  of  $v$  are also important. Motivated by this, we define  $g_v$  such that intuitively we share the contribution of each vertex with its neighbors. That is, each vertex passes a portion of its contribution to its neighbors, and as a result also receives a portion of the contribution of them. This dynamic allows us to argue that scaling does indeed help our protocol.

**The formal analysis via “contribution sharing”.** Consider function  $\ell(x)$  defined as

$$\ell(x) := \max\left\{\frac{x - 2/3}{6}, 0\right\}. \quad (8)$$

This function  $\ell$  is the sharing function and the reason that is defined this way will be apparent later in the analysis. For each vertex  $v$ , define

$$g_v := \frac{1}{2}z_v + (1 - z_v)b_v - \ell(a_v)a_v + \sum_u \ell(a_u)a_{uv}. \quad (9)$$

The following lemma whose proof is deferred to the full version, states that the expected size of fractional matching  $\mathbf{x}$  conditioned on  $E^A$ , is equal to  $\sum g_v$ . Therefore, intuitively, we can think of  $g_v$  as the amount that vertex  $v$  contributes to the size of  $\mathbf{x}$  in expectation.<sup>4</sup>

► **Lemma 13.**  $\mathbf{E}[|\mathbf{x}| \mid E^A] = \sum_v g_v$ .

To show that  $\mathbf{x}$  tends to be large, Lemma 13 above implies that it suffices to show  $g_v$  is large. The next definition and the lemma that follows it are used for this purpose.

► **Definition 14.** Let  $a, b \in [0, 1]$ . We define:

$$f(a, b, x) := b + \left(\left(\frac{1}{2} - b\right)h(a, x) + \ell(x) - \ell(a)\right) \cdot a \quad \text{and} \quad f(a, b) := \min_{x \in [0, 1]} f(a, b, x),$$

► **Lemma 15.** For any vertex  $v$ , it holds that  $g_v \geq f(a_v, b_v)$ .

The proof of Lemma 15 is also deferred to the full version due to space constraints. This lower bound is particularly useful since  $f(a_v, b_v)$  only depends on the values of  $a_v$  and  $b_v$ , whereas  $g_v$  also depends on  $a_u$  of neighbors  $u$  of  $v$ . Having this, if we in fact prove that  $f(a_v, b_v) \geq \alpha(\frac{1}{2}a_v + b_v)$  for all  $v$ , then we get that Algorithm 2 achieves an approximation ratio of at least  $\alpha$  since

$$\frac{\mathbf{E}[|\mathbf{x}| \mid E^A]}{\mathbf{E}[\mu(G) \mid E^A]} \stackrel{\text{Claim 12, Lemma 13}}{=} \frac{\sum_v g_v}{\sum_v \frac{1}{2}a_v + b_v} \stackrel{\text{Lemma 15}}{\geq} \frac{\sum_v f(a_v, b_v)}{\sum_v \frac{1}{2}a_v + b_v} \geq \frac{\sum_v \alpha(\frac{1}{2}a_v + b_v)}{\sum_v \frac{1}{2}a_v + b_v} \geq \alpha.$$

<sup>4</sup> We note that in fact Lemma 13 holds for *any* possible definition of function  $\ell$ . That is, in the proof of Lemma 13, we do not use the value of  $\ell(x)$  defined in Eq (8).

## 48:12 On the Robust Communication Complexity of Bipartite Matching

Note that up to this point of the analysis, we have not used the fact that the edges are partitioned randomly between Alice and Bob. Therefore, in light of the lower bound of [25] which proves achieving a better-than-(2/3)-approximation for requires  $n^{1+\Omega(1/\log \log n)}$  communication, we get that Algorithm 2 cannot achieve a better-than-(2/3)-approximation under an adversarial partitioning of the input graph. As a result, there should be a choice of  $a_v, b_v$  such that  $f(a_v, b_v) \leq \frac{2}{3}(\frac{1}{2}a_v + b_v)$ . Indeed one can confirm that for  $a_v = \frac{1}{2}$  and  $b_v = \frac{1}{2}$ ,  $f(\frac{1}{2}, \frac{1}{2}) = f(\frac{1}{2}, \frac{1}{2}, 0) = 0.5 = \frac{2}{3}(\frac{1}{2}a_v + b_v)$ .

**How random partitioning helps.** Our main insight in bypassing the  $2/3$ -barrier highlighted above is that for an average vertex  $v$ , it cannot always occur that  $a_v = b_v = \frac{1}{2}$  under a random partitioning. Formally, for a vertex  $u$  chosen uniformly at random from  $V$ ,

$$\mathbf{E}_{u \sim V}[a_u] = \frac{1}{n} \sum_v \mathbf{E}[a_v] = \frac{2}{n} \mathbf{E}[\mathbf{a}] = \frac{2}{n} \mathbf{E}[\text{MM}(G) \cap E^A] \stackrel{(\star)}{=} \frac{2}{n} \cdot \frac{\mathbf{E}[\mu(G)]}{2} = \frac{1}{n} \mathbf{E}[\mu(G)],$$

$$\mathbf{E}_{u \sim V}[b_u] = \frac{1}{n} \sum_v \mathbf{E}[b_v] \stackrel{\text{Definition 11}}{=} \frac{1}{n} \mathbf{E}[\text{MM}(G) \cap E^B] \stackrel{(\star)}{=} \frac{1}{n} \cdot \frac{\mathbf{E}[\mu(G)]}{2} = \frac{1}{2n} \mathbf{E}[\mu(G)],$$

where the equalities marked with  $(\star)$  use the fact that each edge is given to Alice/Bob with probability  $1/2$ . This implies that  $\mathbf{E}[a_u] = 2\mathbf{E}[b_u]$  which formalizes our earlier claim that  $a_u = b_u = \frac{1}{2}$  cannot always happen for an average vertex  $u$ .

To turn the intuition above into an actual analysis of the approximation factor for Algorithm 2 under a random partitioning, we write a factor revealing program formalized as Program 1. We prove that the solution to Program 1 is indeed a lower bound for the approximation ratio of Algorithm 2. The proof can be found in the full version of the paper and is based on our intuition above regarding the relation between  $\mathbf{E}[a_u]$  and  $\mathbf{E}[b_u]$  for a vertex  $u$  chosen at random.

We note that for generality Program 1 is written with a parameter  $p$  which is  $1/2$  (more generally  $p$  can be thought of as the probability that each edge is given to Alice).

► **Lemma 16.** *Let  $r$  be the solution of Program 1 below for  $p = \frac{1}{2}$ ; then  $\mathbf{E}[\mathbf{x}] \geq r \cdot \mathbf{E}[\mu(G)]$ .*

► **Program 1.** A factor revealing (non-linear) program for the performance of Algorithm 2.

$$\begin{aligned} & \text{find} && \text{a distribution } \mathcal{S} \text{ for } (a, b) \text{ over } [0, 1] \times [0, 1] \\ & \text{minimizing} && \mathbf{E}_{\mathcal{S}}[f(a, b)] / \mathbf{E}_{\mathcal{S}}[\frac{1}{2}a + b] \\ & \text{subject to} && \mathbf{E}_{\mathcal{S}}[a] = \frac{2p}{1-p} \mathbf{E}_{\mathcal{S}}[b] \\ & && \Pr_{\mathcal{S}}[a + b \leq 1] = 1 \\ & && \Pr_{\mathcal{S}}[a, b \geq 0] = 1 \end{aligned}$$

In order to find the solution of Program 1, we simplify it and then write a factor revealing LP, showing that  $r \geq 0.7167$  which implies the same bound on the approximation ratio of Algorithm 2. Due to space limits, we omit the details of this step, referring interested readers to the full version of the paper.

## 5 Lifting Knowledge of Distribution via Minimax Theorems

As discussed before, our protocol of Section 4 achieves its claimed approximation guarantee assuming that the input graph  $G$  is drawn from some distribution  $\mathcal{G}$  that is known to the algorithm a priori. In the standard communication complexity model, however, we do not have

access to distribution  $\mathcal{G}$  and the algorithm should work against every possible input graph. In this section, we show how one can use minimax theorems to lift the assumption on knowledge of the distribution  $\mathcal{G}$  in our protocols, without incurring any loss to the approximation guarantee. The following theorem formalizes our main result in the Introduction.

► **Theorem 17.** *There is a randomized one-way protocol that given any arbitrary input graph  $G$  partitioned randomly between Alice and Bob, outputs a matching  $M(G)$  in  $G$  such that  $\mathbf{E} |M(G)| \geq 0.716 \cdot \mu(G)$ . The protocol requires  $O(n)$  communication from Alice to Bob.*

Consider a *deterministic* protocol  $\mathcal{A}$  and let us use  $\mathcal{A}(G^A, G^B)$  to denote the size of the matching returned by the protocol  $\mathcal{A}$  when Alice receives subgraph  $G^A$  and Bob receives subgraph  $G^B$ . Recall that in our discussion of Section 4, we say protocol  $\mathcal{A}$  obtains an  $\alpha$ -approximation if

$$\mathbf{E}_{G \sim \mathcal{G}, (G^A, G^B)} [\mathcal{A}(G^A, G^B)] \geq \alpha \cdot \mathbf{E}_{G \sim \mathcal{G}} [\mu(G)], \quad (10)$$

where here and throughout this section, by subscript  $(G^A, G^B)$  we mean the random process of partitioning the edges of  $G$  into  $G^A$  and  $G^B$  independently and uniformly at random. This guarantee is inherently different from that of Theorem 17. In the following, we first show how one can remedy this part and then give the argument for lifting the assumption on the knowledge of  $\mathcal{G}$ .

**Step 1: Getting an Instance-Wise Approximation Guarantee.** In order to remove the assumption on the knowledge of the distribution  $\mathcal{G}$  we first show that we can slightly modify our protocols to get an *instance-wise* expected approximation guarantee.

► **Lemma 18.** *Suppose that given any input distribution  $\mathcal{G}$  on  $n$ -vertex graphs, there is an  $\alpha$ -approximate maximum matching protocol  $\mathcal{A}$  (i.e.,  $\mathcal{A}$  satisfies Eq (10)) with communication cost  $O(n)$ . For any input distribution  $\mathcal{G}$  and any parameter  $\varepsilon > 0$ , there is another deterministic protocol  $\mathcal{A}'$  with communication cost  $O(\frac{n}{\varepsilon})$  such that*

$$\mathbf{E}_{G \sim \mathcal{G}, (G^A, G^B)} [\mathcal{A}'(G^A, G^B) / \mu(G)] \geq (1 - \varepsilon - o(1)) \cdot \alpha.$$

Due to space constraints, we omit the proof of Lemma 18. However, we note that it is easy to prove if one allows  $O(n \log n / \varepsilon)$  communication instead of  $O(n / \varepsilon)$ . To see this, note that if distribution  $\mathcal{G}$  was such that  $\mu(G)$  was the same in all outcomes of  $G$ , then Lemma 18 would be trivial. Therefore, one can make  $O(\log n / \varepsilon)$  geometrically increasing guesses for the value of  $\mu(G)$ , condition on each separately, and run protocol  $\mathcal{A}$  on them in parallel. In the full version of the paper, we show how one can avoid the extra  $O(\log n)$  factor and achieve this with just  $O(n / \varepsilon)$  communication.

**Step 2: Using Yao's Minimax.** Now that we have an instance-wise approximation guarantee using Lemma 18, we show how one can use Yao's minimax principle [41] to give a single randomized protocol that works against all possible input graphs without knowledge of the distribution  $\mathcal{G}$  from which the graph is drawn. The discussion of this section is essentially a straightforward extension of Yao's minimax principle [41] (see, e.g., [38, Section 2.2] or [36]) for the random partition model. The proof of this proposition is almost identical to that of the original Yao's minimax principle and we claim no novelty for this proof.

► **Proposition 19.** *Let  $C$  and  $\alpha$  be two parameters. Suppose for every distribution  $\mathcal{G}$  on  $n$ -vertex graphs, there exists a deterministic protocol  $\mathcal{A}_{\mathcal{G}}$  with communication cost  $C$  with an instance-wise approximation guarantee  $\mathbf{E}_{G \sim \mathcal{G}, (G^A, G^B)} [\mathcal{A}_{\mathcal{G}}(G^A, G^B) / \mu(G)] \geq \alpha$  where here*

## 48:14 On the Robust Communication Complexity of Bipartite Matching

$(G^A, G^B)$  is a random partitioning of  $G$ . Then, there exists a randomized protocol  $\mathcal{A}_*$  with communication cost  $C$  such that for every graph  $G$ ,  $\mathbf{E}_{\mathcal{A}_*, (G^A, G^B) \sim G}[\mathcal{A}_*(G^A, G^B)] \geq \alpha \mu(G)$ , where the expectation here is taken over both the randomness of the protocol and the random partitioning of the edge-set of  $G$  between the players.

**Proof.** Consider a game between two players called the Input player and the Algorithm player. The set of strategies of the Input player are all bipartite graphs on  $n$  vertices, denoted by  $\mathbb{G}(n)$ , and the set of strategies of the Algorithm player are all deterministic one-way protocols with communication cost  $C$ , denoted by  $\mathbb{P}(C)$ ; for fixed  $n$  and  $C$ , both sets are finite.

For any graph  $G \in \mathbb{G}(n)$  as a strategy of the Input player and deterministic protocol  $\mathcal{A} \in \mathbb{P}(C)$  as the strategy of the Algorithm player, we define:

$$\text{val}(G, \mathcal{A}) := \mathbf{E}_{(G^A, G^B) \sim G}[\mathcal{A}(G^A, G^B)/\mu(G)].$$

On a choice of (pure) strategies  $G$  and  $\mathcal{A}$  by the players, we define the payoff of the Algorithm player as  $\text{val}(G, \mathcal{A})$  and for the Input player as  $-\text{val}(G, \mathcal{A})$ . Alternatively, the Algorithm player would like to maximize  $\text{val}(G, \mathcal{A})$  (by choosing  $\mathcal{A}$ ), while the Input player tries to minimize it (by choosing  $G$ ). Thus, this is a zero-sum game.

Let  $\Delta_G$  denote the set of all distributions on strategies (graphs) of the Input player and  $\Delta_P$  denote the set of all distributions on strategies (deterministic protocols) of the Algorithm player. This way,  $\Delta_G$  and  $\Delta_P$  denote the set of all mixed strategies for the Input player and Algorithm player, respectively. Considering this is a zero-sum game, Von Neumann's Minimax Theorem asserts that,

$$\min_{\mathcal{G} \in \Delta(G)} \max_{\mathcal{A} \in \mathbb{P}(C)} \mathbf{E}_{G \sim \mathcal{G}}[\text{val}(G, \mathcal{A})] = \max_{\mathcal{A}_R \in \Delta_P} \min_{G \in \mathbb{G}(n)} \mathbf{E}_{\mathcal{A} \sim \mathcal{A}_R}[\text{val}(G, \mathcal{A})].$$

Replacing the value of  $\text{val}(G, \mathcal{A})$  with its definition on both sides, we have

$$\min_{\mathcal{G} \in \Delta(G)} \max_{\mathcal{A} \in \mathbb{P}(C)} \mathbf{E}_{G, (G^A, G^B) \sim G} \left[ \frac{\mathcal{A}(G^A, G^B)}{\mu(G)} \right] = \max_{\mathcal{A}_R \in \Delta_P} \min_{G \in \mathbb{G}(n)} \mathbf{E}_{\mathcal{A} \sim \mathcal{A}_R, (G^A, G^B)} \left[ \frac{\mathcal{A}(G^A, G^B)}{\mu(G)} \right]. \quad (11)$$

The LHS in Eq (11) corresponds to picking any possible distribution on inputs and then running the “best” deterministic protocol on this distribution and measuring the instance-wise expected approximation ratio of the protocol. Thus, by the statement of the proposition, the LHS is  $\geq \alpha$ .

The RHS in Eq (11) corresponds to picking any distribution over deterministic protocols, i.e., a (public-coin) randomized protocol, and then running this protocol on the “worst” input graph and measure the expected ratio of the protocol. By the lower bound on LHS and Eq (11), this is at least  $\alpha$ , which means that there exists a randomized protocol  $\mathcal{A}_*$  with communication cost  $C$  (the arg max of RHS in Eq (11)) that achieves an  $\alpha$ -approximation in expectation for every input graph partitioned randomly between Alice and Bob. This concludes the proof.  $\blacktriangleleft$

Theorem 17 now follows immediately from Theorem 8, Lemma 18 and Proposition 19.

---

### References

- 1 Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab S. Mirrokni, and Cliff Stein. Coresets meet EDCS: algorithms for matching and vertex cover on massive graphs. In



- Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1616–1635, 2019.
- 2 Sepehr Assadi and Soheil Behnezhad. Beating two-thirds for random-order streaming matching. *CoRR*, abs/2102.07011. To appear in ICALP 2021, 2021.
  - 3 Sepehr Assadi and Aaron Bernstein. Towards a unified theory of sparsification for matching problems. In *2nd Symposium on Simplicity in Algorithms, SOSA@SODA 2019, January 8-9, 2019 - San Diego, CA, USA*, pages 11:1–11:20, 2019.
  - 4 Sepehr Assadi, Sanjeev Khanna, and Yang Li. The stochastic matching problem with (very) few queries. In *Proceedings of the 2016 ACM Conference on Economics and Computation, EC '16, Maastricht, The Netherlands, July 24-28, 2016*, pages 43–60, 2016.
  - 5 Sepehr Assadi, Sanjeev Khanna, and Yang Li. On estimating maximum matching size in graph streams. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1723–1742, 2017.
  - 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 Ran Raz. Near-quadratic lower bounds for two-pass graph streaming algorithms. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 342–353. IEEE, 2020.
  - 9 Maria-Florina Balcan, Yi Li, David P. Woodruff, and Hongyang Zhang. Testing matrix rank, optimally. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 727–746, 2019.
  - 10 Soheil Behnezhad and Mahsa Derakhshan. Stochastic weighted matching:  $(1-\epsilon)$  approximation. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1392–1403, 2020.
  - 11 Soheil Behnezhad, Mahsa Derakhshan, and MohammadTaghi Hajiaghayi. Stochastic matching with few queries:  $(1-\epsilon)$  approximation. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1111–1124, 2020.
  - 12 Soheil Behnezhad, Alireza Farhadi, MohammadTaghi Hajiaghayi, and Nima Reyhani. Stochastic matching with few queries: New algorithms and tools. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2855–2874, 2019.
  - 13 András A. Benczúr and David R. Karger. Approximating  $s$ - $t$  minimum cuts in  $\tilde{O}(n^2)$  time. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 47–55, 1996.
  - 14 András A. Benczúr and David R. Karger. Randomized approximation schemes for cuts and flows in capacitated graphs. *SIAM J. Comput.*, 44(2):290–319, 2015.
  - 15 Aaron Bernstein. Improved bounds for matching in random-order streams. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, pages 12:1–12:13, 2020.
  - 16 Aaron Bernstein and Cliff Stein. Fully dynamic matching in bipartite graphs. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, July 6-10, 2015, Proceedings, Part I*, pages 167–179, 2015.

- 17 Aaron Bernstein and Cliff Stein. Faster fully dynamic matchings with small approximation ratios. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, January 10-12, 2016*, pages 692–711, 2016.
- 18 Amit Chakrabarti, Graham Cormode, and Andrew McGregor. Robust lower bounds for communication and stream computation. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, May 17-20, 2008*, pages 641–650, 2008.
- 19 Graham Cormode, Jacques Dark, and Christian Konrad. Independent sets in vertex-arrival streams. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, pages 45:1–45:14, 2019.
- 20 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.
- 21 Shahar Dobzinski, Noam Nisan, and Sigal Oren. Economic efficiency requires interaction. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 233–242, 2014.
- 22 Alireza Farhadi, Mohammad Taghi Hajiaghayi, Tung Mai, Anup Rao, and Ryan A. Rossi. Approximate maximum matching in random streams. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1773–1785, 2020.
- 23 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005.
- 24 Buddhima Gamlath, Sagar Kale, Slobodan Mitrovic, and Ola Svensson. Weighted matchings via unweighted augmentations. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 491–500, 2019.
- 25 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '12*, pages 468–485. SIAM, 2012. URL: <http://dl.acm.org/citation.cfm?id=2095116.2095157>.
- 26 Venkatesan Guruswami and Krzysztof Onak. Superlinear lower bounds for multipass graph processing. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 287–298, 2013.
- 27 András Hajnal, Wolfgang Maass, and György Turán. On the communication complexity of graph properties. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 186–191, 1988.
- 28 John E. Hopcroft and Richard M. Karp. An  $n^{5/2}$  algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.
- 29 Zengfeng Huang, Bozidar Radunovic, Milan Vojnovic, and Qin Zhang. Communication complexity of approximate matching in distributed graphs. In *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, pages 460–473, 2015.
- 30 Gábor Ivanyos, Hartmut Klauck, Troy Lee, Miklos Santha, and Ronald de Wolf. New bounds on the classical and quantum communication complexity of some graph properties. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India*, pages 148–159, 2012.
- 31 Michael Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1679–1697, 2013. doi:10.1137/1.9781611973105.121.
- 32 Michael Kapralov. Space lower bounds for approximating maximum matching in the edge arrival model. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on*

- Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1874–1893. SIAM, 2021.
- 33 Michael Kapralov, Gilbert Maystre, and Jakab Tardos. Communication efficient coresets for maximum matching. In Hung Viet Le and Valerie King, editors, *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, pages 156–164. SIAM, 2021.
  - 34 Christian Konrad. A simple augmentation method for matchings with applications to streaming algorithms. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, pages 74:1–74:16, 2018.
  - 35 Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, pages 231–242, 2012.
  - 36 Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, 1997.
  - 37 Euiwoong Lee and Sahil Singla. Maximum matching in the online batch-arrival model. In *Integer Programming and Combinatorial Optimization - 19th International Conference, IPCO 2017, Waterloo, ON, Canada, June 26-28, 2017, Proceedings*, pages 355–367, 2017.
  - 38 Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
  - 39 Ilan Newman. Private vs. common random bits in communication complexity. *Inf. Process. Lett.*, 39(2):67–71, 1991.
  - 40 Ran Raz and Boris Spieker. On the "log rank"-conjecture in communication complexity. In *34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993*, pages 168–176, 1993.
  - 41 Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 222–227, 1977.
  - 42 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 209–213, 1979.



# L1 Regression with Lewis Weights Subsampling

Aditya Parulekar ✉

Department of Computer Science, University of Texas at Austin, TX, USA

Advait Parulekar ✉

Department of Electrical and Computer Engineering, University of Texas at Austin, TX, USA

Eric Price ✉

Department of Computer Science, University of Texas at Austin, TX, USA

---

## Abstract

We consider the problem of finding an approximate solution to  $\ell_1$  regression while only observing a small number of labels. Given an  $n \times d$  unlabeled data matrix  $X$ , we must choose a small set of  $m \ll n$  rows to observe the labels of, then output an estimate  $\hat{\beta}$  whose error on the original problem is within a  $1 + \varepsilon$  factor of optimal. We show that sampling from  $X$  according to its Lewis weights and outputting the empirical minimizer succeeds with probability  $1 - \delta$  for  $m > O(\frac{1}{\varepsilon^2} d \log \frac{d}{\varepsilon \delta})$ . This is analogous to the performance of sampling according to leverage scores for  $\ell_2$  regression, but with exponentially better dependence on  $\delta$ . We also give a corresponding lower bound of  $\Omega(\frac{d}{\varepsilon^2} + (d + \frac{1}{\varepsilon^2}) \log \frac{1}{\delta})$ .

**2012 ACM Subject Classification** Computing methodologies → Active learning settings

**Keywords and phrases** Active regression, Lewis weights

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.49

**Category** RANDOM

**Related Version** *Full Version:* <https://arxiv.org/abs/2105.09433>

**Funding** *Advait Parulekar:* This author was supported by NSF Grant 2019844.

*Eric Price:* This author was supported in part by NSF Award CCF-1751040 (CAREER).

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

## 1 Introduction

The standard linear regression problem is, given a data matrix  $X \in \mathbb{R}^{n \times d}$  and corresponding values  $y \in \mathbb{R}^n$ , to find a vector  $\beta \in \mathbb{R}^d$  minimizing  $\|X\beta - y\|_p$ . Least squares regression ( $p = 2$ ) is the most common, but least absolute deviation regression ( $p = 1$ ) is sometimes preferred for its robustness to outliers and heavy-tailed noise. In this paper we focus on  $\ell_1$  regression:

$$\beta^* = \arg \min_{\beta \in \mathbb{R}^d} \|X\beta - y\|_1 \tag{1}$$

But what happens if the unlabeled data  $X$  is cheap but the labels  $y$  are expensive? Can we choose a small subset of indices, only observe the corresponding labels, and still recover a good estimate  $\hat{\beta}$  of the true solution? We would like an algorithm that works with probability  $1 - \delta$  for any input  $(X, y)$ ; this necessitates that our choice of indices be randomized, so the adversary cannot concentrate the noise on them. Formally we define the problem as follows:



© Aditya Parulekar, Advait Parulekar, and Eric Price;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 49; pp. 49:1–49:21



Leibniz International Proceedings in Informatics

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

► **Problem 1** (Active L1 regression). *There is a known matrix  $X \in \mathbb{R}^{n \times d}$  and a fixed unknown vector  $y$ . A learner interacts with the instance by querying rows indexed  $\{i_k\}_{k \in [m]}$  adaptively, and is shown labels  $\{y_{i_k}\}_{k \in [m]}$  corresponding to the rows queried. The learner must return  $\hat{\beta}$  such that with probability  $1 - \delta$  over the learner’s randomness,*

$$\|X\hat{\beta} - y\|_1 \leq (1 + \varepsilon) \min_{\beta} \|X\beta - y\|_1. \quad (2)$$

Some rows of  $X$  may be more important than others. For example, if one row is orthogonal to all the others, we need to query it to have any knowledge of the corresponding  $y$ ; but if many rows are in the same direction it should suffice to label a few of them to predict the rest.

A natural approach to this problem is to attach some notion of “importance”  $p_1, \dots, p_n$  to each row of  $X$ , then sample rows proportional to  $p_i$ . We can represent this as a “sampling-and-reweighting” sketch  $S \in \mathbb{R}^{m \times n}$ , where each row is  $\frac{1}{p_i} e_i$  with probability proportional to  $p_i$ . This reweighting is such that  $\mathbb{E}_S[\|Sv\|_1] \propto \|v\|_1$  for any vector  $v$ . By querying  $m$  rows we can observe  $Sy$ , and so can output the empirical risk minimizer (ERM)

$$\hat{\beta} := \arg \min \|SX\beta - Sy\|_1. \quad (3)$$

For fixed  $\beta$ ,  $\mathbb{E}_S \|SX\beta - Sy\|_1 \propto \|X\beta - y\|_1$ . The hope is that, if the  $p_i$  are chosen carefully, the ERM  $\hat{\beta}$  will satisfy (2) with relatively few samples. Our main result is that this is true if the  $p_i$  are drawn according to the  $\ell_1$  Lewis weights:

► **Theorem 1** (Informal). *Problem 1 can be solved with  $m = O(\frac{1}{\varepsilon^2} d \log \frac{d}{\varepsilon \delta})$  queries. For constant  $\delta = \Theta(1)$ ,  $m = O(\frac{1}{\varepsilon^2} d \log d)$  suffices.*

Note that, while the model allows for adaptive queries, this algorithm is nonadaptive.

We next show that our sample complexity is near-optimal by demonstrating the following lower bound on the number of queries needed by any algorithm to obtain an accurate estimate. Whether the multiplicative factor of  $\log d$  is necessary is an open question.

► **Theorem 2** (Informal). *Any algorithm satisfying Problem 1 must query  $\Omega(d \log \frac{1}{\delta} + \frac{d}{\varepsilon^2} + \frac{1}{\varepsilon^2} \log \frac{1}{\delta})$  rows on some instances  $(X, y)$ .*

For small  $\delta$ , the upper bound is the product of  $d$ ,  $\frac{1}{\varepsilon^2}$ , and  $\log(1/\delta)$  while the lower bound is the product of each pair.

## 1.1 Related Work

### If all the labels are known

LAD regression cannot be solved in closed form. It can be written as a linear program, but this is relatively slow to solve. One approach to speeding up LAD regression is “sketch-and-solve,” which replaces (1) with (3), which has fewer constraints and so can be solved faster. The key idea here is to acquire regression guarantees by ensuring that  $S$  is a *subspace embedding* for the column space of  $[X \ y]$ .

For a survey on techniques to do this, we direct the reader to [17],[13], [3]. In [17], the emphasis is on *oblivious* sketches – distributions which do not require knowledge of  $[X \ y]$ . On the other hand, [13], [3] discuss sketches that depend on  $[X \ y]$ . Most relevant to us [9], which shows that sampling-and-reweighting matrices  $S$  using Lewis weights of  $[X \ y]$  suffice; we give a simple proof of this in Remark 4. The problem is that *figuring out which labels are important* involves looking at all the labels.

### Active $\ell_2$ regression

Here we return to our setting, where only a subset of the labels is available to us. A number of works have studied this problem, including [6, 7, 8]. The  $\ell_2$  version of the problem was solved optimally in [2], where an algorithm was given using  $O(\frac{d}{\epsilon})$  queries to find  $\hat{\beta}$  satisfying  $\mathbb{E} \left[ \|X\hat{\beta} - y\|_2^2 \right] \leq (1 + \epsilon) \|X\beta^* - y\|_2^2$ . Independent, identical sampling using leverage scores achieves the same guarantee using  $O(d \log d + \frac{d}{\epsilon})$  queries. Note that these results for  $\ell_2$  ERM only work in expectation, while our results hold with high probability.

### Subspace embedding for $\ell_1$ norms

Subspace embeddings for the  $\ell_1$  norm have been studied in a long line of work including [15], [16], [12], [5], and [4], the most recent of which describes an iterative algorithm to approximate *Lewis weights*, which are the analogue of leverage scores for importance sampling preserving  $\ell_1$  norms. The [4] result shows that, for the same  $m = O(\frac{1}{\epsilon^2} d \log \frac{d}{\epsilon \delta})$  sample complexity as given in Theorem 11, a sampler sketch  $S$  based on the Lewis weights of  $X$  will have  $\|SX\beta\|_1 \approx_\epsilon \|X\beta\|_1$  for all  $\beta \in \mathbb{R}^d$ .

### Our approach

At a very high level the goal of this paper is to replace the  $\ell_2$  leverage score analysis of [2] (which looks at the sample complexity of  $\ell_2$  regression in setting of this paper) with the  $\ell_1$  Lewis weight analysis in [4] (which, among other results, demonstrates that i.i.d. importance sampling with Lewis weights results in a subspace embedding). However, the differences between  $\ell_1$  and  $\ell_2$  are significant enough that very little of the [2] proof approach remains.

Per [4], the Lewis weight sampling-and-embedding matrix  $S$  preserves  $\|X\beta\|_1$  for all  $\beta$ . The problem is that it *doesn't* preserve  $\|X\beta - y\|_1$ : if  $y$  has outliers, we have no idea where they are to sample them. In the  $\ell_2$  setting, this difficulty is addressed using the closed-form solution  $\beta^* = X^\dagger y$ , and the fact that the residual vector  $X\beta^* - y$  is orthogonal to the column space of  $X$ . If  $S$  is a subspace embedding it will preserve  $\|X\beta - X\beta^*\| \approx \|S(X\beta - X\beta^*)\|$ , while orthogonality of  $X\beta^* - y$  and  $X\beta^*$  ensures that  $\|S(X\tilde{\beta} - X\beta^*)\| \ll \|S(X\tilde{\beta} - y)\| \approx \|X\tilde{\beta} - y\|$  (here the last approximation is not because of embeddings but rather Markov's inequality). In the  $\ell_1$  setting, not only is  $\beta^*$  not expressible in closed form, but there can be many equally valid minimizers  $\beta^*$  that are far from each other. In Appendix A we show how this approach extends to the  $\ell_1$  setting to give a simple proof of Theorem 1 for a constant factor approximation (i.e.,  $\epsilon = O(1)$ ); but the existence of multiple  $\beta^*$  makes  $\epsilon < 1$  seem unobtainable by this approach.

Instead, we massage the [2] subspace embedding proof into the appropriate form, as we discuss in Section 3. While  $S$  doesn't preserve the total error  $\|X\beta - y\|_1$ , it does preserve *relative* error  $\|X\beta - y\|_1 - \|X\beta^* - y\|_1$ ; the effect of outliers is canceled out, so that this concentrates similarly well to  $\|X\beta - X\beta^*\|_1$ .

### Concurrent work

A very similar set of results appears concurrently and independently in [1]. Their main result is identical to ours, with a similar proof. They also extend the result to  $1 < p < 2$ , but with a significantly weaker  $m = \tilde{O}(d^2/\epsilon^2)$  bound. They do not have the  $\Omega(d \log \frac{1}{\delta})$  lower bound.

## 2 Preliminaries: Subspace Embeddings and Importance Sampling

A key idea used in our analysis is that of a  $\ell_1$  subspace embedding, which is a linear sketch of a matrix that preserves  $\ell_1$  norms within the column space of a matrix:

► **Definition 3** (Subspace Embeddings). *A subspace embedding for the column space of the matrix  $X \in \mathbb{R}^{n \times d}$  is a matrix  $S$  such that for all  $\beta \in \mathbb{R}^d$ ,*

$$\|SX\beta\| = (1 \pm \varepsilon)\|X\beta\|$$

► **Remark 4.** Consider the simpler setting in which we had access to all of  $y$ , but we still want to subsample rows to improve computational complexity. We can view the regression loss  $\|X\beta - y\|_1$  as the  $\ell_1$  norm of the point  $[X \ y] \begin{bmatrix} \beta \\ -1 \end{bmatrix}$  in the column space of  $[X \ y]$ . Indeed, suppose  $\beta^* = \arg \min \|X\beta - y\|_1$  as before and let  $\hat{\beta} = \arg \min \|SX\beta - Sy\|_1$ . Then,  $\hat{\beta}$  solves problem 1 because, for  $\varepsilon < \frac{1}{3}$ ,

$$\|X\hat{\beta} - y\|_1 \leq \frac{1}{1 - \varepsilon} \|SX\hat{\beta} - Sy\|_1 \leq \frac{1}{1 - \varepsilon} \|SX\beta^* - Sy\|_1 \leq \frac{1 + \varepsilon}{1 - \varepsilon} \|X\beta^* - y\|_1 \leq (1 + 4\varepsilon)\|X\beta^* - y\|_1.$$

One way to construct a subspace embedding is by sampling rows and rescaling them appropriately:

► **Definition 5** (Sampling and Reweighting with  $\{p_i\}_{i=1}^n$ ). *For any sequence  $\{p_i\}_{i=1}^n$ , let  $N = \sum_i p_i$ . Then, the sampling-and-reweighting distribution  $\mathcal{S}(\{p_i\}_{i=1}^n)$  over the set of matrices  $S \in \mathbb{R}^{N \times n}$  is such that each row of  $S$  is independently the  $i$ th standard basis vector with probability  $\frac{p_i}{N}$ , scaled by  $\frac{1}{p_i}$ . For any  $k \in [N]$ , let  $i_k$  denote the index such that  $S_{k, i_k} = \frac{1}{p_{i_k}}$ .*

When working in  $\ell_2$ , there is a natural choice for re-weighting: the leverage scores of the rows [17].

► **Definition 6** (Leverage Scores). *The leverage score of the  $i$ th row of a matrix  $X$ ,  $l_i(X)$  is defined as  $x_i^\top (X^\top X)^{-1} x_i$ .*

For  $\ell_1$  subspace embeddings, the analogous weights are the  $\ell_1$  Lewis weights, defined implicitly as the unique weights  $\{w_i(X)\}_{i=1}^n$  that satisfy  $w_i(X) = l_i(WX)$  where  $W$  is a diagonal matrix with  $i$ th diagonal entry  $\frac{1}{\sqrt{w_i(X)}}$ . We will drop the explicit dependence on  $X$  whenever it is clear from context.

► **Definition 7** (Lewis Weights). *The  $\ell_1$  Lewis weights of a matrix  $X$  are the unique weights  $\{w_i\}_{i=1}^n$  that satisfy  $w_i^2 = x_i^\top (\sum_{j=1}^n \frac{1}{w_j} x_j x_j^\top)^{-1} x_i$  for all  $i$ .*

Lewis weights are defined in general for general  $\ell_p$  norms, but we will only need the  $\ell_1$  Lewis weights. For basic properties of Lewis weights, we direct the reader to [4]. Using these definitions, we now state the main consequence of using Lewis weights. This result comes from a line of work on embeddings from subspaces of  $L_1[0, 1]$  to  $\ell_1^m$  such as [15], but is reproduced here similar to how it is presented in [4].

► **Theorem 8** ([4] Theorem 2.3). *Sampling at least  $O(\frac{d \log d}{\varepsilon^2})$  rows according to the  $\ell_1$  Lewis weights  $\{w_i\}_{i=1}^n$  of a matrix  $X \in \mathbb{R}^{n \times d}$  results in a subspace embedding for  $X$  with at least some constant probability. If at least  $O(\frac{d \log \frac{d}{\varepsilon \delta}}{\varepsilon^2})$  rows are sampled, then we have a subspace embedding with probability at least  $1 - \delta$ .*



## 2.1 Properties of Lewis Weights

We will need some properties of Lewis weights, particularly of how they change when the matrix  $X$  is modified.

► **Lemma 9** ([4] Lemma 5.5). *The  $\ell_1$  Lewis weights of a matrix do not increase when rows are added.*

► **Lemma 10.** *Let  $X \in \mathbb{R}^{n \times d}$ , and let  $X' \in \mathbb{R}^{kn \times d}$  be  $X$  stacked on itself  $k$  times, with each row scaled down by  $k$ . Then, each of the Lewis weights is reduced by a factor of  $k$ .*

### 3 Proof Overview

► **Theorem 11.** *Let  $X \in \mathbb{R}^{n \times d}$  have  $\ell_1$  Lewis weights  $\{w_i\}_{i \in [n]}$ , and let  $0 < \varepsilon, \delta < 1$ . Then, for any  $N$  that is at least  $O\left(\frac{d}{\varepsilon^2} \log \frac{d}{\varepsilon \delta}\right)$ , there is a sampling-and-reweighting distribution  $\mathcal{S}(\{p_i\}_{i=1}^n)$  satisfying  $\sum_i p_i = N$  such that for all  $y$ , if  $S \sim \mathcal{S}(\{p_i\}_{i=1}^n)$  and  $\hat{\beta} = \arg \min \|SX\beta - Sy\|_1$ , we have*

$$\|X\hat{\beta} - y\|_1 \leq (1 + \varepsilon) \min_{\beta} \|X\beta - y\|_1$$

with probability  $1 - \delta$ . If  $\delta = O(1)$  is some constant, then  $N$  at least  $O\left(\frac{1}{\varepsilon^2} d \log d\right)$  rows suffice.

#### Regression guarantees from column-space embeddings

As noted in Remark 4, it would suffice to show that  $\|SX\beta - Sy\|_1 \approx \|X\beta - y\|_1$  for all  $\beta$ . The problem is that this is impossible without knowing  $y$ : if one random entry of  $y$  is very large, we would need to sample it to estimate  $\|X\beta - y\|_1$  accurately. However, we don't actually need to estimate  $\|X\beta - y\|_1$ ; we just need to be able to distinguish values of  $\beta$  for which  $\|X\beta - y\|_1$  is far from  $\|X\beta^* - y\|_1$  from values for which it is close. That is, it would suffice to accurately

$$\text{estimate } \|X\hat{\beta} - y\|_1 - \|X\beta^* - y\|_1 \quad \text{with} \quad \|SX\hat{\beta} - Sy\|_1 - \|SX\beta^* - Sy\|_1 \quad (4)$$

for every possible  $\beta$ . In the above example where  $y$  has a single large outlier coordinate, sampling this coordinate or not will dramatically affect *both* terms, but will not affect the difference very much. As such, our key lemma, Lemma 28, states that  $\ell_1$  Lewis weight sampling achieves (4) with high probability. In particular, using at least  $m \geq O\left(\frac{d}{\varepsilon^2} \log \frac{d}{\varepsilon \delta}\right)$  rows we have

$$(\|SX\beta^* - Sy\|_1 - \|SX\beta - Sy\|_1) - (\|X\beta^* - y\|_1 - \|X\beta - y\|_1) < \varepsilon \|X(\beta^* - \beta)\|_1 \quad (5)$$

for all  $\beta$  with probability at least  $1 - \delta$ . We do this by adapting the argument of [4] which shows that  $S$  is a column-space embedding with high probability. We have summarized this argument below.

#### Column-space embedding using Lewis weights ([4])

An important result in [4], which directly implies the high probability subspace embedding, and which will be useful to us later is the following moment bound on deviations of  $\|SX\beta\|_1$ .

## 49:6 L1 Regression with Lewis Weights Subsampling

► **Lemma 12** ([4] Lemma 7.4). *If  $N$  is at least  $O\left(\frac{d}{\varepsilon^2} \log \frac{d}{\varepsilon\delta}\right)$ , and  $S \in \mathbb{R}^{N \times n}$  is drawn from the sampling-and-reweighting distribution  $\mathcal{S}(\{p_i\}_{i=1}^n)$  with  $\sum_i p_i = N$  and  $\{p_i\}_{i=1}^n$  proportional to Lewis weights  $\{w_i\}_{i=1}^n$ , then*

$$\mathbb{E}_S \left[ \left( \max_{\|X\beta\|_1=1} \|SX\beta\|_1 - \|X\beta\|_1 \right)^l \right] \leq \varepsilon^l \delta$$

The proof follows from this chain of inequalities:

$$\begin{aligned} \mathbb{E}_S \left[ \left( \max_{\|X\beta\|_1=1} \|SX\beta\|_1 - \|X\beta\|_1 \right)^l \right] &\stackrel{(A)}{\leq} 2^l \mathbb{E}_{\sigma, S} \left[ \left( \max_{\|X\beta\|_1=1} \left| \sum_k \sigma_k \frac{|x_{i_k}^\top \beta|}{p_{i_k}} \right| \right)^l \right] \\ &\stackrel{(B)}{\leq} 2^l \mathbb{E}_{\sigma, S} \left[ \left( \max_{\|X\beta\|_1=1} \sum_k \sigma_k \frac{x_{i_k}^\top \beta}{p_{i_k}} \right)^l \right] \\ &\stackrel{(C)}{\leq} \varepsilon^l \delta \end{aligned}$$

where the  $\sigma_k$  are independent Rademacher variables, which are  $\pm 1$  with probability  $1/2$  each, and  $p_{i_k}$  is proportional to the  $\ell_1$  Lewis weight of row  $i_k$ . (A) follows by symmetrizing the objective  $F := \max_{\|X\beta\|_1=1} \|SX\beta\|_1 - \|X\beta\|_1$ . (B) follows from a contraction lemma. (C) is shown by constructing a related matrix with bounded Lewis weights and applying Lemma 32 from [15] reproduced below.

► **Lemma 13.** *There exists constant  $C$  such that for any  $X \in \mathbb{R}^{n \times d}$  with all  $\ell_1$  Lewis weights less than  $C \frac{\varepsilon^2}{\log(\frac{n}{\delta})}$  and  $l = \log(2n/\delta)$ , we have*

$$\mathbb{E}_\sigma \left[ \left( \max_{\|X\beta\|_1=1} \left| \sum_{i=1}^n \sigma_i x_i^\top \beta \right| \right)^l \right] \leq \frac{\varepsilon^l \delta}{2} \quad (6)$$

### Regression guarantees using Lewis weight sampling

In this work, we show the following chain of inequalities.

$$\begin{aligned} \mathbb{E}_S \left[ \left( \max_{\|X\beta^* - X\beta\|_1=1} |(\|SX\beta^* - Sy\|_1 - \|SX\beta - Sy\|_1) - (\|X\beta^* - y\|_1 - \|X\beta - y\|_1)| \right)^l \right] \\ &\stackrel{(A)}{\leq} 2^l \mathbb{E}_{S, \sigma} \left[ \left( \max_{\|X\beta^* - X\beta\|_1=1} \left| \sum_k \sigma_k \left( \frac{|x_{i_k}^\top \beta^* - y_{i_k}|}{p_{i_k}} - \frac{|x_{i_k}^\top \beta - y_{i_k}|}{p_{i_k}} \right) \right| \right)^l \right] \\ &\stackrel{(B)}{\leq} 2^{2l+1} \mathbb{E}_{S, \sigma} \left[ \left( \max_{\|X(\beta^* - \beta)\|_1=1} \left| \sum_k \sigma_{i_k} \frac{x_{i_k}^\top}{p_{i_k}} (\beta^* - \beta) \right| \right)^l \right] \\ &\stackrel{(C)}{\leq} \varepsilon^l \delta \end{aligned} \quad (7)$$

Here, for (A), we symmetrize the left hand side of (5) in Lemma 29. For (B), we apply a different contraction lemma, Lemma 30, that allows us to remove  $y$  from our expression, and then end up with the same moment bound for (C). Step (C) is essentially an application of Lemma 32 to  $SX$ , however, because we cannot immediately bound the Lewis weights of  $SX$  to confirm the constraints of the Lemma, we instead construct another matrix  $X''$  which does not significantly alter the right hand side of inequality (7) while having bounded Lewis weights. This is done in Lemmas 33 and 34.

### 3.1 Lower Bounds

We will show that any algorithm must see  $\Omega(d \log \frac{1}{\delta} + \frac{1}{\varepsilon^2} \log \frac{1}{\delta} + \frac{d}{\varepsilon^2})$  labels to return  $\hat{\beta}$  satisfying  $\|X\hat{\beta} - y\|_1 \leq (1 + \varepsilon)\|X\beta^* - y\|_1$  with probability greater than  $1 - \delta$ .

For the lower bound proof it is convenient to consider a *distributional* version of the problem:

► **Problem 2** (Distributional active L1 regression). *There is an unknown joint distribution  $P$  over a finite set  $\mathcal{X} \times \mathcal{Y} \subset \mathbb{R}^d \times \mathbb{R}$ , with  $|\mathcal{Y}| = 2$ . The learner is allowed to adaptively observe  $N$  i.i.d. samples from  $P(\cdot | X = x)$  for the learner's choice of  $N$  values  $x \in \mathcal{X}$ . The learner must return  $\hat{\beta}$  satisfying*

$$\mathbb{E}_{(X,Y) \sim P} [|X^\top \hat{\beta} - Y|] \leq (1 + \varepsilon) \inf_{\beta} \mathbb{E}_{(X,Y) \sim P} [|X^\top \beta - Y|]. \quad (8)$$

with probability at least  $1 - \delta$ .

We begin with a lemma that shows that solving the original, Problem 1, for some  $n$  polynomial in the parameters  $d, \varepsilon, \delta$  is harder than solving the distributional version, Problem 2.

► **Lemma 14.** *A randomized algorithm that solves Problem 1 for  $n = \frac{2}{\varepsilon^2} (\log \frac{2}{\delta} + d \log \frac{3d}{\varepsilon})$  with accuracy  $\varepsilon$  and failure probability  $\delta$  can be used to solve any instance of Problem 2, where  $\mathcal{X}, \mathcal{Y}$ , in the unit  $\ell_\infty$  ball, with accuracy  $6\varepsilon$  and failure probability  $2\delta$ , for small  $\varepsilon$ .*

**Proof.** Let  $n = \frac{8}{\varepsilon^2} (\log \frac{2}{\delta} + d \log \frac{4d}{\varepsilon})$ . Construct an instance of Problem 1 in which the rows of feature matrix  $\mathbf{X}$  and the corresponding label vector  $y$  are drawn i.i.d. from  $P$ . Let  $H$  be the unit  $\ell_\infty$  ball. We have the following:

▷ **Claim 15.** For all  $\beta \in H$ , with probability at least  $1 - \delta$ ,

$$(1 - \varepsilon) \mathbb{E}_{(X,Y) \sim P} [|X^\top \beta - Y|] \leq \frac{1}{n} \|\mathbf{X}\beta - y\|_1 \leq (1 + \varepsilon) \mathbb{E}_{(X,Y) \sim P} [|X^\top \beta - Y|]$$

Let  $\beta^\circ$  denote the minimizer  $\inf_{\beta} \mathbb{E}_{(X,Y) \sim P} [|X^\top \beta - Y|]$ . Let  $\beta^*$  denote the minimizer of the matrix instance  $\inf_{\beta} \|\mathbf{X}\beta - y\|_1$ , and let  $\hat{\beta}$  denote the output of the algorithm on the instance generated. Then we have

$$\begin{aligned} (1 - \varepsilon) \mathbb{E}_{(X,Y) \sim P} [|X^\top \hat{\beta} - Y|] &\leq \frac{1}{n} \|\mathbf{X}\hat{\beta} - y\|_1 \\ &\leq (1 + \varepsilon) \frac{1}{n} \|\mathbf{X}\beta^* - y\|_1 && \text{with probability } 1 - \delta \\ &\leq (1 + \varepsilon) \frac{1}{n} \|\mathbf{X}\beta^\circ - y\|_1 \\ &\leq (1 + \varepsilon)^2 \mathbb{E}_{(X,Y) \sim P} [|X^\top \beta^\circ - Y|] \end{aligned}$$

So with probability  $1 - 2\delta$ ,

$$\mathbb{E}_{(X,Y) \sim P} [|X^\top \hat{\beta} - Y|] \leq (1 + 6\varepsilon) \mathbb{E}_{(X,Y) \sim P} [|X^\top \beta^\circ - Y|]. \quad \blacktriangleleft$$

We then prove lower bounds on the accuracy for any algorithm on Problem 2. We prove three theorems that allow us to show Theorem 26: Theorems 16, 20, and 23. To do this, we make several Claims, which are proved in section 7.1.

In all our lower bounds,  $x$  is a uniform  $e_i$ , and  $y \in \{0, 1\}$  while  $y$  is a Bernoulli random variable. For  $\Omega(\frac{d}{\varepsilon^2})$ , we set  $P(y = 1 | x = e_i)$  to  $\frac{1}{2} + \varepsilon$  uniformly at random independently for each  $i$ ; getting an  $\varepsilon$ -approximate solution requires getting most of the biases correct,

## 49:8 L1 Regression with Lewis Weights Subsampling

which requires  $\frac{1}{\varepsilon^2}$  samples from most of the coordinates  $e_i$ . The  $\Omega(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$  instance sets  $P(y = 1 | x = e_i)$  to  $\frac{1}{2} + \varepsilon$  with the same bias for each  $i$ ; solving this is essentially distinguishing a  $\varepsilon$  biased coin from a  $-\varepsilon$ -biased coin. Finally, for  $\Omega(d \log \frac{1}{\delta})$  we set  $P(y = 1 | x = e_i) = 0$  except for a random hidden  $i^*$  with  $P(y = 1 | x = e_{i^*}) = \frac{3}{4}$ . Solving this instance requires finding  $i^*$ , but there's a  $\delta$  chance the first  $d \log \frac{1}{\delta}$  queries are all zero.

► **Theorem 16.** *For any  $d \geq 2$  and  $\varepsilon < \frac{1}{10}$ , there exist families  $\mathcal{X} \in \mathbb{R}^d, \mathcal{Y} \in \mathbb{R}$  of inputs and labels respectively such that any algorithm which solves Problem 2 with  $\delta < \frac{1}{4}$  requires at least  $m = \frac{3d}{2000\varepsilon^2}$  samples.*

We take  $\mathcal{X}$  to be the set of standard basis vectors, and the distribution over  $\mathcal{X}$  to be uniform. We will define a set  $\mathcal{B}$  as being a subset of the unit hypercube  $\{-1, 1\}^d$  such that every element is sufficiently far from every other.

▷ **Claim 17.** There is a set  $\mathcal{B} \subset \mathcal{H}$  with  $|\mathcal{B}| \geq 2^{0.2d}$  such that for any two  $\beta_1, \beta_2 \in \mathcal{B}$ , we have  $|\beta_1 - \beta_2| > 0.2d$

Proof. Here we just need an error correcting code with constant rate and constant relative (Hamming) distance. The existence of such a code follows from the Gilbert-Varshamov bound [10]. ◁

Fix some unknown  $\beta^*$ . We will have  $Y = ZX^\top \beta^*$  where  $Z$  is an independent random variable with probability  $\frac{1}{2} + \varepsilon$  of being 1, and  $\frac{1}{2} - \varepsilon$  of being  $-1$ . This completes our description of  $P$ . We define  $l(\beta)$  to be the  $\ell_1$  norm of the residuals for  $\beta$ , that is,  $l(\beta) = \mathbb{E}_{(X,Y) \sim P} [|X^\top \beta - Y|]$ . We have the following properties of  $l(\beta)$ .

▷ **Claim 18.** For  $D, \mathcal{B}$  as chosen above,  $l(\beta^*) = 1 - 2\varepsilon$ .

▷ **Claim 19.** For  $D, \mathcal{B}$  as chosen above, we have for all  $\beta \in \mathcal{B}$ ,  $l(\beta) - l(\beta^*) = \frac{2\varepsilon}{d} \|\beta - \beta^*\|_1$ .

**Proof of Theorem 16.** Suppose some algorithm returns  $\hat{\beta}$  with  $l(\hat{\beta}) < (1 + \frac{\varepsilon}{5})l(\beta^*) \implies \|\beta^* - \hat{\beta}\|_1 < 0.1d$  with probability  $\frac{3}{4}$ . By Fano's inequality,

$$H(\beta^* | \hat{\beta}) < H\left(\frac{1}{4}\right) + \frac{\log |\mathcal{B}| - 1}{4} < 0.05d,$$

and we have a lower bound on the mutual information between the output of our algorithm and the true parameter:  $I(\hat{\beta}; \beta^*) = H(\beta^*) - H(\beta^* | \hat{\beta}) \geq 0.15d$ . For an upper bound on the mutual information after seeing  $m$  samples, we use the data processing inequality.

$$\begin{aligned} I(\beta^*; \hat{\beta}) &\leq I(\beta^*; (Y_i)_{i \in [m]}) \leq \sum_{i=1}^m I(\beta^*; Y_i | (Y_j)_{j \in [i-1]}) \\ &= \sum_{i=1}^m H(Y_i | (Y_j)_{j \in [i-1]}) - H(Y_i | \beta^*, (Y_j)_{j \in [i-1]}) \\ &\leq \sum_{i=1}^m 1 - H(Y_i | \beta^*, I_i) \\ &\leq 4\varepsilon^2 m \end{aligned}$$

Here we have used that

$$\begin{aligned} H(Y_i | \beta^*, (Y_j)_{j \in [i-1]}) &\geq H(Y_i | \beta^*, I_i, (Y_j)_{j \in [i-1]}) \\ &= H(Y_i | \beta^*, I_i) \end{aligned}$$

and that the distribution of  $Y_i$  conditioned on  $\beta^*, I_i$  is just an independent Bernoulli with parameter  $\frac{1}{2} + \varepsilon$  and so

$$\begin{aligned} \sum_{i=1}^m 1 - H(Y_i|\beta^*, I_i) &\leq \sum_{i=1}^m \left[ 1 + \left(\frac{1}{2} + \varepsilon\right) \log\left(\frac{1}{2} + \varepsilon\right) + \left(\frac{1}{2} - \varepsilon\right) \log\left(\frac{1}{2} - \varepsilon\right) \right] \\ &\leq 4\varepsilon^2 m \end{aligned}$$

So  $0.15d \leq I(\beta^*; \hat{\beta}) \leq 4\varepsilon^2 m$ , and so we need  $m \geq \frac{3d}{80\varepsilon^2}$ . The result follows by replacing  $\varepsilon$  with  $5\varepsilon$ .  $\blacktriangleleft$

We can use the same instance to give a high probability lower bound of  $\Omega(\log \frac{1}{\delta}/\varepsilon^2)$ .

► **Theorem 20.** *For any  $d$  and  $\varepsilon < \frac{1}{10}$ , there exist sets  $\mathcal{X} \in \mathbb{R}, \mathcal{Y} \in \mathbb{R}$  of inputs and labels respectively, and a distribution  $P$  on  $\mathcal{X} \times \mathcal{Y}$  such that any algorithm which solves problem 2 requires at least  $m = \frac{1}{4\varepsilon^2} \log \frac{1}{\delta}$  samples.*

**Proof.** Consider two instances, denoted by subscripts (1) and (2) with  $\beta_{(1)}^* = -\mathbb{1}_d$  and  $\beta_{(2)}^* = \mathbb{1}_d$ , where  $\mathbb{1}_d \in \mathbb{R}^d$  is the all-ones vector. Denote by  $P_{(i)}$  the distribution over  $\mathcal{X}, \mathcal{Y}$  for instance (i), and let  $l_{\beta_{(i)}^*}(\beta) = \mathbb{E}_{(X,Y) \sim P_{(i)}}[|X^\top \beta - Y|]$  for  $i \in \{1, 2\}$ .

▷ **Claim 21.** For any  $\beta$ ,  $\max\{l_{\beta_{(1)}^*}(\beta) - l_{\beta_{(1)}^*}(\beta_{(1)}^*), l_{\beta_{(2)}^*}(\beta) - l_{\beta_{(2)}^*}(\beta_{(2)}^*)\} > 2\varepsilon$

From this claim together with Claim 18, we have for some  $i \in \{1, 2\}$ ,  $l_{\beta_{(i)}^*}(\beta) \geq (1 + 2\varepsilon)l_{\beta_{(i)}^*}(\beta_{(i)}^*)$ , for all  $\beta$ .

Denote by  $\hat{\beta}$  the output of the algorithm. Denote by  $\mathbb{P}_{(1)}$  the distribution over outputs by a algorithm interacting instance (1), and by  $\mathbb{P}_{(2)}$  the distribution over outputs by a algorithm interacting instance (2). Denote by  $A$  the event that  $l_{\beta_{(1)}^*}(\hat{\beta}) - l_{\beta_{(1)}^*}(\beta_{(1)}^*) \geq 2\varepsilon$ . Note that under  $A^c$ , we have  $l_{\beta_{(2)}^*}(\hat{\beta}) - l_{\beta_{(2)}^*}(\beta_{(2)}^*) \geq 2\varepsilon$ . Because the algorithm fails with probability at most  $\delta$  on any instance, we have  $2\delta \geq \mathbb{P}_{(1)}(A) + \mathbb{P}_{(2)}(A^c)$ . On the other hand,  $\mathbb{P}_{(1)}(A) + \mathbb{P}_{(2)}(A^c) \geq e^{-D(\mathbb{P}_{(1)}\|\mathbb{P}_{(2)})}$ . We can bound the KL-divergence of the two distributions as an aggregate KL-divergence over the course of acquiring the samples.

► **Theorem 22 (Lemma 15.1, [11]).** *If a learner interacts with two environments (1) and (2) through a policy  $\pi(\cdot|I_1, Y_1, I_2, Y_2, \dots, Y_{i-1})$  which dictates a distribution over actions  $I_i$  conditioned on the past  $(I_1, Y_1, \dots, Y_{i-1})$ , and sees label  $Y_i$  distributed according to some label distribution  $P_{(1), I_i}$  and  $P_{(2), I_i}$ , then the KL-divergence between the output of the learner on instance (1) and (2),  $\mathbb{P}_{(1)}$  and  $\mathbb{P}_{(2)}$  is given by*

$$D(\mathbb{P}_{(1)}\|\mathbb{P}_{(2)}) = \sum_{k=1}^d \mathbb{E}_{(1)} \left[ \sum_{i=1}^N \mathbb{1}\{I_i = k\} \cdot D(P_{(1), I_i}\|P_{(2), I_i}) \right]$$

Now,  $P_{(1), k}$  is a Bernoulli with parameter  $\frac{1}{2} + \varepsilon$ , and  $P_{(2), k}$  is a Bernoulli with parameter  $\frac{1}{2} - \varepsilon$ , so  $D(P_{(1), k}\|P_{(2), k}) \leq 16\varepsilon^2$ , and so we have

$$\begin{aligned} \sum_{k=1}^d \mathbb{E}_{(1)} \left[ \sum_{i=1}^N \mathbb{1}\{I_i = k\} \cdot D(P_{(1), I_i}\|P_{(2), I_i}) \right] &\leq \sum_{k=1}^d \mathbb{E}_{(1)} \left[ \sum_{i=1}^N \mathbb{1}\{I_i = k\} \cdot 16\varepsilon^2 \right] \\ &= 16\varepsilon^2 \cdot \mathbb{E}_{(1)} \left[ \sum_{k=1}^d \sum_{i=1}^N \mathbb{1}\{I_i = k\} \right] = 16\varepsilon^2 m \end{aligned}$$

Putting this together, we have  $\delta \geq e^{-16\varepsilon^2 m} \implies m \geq \frac{1}{16\varepsilon^2} \log \frac{1}{\delta}$ , and the result follows by replacing  $\varepsilon$  with  $\frac{1}{2}\varepsilon$ .  $\blacktriangleleft$

## 49:10 L1 Regression with Lewis Weights Subsampling

► **Theorem 23.** For any  $d \geq 2$ , there exist sets  $\mathcal{X} \in \mathbb{R}^d, \mathcal{Y} \in \mathbb{R}$  of inputs and labels, and a distribution  $P$  on  $\mathcal{X} \times \mathcal{Y}$  such that any algorithm which solves Problem 2, with  $\varepsilon = 1$ , requires at least  $m = \frac{d}{3} \log \frac{1}{8\delta}$  samples.

**Proof.** All logarithms are base 4. Consider instances in which  $\mathcal{X} = \{e_1, e_2, \dots, e_d\}$  where  $e_i$  denotes the  $i$ th standard basis vector and the distribution over  $\mathcal{X}$  is uniform. We take  $Y = ZX^\top \beta^*$  for some  $\beta^*$ , where  $Z$  is an independent Bernoulli random variable which is 1 with probability  $\frac{3}{4}$ , and 0 otherwise. Consider  $d$  instances labelled with subscripts (1), (2),  $\dots$ , ( $d$ ), one in which each of the  $d$  standard basis is  $\beta^*$ , that is,  $\beta_{(i)}^* = e_i$ . Denote by  $\beta_j$  the  $j$ th coordinate of  $\beta$ . For each instance, we have

▷ **Claim 24.** For all  $i \in [d], \beta \in \mathbb{R}^d$ , we have  $\ell_{\beta_{(i)}^*}(\beta) \geq \frac{1}{4d}$  with equality when  $\beta = \beta_{(i)}^*$

We would like our algorithm to return an estimate  $\hat{\beta}$  which satisfies  $\ell_{\beta^*}(\hat{\beta}) < \frac{1}{2d}$ . We first note that any choice of  $\beta$  only succeeds to be this close to the optimal on a single instance.

▷ **Claim 25.** Any  $\beta \in \mathbb{R}^d$  can only satisfy  $\ell_{\beta_{(i)}^*}(\hat{\beta}) < \frac{1}{2d}$  for one  $i \in [d]$ .

So, we may as well enforce that the algorithm return one of  $e_1, e_2, \dots, e_d$ , since any other output can be mapped to one of these to improve the performance of the algorithm.

We will allow our algorithm to sample  $N = \frac{d}{3} \log \frac{1}{\delta}$  rows total. Let  $\mathcal{E}$  be the event that  $Y_1, Y_2, \dots, Y_N$  are all zero. Given any algorithm  $\mathcal{A}$ , let  $F_{\mathcal{A}}$  denote the set of rows it samples fewer than  $\log \frac{1}{\delta}$  times with probability at least  $\frac{1}{2}$ , in event  $\mathcal{E}$ . Because the total number of rows sampled is  $\frac{d}{3} \log \frac{1}{\delta}$ , there must be at least  $\frac{2d}{3}$  rows which are sampled fewer than  $\frac{1}{2} \log \frac{1}{\delta}$  times in expectation.

By Markov's inequality, these rows are sampled fewer than  $\log \frac{1}{\delta}$  times with probability at least  $\frac{1}{2}$ , and are thus all in  $F_{\mathcal{A}}$ . Let  $B_{\mathcal{A}}$  denote the distribution over outputs  $\hat{\beta}$  of  $\mathcal{A}$  in event  $\mathcal{E}$ . Let  $i_{\mathcal{A}} = \arg \min_{j \in F_{\mathcal{A}}} B_{\mathcal{A}}(j)$ . Denote by  $G_{\mathcal{A}}$  the event that row  $i_{\mathcal{A}}$  is sampled fewer than  $\log \frac{1}{\delta}$  times; by construction we have  $\mathbb{P}(G_{\mathcal{A}}) > \frac{1}{2}$ .

The subscripts are explicit because  $F_{\mathcal{A}}, B_{\mathcal{A}}, i_{\mathcal{A}}, \mathbb{P}[G_{\mathcal{A}}]$  are properties of the algorithm and are independent of the instance with which it interacts. Consider the performance of this algorithm against the instance  $\beta_{(i_{\mathcal{A}})}^*$ .

Let  $Y_{(i_{\mathcal{A}}), j, k}$  denote the label returned to the algorithm when it queries  $e_j$  for the  $k$ th time. Let  $T_{(i_{\mathcal{A}})} = \min\{t | Y_{(i_{\mathcal{A}}), i_{\mathcal{A}}, t} = 1\}$ . Denote by  $E_{(i_{\mathcal{A}})}$  the event that  $T_{(i_{\mathcal{A}})} > \log \frac{1}{\delta}$ . Because  $T_{(i_{\mathcal{A}})}$  is a geometric random variable, we have  $\mathbb{P}[E_{(i_{\mathcal{A}})}] > \delta$ .

Now condition on the event  $G_{\mathcal{A}} \cap E_{i_{\mathcal{A}}}$ , which is an event with probability  $\frac{1}{2}\delta$ . Here our algorithm samples  $i_{\mathcal{A}}$  fewer than  $T_{i_{\mathcal{A}}}$  times, so it never sees a 1 and its output distribution is  $B_{\mathcal{A}}$ . It returns  $i \in F_{\mathcal{A}} \setminus \{i_{\mathcal{A}}\}$  with probability at least  $1 - B_{\mathcal{A}}(i_{\mathcal{A}}) \geq 1 - \frac{1}{|F_{\mathcal{A}}|} \geq 1 - \frac{3}{2d} \geq \frac{1}{4}$ . In summary, even after  $\frac{d}{3} \log \frac{1}{\delta}$  queries, no algorithm can return  $\hat{\beta}$  with  $\|X\hat{\beta} - y\| < (1 + \varepsilon)\|X\beta^* - y\|$  with probability greater than  $\frac{1}{8}\delta$ . The result follows by replacing  $\delta$  by  $8\delta$ . ◀

Putting these together we have:

► **Theorem 26.** For any  $d \geq 2, \varepsilon < \frac{1}{10}, \delta < \frac{1}{4}$ , there exist sets  $\mathcal{X} \in \mathbb{R}^d, \mathcal{Y} \in \mathbb{R}$  of inputs and labels, and a distribution  $P$  on  $\mathcal{X} \times \mathcal{Y}$  such that any algorithm which solves Problem 2, with  $\varepsilon = 1$ , requires at least  $m = \Omega(\frac{d}{\varepsilon^2} + \frac{1}{\varepsilon^2} \log \frac{1}{\delta} + d \log \frac{1}{\delta})$  samples.

► **Corollary 27.** Any algorithm that solves Problem 1 takes at least  $\Omega(d \log \frac{1}{\delta} + \frac{d}{\varepsilon^2} + \frac{1}{\varepsilon^2} \log \frac{1}{\delta})$  samples for some  $n = O(\frac{d \log \frac{4}{\delta}}{\varepsilon})$ .

**Proof.** Each of the instances that demonstrate the lower bounds above, in Lemmas 16, 20, and 23, take  $|\mathcal{X}| = d$ , the results follows from Lemma 14. ◀

## 4 Proof of Theorem 11

► **Lemma 28.** *Let  $X \in \mathbb{R}^{n \times d}$  have  $\ell_1$  Lewis weights  $\{w_i\}_{i \in [n]}$ . Then, for any  $N$  that is at least  $O\left(\frac{d}{\varepsilon^2} \log \frac{d}{\varepsilon \delta}\right)$ , there is a sampling-and-reweighting distribution  $\mathcal{S}(\{p_i\}_{i=1}^n)$  satisfying  $\sum_i p_i = N$  such that for all  $y$ , if  $S \sim \mathcal{S}(\{p_i\}_{i=1}^n)$  and  $\beta^* = \arg \min \|X\beta - y\|_1$ , we have for all  $\beta$*

$$(\|SX\beta^* - Sy\|_1 - \|SX\beta - Sy\|_1) - (\|X\beta^* - y\|_1 - \|X\beta - y\|_1) \leq \varepsilon \cdot \|X\beta^* - X\beta\|_1 \quad (9)$$

with probability at least  $1 - \delta$ . Further, for constant  $\delta$ ,  $m = O(d \log d / \varepsilon^2)$  rows suffice.

This lemma is proved for high probability in Section 4.1, and for constant probability in the full version of this paper, [14]. Given this, we can prove the main theorem.

**Proof of Theorem 11.** Applying Lemma 28 to  $\hat{\beta} := \arg \min \|SX\beta - Sy\|_1$ , we get

$$\left(\|SX\beta^* - Sy\|_1 - \|SX\hat{\beta} - Sy\|_1\right) \leq \left(\|X\beta^* - y\|_1 - \|X\hat{\beta} - y\|_1\right) + \varepsilon \cdot \|X\beta^* - X\hat{\beta}\|_1$$

Since  $\hat{\beta}$  is the minimizer of  $\|SX\beta - Sy\|_1$ , the left side is non-negative. So,

$$\begin{aligned} \|X\hat{\beta} - y\|_1 &\leq \|X\beta^* - y\|_1 + \varepsilon \cdot \|X\beta^* - X\hat{\beta}\|_1 \\ &\leq \|X\beta^* - y\|_1 + \varepsilon \cdot (\|X\beta^* - y\|_1 + \|X\hat{\beta} - y\|_1) \end{aligned}$$

Rearranging, and assuming  $\varepsilon < 1/2$ ,

$$\begin{aligned} \|X\hat{\beta} - y\|_1 &\leq \frac{1 + \varepsilon}{1 - \varepsilon} \|X\beta^* - y\|_1 \\ &\leq (1 + 4\varepsilon) \|X\beta^* - y\|_1 \end{aligned}$$

Using  $\varepsilon' = \varepsilon/4$  proves the theorem. ◀

### 4.1 Proof of Lemma 28

This argument is similar to that in Appendix B of [4]. In order to prove Lemma 28, by Markov's inequality, it is sufficient to show that for some  $l$ ,

$$M := \mathbb{E}_S \left[ \left( \max_{\|X\beta^* - X\beta\|_1=1} |(\|SX\beta^* - Sy\|_1 - \|SX\beta - Sy\|_1) - (\|X\beta^* - y\|_1 - \|X\beta - y\|_1)| \right)^l \right] \leq \varepsilon^l \delta$$

To show this, we will symmetrize, then use a contraction lemma to cancel the  $y$  terms. Then, with all the terms being within the column space of  $SX$ , we use the fact that  $S$  is a subspace embedding with high probability. We present two different bounds, one used for the constant probability and one for the high probability cases, but the following intermediate bound is the same for the two:

► **Lemma 29.** *Given a matrix  $X \in \mathbb{R}^{n \times d}$ , let  $\mathcal{S}(\{p_i\}_{i \in [n]})$  be any sampling-and-reweighting distribution, and let  $i_k$  be the row-indices chosen by this sampling matrix such that  $S_{k,i_k} = \frac{1}{p_{i_k}}$ . Let  $\sigma_k$  be independent Rademacher variables that are  $\pm 1$  each with probability 0.5. Then,*

$$M \leq 2^l \mathbb{E}_{S, \sigma} \left[ \left( \max_{\|X\beta^* - X\beta\|_1=1} \left| \sum_k \sigma_k \left( \frac{|x_{i_k}^\top \beta^* - y_{i_k}|}{p_{i_k}} - \frac{|x_{i_k}^\top \beta - y_{i_k}|}{p_{i_k}} \right) \right| \right)^l \right] \quad (10)$$

## 49:12 L1 Regression with Lewis Weights Subsampling

This is essentially standard symmetrization; the proof is in Appendix B. To simplify the expression and eliminate the terms involving the labels, we then use a theorem from [12]:

► **Lemma 30** ([12] Theorem 5). *Let  $\Phi : \mathbb{R}^+ \rightarrow \mathbb{R}^+$  be convex and increasing, and let  $\phi_k : \mathbb{R} \rightarrow \mathbb{R}$  be contractions such that  $\phi_k(0) = 0$  for all  $k$ . Let  $\mathcal{F}$  be a class of functions on  $\{1, 2, 3, \dots, n\}$ , and  $\|g(f)\|_{\mathcal{F}} = \sup_{f \in \mathcal{F}} |g(f)|$ . Then,*

$$\mathbb{E}_{\sigma} \left[ \Phi \left( \frac{1}{2} \left\| \sum_k \sigma_k \phi_k(f(k)) \right\|_{\mathcal{F}} \right) \right] \leq \frac{3}{2} \mathbb{E}_{\sigma} \left[ \Phi \left( \left\| \sum_k \sigma_k f(k) \right\|_{\mathcal{F}} \right) \right]$$

► **Lemma 31.** *For any  $y \in \mathbb{R}^n$ , we have*

$$\begin{aligned} & \mathbb{E}_{S, \sigma} \left[ \left( \max_{\|X\beta^* - X\beta\|_1=1} \left| \sum_k \sigma_k \left( \frac{|x_{i_k}^{\top} \beta^* - y_{i_k}|}{p_{i_k}} - \frac{|x_{i_k}^{\top} \beta - y_{i_k}|}{p_{i_k}} \right) \right| \right)^l \right] \\ & \leq 2^{l+1} \mathbb{E}_{S, \sigma} \left[ \left( \max_{\|X\beta^* - X\beta\|_1=1} \left| \sum_k \sigma_k \left( \frac{x_{i_k}^{\top} \beta^* - x_{i_k}^{\top} \beta}{p_{i_k}} \right) \right| \right)^l \right] \end{aligned} \quad (11)$$

**Proof.** We take  $\Phi(x) = x^l$ , which is convex and increasing for  $l > 1$ , let  $\mathcal{F}$  be the set of functions  $f_{\beta}$  where  $f_{\beta}(k) = \frac{x_{i_k}^{\top} \beta^* - x_{i_k}^{\top} \beta}{p_{i_k}}$  and  $\beta$  satisfies  $\|X\beta^* - X\beta\|_1 = 1$ , and let  $\phi_k$  be defined as

$$\phi_k(z) = \frac{|x_{i_k}^{\top} \beta^* - y_{i_k}|}{p_{i_k}} - \frac{|x_{i_k}^{\top} \beta^* - zp_{i_k} - y_{i_k}|}{p_{i_k}}.$$

This satisfies

$$\phi_k(f_{\beta}(k)) = \phi_k \left( \frac{x_{i_k}^{\top} \beta^* - x_{i_k}^{\top} \beta}{p_{i_k}} \right) = \frac{|x_{i_k}^{\top} \beta^* - y_{i_k}|}{p_{i_k}} - \frac{|x_{i_k}^{\top} \beta - y_{i_k}|}{p_{i_k}}.$$

This is a contraction, since

$$\begin{aligned} |\phi_k(z_1) - \phi_k(z_2)| &= \left| \frac{|x_{i_k}^{\top} \beta^* - z_2 p_{i_k} - y_{i_k}|}{p_{i_k}} - \frac{|x_{i_k}^{\top} \beta^* - z_1 p_{i_k} - y_{i_k}|}{p_{i_k}} \right| \\ &\leq \frac{|z_1 p_{i_k} - z_2 p_{i_k}|}{p_{i_k}} \leq |z_1 - z_2| \end{aligned}$$

Applying Lemma 30 with these parameters, we have

$$\begin{aligned} & \mathbb{E}_{\sigma} \left[ \left( \frac{1}{2} \max_{\|X\beta^* - X\beta\|_1=1} \left| \sum_k \sigma_k \left( \frac{|x_{i_k}^{\top} \beta^* - y_{i_k}|}{p_{i_k}} - \frac{|x_{i_k}^{\top} \beta - \text{true}y_{i_k}|}{p_{i_k}} \right) \right| \right)^l \right] \\ & \leq \frac{3}{2} \mathbb{E}_{\sigma} \left[ \left( \max_{\|X\beta^* - X\beta\|_1=1} \left| \sum_k \sigma_k \left( \frac{x_{i_k}^{\top} \beta^* - x_{i_k}^{\top} \beta}{p_{i_k}} \right) \right| \right)^l \right] \end{aligned}$$

After taking the expectation with respect to  $S$  and multiplying both sides by  $2^l$ , this gives the statement of the lemma. ◀

From here, we use two separate results to show the appropriate row counts for the constant and high probability cases. The constant probability case is left for the full version of this paper, [14].

For high probability row-counts, we use a lemma from [4]:



► **Lemma 32** (8.2, 8.3, 8.4 in [4]). *There exists constant  $C$  such that for any  $X \in \mathbb{R}^{n \times d}$  with all  $\ell_1$  Lewis weights less than  $C \frac{\varepsilon^2}{\log(\frac{n}{\delta})}$  and  $l = \log(2n/\delta)$ , then*

$$\mathbb{E}_\sigma \left[ \left( \max_{\|X\beta\|_1=1} \left| \sum_{i=1}^n \sigma_i x_i^\top \beta \right| \right)^l \right] \leq \frac{\varepsilon^l \delta}{2} \quad (12)$$

We want a similar statement, but for arbitrary matrices, with no bounds placed on the Lewis weights. To do this, we construct a new, related matrix using the following lemma, which is proved in Appendix B:

► **Lemma 33** (Similar to [4] Lemma B.1). *Let  $X$  be any matrix, and let  $W$  be the matrix that has the Lewis weights of  $X$  in the diagonal entries. Let  $N \geq \frac{d}{\varepsilon^2} \log \frac{d}{\varepsilon \delta}$ . There exist constants  $C_1, C_2, C_3$  such that we can construct a matrix  $X'$  such that*

- $X'$  has  $C_1 d N$  rows,
- $X'^\top W'^{-1} X' \succeq X^\top W^{-1} X$ , (where  $W'$  is the matrix that has the Lewis weights of  $X'$  in the diagonal entries),
- $\|X'\beta\|_1 \leq C_2 \|X\beta\|_1$  for all  $\beta$ ,
- the Lewis weights of  $X'$  are bounded by  $\frac{C_3}{N}$ .

► **Lemma 34.** *Consider  $X \in \mathbb{R}^{n \times d}$  with  $\ell_1$  Lewis weights  $w_i$ . Let  $p_i$  be some set of sampling values such that  $N = \sum_i p_i$  and, for some constants  $C, C_1, C_4$ ,*

$$p_i \geq \frac{\log\left(\frac{N+C_1 Nd}{\delta}\right)}{C\varepsilon^2} w_i$$

*Then, if  $N \geq C_4 \frac{d}{\varepsilon^2} \log \frac{d}{\varepsilon \delta}$  and if  $S \sim \mathcal{S}(\{p_i\}_{i \in [n]})$ , then*

$$\mathbb{E}_{S, \sigma} \left[ \left( \max_{\|X\beta\|_1=1} \left| \sum_{k=1}^N \sigma_k \frac{x_{i_k}^\top \beta}{p_{i_k}} \right| \right)^l \right] \leq \frac{\varepsilon^l \delta}{2} \quad (13)$$

**Proof of Lemma 34.** Ideally the Lewis weights of  $SX$  would be bounded by  $C \frac{\varepsilon^2}{\log \frac{N}{\delta}}$  and we could directly apply Lemma 32 to  $SX$  to obtain a bound on the moment. However, we do not know this. Instead, we first construct  $X'$  using  $X$  as described in Lemma 33. We then construct a new matrix  $X''$  by stacking  $X'$  on top of  $SX$ . Define  $W''$  to be the diagonal matrix consisting of the  $\ell_1$  Lewis weights of  $X''$ . Define, for convenience,  $R = N + C_1 Nd$ , which is the number of rows  $X''$  has.

We can bound the term on the left side of (13) by the same term, summing over the rows of  $X''$  instead. That is,

$$\mathbb{E}_{S, \sigma} \left[ \left( \max_{\|X\beta\|_1=1} \left| \sum_{k=1}^N \sigma_k \frac{x_{i_k}^\top \beta}{p_{i_k}} \right| \right)^l \right] \leq \mathbb{E}_{S, \sigma} \left[ \left( \max_{\|X\beta\|_1=1} \left| \sum_{i=1}^R \sigma_i x_i''^\top \beta \right| \right)^l \right]$$

Our goal is to apply Lemma 32 to the right side. To do this, we need to show the correct bound on its Lewis weights, and then have the term be a maximum over  $\|X''\beta\|_1 = 1$ , rather than  $\|X\beta\|_1 = 1$ .

## 49:14 L1 Regression with Lewis Weights Subsampling

**Bounding the Lewis weights of  $X''$ .** By Lemma 9, the  $\ell_1$  Lewis weights of a matrix do not increase when more rows are added. So, the rows in  $X''$  that are from  $X'$  have Lewis weights that are bounded above by  $C_3 \frac{\varepsilon^2}{\log(\frac{d}{\varepsilon\delta})}$ . Further,

$$\begin{aligned} X''^\top W''^{-1} X'' &= \sum_{i=1}^R \frac{1}{w_i''} x_i'' (x_i'')^\top \\ &\succeq \sum_{i=1}^{R-N} \frac{1}{w_k''} x_k'' (x_k'')^\top && \text{since } \sum_{i=kC_1 d^2+1}^N \frac{1}{w_i''} x_i'' (x_i'')^\top \succeq 0 \\ &= X'^\top W'^{-1} X' \succeq X^\top W^{-1} X. \end{aligned}$$

So, any row  $y_i = x_i/p_i$  in  $X''$  that is from  $SX$  satisfies

$$\begin{aligned} w_i''^2 &= y_i^\top (X''^\top W''^{-1} X'')^{-1} y_i \leq y_i^\top (X^\top W^{-1} X)^{-1} y_i = \frac{1}{p_i^2} x_i^\top (X^\top W^{-1} X)^{-1} x_i \\ &\leq \left( \frac{C\varepsilon^2}{\log(\frac{R}{\delta})} \frac{1}{w_i} \right)^2 \cdot w_i^2 = \left( \frac{C\varepsilon^2}{\log(\frac{R}{\delta})} \right)^2 \end{aligned}$$

which means that all of the Lewis weights of  $X''$  are less than the larger of  $C \frac{\varepsilon^2}{\log(\frac{R}{\delta})}$  and  $C_3 \frac{\varepsilon^2}{\log(\frac{d}{\varepsilon\delta})}$ . Now, for small enough  $\varepsilon, \delta$ ,  $\log \frac{R}{\delta} \leq \frac{C}{C_3} \log \frac{d}{\varepsilon\delta}$ , we have the Lewis weight upper bound for all rows of  $X''$  is  $C \frac{\varepsilon^2}{\log(\frac{R}{\delta})}$

**Renormalizing to maximize over  $\|X''\beta\|_1 = 1$ .** If we define the following

$$F := \max_{\|X\beta\|_1=1} \left| \|SX\beta\|_1 - \|X\beta\|_1 \right|$$

then,

$$\|X''\beta\|_1 = \|SX\beta\|_1 + \|X'\beta\|_1 \leq (1 + C_2 + F)\|X\beta\|_1$$

So, we get

$$\begin{aligned} \left( \max_{\|X\beta\|_1=1} \left| \sum_{k=1}^R \sigma_k x_k''^\top \beta \right| \right)^l &\leq (1 + C_2 + F)^l \left( \max_{\|X''\beta\|_1=1} \left| \sum_{k=1}^R \sigma_k x_k''^\top \beta \right| \right)^l \\ &\leq 2^{l-1} ((1 + C_2)^l + F^l) \left( \max_{\|X''\beta\|_1=1} \left| \sum_{k=1}^R \sigma_k x_k''^\top \beta \right| \right)^l \end{aligned}$$

Taking expectations of either side over just the Rademacher variables,

$$\mathbb{E}_\sigma \left[ \left( \max_{\|X\beta\|_1=1} \left| \sum_{k=1}^R \sigma_k x_k''^\top \beta \right| \right)^l \right] \leq 2^{l-1} ((1 + C_2)^l + F^l) \mathbb{E}_\sigma \left[ \left( \max_{\|X''\beta\|_1=1} \left| \sum_{k=1}^R \sigma_k x_k''^\top \beta \right| \right)^l \right]$$

**Applying Lemma 32 to  $X''$ .** Since  $X''$  has  $R$  rows, and the correct Lewis weight bound, we can simply apply Lemma 32 to the right side above

$$\mathbb{E}_\sigma \left[ \left( \max_{\|X\beta\|_1=1} \left| \sum_{k=1}^R \sigma_k x_k''^\top \beta \right| \right)^l \right] \leq 2^{l-1} ((1 + C_2)^l + F^l) \frac{\varepsilon^l \delta}{2}$$

Now, by Lemma 12, we know that  $\mathbb{E}_S[F^l] \leq \varepsilon^l \delta$ . So, taking the expectation with respect to the sampling matrices of either side of the above, we get, for small enough  $\varepsilon, \delta$ ,

$$\mathbb{E}_{S, \sigma} \left[ \left( \max_{\|X\beta\|=1} \left| \sum_{k=1}^{kC_1 d^2 + N} \sigma_k x_k''^\top \beta \right| \right)^l \right] \leq 2^{l-1} ((1 + C_2)^l + \varepsilon^l \delta) \frac{\varepsilon^l \delta}{2} \leq 2^l (1 + C_2)^l \frac{\varepsilon^l \delta}{2}$$

So, solving the problem for  $\varepsilon' = \frac{\varepsilon}{2+2C_2}$  gives the correct bound.  $\blacktriangleleft$

Finally, we can show Lemma 28

**Proof of Lemma 28.** Take  $l = \log(2n/\delta)$ ,  $N = 5 \frac{(1+C_1)C_3}{C} \frac{d}{\varepsilon^2} \log \frac{d}{\varepsilon \delta}$ . Then, we apply Lemma 29, Lemma 31, and Lemma 34 to get

$$M \leq 2^{2l} \varepsilon^l \delta$$

which, solving the problem for  $\varepsilon/4$ , gives the correct bound. Then, applying Markov's inequality, we get that with probability  $\delta$ ,

$$\max_{\|X\beta^* - X\beta\|=1} |(\|SX\beta^* - Sy\|_1 - \|SX\beta - Sy\|_1) - (\|X\beta^* - y\|_1 - \|X\beta - y\|_1)| \leq \varepsilon$$

Finally, scaling up appropriately gives, in generality,

$$|(\|SX\beta^* - Sy\|_1 - \|SX\beta - Sy\|_1) - (\|X\beta^* - y\|_1 - \|X\beta - y\|_1)| \leq \varepsilon \|X\beta^* - X\beta\|_1 \quad \blacktriangleleft$$

---

## References

- 1 Xue Chen and Michał Dereziński. Query complexity of least absolute deviation regression via robust uniform convergence. *arXiv preprint arXiv:2102.02322*, 2021.
- 2 Xue Chen and Eric Price. Active regression via linear-sample sparsification. 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 663–695, Phoenix, USA, June 25–28 2019. PMLR. URL: <http://proceedings.mlr.press/v99/chen19a.html>.
- 3 Kenneth L. Clarkson. Subgradient and sampling algorithms for  $\ell_1$  regression. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, page 257–266, USA, 2005. Society for Industrial and Applied Mathematics.
- 4 Michael B. Cohen and Richard Peng. Lp row sampling by lewis weights. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, page 183–192, New York, NY, USA, 2015. Association for Computing Machinery. doi:10.1145/2746539.2746567.
- 5 Anirban Dasgupta, Petros Drineas, Boulos Harb, Ravi Kumar, and Michael W Mahoney. Sampling algorithms and coresets for  $\ell_p$  regression. *SIAM Journal on Computing*, 38(5):2060–2078, 2009.
- 6 Michał Dereziński and Michael W Mahoney. Determinantal point processes in randomized numerical linear algebra. *Notices of the American Mathematical Society*, 68(1), 2021.
- 7 Michał Dereziński and Manfred K Warmuth. Unbiased estimates for linear regression via volume sampling. *arXiv preprint arXiv:1705.06908*, 2017.
- 8 Petros Drineas, Michael W Mahoney, and Shan Muthukrishnan. Sampling algorithms for  $\ell_2$  regression and applications. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1127–1136, 2006.
- 9 David Durfee, Kevin A. Lai, and Saurabh Sawlani.  $\ell_1$  regression using lewis weights preconditioning and stochastic gradient descent. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 1626–1656. PMLR, July 06–09 2018. URL: <http://proceedings.mlr.press/v75/durfee18a.html>.

- 10 E. N. Gilbert. A comparison of signalling alphabets. *The Bell System Technical Journal*, 31(3):504–522, 1952. doi:10.1002/j.1538-7305.1952.tb01393.x.
- 11 T. Lattimore and C. Szepesvari. *Bandit Algorithms*. Cambridge University Press, 2020.
- 12 M. Ledoux and M. Talagrand. Comparison theorems, random geometry and some limit theorems for empirical processes. *Ann. Probab.*, 17(2):596–631, April 1989. doi:10.1214/aop/1176991418.
- 13 Michael W. Mahoney. Randomized algorithms for matrices and data. *Found. Trends Mach. Learn.*, 3(2):123–224, 2011. doi:10.1561/22000000035.
- 14 Aditya Parulekar, Advait Parulekar, and Eric Price. L1 regression with lewis weights subsampling. *CoRR*, abs/2105.09433, 2021. arXiv:2105.09433.
- 15 Michel Talagrand. Embedding subspaces of  $\ell_1$  into  $\ell_n$ . *Proceedings of the American Mathematical Society*, 108(2):363–369, 1990. URL: <http://www.jstor.org/stable/2048283>.
- 16 Michel Talagrand. Embedding subspaces of  $\ell_p$  in  $\ell_{pn}$ . In J. Lindenstrauss and V. Milman, editors, *Geometric Aspects of Functional Analysis*, pages 311–326, Basel, 1995. Birkhäuser Basel.
- 17 David P. Woodruff. Sketching as a tool for numerical linear algebra. *Found. Trends Theor. Comput. Sci.*, 10(1–2):1–157, 2014. doi:10.1561/04000000060.

## A

 Constant-factor approximation

► **Theorem 35.** Let  $X \in \mathbb{R}^{n \times d}$  have  $\ell_1$  Lewis weights  $\{w_i\}_{i \in [n]}$ . Then, for any  $N$  that is at least  $O(d \log d)$ , there is a sampling-and-reweighting distribution  $\mathcal{S}(\{p_i\}_{i=1}^n)$  satisfying  $\sum_i p_i = N$  such that for all  $y$ , if  $S \sim \mathcal{S}(\{p_i\}_{i=1}^n)$  and  $\hat{\beta} = \arg \min \|SX\beta - Sy\|_1$ , we have

$$\|X\hat{\beta} - y\|_1 \leq 41 \min_{\beta} \|X\beta - y\|_1$$

with probability 0.9.

**Proof.** Since we just want a constant factor approximation, we can take  $\mathcal{S}$  to be the distribution over constant probability Lewis weight  $\ell_1$ -subspace embeddings, so that  $\|X\beta\|_1 \leq 2\|SX\beta\|_1$  with probability at least 0.9. We have

$$\begin{aligned} \|X\hat{\beta} - y\|_1 &\leq \|X\hat{\beta} - X\beta^*\|_1 + \|X\beta^* - y\|_1 \\ &\leq 2\|SX\hat{\beta} - SX\beta^*\|_1 + \|X\beta^* - y\|_1 \\ &\leq 2(\|SX\hat{\beta} - Sy\|_1 + \|SX\beta^* - Sy\|_1) + \|X\beta^* - y\|_1 \\ &\leq 4(\|SX\beta^* - Sy\|_1) + \|X\beta^* - y\|_1 \end{aligned}$$

where in the last inequality, we have used the fact that  $\hat{\beta}$  is the minimizer of  $\|SX\beta - Sy\|_1$ . Now, by Markov's inequality, with probability 0.9,  $\|SX\beta^* - Sy\|_1 \leq 10\|X\beta^* - y\|_1$ . So, we have with probability 0.81,

$$\|X\hat{\beta} - y\|_1 \leq 41\|X\beta^* - y\|_1$$

Since we only used a constant-factor subspace embedding, the row count would be  $O(d \log d)$ . ◀

## B

 Proofs of Lemmas

► **Lemma 10.** Let  $X \in \mathbb{R}^{n \times d}$ , and let  $X' \in \mathbb{R}^{kn \times d}$  be  $X$  stacked on itself  $k$  times, with each row scaled down by  $k$ . Then, each of the Lewis weights is reduced by a factor of  $k$ .

**Proof.** Let  $\{w_i\}_{i=1}^n$  be the Lewis weights of  $X$ , and let  $\{w'_i\}_{i=1}^{kn}$  be the Lewis weights of  $X'$ . Let  $x_i$  be the  $i$ th row of  $X$ , and similarly let  $x'_i$  be the  $i$ th row of  $X'$ . Let the ordering of the rows be such that  $x'_{jn+i} = \frac{1}{k}x_i$  for  $0 \leq j < k$ . Let  $W$  be the diagonal matrix where  $W_{ii} = w_i$ . Since Lewis weights are defined circularly, we just need to check that the suggested weights work, and by uniqueness, they will be correct.

We know that  $w_i^2 = x_i^\top (X^\top W^{-1}X)^{-1}x_i$ . Therefore, if we take  $W'$  to be the diagonal matrix of size  $kn \times kn$ , and set the diagonal entries to be the Lewis weights of  $X$  divided by  $k$ , repeated  $k$  times, then we have

$$X'^\top W'^{-1}X' = \sum_{i=1}^{kn} \frac{1}{w'_i} x'_i x'^\top_i = \sum_{i=1}^{kn} \frac{k}{w_i} x'_i x'^\top_i = k \sum_{i=1}^n \frac{k}{w_i} \cdot \frac{1}{k^2} x_i x_i^\top$$

In the last expression above, we are only summing over the first set of rows in  $X'$ , which are the scaled rows of  $X$ , and then multiplying by  $k$  since they are repeated  $k$  times. Now,

$$k \sum_{i=1}^n \frac{k}{w_i} \cdot \frac{1}{k^2} x_i x_i^\top = \sum_{i=1}^n \frac{1}{w_i} x_i x_i^\top = X^\top W^{-1}X$$

So, finally, for an arbitrary row  $x'_{jn+i}$ , which corresponds to row  $x_i$  in the original matrix, we get its Lewis weight:

$$w'^2_{jn+i} = x'^\top_{jn+i} (X'^\top W'^{-1}X')^{-1} x'_{jn+i} = \frac{1}{k^2} x_i^\top (X^\top W^{-1}X)^{-1} x_i = \frac{w_i^2}{k^2}$$

which proves that our suggested Lewis weights are consistent.  $\blacktriangleleft$

**► Lemma 29.** Given a matrix  $X \in \mathbb{R}^{n \times d}$ , let  $\mathcal{S}(\{p_i\}_{i \in [n]})$  be any sampling-and-reweighting distribution, and let  $i_k$  be the row-indices chosen by this sampling matrix such that  $S_{k,i_k} = \frac{1}{p_{i_k}}$ . Let  $\sigma_k$  be independent Rademacher variables that are  $\pm 1$  each with probability 0.5. Then,

$$M \leq 2^l \mathbb{E}_{S,\sigma} \left[ \left( \max_{\|X\beta^* - X\beta\|=1} \left| \sum_k \sigma_k \left( \frac{|x_{i_k}^\top \beta^* - y_{i_k}|}{p_{i_k}} - \frac{|x_{i_k}^\top \beta - y_{i_k}|}{p_{i_k}} \right) \right| \right)^l \right] \quad (10)$$

**Proof.** We proceed by symmetrization. Since the matrix  $S$  scales the rows by the probability they are picked with, the expectation of  $\|SM\beta\|_1$  is just  $\|M\beta\|_1$ , for any matrix  $M$  and vector  $\beta$ . So, adding or subtracting the same term with a different sampling matrix  $S'$ ,  $(\|S'X\beta^* - S'y\|_1 - \|S'X\beta - S'y\|_1) - (\|X\beta^* - y\|_1 - \|X\beta - y\|_1)$ , is just adding a mean zero term, and since taking the  $l$ th power of a maximum is convex, this can only increase the expectation. That is,

$$\begin{aligned} & \mathbb{E}_{S,S'} \left[ \left( \max_{\|X\beta^* - X\beta\|=1} |(\|SX\beta^* - Sy\|_1 - \|SX\beta - Sy\|_1) - (\|X\beta^* - y\|_1 - \|X\beta - y\|_1)| \right)^l \right] \\ & \leq \mathbb{E}_{S,S'} \left[ \left( \max_{\|X\beta^* - X\beta\|=1} |((\|SX\beta^* - Sy\|_1 - \|SX\beta - Sy\|_1) - (\|X\beta^* - y\|_1 - \|X\beta - y\|_1)) \right. \right. \\ & \quad \left. \left. - ((\|S'X\beta^* - S'y\|_1 - \|S'X\beta - S'y\|_1) - (\|X\beta^* - y\|_1 - \|X\beta - y\|_1)) \right| \right)^l \right] \end{aligned}$$

So, we can bound  $M$  as

$$M \leq \mathbb{E}_{S,S'} \left[ \left( \max_{\|X\beta^* - X\beta\|=1} |(\|SX\beta^* - Sy\|_1 - \|SX\beta - Sy\|_1) - (\|S'X\beta^* - S'y\|_1 - \|S'X\beta - S'y\|_1)| \right)^l \right]$$

## 49:18 L1 Regression with Lewis Weights Subsampling

Let  $i_k$  be the indices chosen by  $S$ , and  $i'_k$  the indices chosen by  $S'$ . Rewriting this as a sum,

$$M \leq \mathbb{E}_{S, S'} \left[ \left( \max_{\|X\beta^* - X\beta\|=1} \left| \sum_k \left( \frac{|x_{i_k}^\top \beta^* - y_{i_k}|}{p_{i_k}} - \frac{|x_{i_k}^\top \beta - y_{i_k}|}{p_{i_k}} \right) - \sum_k \left( \frac{|x_{i'_k}^\top \beta^* - y_{i'_k}|}{p_{i'_k}} - \frac{|x_{i'_k}^\top \beta - y_{i'_k}|}{p_{i'_k}} \right) \right| \right)^l \right]$$

Now, since  $i_k$  and  $i'_k$  are independent and identically distributed, randomly swapping elements from either sum does not change the distribution. This amounts to adding a random sign  $\sigma_k$  to the terms, where  $\sigma_k = \pm 1$  independently with probability  $1/2$ . So,

$$\begin{aligned} M &\leq \mathbb{E}_{S, S', \sigma} \left[ \left( \max_{\|X\beta^* - X\beta\|=1} \left| \sum_k \sigma_k \left( \frac{|x_{i_k}^\top \beta^* - y_{i_k}|}{p_{i_k}} - \frac{|x_{i_k}^\top \beta - y_{i_k}|}{p_{i_k}} \right) - \sum_k \sigma_k \left( \frac{|x_{i'_k}^\top \beta^* - y_{i'_k}|}{p_{i'_k}} - \frac{|x_{i'_k}^\top \beta - y_{i'_k}|}{p_{i'_k}} \right) \right| \right)^l \right] \\ &\leq \mathbb{E}_{S, S', \sigma} \left[ \left( \max_{\|X\beta^* - X\beta\|=1} \left| \sum_k \sigma_k \left( \frac{|x_{i_k}^\top \beta^* - y_{i_k}|}{p_{i_k}} - \frac{|x_{i_k}^\top \beta - y_{i_k}|}{p_{i_k}} \right) \right| + \max_{\|X\beta^* - X\beta\|=1} \left| \sum_k \sigma_k \left( \frac{|x_{i'_k}^\top \beta^* - y_{i'_k}|}{p_{i'_k}} - \frac{|x_{i'_k}^\top \beta - y_{i'_k}|}{p_{i'_k}} \right) \right| \right)^l \right] \\ &\leq 2^l \mathbb{E}_{S, \sigma} \left[ \left( \max_{\|X\beta^* - X\beta\|=1} \left| \sum_k \sigma_k \left( \frac{|x_{i_k}^\top \beta^* - y_{i_k}|}{p_{i_k}} - \frac{|x_{i_k}^\top \beta - y_{i_k}|}{p_{i_k}} \right) \right| \right)^l \right] \end{aligned}$$

Where the final inequality follows from  $(a + b)^l \leq 2^{l-1}(a^l + b^l)$ . Putting these together,

$$M \leq 2^l \mathbb{E}_{S, \sigma} \left[ \left( \max_{\|X\beta^* - X\beta\|=1} \left| \sum_k \sigma_k \left( \frac{|x_{i_k}^\top \beta^* - y_{i_k}|}{p_{i_k}} - \frac{|x_{i_k}^\top \beta - y_{i_k}|}{p_{i_k}} \right) \right| \right)^l \right] \quad (14)$$

◀

► **Lemma 33** (Similar to [4] Lemma B.1). *Let  $X$  be any matrix, and let  $W$  be the matrix that has the Lewis weights of  $X$  in the diagonal entries. Let  $N \geq \frac{d}{\varepsilon^2} \log \frac{d}{\varepsilon \delta}$ . There exist constants  $C_1, C_2, C_3$  such that we can construct a matrix  $X'$  such that*

- $X'$  has  $C_1 d N$  rows,
- $X'^\top W'^{-1} X' \succeq X^\top W^{-1} X$ , (where  $W'$  is the matrix that has the Lewis weights of  $X'$  in the diagonal entries),
- $\|X'\beta\|_1 \leq C_2 \|X\beta\|_1$  for all  $\beta$ ,
- the Lewis weights of  $X'$  are bounded by  $\frac{C_3}{N}$ .

**Proof.** Given matrix  $X$ , we can use Lemma B.1 from [4] to construct a new matrix  $X_1$  that satisfies

- $X_1$  has  $C_1 d^2$  rows,
- $X_1^\top W_1^{-1} X_1 \succeq X^\top W^{-1} X$ , (where  $W_1$  is the matrix that has the Lewis weights of  $X_1$  in the diagonal entries),
- $\|X_1\beta\|_1 \leq C_2 \|X\beta\|_1$  for all  $\beta$ ,
- the Lewis weights of  $X_1$  are bounded by  $\frac{C_3}{d}$ .

So, we can take this matrix and stack it on itself  $k = \frac{N}{d}$  times, while scaling each row down by the same  $k$ . This will be our matrix  $X'$ .  $X'$  will then have  $k = C_1 N d$  rows, which satisfies the first bullet. Also, by Lemma 10, this shrinks the Lewis weights by a factor of  $k$ , which changes the Lewis weight upper bound to  $\frac{C_3}{kd} = \frac{C_3}{N}$  which is what we need. Now, since we are repeating rows  $k$  times, but each row is scaled down by  $k$ , we have  $\|X_1 \beta\|_1 = \|X' \beta\|_1$  for all  $\beta$ . Therefore,  $\|X' \beta\|_1 \leq C_2 \|X \beta\|_1$  for all  $\beta$ . Finally, as in the proof of Lemma 10, we know that since we have duplicated the rows of  $X_1$   $k$  times but scaled them down by  $k$ ,  $X_1^\top W_1^{-1} X_1 = X'^\top W'^{-1} X'$ , and so we are done.  $\blacktriangleleft$

## B.1 Proof of Claims 15, 18, 19, 24, and 25

▷ **Claim 15.** For all  $\beta \in H$ , with probability at least  $1 - \delta$ ,

$$(1 - \varepsilon) \mathbb{E}_{(X,Y) \sim P} [|X^\top \beta - Y|] \leq \frac{1}{n} \|\mathbf{X}\beta - y\|_1 \leq (1 + \varepsilon) \mathbb{E}_{(X,Y) \sim P} [|X^\top \beta - Y|]$$

Proof of Claim 15. By assumption, we know that  $X^\top \beta, Y \in [-1, 1]$ , so,  $|X^\top \beta - Y| \in [0, 2]$ . So, for fixed  $\beta$ , by Hoeffding's on the rows of  $\mathbf{X}\beta - y$ , we have that if  $n \geq \frac{8}{\varepsilon^2} \log \frac{2}{\delta'}$ , then with probability at least  $1 - \delta'$ ,

$$\left(1 - \frac{\varepsilon}{2}\right) \mathbb{E}_{(X,Y) \sim P} [|X^\top \beta - Y|] \leq \frac{1}{n} \|\mathbf{X}\beta - y\|_1 \leq \left(1 + \frac{\varepsilon}{2}\right) \mathbb{E}_{(X,Y) \sim P} [|X^\top \beta - Y|] \quad (15)$$

Now, we construct a  $\frac{\varepsilon}{2d}$ -covering  $S$  of the unit  $\ell_\infty$  ball  $H$ , with fewer than  $\left(\frac{4d}{\varepsilon}\right)^d$  elements, so that for any  $\beta$ , there is some  $\beta_c \in S$  such that  $\|\beta - \beta_c\|_\infty \leq \frac{\varepsilon}{2d}$ . To do this, simply take  $S = \{\beta : \beta_i = k \frac{\varepsilon}{2d}, k \in \mathbb{Z} \cap [-2d/\varepsilon, 2d/\varepsilon]\}$ .

Note that  $\mathbf{X}$  has rows on the hypercube. So, if we denote  $x_{i,j}$  to be the entry of  $\mathbf{X}$  in the  $i$ th row and  $j$ th column, then  $x_{i,j} \in \{-1, 1\}$ . Therefore, for any  $\beta$ ,

$$\|\mathbf{X}\beta\|_1 = \sum_{i=1}^n |x_i^\top \beta| \leq \sum_{i=1}^n \sum_{j=1}^d |x_{i,j} \beta_j| \leq \sum_{i=1}^n \sum_{j=1}^d |\beta_j| \leq nd \|\beta\|_\infty$$

Therefore, we can apply Hoeffding's, as in (15), with  $\delta' = \delta \left(\frac{\varepsilon}{4d}\right)^d$ , and union bound over the set  $S$ , to get that for any  $\beta \in S$ , with probability at least  $1 - \delta$ , (15) holds.

Then, for any  $\beta \in H$ , by the covering property, we can find some  $\beta_c \in S$  such that

$$\|\beta - \beta_c\|_\infty \leq \frac{\varepsilon}{d} \implies \|\mathbf{X}\beta - \mathbf{X}\beta_c\|_1 \leq n\varepsilon. \quad (16)$$

We have

$$\|\mathbf{X}\beta_c - y\|_1 - \|\mathbf{X}\beta_c - \mathbf{X}\beta\|_1 \leq \|\mathbf{X}\beta - y\|_1 \leq \|\mathbf{X}\beta - \mathbf{X}\beta_c\|_1 + \|\mathbf{X}\beta_c - y\|_1$$

So, combining (15) and (16), and dividing by  $n$ , we finally have that if  $n \geq \frac{8}{\varepsilon^2} \left(\log \frac{2}{\delta} + d \log \frac{4d}{\varepsilon}\right)$ , then for all  $\beta \in H$ ,

$$(1 - \varepsilon) \mathbb{E}_{(X,Y) \sim P} [|X^\top \beta - Y|] \leq \frac{1}{n} \|\mathbf{X}\beta - y\|_1 \leq (1 + \varepsilon) \mathbb{E}_{(X,Y) \sim P} [|X^\top \beta - Y|] \quad \triangleleft$$

▷ **Claim 18.** For  $D, \mathcal{B}$  as chosen above,  $l(\beta^*) = 1 - 2\varepsilon$ .

## 49:20 L1 Regression with Lewis Weights Subsampling

Proof of Claim 18. The  $\ell_1$  error for the correct  $\beta$  is given by

$$\begin{aligned}
& \mathbb{E}_{(X,Y) \sim P} |X^\top \beta^* - Y| \\
&= \mathbb{E}_X [E_{Y \sim P(\cdot|X)} |X^\top \beta^* - Y|] && \text{by independence} \\
&= \mathbb{E}_X \left[ \left( \frac{1}{2} + \varepsilon \right) |X^\top \beta^* - X^\top \beta^*| + \left( \frac{1}{2} - \varepsilon \right) |X^\top \beta^* + X^\top \beta^*| \right] \\
&= \mathbb{E}_X [(1 - 2\varepsilon) |X^\top \beta^*|] && \beta^* \in \mathcal{H} \\
&= 1 - 2\varepsilon && \triangleleft
\end{aligned}$$

▷ **Claim 19.** For  $D, \mathcal{B}$  as chosen above, we have for all  $\beta \in \mathcal{B}$ ,  $l(\beta) - l(\beta^*) = \frac{2\varepsilon}{d} \|\beta - \beta^*\|_1$ .

Proof of Claim 19.

$$\begin{aligned}
& \mathbb{E}_{(X,Y) \sim P} |X^\top \beta - Y| \\
&= \mathbb{E}_X [E_{Y \sim P(\cdot|X)} |X^\top \beta - Y|] \\
&= \mathbb{E}_X \left[ \left( \frac{1}{2} + \varepsilon \right) |X^\top \beta - X^\top \beta^*| + \left( \frac{1}{2} - \varepsilon \right) |X^\top \beta + X^\top \beta^*| \right] \\
&= (1 - 2\varepsilon) + 2\varepsilon \mathbb{E}_X [X^\top \beta - X^\top \beta^*] \\
&= (1 - 2\varepsilon) + 2\varepsilon \frac{1}{d} \|\beta - \beta^*\|_1 && \triangleleft
\end{aligned}$$

▷ **Claim 21.** For any  $\beta$ ,  $\max\{\ell_{\beta_{(1)}^*}(\beta) - \ell_{\beta_{(1)}^*}(\beta_{(1)}^*), \ell_{\beta_{(2)}^*}(\beta) - \ell_{\beta_{(2)}^*}(\beta_{(2)}^*)\} > 2\varepsilon$

Proof of Claim 21.

$$\begin{aligned}
l(\beta) + l(\beta) &= 2 - 4\varepsilon + \frac{2\varepsilon}{d} \|\beta_{(1)}^* - \beta\|_1 + \frac{2\varepsilon}{d} \|\beta_{(2)}^* - \beta\|_1 \\
&\geq 2 - 4\varepsilon + \frac{2\varepsilon}{d} \|\beta_{(2)}^* - \beta_{(1)}^*\|_1 \\
&= 2 \\
\implies \max\{\ell_{\beta_{(1)}^*}(\beta) - \ell_{\beta_{(1)}^*}(\beta_{(1)}^*), \ell_{\beta_{(2)}^*}(\beta) - \ell_{\beta_{(2)}^*}(\beta_{(2)}^*)\} &> 2\varepsilon, && \forall \beta \in \mathbb{R}^d \\
&&& \triangleleft
\end{aligned}$$

▷ **Claim 24.** For all  $i \in [d], \beta \in \mathbb{R}^d$ , we have  $\ell_{\beta_{(i)}^*}(\beta) \geq \frac{1}{4d}$  with equality when  $\beta = \beta_{(i)}^*$

Proof of Claim 24.

$$\begin{aligned}
\ell_{\beta_{(i)}^*}(\beta) &= \frac{1}{d} \sum_{j \neq i} |\beta_j| + \frac{\frac{1}{2} + \varepsilon}{d} |1 - \beta_i| + \frac{\frac{1}{2} - \varepsilon}{d} |\beta_i| \\
&\geq \frac{\frac{1}{2} - \varepsilon}{d} (|\beta_i| + |1 - \beta_i|) + \frac{2\varepsilon}{d} |1 - \beta_i| \geq \frac{\frac{1}{2} - \varepsilon}{d} && \triangleleft
\end{aligned}$$

▷ **Claim 25.** Any  $\beta \in \mathbb{R}^d$  can only satisfy  $\ell_{\beta_{(i)}^*}(\hat{\beta}) < \frac{1}{2d}$  for one  $i \in [d]$ .



Proof of Claim 25. Indeed, suppose  $\beta$  was such that  $\ell_{\beta_{(I)}}^*(\beta), \ell_{\beta_{(J)}}^*(\beta) < \frac{1}{2d}$ . Then we must have

$$\begin{aligned}
\frac{1}{2d} &\geq \ell_{\beta_{(I)}}^*(\beta) \\
&= \frac{1}{d} \sum_{j \neq I} |\beta_j| + \frac{\frac{1}{2} - \varepsilon}{d} (|\beta_I| + |1 - \beta_I|) + \frac{2\varepsilon}{d} |1 - \beta_I| \\
&\geq \frac{1}{d} \sum_{j \neq I} |\beta_j| + \frac{\frac{1}{2} - \varepsilon}{d} + \frac{2\varepsilon}{d} |1 - \beta_I| \\
\iff \varepsilon &\geq \sum_{j \neq I} |\beta_j| + 2\varepsilon |1 - \beta_I| \\
&\geq \sum_{j \neq I} |\beta_j| + 2\varepsilon - 2\varepsilon |\beta_I| \\
\iff 2|\beta_I| &\geq \|\beta\|_1 + 2\varepsilon
\end{aligned}$$

Similarly for  $J$ , so we would have  $\|\beta\| \geq |\beta_I| + |\beta_J| \geq \|\beta\|_1 + 2\varepsilon$ . ◁



# Hitting Sets for Orbits of Circuit Classes and Polynomial Families

Chandan Saha ✉

Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India

Bhargav Thankey ✉

Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India

---

## Abstract

---

The orbit of an  $n$ -variate polynomial  $f(\mathbf{x})$  over a field  $\mathbb{F}$  is the set  $\{f(A\mathbf{x} + \mathbf{b}) : A \in \text{GL}(n, \mathbb{F}) \text{ and } \mathbf{b} \in \mathbb{F}^n\}$ . In this paper, we initiate the study of explicit hitting sets for the *orbits* of polynomials computable by several natural and well-studied circuit classes and polynomial families. In particular, we give quasi-polynomial time hitting sets for the orbits of:

1. Low-individual-degree polynomials computable by *commutative ROABPs*. This implies quasi-polynomial time hitting sets for the orbits of the *elementary symmetric polynomials*.
2. Multilinear polynomials computable by *constant-width ROABPs*. This implies a quasi-polynomial time hitting set for the orbits of the family  $\{\text{IMM}_{3,d}\}_{d \in \mathbb{N}}$ , which is complete for arithmetic formulas.
3. Polynomials computable by *constant-depth, constant-occur formulas*. This implies quasi-polynomial time hitting sets for the orbits of *multilinear depth-4 circuits with constant top fan-in*, and also polynomial-time hitting sets for the orbits of the *power symmetric* and the *sum-product polynomials*.
4. Polynomials computable by *occur-once formulas*.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Algebraic complexity theory

**Keywords and phrases** Hitting Sets, Orbits, ROABPs, Rank Concentration

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.50

**Category** RANDOM

**Related Version** *Full Version*: <https://eccc.weizmann.ac.il/report/2021/015/>

**Acknowledgements** We thank Rohit Gurjar, Ankit Garg, Neeraj Kayal, and Vishwas Bhargava for several stimulating discussions at the onset of this work.

## 1 Introduction

Polynomial identity testing (PIT) is a fundamental problem in arithmetic circuit complexity. PIT is the problem of deciding if a given arithmetic circuit computes an identically zero polynomial. It is one of the few natural problems in BPP (in fact, in co-RP) for which we do not know of deterministic polynomial-time algorithms. A probabilistic polynomial-time algorithm for PIT follows from the DeMillo-Lipton-Schwartz-Zippel lemma [15, 71, 78]. PIT has connections to other interesting problems like perfect matching [19, 41, 49, 53, 75], the linear matroid intersection [33, 55], and the maximum rank matrix completion [33, 54]. The deterministic primality testing algorithm in [4] derandomizes a particular instance of PIT over a ring [2]. Also, multivariate polynomial factorization for general circuits can be efficiently reduced to PIT and factoring univariate polynomials [37, 38, 48]. Moreover, derandomizing PIT or the black-box version of PIT<sup>1</sup> is essentially equivalent to proving arithmetic circuit lower bounds.

---

<sup>1</sup> An algorithm for the black-box PIT problem takes as input black-box access to a circuit. The algorithm cannot “see” the circuit but can query it at any point [1, 34, 36, 57]. The black-box PIT problem for a circuit class  $\mathcal{C}$  is also known as the problem of constructing *hitting sets* for  $\mathcal{C}$



© Chandan Saha and Bhargav Thankey;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 50; pp. 50:1–50:26



Leibniz International Proceedings in Informatics

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

In the past two decades, PIT algorithms and hitting set constructions have been studied for various restricted classes/models of circuits. Bounding the read of every variable is a natural restriction that has received a lot of attention. Two constant-read models, viz. *read-once oblivious algebraic branching programs* (ROABPs) and *constant-read* (more generally, *constant-occur*) *formulas*. These models are quiet powerful and capture many interesting circuit classes. A polynomial-time PIT algorithm and a quasi-polynomial time hitting set construction for ROABPs are known [3, 23, 61]. A quasi-polynomial time hitting set construction for multilinear constant-read formulas was given by [10]. [5] obtained polynomial-time constructible hitting sets for constant-depth, constant-occur formulas.

**Hitting sets for orbits.** In this paper, we study hitting set constructions for the *orbits* of ROABPs and constant-occur formulas. The orbit of a polynomial  $f$  is the set of polynomials obtained by applying invertible affine transformations on the variables of  $f$ , i.e., by replacing the variables of  $f$  with linearly independent affine forms. The orbit of a circuit class is the union of the orbits of the polynomials computable by the circuits in the class. Our reasons for studying hitting sets for the orbits of ROABPs and constant-occur formulas are threefold:

1. *The power of orbit closures:* The set of affine projections of an  $n$ -variate polynomial  $f(\mathbf{x})$  over a field  $\mathbb{F}$  is  $\text{aproj}(f) := \{f(A\mathbf{x} + \mathbf{b}) : A \in \mathbb{F}^{n \times n} \text{ and } \mathbf{b} \in \mathbb{F}^n\}$ ; the orbit of  $f$  is the set  $\text{orb}(f) = \{f(A\mathbf{x} + \mathbf{b}) : A \in \text{GL}(n, \mathbb{F}) \text{ and } \mathbf{b} \in \mathbb{F}^n\} \subseteq \text{aproj}(f)$ . Affine projections of polynomials computable by polynomial-size ROABPs or constant-occur formulas have great expressive power. All polynomials computed by algebraic branching programs and arithmetic formulas are affine projections of polynomial width ROABPs and constant ROABPs, respectively. Similarly, all polynomials computed by depth-3 arithmetic circuits (which are quiet powerful [7, 30, 46, 76, 77]) and arithmetic formulas are affine projections of read once formulas. The orbit of  $f$  being a mathematically interesting subset of  $\text{aproj}(f)$ , it is natural to ask if we can construct efficient hitting sets for the orbits of the above-mentioned circuit classes. Moreover,  $\text{orb}(f)$  is not “much smaller” than  $\text{aproj}(f)$ , as the latter is contained in the *orbit closure* of  $f$  if  $\text{char}(\mathbb{F}) = 0$  (see the full version [64] for more details).
2. *Geometry of the circuit classes:* Consider an  $n$ -variate polynomial  $f \in \mathbb{R}[\mathbf{x}]$  and let  $\mathbb{V}(f)$  be the variety (i.e., the zero locus) of  $f$ . The geometry of  $\mathbb{V}(f)$  is preserved by any rigid transformation on  $\mathbb{R}^n$ . Computation of a set  $\mathcal{H} \subseteq \mathbb{R}^n$  that is not contained in  $T(\mathbb{V}(f))$ , for every rigid transformation  $T$ , would have to be “mindful” of the geometry of  $\mathbb{V}(f)$  and oblivious to the choice of the coordinate system. Computing such an  $\mathcal{H}$  is exactly the problem of constructing a hitting set for the polynomials  $\{f(R\mathbf{x} + \mathbf{b}) : R \in O(n, \mathbb{R}) \text{ and } \mathbf{b} \in \mathbb{R}^n\}$ . We can generalize the problem slightly by replacing  $R \in O(n, \mathbb{R})$  with  $A \in \text{GL}(n, \mathbb{R})$ . A hitting set for ROABPs or constant-occur formulas does not immediately give a hitting set for  $\{f(A\mathbf{x} + \mathbf{b}) : A \in \text{GL}(n, \mathbb{R}) \text{ and } \mathbf{b} \in \mathbb{R}^n\}$ , as the definitions of an ROABP and a constant-occur formula are tied to the choice of the coordinate system. It is thus natural to ask if there is anything special about the geometry of  $\mathbb{V}(f)$  which can facilitate efficient constructions of hitting sets for  $\text{orb}(f)$ .
3. *Strengthening existing techniques:* Finally, it is worth investigating whether the techniques used to design hitting sets for ROABPs and constant-occur formulas can be applied or strengthened or combined to give hitting sets for the orbits of these circuit classes.

## 1.1 The models

Unless otherwise stated, we will assume that polynomials are over a field  $\mathbb{F}$ . Read Once Algebraic Branching Programs (ROABPs) are the read once versions of Algebraic Branching Programs defined by Nisan [56]. While Nisan defined ABPs using directed graphs, we use the following equivalent and conventional definition of an ROABP.

► **Definition 1** (ROABP [23]). *An  $n$ -variate, width- $w$  read-once oblivious algebraic branching program (ROABP) is a product of the form  $\mathbf{1}^T \cdot M_1(x_1)M_2(x_2) \cdots M_n(x_n) \cdot \mathbf{1}$ , where  $\mathbf{1}$  is the  $w \times 1$  vector of all ones, and for every  $i \in [n]$ ,  $M_i(x_i)$  is a  $w \times w$  matrix whose entries are in  $\mathbb{F}[x_i]$ .*

► **Definition 2** (Commutative ROABP). *An  $n$ -variate, width- $w$  commutative ROABP is an  $n$ -variate, width- $w$  ROABP  $\mathbf{1}^T \cdot M_1(x_1)M_2(x_2) \cdots M_n(x_n) \cdot \mathbf{1}$ , where for all  $i, j \in [n]$ ,  $M_i(x_i)$  and  $M_j(x_j)$  commute with each other.*

A polynomial  $f$  is  $s$ -sparse if it has at most  $s$  monomials with non-zero coefficients; these monomials will be referred to as the *monomials of  $f$* . A degree- $d$   $s$ -sparse polynomial can be computed by a depth-2 circuit of size  $sd$  as well as by a width- $s$  commutative ROABP.

► **Definition 3** (Occur- $k$  formula [5]). *An occur- $k$  formula is a rooted tree whose leaves are labelled by  $s$ -sparse polynomials and whose internal nodes are sum (+) gates or product-power ( $\times\lambda$ ) gates. Each variable appears in at most  $k$  of the sparse polynomials that label the leaves. The edges feeding into a + gate are labelled by field elements and have 1 as edge weights, whereas the edges feeding into a  $\times\lambda$  gate have natural numbers as edge weights. A leaf node computes the  $s$ -sparse polynomial that labels it. A + gate with inputs from nodes that compute  $f_1, \dots, f_m$  and with the corresponding input edge labels  $\alpha_1, \dots, \alpha_m$ , computes  $\alpha_1 f_1 + \cdots + \alpha_m f_m$ . A  $\times\lambda$  gate with inputs from nodes that compute  $f_1, \dots, f_m$  and with the corresponding input edge weights  $e_1, \dots, e_m$ , computes  $f_1^{e_1} \cdots f_m^{e_m}$ . The formula computes the polynomial that is computed by the root node.*

*The size of an occur- $k$  formula is the weighted sum of all the edges in it (i.e., an edge is counted as many times as its edge weight) plus the sizes of the depth-2 circuits computing the  $s$ -sparse polynomials at the leaves. The depth of an occur- $k$  formula is equal to the depth of the underlying tree plus 2, to account for the depth of the circuits computing the sparse polynomials at the leaves.*

Read- $k$  formulas have been studied intensely in the literature (see Section 1.4). Occur- $k$  formulas generalize read- $k$  formulas in two ways – the leaves are labelled by arbitrary sparse polynomials instead of just variables, and powering gates are included along with the usual sum and product gates. These generalizations help make the occur- $k$  model complete<sup>2</sup>, and capture other interesting circuit classes (such as multilinear depth-4 circuits with constant top fan-in [39, 65]) and polynomial families (such as the power symmetric polynomials). Besides, unlike some prior work [10, 39, 65], there is no restriction of multilinearity on the model. We will identify the variable set  $\mathbf{x} = \{x_1, \dots, x_n\}$  with the column vector  $(x_1 \ x_2 \ \cdots \ x_n)^T$ .

► **Definition 4** (Orbits of polynomials). *Let  $f(\mathbf{x})$  be an  $n$ -variate polynomial over a field  $\mathbb{F}$ . The orbit of  $f$ , denoted by  $\text{orb}(f)$ , is the set  $\{f(A\mathbf{x}) : A \in \text{GL}(n, \mathbb{F})\}$ . The orbit of a set of polynomials  $\mathcal{C}$ , denoted by  $\text{orb}(\mathcal{C})$ , is the union of the orbits of the polynomials in  $\mathcal{C}$ .*

<sup>2</sup> For example, the power symmetric polynomial  $x_1^n + \dots + x_n^n$  cannot be computed by a read- $k$  formula for any  $k < n$ , but it can be computed by an occur-once formula.

The results we present in this paper hold even if we define the orbit of an  $n$ -variate polynomial  $f$  as  $\text{orb}(f) = \{f(A\mathbf{y} + \mathbf{b}) : |\mathbf{y}| = m \geq n, A \in \mathbb{F}^{n \times m} \text{ has rank } n, \text{ and } \mathbf{b} \in \mathbb{F}^n\}$ . However, we work with this slightly conventional definition of  $\text{orb}(f)$  for simplicity of exposition, and because the proofs in the general setting are nearly the same as the proofs we present here. By the “orbit of a circuit class  $\mathcal{C}$ ”, we mean the union of the orbits of the polynomials computable by the circuits in the class  $\mathcal{C}$ . Our main results are efficient constructions of hitting sets for the orbits of commutative ROABPs and constant-width ROABPs (under low individual degree restriction), and the orbits of constant-depth constant-occur formulas and occur-once formulas.

## 1.2 Our results

► **Definition 5** (Hitting set). *Let  $\mathcal{C}$  be a set of  $n$ -variate polynomials. A set of points  $\mathcal{H} \subseteq \mathbb{F}^n$  is a hitting set for  $\mathcal{C}$  if for every non-zero  $f \in \mathcal{C}$ , there is a point  $\mathbf{a} \in \mathcal{H}$  such that  $f(\mathbf{a}) \neq 0$ .*

By a “ $T$ -time hitting set”, we mean that the hitting set can be computed in  $T$  time. The *individual degree* of a monomial is the largest of the exponents of the variables that appear in it. The individual degree of a polynomial is the largest of the individual degrees of its monomials. We are now ready to state our results.

► **Theorem 6** (Hitting sets for the orbits of commutative ROABPs with low individual degree). *Let  $\mathcal{C}$  be the set of  $n$ -variate polynomials with individual degree at most  $d$  that are computable by width- $w$  commutative ROABPs. If  $|\mathbb{F}| > n^2d$ , then a hitting set for  $\text{orb}(\mathcal{C})$  can be computed in  $(nd)^{O(d \log w)}$  time.*

We say an  $n$ -variate polynomial  $f(x_1, x_2, \dots, x_n)$  can be expressed as a sum of  $s$  products of univariates if  $f = \sum_{i \in [s]} \prod_{j \in [n]} f_{i,j}(x_j)$ , where each  $f_{i,j}(x_j)$  is a univariate polynomial in  $x_j$ . This model is subsumed by commutative ROABPs and has found important applications in several other works [30, 63, 66]. The above theorem implies a  $nd^{O(d \log s)}$  time hitting set for this model. As the elementary symmetric polynomials and low individual degree sparse polynomials are special cases of low individual degree sum of products of univariates, we also get quasi-polynomial hitting sets for these models. It turns out though that for the particular case of sparse polynomials it is possible to remove the individual degree restriction from the above theorem. This is due to an independent and simultaneous work by [51]. We state their result next.

► **Theorem 7** (Hitting sets for the orbits of sparse polynomials [51]). *Let  $\mathcal{C}$  be the set of  $n$ -variate,  $s$ -sparse polynomials of degree at most  $d$ . If  $|\mathbb{F}| > nd$  and  $\text{char}(\mathbb{F}) = 0$  or  $> d$ , then a hitting set for  $\text{orb}(\mathcal{C})$  can be computed in  $(nd)^{O(\log s)}$  time.*

The above theorem plays a basic role in the proofs of Theorem 9 and Theorem 10. There, we apply the algebraic independence based analysis from [5, 11] and the Shpilka-Volkovich (SV) generator based argument from [73], respectively, to reduce to the case of constructing hitting sets for the orbits of sparse polynomials. While in the original version of our work [64] we applied Theorem 6 in the base case of the proofs of Theorem 9 and 10, here we plug-in Theorem 7 in the base case. This helps us forgo the low individual degree restriction that was present in these theorems in the original version.

► **Theorem 8** (Hitting sets for the orbits of multilinear constant-width ROABPs). *Let  $\mathcal{C}$  be the set of  $n$ -variate multilinear polynomials that are computable by width- $w$  ROABPs. If  $|\mathbb{F}| > n^{O(w^4)}$ , then a hitting set for  $\text{orb}(\mathcal{C})$  can be computed in  $n^{O(w^6 \cdot \log n)}$  time.*

The theorem gives a quasi-polynomial time hitting set for  $\text{orb}(\text{IMM}_{3,d})$ , which is complete for the class of arithmetic formulas under affine projections (in fact, under  $p$ -projections) [12]. The set of affine projections of  $\text{IMM}_{2,d}$  is also quite rich, despite the fact that there are simple quadratic polynomials that are not in  $\text{aproj}(\text{IMM}_{2,d})$  for *any*  $d$  [8, 62]. This is because hitting sets for  $\text{aproj}(\text{IMM}_{2,d})$  give hitting sets for depth-3 circuits [62]. Moreover,  $\overline{\text{orb}(\text{IMM}_{2,d})}$  captures the orbit closures of arithmetic formulas [13]. The above theorem implies a quasi-polynomial time hitting set for  $\text{orb}(\text{IMM}_{2,d})$ .

► **Theorem 9** (Hitting sets for the orbits of constant-depth, constant-occur formulas). *Let  $\mathcal{C}$  be the set of  $n$ -variate, degree- $D$  polynomials that are computable by depth- $\Delta$ , occur- $k$  formulas of size  $s$ . Let  $R := (2k)^{2\Delta \cdot 2^\Delta}$ . If  $\text{char}(\mathbb{F}) = 0$  or  $> (2ks)^{\Delta^3 R}$ , then a hitting set for  $\text{orb}(\mathcal{C})$  can be computed in  $(nRD)^{O(R(\log R + \Delta \log k + \Delta \log s) + \Delta R)}$  time. If the leaves are labelled by  $b$ -variate polynomials, then a hitting set for  $\text{orb}(\mathcal{C})$  can be computed in  $(nRD)^{O(Rb + \Delta R)}$  time. In particular, if  $\Delta$  and  $k$  are constants, then the hitting sets can be constructed in time  $(nD)^{O(\log s)}$  and  $(nD)^{O(b)}$ , respectively.*

The above theorem gives quasi-polynomial hitting sets for the orbits of two other interesting models viz. multilinear depth-4 circuits with constant top fan-in and the class of polynomials  $C(f_1, \dots, f_m)$ , where  $C$  is a low-degree circuit and  $f_1, \dots, f_m$  are sparse polynomials with bounded transcendence degree [11]. The theorem also yields polynomial-time hitting sets for the orbits of the power symmetric polynomial and the sum-product polynomial  $\text{SP}_{n,D} = \sum_{i \in [n]} \prod_{j \in [D]} x_{i,j}$ . Prior to our work, [47] gave a polynomial-time hitting set for the orbit of power symmetric polynomials using a different argument.

► **Theorem 10** (Hitting sets for the orbits of occur-once formulas). *Let  $\mathcal{C}$  be the set of  $n$ -variate, degree- $D$  polynomials that are computable by occur-once formulas whose leaves are labelled by  $s$ -sparse polynomials. If  $|\mathbb{F}| > nD$  and  $\text{char}(\mathbb{F}) = 0$  or  $> D$ , then a hitting set for  $\text{orb}(\mathcal{C})$  can be computed in  $(nD)^{O(\log n + \log s)}$  time. If the leaves are labelled by  $b$ -variate polynomials, then a hitting set for  $\text{orb}(\mathcal{C})$  can be computed in  $(nD)^{O(\log n + b)}$  time.*

The independent and concurrent work [51] gave (among other results) a quasi-polynomial time hitting set for the orbits of read-once formulas. We note that this result also follows from the second part of the above theorem which is already present in the original version of this work [64]. The proofs of Theorems 9 and 10 can be found in the full version [64].

### 1.3 Proof techniques

Let us briefly discuss the techniques that go into proving the above results.

**Commutative ROABPs with low individual degree.** Theorem 6 is proved by adapting the rank concentration by translation technique of [6] to work for the orbits of commutative ROABPs. Let  $f = \mathbf{1}^T \cdot M_1(x_1)M_2(x_2) \cdots M_n(x_n) \cdot \mathbf{1}$  be a commutative ROABP and  $F = M_1(x_1)M_2(x_2) \cdots M_n(x_n)$ . For any  $A \in \text{GL}(n, \mathbb{F})$ , let  $g = f(A\mathbf{x})$  and  $G = F(A\mathbf{x})$ . Suppose that  $A$  maps  $x_i$  to a linear form  $\ell_i(\mathbf{x})$  for every  $i \in [n]$ , and let  $y_i = \ell_i(\mathbf{x})$ . Then,  $g = \mathbf{1}^T \cdot M_1(y_1)M_2(y_2) \cdots M_n(y_n) \cdot \mathbf{1}$  and  $G = M_1(y_1)M_2(y_2) \cdots M_n(y_n)$ . We show that if  $g \neq 0$ , then there exist *explicit* “low” degree polynomials  $t_1(\mathbf{z}), \dots, t_n(\mathbf{z})$ , where  $\mathbf{z}$  is a “small” set of variables, such that  $g(x_1 + t_1(\mathbf{z}), \dots, x_n + t_n(\mathbf{z}))$  has a “low” support monomial. This is done by proving that  $G(x_1 + t_1(\mathbf{z}), \dots, x_n + t_n(\mathbf{z}))$  has low support rank concentration over  $\mathbb{F}(\mathbf{z})$  in the “ $\mathbf{y}$ -variables” (see Section 2.2 for the meaning of low support rank concentration.). That done, we use the assumption that  $f$  has low individual degree to argue that  $g(x_1 + t_1(\mathbf{z}), \dots, x_n + t_n(\mathbf{z}))$  also has a low support  $\mathbf{x}$ -monomial. This and the fact that  $|\mathbf{z}|$  is small imply that  $g(x_1 + t_1(\mathbf{z}), \dots, x_n + t_n(\mathbf{z}))$ , when viewed as a polynomial in  $\mathbb{F}[\mathbf{x}, \mathbf{z}]$ , has a low support monomial. Finally, we use the SV generator to hit  $g$ .

Our analysis differs from that in [6] at a crucial point: In [6], it was shown that  $F(\mathbf{x} + \mathbf{t}) = M_1(x_1 + t_1)M_2(x_2 + t_2) \cdots M_n(x_n + t_n)$  has low support rank concentration over  $\mathbb{F}(\mathbf{t})$  if the nonzeroness of every polynomial in a certain collection of polynomials – each in a “small” set of  $\mathbf{t}$ -variables – is preserved. As each polynomial in the collection has “few”  $\mathbf{t}$ -variables, a substitution  $t_i \leftarrow t_i(\mathbf{z})$  that preserves its nonzeroness is relatively easy to construct. But the collection of polynomials that we need to preserve to show low support rank concentration for  $G(\mathbf{x} + \mathbf{t})$  is such that every polynomial in the collection has potentially all the  $\mathbf{t}$ -variables. However, we are able to argue that each of these polynomials still has a low support  $\mathbf{t}$ -monomial. This then helps us construct a substitution  $t_i \mapsto t_i(\mathbf{z})$  that preserves the nonzeroness of these polynomials.

**Multilinear constant-width ROABPs.** Theorem 8 is proved by combining the rank concentration by translation technique of [6] with the merge-and-reduce idea from [23] and [21]. Let  $f = \mathbf{1}^T \cdot M_1(x_1)M_2(x_2) \cdots M_n(x_n) \cdot \mathbf{1}$  be a multilinear, width- $w$  ROABP; here  $M_i(x_i) \in \mathbb{F}^{w \times w}[x_i]$  for all  $i \in [n]$ . Also, let  $F = M_1(x_1)M_2(x_2) \cdots M_n(x_n)$ . For any  $A \in \text{GL}(n, \mathbb{F})$ , let  $g = f(A\mathbf{x})$  and  $G = F(A\mathbf{x})$ . For  $i \in [n]$ , suppose that  $A$  maps  $x_i \mapsto \ell_i(\mathbf{x})$ , where  $\ell_i$  is a linear form, and let  $y_i = \ell_i(\mathbf{x})$  and  $\mathbf{y} = \{y_1, \dots, y_n\}$ . Then,  $g = \mathbf{1}^T \cdot M_1(y_1)M_2(y_2) \cdots M_n(y_n) \cdot \mathbf{1}$  and  $G = M_1(y_1)M_2(y_2) \cdots M_n(y_n)$ . Much like in the case of commutative ROABPs, we show that if  $g \neq 0$ , then there exist explicit “low” degree polynomials  $t_1(\mathbf{z}), \dots, t_n(\mathbf{z})$ , where  $\mathbf{z}$  is a “small” set of variables such that  $G(x_1 + t_1(\mathbf{z}), \dots, x_n + t_n(\mathbf{z}))$  has “low” support rank concentration in the “ $\mathbf{y}$ -variables”. While in the rank concentration argument for commutative ROABPs the  $\mathbf{x}$ -variables were translated only once, here the translations can be thought of as happening sequentially and in stages. There will be  $\lceil \log n \rceil$  stages with each stage also consisting of multiple translations. After the  $p$ -th stage, the product of any  $2^p$  consecutive matrices in  $G$  will have low support rank concentration in the  $\mathbf{y}$ -variables. Thus, after  $\lceil \log n \rceil$  stages, we will have low support rank concentration in the  $\mathbf{y}$ -variables for  $G(x_1 + t_1(\mathbf{z}), \dots, x_n + t_n(\mathbf{z}))$ .

As in the case of commutative ROABPs, we show that  $G(\mathbf{x} + \mathbf{t})$  has low support rank concentration if each polynomial in a certain collection of non-zero polynomials in the  $\mathbf{t}$ -variables is kept non-zero by the substitution  $t_i \mapsto t_i(\mathbf{z})$ . However, in this case, it is trickier to show that these polynomials have low support  $\mathbf{t}$ -monomials. We do this by arguing that each such polynomial can be expressed as a ratio of a polynomial that contains a low support  $\mathbf{t}$ -monomial and a product of linear forms in the  $\mathbf{t}$ -variables.

**Constant-depth, constant-occur formulas.** We prove Theorem 9 by combining the algebraic independence based technique in [5] with Theorem 7. Let  $f$  be a constant-depth, constant-occur formula. We first show that it can be assumed without loss of generality that the top-most gate of  $f$  is a  $+$  gate whose fan-in is upper bounded by the occur of  $f$ , say  $k$ . In [5], they were able to upper bound the top fan-in by simply translating a variable by 1 and subtracting the original formula. However, the same idea does not quite work here, because we have only access to a polynomial in the *orbit* of  $f$ . To upper bound the top fan-in, we show that there exists a variable  $x_i$  such that  $\frac{\partial f}{\partial x_i}$  is a constant-depth, constant-occur formula with top fan-in bounded by  $k$ . Then, using the chain rule of differentiation, we show that one can construct a hitting set generator for  $\text{orb}(f)$  from a generator for  $\text{orb}\left(\frac{\partial f}{\partial x_i}\right)$ ; this means that we can shift our attention to  $f' = \frac{\partial f}{\partial x_i}$ , which we shall henceforth refer to as  $f$ .

Let  $f = f_1 + \dots + f_k$ ,  $A \in \text{GL}(n, \mathbb{F})$ ,  $g = f(A\mathbf{x})$ ,  $g = g_1 + \dots + g_k$  where for all  $i \in [k]$ ,  $g_i = f_i(A\mathbf{x})$ . It was shown in [5] that a homomorphism, which is faithful (see Definition 17) to  $f_1, \dots, f_k$ , is a hitting set generator for  $f$ . In our case, this translates to ‘a



homomorphism that is faithful to  $g_1, \dots, g_k$  is a hitting set generator for  $g$ . [5] also showed that the problem of constructing a homomorphism  $\phi$  that is faithful to  $f_1, \dots, f_k$  reduces to constructing a homomorphism  $\psi$  that preserves the determinant of a certain matrix. This matrix is an appropriate sub-matrix of the Jacobian of  $f_1, \dots, f_k$ . Also, it was argued that its determinant is a product of sparse polynomials and so  $\psi$  was obtained from [45]. We use a similar argument, along with the chain rule, to show that the problem of constructing a homomorphism  $\phi$  that is faithful to  $g_1, \dots, g_k$  reduces to constructing a homomorphism  $\psi$  that preserves the determinant of a sub-matrix of the same Jacobian *evaluated at*  $A\mathbf{x}$ . As this determinant is a product of polynomials in the orbit of sparse polynomials, we can use Theorem 7 to construct such a  $\psi$ .

**Occur-once formulas.** We prove Theorem 10 by building upon the arguments in [73] and linking it with Theorem 7. At first, we show two structural results for occur-once formulas. These lemmas are generalizations of similar structural results for read-once formulas shown in [73]. Much like in [73], the structural results help us show that for a “typical” occur-once formula  $f$  with a  $+$  gate as the root node, there exists a variable  $x_i$  such that  $\frac{\partial f}{\partial x_i}$  is a product of occur-once formulas, each of which has at most half as many non-constant leaves as  $f$ . We then use this fact to show that a hitting-set generator for  $\text{orb}(f)$  can be constructed from a generator for  $\text{orb}\left(\frac{\partial f}{\partial x_i}\right)$ . [73] uses the derivatives of  $f$  in a similar way to show that a generator for  $f$  can be constructed from that for  $\frac{\partial f}{\partial x_i}$  using the SV generator (see Definition 12). However, in our case, we want a generator for  $\text{orb}(f)$  and not just for  $f$ . For this reason, we first use the chain rule for derivatives to relate the gradient of a  $g \in \text{orb}(f)$  with that of  $f$ , and then argue that there exists a  $x_j$  such that a generator for  $\text{orb}\left(\frac{\partial f}{\partial x_i}\right)$  is also a generator for  $\frac{\partial g}{\partial x_j}$ . Finally, we use this generator for  $\frac{\partial g}{\partial x_j}$  to construct a generator for  $g$ . The argument then proceeds by induction on the number of non-constant leaves. In the base case, we need a hitting set generator for orbits of sparse polynomials which we get from Theorem 7.

## 1.4 Related work

We give a brief account of known results on PIT and hitting sets for arithmetic circuits. The results on hitting sets for the constant-read models are most relevant to our work here. However, for the sake of completeness, we mention a few other prominent results.

**Constant-read models.** [73] gave a polynomial-time PIT algorithm and a quasi-polynomial time hitting set construction for sums of constantly many *preprocessed* read-once formulas (PROFs). [52] later gave a polynomial time hitting set for the same model. [10] gave a quasi-polynomial time hitting set construction for multilinear *sparse-substituted* read- $k$  formulas, wherein the leaves are replaced by sparse polynomials and every variable appears in at most  $k$  of the sparse polynomials. Observe that the models studied in all three works are special cases of constant occur formulas.

A polynomial-time PIT for ROABPs follows from the PIT algorithm for non-commutative formulas [61]. [23] gave quasi-polynomial time hitting sets for ROABPs, when the order of the variables is known. Building on the rank concentration by translation technique from [6] and the merge-and-reduce idea from [23], [21] gave a quasi-polynomial time hitting set construction for low individual degree ROABPs. Finally, [3] obtained a quasi-polynomial time constructible hitting set for ROABPs using a different and simpler method, namely *basis isolation*, which can be thought of as a generalization of the monomial isolation method in [45]. [32] designed hitting sets for sums of constantly many ROABPs

in quasi-polynomial time; they also gave a polynomial-time PIT algorithm for the same model. Recently, more efficient constructions of hitting sets for ROABPs have been obtained [27], sometimes under additional restrictions on the model such as commutativity and constant-width [31]. For read- $k$  oblivious ABPs, [9] obtained a subexponential-time PIT algorithm.

**Orbits and orbit closures.** A polynomial-time hitting set for the *orbit* of the power symmetric polynomial  $\text{PSym}_{n,d} = x_1^d + \dots + x_n^d$  was given by [47]. As that  $\text{PSym}$  is computable by a depth-2 occur-once formula, Theorem 9 subsumes this result. Our hitting-set construction is different from the one in [47] which involves the Hessian matrix, whereas the proofs here work with just the first order derivatives. Very recently and independent of our work, [51] gave quasi-polynomial time hitting sets for the orbits of sparse polynomials and read-once formulas. For the orbit closures of polynomials that are computable by low-degree, polynomial-size circuits (i.e., VP circuits), [24, 28] gave PSPACE constructions of hitting sets.

**Constant-depth models.** The polynomial-time hitting set construction for depth-2 circuits (i.e., sparse polynomials) in [45] is one of the widely used results in black-box PIT. [16] gave a quasi-polynomial time PIT algorithm for depth-3 circuits with constant top fan-in. Later [44] improved the complexity to polynomial-time. Using ideas developed in [16], and [25], [40, 43, 70] gave polynomial-time constructible hitting sets for depth-3 circuits with constant top fan-in over  $\mathbb{Q}$ . Ultimately, a combination of ideas from the [44] and [25] led to a polynomial-time hitting set construction for the same model over any field [69, 70]. Meanwhile, [42, 66] gave polynomial-time PIT for depth-3 powering circuits. Using ideas from [44] and [66], [63] gave polynomial-time PIT for the sum of a depth-3 circuit with constant top fan-in and a *semi-diagonal* circuit (which is a special kind of a depth-4 circuit). [62] showed that polynomial-time PIT (resp. hitting sets) for  $\text{aproj}(\text{IMM}_{2,d})$  implies polynomial-time PIT (resp. hitting sets) for depth-3 circuits.

A quasi-polynomial time hitting set for set-multilinear depth-3 circuits with known variable-partition was given by [22]. Independently and simultaneously, [6] gave a quasi-polynomial time hitting set for set-multilinear depth-3 circuits with *unknown* variable-partition (and more generally, for constant-depth *pure* formulas [58]) using a different technique, namely *rank concentration by translation*. Set-multilinear depth-3 circuits (in fact, pure formulas) form a subclass of ROABPs. [14] gave subexponential-time hitting sets for multilinear depth-3 and depth-4 formulas (and more generally, for constant-depth multilinear regular formulas) by reducing the problem to constructing hitting sets for ROABPs. For multilinear depth-4 circuits with constant top fan-in, [39] gave a quasi-polynomial time hitting set. This was improved to a polynomial-time hitting set in [65]. Multilinear depth-4 circuits with constant top fan-in form a subclass of depth-4 constant-occur formulas. [5] gave a unifying method based on algebraic independence to design polynomial-time hitting sets for both depth-3 circuits with constant top fan-in and constant-depth, constant-occur formulas. A generalization of depth-3 powering circuits to depth-4 is sums of powers of constant degree polynomials; [20] gave a quasi-polynomial time hitting set for this model. Recently, a sequence of work [59, 60, 72] led to a polynomial-time hitting set for depth-4 circuits with top fan-in at most 3 and bottom fan-in at most 2 via a resolution of a conjecture of [11, 29] on the algebraic rank of the factors appearing in such circuits.

**Edmonds' model.** An important special case of PIT is the following problem: given  $f = \det(A_0 + \sum_{i \in [n]} x_i A_i)$ , where  $A_i \in \mathbb{F}^{n \times n}$  is a rank-1 matrix for every  $i \in [n]$  and  $A_0 \in \mathbb{F}^{n \times n}$  is an arbitrary matrix, check if  $f = 0$  [17]. This case of PIT, played an instrumental

role in devising fast parallel algorithms for several problems such as perfect matching, linear matroid intersection and maximum rank matrix completion [19, 33, 41, 49, 53–55, 75]. A polynomial-time PIT for this model is known [18, 26, 35, 50, 54]. [33] gave a quasi-polynomial time hitting set via a certain derandomization of the Isolation Lemma [53].

We refer the reader to the surveys [67, 68, 74] for more details on some of the results and the models mentioned above.

## 2 Preliminaries

► **Definition 11** (Hitting set generator). *Let  $\mathcal{C}$  be a set of  $n$ -variate polynomials and  $t \in \mathbb{N}$ . A polynomial map  $\mathcal{G} : \mathbb{F}^t \rightarrow \mathbb{F}^n$  is a hitting set generator for  $\mathcal{C}$  if  $\forall f \in \mathcal{C} \setminus \{0\}$ , we have  $f \circ \mathcal{G} \neq 0$ .*

We say the number of variables of  $\mathcal{G}$  is  $t$ , and the degree of  $\mathcal{G}$  – denoted by  $\deg(\mathcal{G})$  – is the maximum of the degrees of the  $n$  polynomials that define  $\mathcal{G}$ . We will denote the  $t$ -variate polynomial  $f \circ \mathcal{G}$  by  $f(\mathcal{G})$ . By treating a matrix  $A \in \mathbb{F}^{n \times n}$  as a linear transformation from  $\mathbb{F}^n$  to  $\mathbb{F}^n$ , we will denote the polynomial map  $A \circ \mathcal{G}$  by  $A\mathcal{G}$  and the  $t$ -variate polynomial  $f \circ A\mathcal{G}$  by  $f(A\mathcal{G})$ . If the defining polynomials of  $\mathcal{G}$  have degree  $d_0$  and the degree of the polynomials in  $\mathcal{C}$  is at most  $D$ , then the degree of  $f(\mathcal{G})$  is at most  $d_0D$ . Thus, if we are given the defining polynomials of  $\mathcal{G}$ , then we can construct a hitting set for  $\mathcal{C}$  in time  $\text{poly}(n, (d_0D)^t)$  using the Schwartz-Zippel lemma, provided also that  $|\mathbb{F}| > d_0D$ .

### 2.1 The Shpilka-Volkovich generator

► **Definition 12** (The Shpilka-Volkovich hitting set generator [73]). *Assume that  $|\mathbb{F}| \geq n$  and let  $\alpha_1, \dots, \alpha_n$  be distinct elements of  $\mathbb{F}$ . For  $i \in [n]$ , let  $L_i(y) := \prod_{j \in [n], j \neq i} \frac{y - \alpha_j}{\alpha_i - \alpha_j}$  be the  $i$ -th Lagrange interpolation polynomial. Then, for  $t \in \mathbb{N}$ , the Shpilka-Volkovich (SV) generator  $\mathcal{G}_t^{SV} : \mathbb{F}^{2t} \rightarrow \mathbb{F}^n$  is defined as  $\mathcal{G}_t^{SV} := \left( \mathcal{G}_t^{(1)}, \dots, \mathcal{G}_t^{(n)} \right)$  where,  $\mathcal{G}_t^{(i)}(y_1, \dots, y_t, z_1, \dots, z_t) = \sum_{k=1}^t L_i(y_k) \cdot z_k$ .*

Notice that  $\deg(\mathcal{G}_t^{(i)}) = n$ , and  $\mathcal{G}_{t+1}^{SV}|_{(y_{t+1}=\alpha_i)} = \mathcal{G}_t^{SV} + \mathbf{e}_i \cdot z_{t+1}$ , where  $\mathbf{e}_i$  is the  $i$ -th standard basis vector of  $\mathbb{F}^n$ . Thus,  $\text{Img}(\mathcal{G}_t^{SV}) \subseteq \text{Img}(\mathcal{G}_{t+1}^{SV})$  and, continuing in this manner,  $\text{Img}(\mathcal{G}_t^{SV}) \subseteq \text{Img}(\mathcal{G}_{t'}^{SV})$  for any  $t' \geq t$ .

► **Observation 13.** *Let  $f \in \mathbb{F}[\mathbf{x}]$  be a non-zero polynomial that depends on only  $b$  of the  $\mathbf{x}$  variables, and  $g \in \text{orb}(f)$ . Then,  $g$  has a monomial of support at most  $b$  and  $g(\mathcal{G}_b^{SV}) \neq 0$ .*

The above observation is proved in the full version [64]. The following observation, which allows us to construct a hitting set generator for  $f$  from a hitting set generator for  $\frac{\partial f}{\partial x_i}$  is used crucially in the proofs of Theorems 9 and 10 and is proved in the full version [64].

► **Observation 14.** *Let  $f \in \mathbb{F}[\mathbf{x}]$  be an  $n$ -variate, degree  $d$  polynomial, and for some  $m \in \mathbb{N}$ , let  $\mathcal{G} : \mathbb{F}^m \rightarrow \mathbb{F}^n$  be a polynomial map of degree at most  $d'$ . If  $|\mathbb{F}| > dd'$  and there is an  $i \in [n]$  such that  $\frac{\partial f}{\partial x_i}(\mathcal{G}) \neq 0$ , then  $f(\mathcal{G} + \mathcal{G}_1^{SV})$  is not a constant.*

### 2.2 Low support rank concentration

Let  $F$  be a polynomial in  $\mathbf{x}$ -variables with coefficients from  $\mathbb{K}^{w \times w}$ , where  $\mathbb{K}$  is a field and  $w \in \mathbb{N}$ . For an  $m \in \mathbb{N}$ , we say that  $F$  has *support- $m$  rank concentration* over  $\mathbb{K}$  if the coefficient of every monomial in  $F$  is in the  $\mathbb{K}$ -span of the coefficients of the monomials of support at most  $m$  in  $F$ . Support of a monomial  $\mathbf{x}^\alpha$  will be denoted as  $\text{Supp}(\mathbf{x}^\alpha)$ . We prove the below observation in the full version [64].

► **Observation 15.** Let  $f = \mathbf{1}^T \cdot M_1(x_1)M_2(x_2) \cdots M_n(x_n) \cdot \mathbf{1} \in \mathbb{F}[\mathbf{x}]$  be computable by an ROABP of width  $w$ , and  $F = M_1(x_1)M_2(x_2) \cdots M_n(x_n)$ . For an  $m \in \mathbb{N}$  and  $t_1(\mathbf{z}), \dots, t_n(\mathbf{z}) \in \mathbb{F}[\mathbf{z}]$ , where  $\mathbf{z}$  is a set of variables different from  $\mathbf{x}$ , suppose that  $F(\mathbf{x} + \mathbf{t}(\mathbf{z})) := M_1(x_1 + t_1(\mathbf{z}))M_2(x_2 + t_2(\mathbf{z})) \cdots M_n(x_n + t_n(\mathbf{z})) \in \mathbb{F}(\mathbf{z})^{w \times w}[\mathbf{x}]$  has support- $m$  rank concentration over  $\mathbb{F}(\mathbf{z})$ . Then,  $f(x_1 + t_1(\mathbf{z}), \dots, x_n + t_n(\mathbf{z}))$ , when viewed as a polynomial in  $\mathbf{x}$ -variables with coefficients from  $\mathbb{F}[\mathbf{z}]$ , has an  $\mathbf{x}$ -monomial of support at most  $m$ , provided  $f \neq 0$ .

### 2.3 Algebraic rank and faithful homomorphisms

We say that polynomials  $f_1, \dots, f_m \in \mathbb{F}[\mathbf{x}]$  are algebraically independent over  $\mathbb{F}$ , if they do not satisfy any non-trivial polynomial equation over  $\mathbb{F}$ , i.e., for any  $p \in \mathbb{F}[y_1, \dots, y_m]$ ,  $p(f_1, \dots, f_m) = 0$  only if  $p = 0$ . For  $\mathbf{f} = (f_1, \dots, f_m)$ , the transcendence degree (i.e., the algebraic rank) of  $\mathbf{f}$  over  $\mathbb{F}$  is the cardinality of any maximal algebraically independent subset of  $\{f_1, \dots, f_m\}$  over  $\mathbb{F}$ . The notion of algebraic rank is well defined as algebraic independence satisfies the matroid properties.

For  $\mathbf{f} = (f_1, \dots, f_m) \in \mathbb{F}[\mathbf{x}]^m$ , let  $J_{\mathbf{x}}(\mathbf{f})$  denote the Jacobian matrix of  $\mathbf{f}$ . The following well-known lemma relates the transcendence degree of  $\mathbf{f}$  over  $\mathbb{F}$  – denoted by  $\text{tr-deg}_{\mathbb{F}}(\mathbf{f})$  – to the rank of the Jacobian.

► **Lemma 16** (The Jacobian criterion). Let  $\mathbf{f} = (f_1, \dots, f_m) \in \mathbb{F}[\mathbf{x}]^m$  be a tuple of polynomials of degree at most  $D$  and  $\text{tr-deg}_{\mathbb{F}}(\mathbf{f}) = r$ . If  $\text{char}(\mathbb{F}) = 0$  or  $\text{char}(\mathbb{F}) > D^r$ , then  $\text{tr-deg}_{\mathbb{F}}(\mathbf{f}) = \text{rank}_{\mathbb{F}(\mathbf{x})} J_{\mathbf{x}}(\mathbf{f})$ .

► **Definition 17** (Faithful homomorphisms). A homomorphism  $\phi : \mathbb{F}[\mathbf{x}] \rightarrow \mathbb{F}[\mathbf{z}]$  is said to be faithful to  $\mathbf{f} = (f_1, \dots, f_m) \in \mathbb{F}[\mathbf{x}]^m$  if  $\text{tr-deg}_{\mathbb{F}}(\mathbf{f}) = \text{tr-deg}_{\mathbb{F}}(\phi(\mathbf{f}))$ .

► **Lemma 18** (Theorem 2.4 in [5]). If a homomorphism  $\phi : \mathbb{F}[\mathbf{x}] \rightarrow \mathbb{F}[\mathbf{z}]$  is faithful to  $\mathbf{f} = (f_1, \dots, f_m) \in \mathbb{F}[\mathbf{x}]^m$ , then for any  $p \in \mathbb{F}[y_1, \dots, y_m]$ ,  $p(\mathbf{f}) = 0$  if and only if  $p(\phi(\mathbf{f})) = 0$ .

The following lemma was proved in [5, 11].

► **Lemma 19** (Lemma 2.7 of [5]). Let  $\mathbf{f} = (f_1, \dots, f_m)$  be a tuple of polynomials of degree at most  $D$ ,  $\text{tr-deg}_{\mathbb{F}}(\mathbf{f}) \leq r$ , and  $\text{char}(\mathbb{F}) = 0$  or  $> D^r$ . Let  $\psi : \mathbb{F}[\mathbf{x}] \rightarrow \mathbb{F}[\mathbf{z}]$  be a homomorphism such that  $\text{rank}_{\mathbb{F}(\mathbf{z})} J_{\mathbf{x}}(\mathbf{f}) = \text{rank}_{\mathbb{F}(\mathbf{z})} \psi(J_{\mathbf{x}}(\mathbf{f}))$ . Then, the map  $\phi : \mathbb{F}[\mathbf{x}] \rightarrow \mathbb{F}[\mathbf{z}, t, y_1, \dots, y_r]$  that, for all  $i \in [n]$ , maps  $x_i \rightarrow \left( \sum_{j=1}^r y_j t^{ij} \right) + \psi(x_i)$  is faithful to  $\mathbf{f}$ .

We will need the following observation in our proofs. It is proved in the full version [64].

► **Observation 20.** Let  $\mathbf{f} = (f_1, \dots, f_m) \in \mathbb{F}[\mathbf{x}]^m$  be a tuple of polynomials with  $\text{tr-deg}_{\mathbb{F}}(\mathbf{f}) = r$ . For any  $A \in \text{GL}(n, \mathbb{F})$ , let  $g_i = f_i(A\mathbf{x}) \forall i \in [m]$  and  $\mathbf{g} = (g_1, \dots, g_m)$ . Then,  $\text{tr-deg}_{\mathbb{F}}(\mathbf{g}) = r$ .

## 3 Hitting sets for the orbits of commutative ROABPs

**The strategy.** (Recap) Let  $f = \mathbf{1}^T \cdot M_1(x_1)M_2(x_2) \cdots M_n(x_n) \cdot \mathbf{1}$  be a width- $w$  commutative ROABP; here  $M_i(x_i) \in \mathbb{F}^{w \times w}[x_i]$  for all  $i \in [n]$ . Also, let  $F = M_1(x_1)M_2(x_2) \cdots M_n(x_n)$ . For any  $A \in \text{GL}(n, \mathbb{F})$ , let  $g = f(A\mathbf{x})$  and  $G = F(A\mathbf{x})$ . For  $i \in [n]$ , suppose that  $A$  maps  $x_i \mapsto \ell_i(\mathbf{x})$ , where  $\ell_i$  is a linear form, and let  $y_i = \ell_i(\mathbf{x})$  and  $\mathbf{y} = \{y_1, \dots, y_n\}$ . Then,  $g = \mathbf{1}^T \cdot M_1(y_1)M_2(y_2) \cdots M_n(y_n) \cdot \mathbf{1}$  and  $G = M_1(y_1)M_2(y_2) \cdots M_n(y_n)$ . We will show that if  $g \neq 0$ , then there exist explicit “low” degree polynomials  $t_1(\mathbf{z}), \dots, t_n(\mathbf{z})$ , where  $\mathbf{z}$  is a “small” set of variables such that  $g(x_1 + t_1(\mathbf{z}), \dots, x_n + t_n(\mathbf{z}))$  has a “low” support monomial. This

will be done by proving that  $G(x_1 + t_1(\mathbf{z}), \dots, x_n + t_n(\mathbf{z}))$  has low support rank concentration in the “ $\mathbf{y}$ -variables”. Applying Observation 15, we will get that  $g(x_1 + t_1(\mathbf{z}), \dots, x_n + t_n(\mathbf{z}))$  has a low support  $\mathbf{y}$ -monomial. This will then imply that  $g(x_1 + t_1(\mathbf{z}), \dots, x_n + t_n(\mathbf{z}))$  has a low support  $\mathbf{x}$ -monomial, provided  $f$  has low individual degree. Finally, we will plug in the SV generator to preserve the non-zerosness of  $g$ . More precisely, we will prove the following theorem at the end of Section 3.2.

► **Theorem 21.** *Let  $f$  be an  $n$ -variate polynomial with individual degree at most  $d$  that is computable by a width- $w$  commutative ROABP. If  $|\mathbb{F}| \geq n$ , then  $\mathcal{G}_{(2^{\lceil \log w^2 \rceil}(d+1)+1)}^{SV}$  is a hitting set generator for  $\text{orb}(f)$ .*

**Notations and conventions.** In the analysis, we will treat  $t_1(\mathbf{z}), \dots, t_n(\mathbf{z})$  as formal variables  $\mathbf{t} = (t_1, \dots, t_n)$  while always keeping in mind the substitution map  $t_i \mapsto t_i(\mathbf{z})$ . For  $i \in [n]$ , let  $r_i = \ell_i(\mathbf{t})$ . For  $S \subseteq [n]$ , define  $\mathbf{r}_S = \{r_i : i \in S\}$ . The  $\mathbb{F}$ -linear independence of  $\ell_1, \dots, \ell_n$  allows us to treat  $\mathbf{y}$  and  $\mathbf{r}$  as sets of formal variables. Notice that in this notation,  $G(\mathbf{x} + \mathbf{t}) = M_1(y_1 + r_1)M_2(y_2 + r_2) \cdots M_n(y_n + r_n)$ . Let  $\mathbb{A}$  denote the matrix algebra  $\mathbb{F}^{w \times w}$ . For  $i \in [n]$ , let  $M_i(y_i) = \sum_{e_i=0}^d u_{i,e_i} y_i^{e_i}$ , where  $u_{i,e_i} \in \mathbb{A}$  and  $M_i(y_i + r_i) = \sum_{b_i=0}^d v_{i,b_i} y_i^{b_i}$ , where  $v_{i,b_i} \in \mathbb{A}[r_i] \subset \mathbb{A}[\mathbf{t}]$ . As  $f$  is a commutative ROABP,  $M_1(y_1), \dots, M_n(y_n)$  commute with each other and hence  $u_{i,e_i}$  and  $u_{j,e_j}$  also commute for  $i \neq j$ . The following observation, which we prove in the full version [64], implies that  $v_{i,e_i}$  and  $v_{j,e_j}$  also commute for  $i \neq j$ .

► **Observation 22.** *For every  $i \in [n]$  and  $b_i, e_i \in \{0, \dots, d\}$ ,  $v_{i,b_i} = \sum_{e_i=0}^d \binom{e_i}{b_i} \cdot r_i^{e_i - b_i} \cdot u_{i,e_i}$  and  $u_{i,e_i} = \sum_{b_i=0}^d \binom{b_i}{e_i} \cdot (-r_i)^{b_i - e_i} \cdot v_{i,b_i}$ , where  $\binom{a}{b} = 0$  if  $a < b$ .*

For a set  $S = \{i_1, i_2, \dots, i_{|S|}\} \subseteq [n]$ , where  $i_1 < i_2 < \dots < i_{|S|}$ , the vector  $(b_{i_1}, b_{i_2}, \dots, b_{i_{|S|}})$  will be denoted by  $(b_i : i \in S)$ . Let  $\text{Supp}(\mathbf{b})$  denote the support of the vector  $\mathbf{b}$  which is defined as the number of non-zero elements in it. Define the parameter  $m := 2 \lceil \log w^2 \rceil + 1$ .

### 3.1 The goal: low support rank concentration

We set ourselves the goal of proving that there exist explicit degree- $n$  polynomials  $t_1(\mathbf{z}), \dots, t_n(\mathbf{z})$ , where  $|\mathbf{z}| = 2m$ , such that  $G(x_1 + t_1(\mathbf{z}), \dots, x_n + t_n(\mathbf{z})) = M_1(y_1 + r_1)M_2(y_2 + r_2) \cdots M_n(y_n + r_n) \in \mathbb{A}[r_1, \dots, r_n][\mathbf{y}]$  has support- $(m - 1)$  rank concentration over  $\mathbb{F}(\mathbf{z})$  in the  $\mathbf{y}$ -variables. We will show in this and the next section that this happens if all polynomials in a certain collection of non-zero polynomials  $\{h_S(\mathbf{r}_S) : S \subseteq \binom{[n]}{m}\} \subseteq \mathbb{F}[r_1, \dots, r_n]$ , remain non-zero under the substitution  $t_i \mapsto t_i(\mathbf{z})$ . The following lemma, proved in the full version [64], will help us achieve this goal.

► **Lemma 23.** *Let  $G, \mathbf{t}, \mathbf{z}, \mathbf{y}$  and  $\mathbf{r}_S$  be as defined above. Suppose that the following two conditions are satisfied:*

1. *For every  $S \subseteq \binom{[n]}{m}$  and  $(b_i : i \in S) \in \{0, \dots, d\}^m$ , there is a non-zero polynomial  $h_S(\mathbf{r}_S)$  such that  $h_S(\mathbf{r}_S) \cdot \prod_{i \in S} v_{i,b_i} \in \mathbb{F}[\mathbf{t}]$ -span  $\left\{ \prod_{i \in S} v_{i,b'_i} : \text{Supp}(b'_i : i \in S) < m \right\}$ .*
  2. *There exists a substitution  $t_i \mapsto t_i(\mathbf{z})$  that keeps  $h_S(\mathbf{r}_S)$  non-zero for all  $S \subseteq \binom{[n]}{m}$ .*
- Then, for every  $\mathbf{b} = (b_i : i \in [n]) \in \{0, \dots, d\}^n$ ,*

$$\prod_{i \in [n]} v_{i,b_i} \in \mathbb{F}(\mathbf{z})\text{-span} \left\{ \prod_{i \in [n]} v_{i,b'_i} : \text{Supp}(b'_i : i \in [n]) < m \right\},$$

*and  $G(x_1 + t_1(\mathbf{z}), \dots, x_n + t_n(\mathbf{z}))$  has support- $(m - 1)$  rank concentration over  $\mathbb{F}(\mathbf{z})$  in  $\mathbf{y}$ -variables.*

### 3.2 Achieving rank concentration

We will now see how to satisfy conditions 1 and 2 of Lemma 23 such that  $\deg_{\mathbf{r}_S}(h_S(\mathbf{r}_S)) \leq md^{m+1}$ ,  $t_i(\mathbf{z})$  is an explicit degree- $n$  polynomial, and  $|\mathbf{z}| = 2m$ . Assume wlog that  $S = [m]$ . For  $\mathbf{b} = (b_1, \dots, b_m)$  and  $\mathbf{e} = (e_1, \dots, e_m)$  in  $\{0, \dots, d\}^m$ , define  $\binom{\mathbf{b}}{\mathbf{e}} := \prod_{i \in [m]} \binom{b_i}{e_i}$ , where,  $\binom{b_i}{e_i} = 0$  if  $b_i < e_i$ . Also, let  $v_{\mathbf{b}} := \prod_{i \in [m]} v_{i, b_i}$  and  $u_{\mathbf{e}} := \prod_{i \in [m]} u_{i, e_i}$ . Define  $\mathbf{r} := (-r_1, \dots, -r_m)$ ,  $\mathbf{r}^{\mathbf{b}} := \prod_{i \in [m]} (-r_i)^{b_i}$  and  $\mathbf{r}^{-\mathbf{e}} := \prod_{i \in [m]} (-r_i)^{-e_i}$ . We now define some vectors and matrices by fixing an arbitrary order on the elements of  $\{0, \dots, d\}^m$ .

Let  $V := (v_{\mathbf{b}} : \mathbf{b} \in \{0, \dots, d\}^m)$  and  $U := (u_{\mathbf{e}} : \mathbf{e} \in \{0, \dots, d\}^m)$ ;  $V$  is a row vector in  $\mathbb{A}[\mathbf{r}]^{(d+1)^m}$  whereas  $U$  is a row vector in  $\mathbb{A}^{(d+1)^m}$ . Let  $C := \text{diag}(\mathbf{r}^{\mathbf{b}} : \mathbf{b} \in \{0, \dots, d\}^m)$  and  $D := \text{diag}(\mathbf{r}^{-\mathbf{e}} : \mathbf{e} \in \{0, \dots, d\}^m)$ ; both  $C$  and  $D$  are  $(d+1)^m \times (d+1)^m$  diagonal matrices. Let  $M$  be a  $(d+1)^m \times (d+1)^m$  matrix whose rows and columns are indexed by  $\mathbf{b} \in \{0, \dots, d\}^m$  and  $\mathbf{e} \in \{0, \dots, d\}^m$  respectively. The entry of  $M$  indexed by  $(\mathbf{b}, \mathbf{e})$  contains  $\binom{\mathbf{b}}{\mathbf{e}}$ . We now make the following claim which is proved in the full version [64].

▷ **Claim 24.** Let  $U, V, C, M$  and  $D$  be as defined above. Then,  $U = VCMD$ .

In [6], a very similar equation was called the *transfer equation* and we will refer to  $U = VCMD$  by the same name. Let  $F := \{\mathbf{b} \in \{0, \dots, d\}^m : \text{Supp}(\mathbf{b}) = m\}$ ; clearly,  $|F| = d^m$ . Also, let us call the set of all vectors  $(n_{\mathbf{e}} : \mathbf{e} \in \{0, \dots, d\}^m) \in \mathbb{F}^{(d+1)^m}$  for which  $\sum_{\mathbf{e} \in \{0, \dots, d\}^m} n_{\mathbf{e}} u_{\mathbf{e}} = 0$  the *null space* of  $U$ . Then, we have the following lemma.

► **Lemma 25.** *There are vectors  $\{\mathbf{n}_{\mathbf{b}} : \mathbf{b} \in F\}$  in the null space of  $U$  such that the following holds: Let  $N$  be the  $(d+1)^m \times d^m$  matrix whose rows are indexed by  $\mathbf{e} \in \{0, \dots, d\}^m$  and whose columns are indexed by  $\mathbf{b} \in F$  and whose column indexed by  $\mathbf{b}$  is  $\mathbf{n}_{\mathbf{b}}$ . Then, the square matrix  $[CMDN]_F$  is invertible, where  $[CMDN]_F$  is the sub-matrix of  $CMDN$  consisting of only those rows of  $CMDN$  that are indexed by  $\mathbf{b} \in F$ .*

We need the value of  $m$  in the proof of the lemma which is given in Appendix A. For now, observe that  $\det([CMDN]_F) \in \mathbb{F}[\mathbf{r}]$ : Every entry of  $[CMDN]_F$  is a  $\mathbb{F}$ -linear combination of some entries of the matrix  $CMD$ . The entry of  $CMD$  indexed by  $(\mathbf{b}, \mathbf{e})$  is  $\binom{\mathbf{b}}{\mathbf{e}} \cdot \mathbf{r}^{\mathbf{b}} \cdot \mathbf{r}^{-\mathbf{e}}$ , which is non-zero only if  $b_i \geq e_i$  for all  $i \in [m]$ . In this case,  $\mathbf{r}^{\mathbf{b}} \cdot \mathbf{r}^{-\mathbf{e}}$  is a monomial in the  $\mathbf{r}$ -variables. Thus,  $\det([CMDN]_F)$  – which is a polynomial in the entries of  $[CMDN]_F$  – is a polynomial in the  $\mathbf{r}$ -variables. This observation leads to the following corollary of the above lemma, which immediately gives a way to satisfy condition 1 of Lemma 23.

► **Corollary 26.** *Let  $h(\mathbf{r}) := \det([CMDN]_F)$ . Then, for every  $\mathbf{b} \in F$ ,*

$$h(\mathbf{r}) \cdot v_{\mathbf{b}} \in \mathbb{F}[\mathbf{t}]\text{-span} \{v_{\mathbf{b}'} : \mathbf{b}' \in \{0, \dots, d\}^m \text{ and } \text{Supp}(\mathbf{b}') < m\}.$$

The above corollary is proved in the full version [64]. The following claim about  $h(\mathbf{r})$  gives us a way to satisfy condition 2 of Lemma 23. It's proof can be found in the full version [64].

▷ **Claim 27.** The polynomial  $h(\mathbf{r})$ , when viewed as a polynomial in the  $\mathbf{t}$ -variables after setting  $r_i = \ell_i(\mathbf{t})$ , has a  $\mathbf{t}$ -monomial of support at most  $m$ .

By substituting  $\mathcal{G}_m^{SV}$  for  $\mathbf{t}$ , the polynomial  $h(\mathbf{r})$  remains non-zero, satisfying condition 2. The number of variables in  $\mathcal{G}_m^{SV}$ , i.e.,  $|\mathbf{z}| = 2m$  and its degree is  $n$ . The proofs of Theorems 21 and 6 using Lemma 23 can be found in Appendix A.

#### 4 Hitting sets for the orbits of multilinear constant-width ROABPs

**The strategy.** (Recap) Let  $f = \mathbf{1}^T \cdot M_1(x_1)M_2(x_2) \cdots M_n(x_n) \cdot \mathbf{1}$  be a multilinear, width- $w$  ROABP; here  $M_i(x_i) \in \mathbb{F}^{w \times w}[x_i]$  for all  $i \in [n]$ . Also, let  $F = M_1(x_1)M_2(x_2) \cdots M_n(x_n)$ . For any  $A \in \text{GL}(n, \mathbb{F})$ , let  $g = f(A\mathbf{x})$  and  $G = F(A\mathbf{x})$ . For  $i \in [n]$ , suppose that  $A$  maps  $x_i \mapsto \ell_i(\mathbf{x})$ , where  $\ell_i$  is a linear form, and let  $y_i = \ell_i(\mathbf{x})$  and  $\mathbf{y} = \{y_1, \dots, y_n\}$ . Then,  $g = \mathbf{1}^T \cdot M_1(y_1)M_2(y_2) \cdots M_n(y_n) \cdot \mathbf{1}$  and  $G = M_1(y_1)M_2(y_2) \cdots M_n(y_n)$ . Just like in the previous section, we will show that if  $g \neq 0$ , then there exist explicit “low” degree polynomials  $t_1(\mathbf{z}), \dots, t_n(\mathbf{z})$ , where  $\mathbf{z}$  is a “small” set of variables such that  $G(x_1 + t_1(\mathbf{z}), \dots, x_n + t_n(\mathbf{z}))$  has “low” support rank concentration in the “ $\mathbf{y}$ -variables”. While in the rank concentration argument in the previous section the  $\mathbf{x}$ -variables were translated only once, here the translations can be thought of as happening sequentially and in stages. There will be  $\lceil \log n \rceil$  stages with each stage also consisting of multiple translations. After the  $p$ -th stage, the product of any  $2^p$  consecutive matrices in  $G$  will have low support rank concentration in the  $\mathbf{y}$ -variables. Thus, after  $\lceil \log n \rceil$  stages, we will have low support rank concentration in the  $\mathbf{y}$ -variables for  $G(x_1 + t_1(\mathbf{z}), \dots, x_n + t_n(\mathbf{z}))$ .

**Notations and conventions.** Much like in the previous section, we will first translate the  $\mathbf{x}$ -variables by the  $\mathbf{t}$ -variables and then substitute the  $\mathbf{t}$ -variables by low degree polynomials in a small set of variables. We will translate the  $\mathbf{x}$ -variables by  $\lceil \log n \rceil$  groups of  $\mathbf{t}$ -variables,  $\mathbf{t}_1, \dots, \mathbf{t}_{\lceil \log n \rceil}$ . For all  $p \in \lceil \log n \rceil$ , the group  $\mathbf{t}_p$  will have  $\mu := w^2 + \lceil \log w^2 \rceil$  sub-groups of  $\mathbf{t}$ -variables,  $\mathbf{t}_{p,1}, \dots, \mathbf{t}_{p,\mu}$ . For all  $p \in \lceil \log n \rceil$  and  $q \in [\mu]$ ,  $\mathbf{t}_{p,q} := \{t_{p,q,1}, \dots, t_{p,q,n}\}$ . Thus, finally the translation will look like  $x_i \rightarrow x_i + \sum_{p \in \lceil \log n \rceil, q \in [\mu]} t_{p,q,i}$  for all  $i \in [n]$ . Finally, we will substitute the  $\mathbf{t}$ -variables as  $t_{p,q,i} \mapsto s_{p,q} \cdot z_{p,q}^{\beta_{p,q}(i)}$ , where  $\beta_{p,q}(i)$  will be fixed later in the analysis. Let  $r_{p,q,i} := \ell_i(\mathbf{t}_{p,q})$ ; notice that for all  $i \in [n]$ ,  $y_i$  is translated as  $y_i \rightarrow y_i + \sum_{p \in \lceil \log n \rceil, q \in [\mu]} \ell_i(\mathbf{t}_{p,q}) = y_i + \sum_{p \in \lceil \log n \rceil, q \in [\mu]} r_{p,q,i}$ .

For the purpose of analysis, we will think of the translation as happening sequentially in the order  $\mathbf{t}_{1,1}, \dots, \mathbf{t}_{1,\mu}, \mathbf{t}_{2,1}, \dots, \mathbf{t}_{2,\mu}, \dots, \mathbf{t}_{n,1}, \dots, \mathbf{t}_{n,\mu}$ , i.e., we will first translate by  $\mathbf{t}_{1,1}$ , then by  $\mathbf{t}_{1,2}$ , and so on. We denote the order thus imposed on the set  $\{(p, q) : p \in \lceil \log n \rceil, q \in [\mu]\}$  by  $\prec$ .

For a set  $S = \{i_1, i_2, \dots, i_{|S|}\} \subseteq [n]$ , where  $i_1 < i_2 < \dots < i_{|S|}$ , the vector  $(b_{i_1}, b_{i_2}, \dots, b_{i_{|S|}})$  will be denoted by  $(b_i : i \in S)$ . Let  $\text{Supp}(\mathbf{b})$  denote the support of the vector  $\mathbf{b}$  which is defined as the number of non-zero elements in it. The inductive argument given on the next two subsections is inspired by the “merge-and-reduce” idea from [21, 23].

#### 4.1 Low support rank concentration: an inductive argument

In this and the next sections, we will prove the following lemma. Let  $\mathbb{A} := \mathbb{F}^{w \times w}$ .

► **Lemma 28.** *There exist  $\{\beta_{p,q}(i) : p \in \lceil \log n \rceil, q \in [\mu], i \in [n]\} \subset \mathbb{Z}_{\geq 0}$ , such that when we treat  $G(x_1 + \sum_{p \in \lceil \log n \rceil, q \in [\mu]} s_{p,q} \cdot z_{p,q}^{\beta_{p,q}(1)}, \dots, x_n + \sum_{p \in \lceil \log n \rceil, q \in [\mu]} s_{p,q} \cdot z_{p,q}^{\beta_{p,q}(n)})$ , as a polynomial in the  $\mathbf{y}$ -variables over  $\mathbb{A}[r_{p,q,i} : p \in \lceil \log n \rceil, q \in [\mu], i \in [n]]$ , has support- $\mu$  rank concentration in  $\mathbf{y}$ -variables over  $\mathbb{F}(s_{p,q}, z_{p,q} : p \in \lceil \log n \rceil, q \in [\mu])$ . The  $\beta_{p,q}(i)$ s can be found in time  $n^{O(w^4)}$  and each  $\beta_{p,q}(i) \leq n^{O(w^4)}$ .*

We will prove this lemma by induction on  $(p, q)$ . Let us call  $\beta_{p,q}(i)$ s *efficiently computable and good* if they can be found in time  $n^{O(w^4)}$  and each  $\beta_{p,q}(i) \leq n^{O(w^4)}$ . Precisely, the induction hypothesis is as follows.

## 50:14 Hitting Sets for Orbits of Circuit Classes and Polynomial Families

**Induction hypothesis.** Just before translating by  $\mathbf{t}_{p^*,q^*}$ -variables, assume that there exist efficiently computable and good  $\{\beta_{p,q}(i) : (p,q) \prec (p^*,q^*)\}$  such that the product of any  $2^{p^*}$  consecutive matrices in

$$G \left( x_1 + \sum_{(p,q) \prec (p^*,q^*)} s_{p,q} \cdot z_{p,q}^{\beta_{p,q}(1)}, \dots, x_n + \sum_{(p,q) \prec (p^*,q^*)} s_{p,q} \cdot z_{p,q}^{\beta_{p,q}(n)} \right)$$

has support- $(2\mu - (q^* - 1))$  rank concentration over  $\mathbb{F}(s_{p,q}, z_{p,q} : (p,q) \prec (p^*,q^*))$  in  $\mathbf{y}$ -variables.

**Base case.** In the base case,  $(p^*,q^*) = (1,1)$ . Observe that we can assume that  $w \geq 2$ ; if  $w = 1$ , then  $g$  is a product of univariates and the existence of a polynomial time hitting set follows from Observation 13. For any  $w \geq 2$ ,  $2 \leq 2\mu$ . As a product of any two consecutive matrices in  $G$  has support  $2 \leq 2\mu$  rank concentration in the  $\mathbf{y}$ -variables over  $\mathbb{F}$ , the base case is satisfied.

**Induction step.** We need to show that there exist  $\{\beta_{p^*,q^*}(i) : i \in [n]\}$  which are efficiently computable and good, such that after translating by  $\mathbf{t}_{p^*,q^*}$  and substituting  $t_{p^*,q^*,i} \rightarrow s_{p^*,q^*} \cdot z_{p^*,q^*}^{\beta_{p^*,q^*}(i)}$ , the product of any  $2^{p^*}$  consecutive matrices in

$$G \left( x_1 + \sum_{(p,q) \preceq (p^*,q^*)} s_{p,q} \cdot z_{p,q}^{\beta_{p,q}(1)}, \dots, x_n + \sum_{(p,q) \preceq (p^*,q^*)} s_{p,q} \cdot z_{p,q}^{\beta_{p,q}(n)} \right)$$

has support- $(2\mu - q^*)$  rank concentration in the  $\mathbf{y}$ -variables over  $\mathbb{F}(s_{p,q}, z_{p,q} : (p,q) \preceq (p^*,q^*))$ . If  $q^* < \mu$ , then this would mean that the induction hypothesis holds immediately before translation by  $\mathbf{t}_{p^*,q^*+1}$ . Otherwise, if  $q^* = \mu$ , then the following easy-to-verify observation implies that the induction hypothesis holds immediately before translation by  $\mathbf{t}_{p^*+1,1}$ .

► **Observation 29.** Suppose that  $\{\beta_{p,q}(i) : (p,q) \preceq (p^*,\mu)\}$  are such that the product of any  $2^{p^*}$  consecutive matrices in

$$G \left( x_1 + \sum_{(p,q) \preceq (p^*,\mu)} s_{p,q} \cdot z_{p,q}^{\beta_{p,q}(1)}, \dots, x_n + \sum_{(p,q) \preceq (p^*,\mu)} s_{p,q} \cdot z_{p,q}^{\beta_{p,q}(n)} \right)$$

has support- $\mu$  rank concentration in  $\mathbf{y}$ -variables over  $\mathbb{F}(s_{p,q}, z_{p,q} : (p,q) \preceq (p^*,\mu))$ . Then the product of any  $2^{p^*+1}$  consecutive matrices in

$$G \left( x_1 + \sum_{(p,q) \preceq (p^*,\mu)} s_{p,q} \cdot z_{p,q}^{\beta_{p,q}(1)}, \dots, x_n + \sum_{(p,q) \preceq (p^*,\mu)} s_{p,q} \cdot z_{p,q}^{\beta_{p,q}(n)} \right)$$

has support- $2\mu$  rank concentration in the  $\mathbf{y}$ -variables over  $\mathbb{F}(s_{p,q}, z_{p,q} : (p,q) \preceq (p^*,\mu))$ .

Simplifying notations for the ease of exposition. By focusing on the induction step, we will henceforth denote  $\mathbb{F}(s_{p,q}, z_{p,q} : (p,q) \prec (p^*,q^*))$  by  $\mathbb{F}$ , and for all  $i \in [n]$ ,

$$M_i \left( y_j + \sum_{(p,q) \prec (p^*,q^*)} \ell_i \left( s_{p,q} \cdot z_{p,q}^{\beta_{p,q}(1)}, \dots, s_{p,q} \cdot z_{p,q}^{\beta_{p,q}(n)} \right) \right)$$

by  $M_i(y_i)$ ,  $t_{p^*,q^*,i}$  by  $t_i$ ,  $r_{p^*,q^*,i}$  by  $r_i$ ,  $s_{p^*,q^*}$  by  $s$ ,  $z_{p^*,q^*}$  by  $z$  and  $\beta_{p^*,q^*}(i)$  by  $\beta(i)$ .



Without loss of generality, we shall consider the product  $M_1(y_1 + r_1) \cdots M_m(y_m + r_m)$  of the first  $m = 2^{p^*}$  matrices. Our goal is to show that there exist efficiently computable and good  $\{\beta(i) : i \in [m]\}$  such that after substituting  $t_i \rightarrow s \cdot z^{\beta(i)}$ , the above product has support- $(2\mu - q^*)$  rank concentration in the  $\mathbf{y}$ -variables over  $\mathbb{F}(s, z)$  assuming that  $M_1(y_1) \cdots M_m(y_m)$  has support- $(2\mu - (q^* - 1))$  rank concentration in the  $\mathbf{y}$ -variables over  $\mathbb{F}$ .

## 4.2 Details of the induction step

**Recalling some notations.** Before we show how to achieve rank concentration, let us recall some notations defined in Section 3. While in Section 3, the individual degree is  $d$ , here the individual degree is 1 and so, we modify the definitions accordingly.  $\mathbb{A}$  is used to denote the matrix algebra  $\mathbb{F}^{w \times w}$ . For  $i \in [m]$ ,  $M_i(y_i) = \sum_{e_i=0}^1 u_{i,e_i} y_i^{e_i}$ , where  $u_{i,e_i} \in \mathbb{A}$  and  $M_i(y_i + r_i) = \sum_{b_i=0}^1 v_{i,b_i} y_i^{b_i}$ , where  $v_{i,b_i} \in \mathbb{A}[r_i] \subset \mathbb{A}[\mathbf{t}]$ . For  $\mathbf{b} = (b_1, \dots, b_m)$  and  $\mathbf{e} = (e_1, \dots, e_m)$  in  $\{0, 1\}^m$ ,  $\binom{\mathbf{b}}{\mathbf{e}} := \prod_{i \in [m]} \binom{b_i}{e_i}$ . Also,  $v_{\mathbf{b}} := \prod_{i \in [m]} v_{i,b_i}$  and  $u_{\mathbf{e}} := \prod_{i \in [m]} u_{i,e_i}$ . Moreover,  $\mathbf{r} := (-r_1, \dots, -r_m)$ ,  $\mathbf{r}^{\mathbf{b}} := \prod_{i \in [m]} (-r_i)^{b_i}$  and  $\mathbf{r}^{-\mathbf{e}} := \prod_{i \in [m]} (-r_i)^{-e_i}$ . Let  $\mathbf{t} := (t_1, \dots, t_n)$ .

The following vectors and matrices are defined by fixing an arbitrary order on the elements of  $\{0, 1\}^m$ .  $V := (v_{\mathbf{b}} : \mathbf{b} \in \{0, 1\}^m)$  and  $U := (u_{\mathbf{e}} : \mathbf{e} \in \{0, 1\}^m)$ ;  $V$  is a row vector in  $\mathbb{A}[\mathbf{r}]^{2^m}$  whereas  $U$  is a row vector in  $\mathbb{A}^{2^m}$ . Both  $C := \text{diag}(\mathbf{r}^{\mathbf{b}} : \mathbf{b} \in \{0, 1\}^m)$  and  $D := \text{diag}(\mathbf{r}^{-\mathbf{e}} : \mathbf{e} \in \{0, 1\}^m)$  are  $2^m \times 2^m$  diagonal matrices. Finally,  $M$  is a  $2^m \times 2^m$  numeric matrix whose rows and columns were indexed by  $\mathbf{b} \in \{0, 1\}^m$  and  $\mathbf{e} \in \{0, 1\}^m$ , respectively. The entry of  $M$  indexed by  $(\mathbf{b}, \mathbf{e})$  contains  $\binom{\mathbf{b}}{\mathbf{e}}$ . The proof of the following transfer equation is same as the proof of Claim 24.

▷ **Claim 30.** Let  $U, V, C, M$  and  $D$  be as defined above. Then,  $U = VCMD$ .

Let  $F := \{\mathbf{b} \in \{0, 1\}^m : \text{Supp}(\mathbf{b}) > 2\mu - q^*\}$ . Also, recall that the the *null space* of  $U$  is the set of all vectors  $(n_{\mathbf{e}} : \mathbf{e} \in \{0, 1\}^m) \in \mathbb{F}^{2^m}$  for which  $\sum_{\mathbf{e} \in \{0, 1\}^m} n_{\mathbf{e}} u_{\mathbf{e}} = 0$ . We have the following lemma.

► **Lemma 31.** *There are vectors  $\{\mathbf{n}_{\mathbf{b}} : \mathbf{b} \in F\}$  in the null space of  $U$  such that the following holds: Let  $N$  be the  $2^m \times |F|$  matrix whose rows are indexed by  $\mathbf{e} \in \{0, 1\}^m$  and whose columns are indexed by  $\mathbf{b} \in F$  and whose  $\mathbf{b}$ -th column is  $\mathbf{n}_{\mathbf{b}}$ . Then, the square matrix  $[CMDN]_F$  is invertible, where  $[CMDN]_F$  is the sub-matrix of  $CMDN$  consisting of only those rows of  $CMDN$  that are indexed by  $F$ . Also,  $\det([CMDN]_F) \in \mathbb{F}[\mathbf{r}] \subset \mathbb{F}[\mathbf{t}]$  can be expressed as the ratio of a polynomial in  $\mathbb{F}[\mathbf{t}]$  that contains a monomial of degree at most  $2w^2\mu$  in the  $\mathbf{t}$ -variables and a product of linear forms in  $\mathbb{F}[\mathbf{t}]$ .*

The proof of this lemma, which uses the value of  $\mu$ , is given in the full version [64]. We now complete the induction step using this lemma. As  $\det([CMDN]_F)$  is a polynomial in  $\mathbb{F}[\mathbf{r}]$  we get the following corollaries.

► **Corollary 32.** *Let  $h(\mathbf{r}) := \det([CMDN]_F)$ . Then, for every  $\mathbf{b} \in F$ ,*

$$h(\mathbf{r}) \cdot v_{\mathbf{b}} \in \mathbb{F}[\mathbf{t}]\text{-span} \{v_{\mathbf{b}'} : \mathbf{b}' \in \{0, 1\}^m \text{ and } \text{Supp}(\mathbf{b}') \leq 2\mu - q^*\}. \quad (1)$$

**Proof.** Same as the proof of Corollary 26. ◀

► **Corollary 33.** *Suppose  $\{\beta(i) : i \in [n]\}$  are such that the substitution  $t_i \mapsto s \cdot z^{\beta(i)}$  keeps all non-zero polynomials in  $\mathbb{F}[\mathbf{t}]$  containing a monomial of degree at most  $2w^2\mu$  in the  $\mathbf{t}$ -variables non-zero. Then, the product  $M_1(y_1 + r_1) \cdots M_m(y_m + r_m)$  has support- $(2\mu - q^*)$  rank concentration in the  $\mathbf{y}$ -variables over  $\mathbb{F}(s, z)$  after substituting  $t_i \rightarrow s \cdot z^{\beta(i)}$ .*

**Proof.** Multiply both sides of (1) by  $(h(\mathbf{r}))^{-1}$  after substituting  $t_i \mapsto s \cdot z^{\beta(i)}$ . ◀

The following claim, proved in the full version [64], allows us to compute  $\{\beta(i) : i \in [n]\}$  efficiently.

▷ **Claim 34.** There exist  $\{\beta(i) : i \in [n]\}$  such that the substitution  $t_i \mapsto s \cdot z^{\beta(i)}$  keeps all non-zero polynomials in  $\mathbb{F}[\mathbf{t}]$  containing a monomial of degree at most  $2w^2\mu$  in the  $\mathbf{t}$ -variables non-zero. Moreover, we can find all the  $\beta(i)$  in time  $n^{O(w^4)}$  and each  $\beta(i) \leq n^{O(w^4)}$ .

This completes the induction step. Lemma 28 and Theorem 8 are proved in Appendix B.

## 5 Conclusion

In this paper, we have given efficient hitting sets for orbits of several well-studied circuit classes such as commutative ROABPs and constant-width ROABPs (under the low individual degree restriction), and constant-depth constant-occur formulas and occur-once formulas. In the process, we have obtained efficiently constructible hitting sets for the orbits of the elementary symmetric and power symmetric and sum-product polynomials as well as the iterated matrix multiplication polynomials of width-3, which is a complete family of polynomials for arithmetic formulas under  $p$ -projections. The hitting set problem for the orbits of these circuit classes and polynomial families is interesting as their affine projections capture much larger circuit classes and orbits are a natural and dense subset of the set of affine projections. However, the following questions still remain open:

- **Removing the low individual degree restriction.** The low individual degree restriction is natural as it subsumes the multilinear case. However, it would be ideal if we get rid of this limitation of our results. In particular, can we give an efficient hitting-set construction for the orbits of general commutative ROABPs and constant-width ROABPs?
- **Lower bound and hitting set for the orbits of ROABPs.** We would also like to remove the requirements of commutativity and constant-width from our results on hitting sets for the orbits of ROABPs. It is worth noting that an explicit hitting set for the orbits of ROABPs implies a lower bound for the same model computing some explicit polynomial [1]. To our knowledge, no explicit lower bound is known for the orbits of ROABPs. Can we prove such a lower bound first?
- **Hitting sets for the orbits of Det and IMM.** The determinant (Det) and the iterated matrix multiplication (IMM) polynomial families are complete for the class of algebraic branching programs under  $p$ -projections. Can we design efficiently constructible hitting sets for the orbits of Det and IMM?

---

## References

- 1 Manindra Agrawal. Proving lower bounds via pseudo-random generators. In Ramaswamy Ramanujam and Sandeep Sen, editors, *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science, 25th International Conference, Hyderabad, India, December 15-18, 2005, Proceedings*, volume 3821 of *Lecture Notes in Computer Science*, pages 92–105. Springer, 2005.
- 2 Manindra Agrawal and Somenath Biswas. Primality and identity testing via Chinese remaindering. *J. ACM*, 50(4):429–443, 2003. Conference version appeared in the proceedings of FOCS 1999.
- 3 Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Hitting-sets for ROABP and sum of set-multilinear circuits. *SIAM J. Comput.*, 44(3):669–697, 2015.

- 4 Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2):781–793, 2004.
- 5 Manindra Agrawal, Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena. Jacobian Hits Circuits: Hitting Sets, Lower Bounds for Depth-D Occur-k Formulas and Depth-3 Transcendence Degree-k Circuits. *SIAM J. Comput.*, 45(4):1533–1562, 2016. Conference version appeared in the proceedings of STOC 2012.
- 6 Manindra Agrawal, Chandan Saha, and Nitin Saxena. Quasi-polynomial hitting-set for set-depth- $\Delta$  formulas. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 321–330. ACM, 2013.
- 7 Manindra Agrawal and V. Vinay. Arithmetic Circuits: A Chasm at Depth Four. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 67–75. IEEE Computer Society, 2008.
- 8 Eric Allender and Fengming Wang. On the power of algebraic branching programs of width two. *Comput. Complex.*, 25(1):217–253, 2016. Conference version appeared in the proceedings of ICALP 2011.
- 9 Matthew Anderson, Michael A. Forbes, Ramprasad Saptharishi, Amir Shpilka, and Ben Lee Volk. Identity Testing and Lower Bounds for Read- $k$  Oblivious Algebraic Branching Programs. *ACM Trans. Comput. Theory*, 10(1):3:1–3:30, 2018. Conference version appeared in the proceedings of CCC 2016.
- 10 Matthew Anderson, Dieter van Melkebeek, and Ilya Volkovich. Deterministic polynomial identity tests for multilinear bounded-read formulae. *Comput. Complex.*, 24(4):695–776, 2015. Conference version appeared in the proceedings of CCC 2011.
- 11 Malte Beecken, Johannes Mittmann, and Nitin Saxena. Algebraic independence and blackbox identity testing. *Inf. Comput.*, 222:2–19, 2013. Conference version appeared in the proceedings of ICALP 2011.
- 12 Michael Ben-Or and Richard Cleve. Computing Algebraic Formulas Using a Constant Number of Registers. *SIAM J. Comput.*, 21(1):54–58, 1992. Conference version appeared in the proceedings of STOC 1988.
- 13 Karl Bringmann, Christian Ikenmeyer, and Jeroen Zuiddam. On algebraic branching programs of small width. *J. ACM*, 65(5):32:1–32:29, 2018. Conference version appeared in the proceedings of CCC 2017.
- 14 Rafael Mendes de Oliveira, Amir Shpilka, and Ben lee Volk. Subexponential Size Hitting Sets for Bounded Depth Multilinear Formulas. *Comput. Complex.*, 25(2):455–505, 2016. Conference version appeared in the proceedings of CCC 2015.
- 15 Richard A. DeMillo and Richard J. Lipton. A Probabilistic Remark on Algebraic Program Testing. *Inf. Process. Lett.*, 7(4):193–195, 1978.
- 16 Zeev Dvir and Amir Shpilka. Locally decodable codes with two queries and polynomial identity testing for depth 3 circuits. *SIAM J. Comput.*, 36(5):1404–1434, 2007. Conference version appeared in the proceedings of STOC 2005.
- 17 Jack Edmonds. Systems of distinct representatives and linear algebra. *Journal of research of the National Bureau of Standards*, 71:241–245, 1967.
- 18 Jack Edmonds. Matroid intersection. In P.L. Hammer, E.L. Johnson, and B.H. Korte, editors, *Discrete Optimization I*, volume 4 of *Annals of Discrete Mathematics*, pages 39–49. Elsevier, 1979.
- 19 Stephen A. Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite perfect matching is in quasi-nc. 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 754–763. ACM, 2016.
- 20 Michael A. Forbes. Deterministic divisibility testing via shifted partial derivatives. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 451–465. IEEE Computer Society, 2015.

- 21 Michael A. Forbes, Ramprasad Saptharishi, and Amir Shpilka. Hitting sets for multilinear read-once algebraic branching programs, in any order. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 867–875. ACM, 2014.
- 22 Michael A. Forbes and Amir Shpilka. On identity testing of tensors, low-rank recovery and compressed sensing. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 163–172. ACM, 2012.
- 23 Michael A. Forbes and Amir Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 243–252. IEEE Computer Society, 2013.
- 24 Michael A. Forbes and Amir Shpilka. A PSPACE construction of a hitting set for the closure of small algebraic circuits. 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 1180–1192. ACM, 2018.
- 25 Ariel Gabizon and Ran Raz. Deterministic extractors for affine sources over large fields. *Comb.*, 28(4):415–440, 2008. Conference version appeared in the proceedings of FOCS 2005.
- 26 James F. Geelen. Maximum rank matrix completion. *Linear Algebra and its Applications*, 288:211 – 217, 1999.
- 27 Zeyu Guo and Rohit Gurjar. Improved explicit hitting-sets for roabps. In Jaroslaw Byrka and Raghu Meka, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*, volume 176 of *LIPICs*, pages 4:1–4:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
- 28 Zeyu Guo, Nitin Saxena, and Amit Sinhababu. Algebraic dependencies and PSPACE algorithms in approximative complexity. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, volume 102 of *LIPICs*, pages 10:1–10:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
- 29 Ankit Gupta. Algebraic geometric techniques for depth-4 PIT & sylvester-gallai conjectures for varieties. *Electron. Colloquium Comput. Complex.*, 21:130, 2014. URL: <http://eccc.hpi-web.de/report/2014/130>.
- 30 Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Arithmetic Circuits: A Chasm at Depth 3. *SIAM J. Comput.*, 45(3):1064–1079, 2016. Conference version appeared in the proceedings of FOCS 2013.
- 31 Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Identity testing for constant-width, and any-order, read-once oblivious arithmetic branching programs. *Theory Comput.*, 13(1):1–21, 2017. Conference version appeared in the proceedings of CCC 2016.
- 32 Rohit Gurjar, Arpita Korwar, Nitin Saxena, and Thomas Thierauf. Deterministic Identity Testing for Sum of Read-Once Oblivious Arithmetic Branching Programs. *Comput. Complex.*, 26(4):835–880, 2017. Conference version appeared in the proceedings of CCC 2015.
- 33 Rohit Gurjar and Thomas Thierauf. Linear matroid intersection is in quasi-nc. *Comput. Complex.*, 29(2):9, 2020. Conference version appeared in the proceedings of STOC 2017.
- 34 Joos Heintz and Claus-Peter Schnorr. Testing polynomials which are easy to compute (extended abstract). In Raymond E. Miller, Seymour Ginsburg, Walter A. Burkhard, and Richard J. Lipton, editors, *Proceedings of the 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA*, pages 262–272. ACM, 1980.
- 35 Gábor Ivanyos, Marek Karpinski, and Nitin Saxena. Deterministic polynomial time algorithms for matrix completion problems. *SIAM J. Comput.*, 39(8):3736–3751, 2010.
- 36 Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Comput. Complex.*, 13(1-2):1–46, 2004. Conference version appeared in the proceedings of STOC 2003.

- 37 Erich Kaltofen. Factorization of polynomials given by straight-line programs. *Adv. Comput. Res.*, 5:375–412, 1989.
- 38 Erich Kaltofen and Barry M. Trager. Computing with Polynomials Given By Black Boxes for Their Evaluations: Greatest Common Divisors, Factorization, Separation of Numerators and Denominators. *J. Symb. Comput.*, 9(3):301–320, 1990. Conference version appeared in the proceedings of FOCS 1988.
- 39 Zohar Shay Karnin, Partha Mukhopadhyay, Amir Shpilka, and Ilya Volkovich. Deterministic Identity Testing of Depth-4 Multilinear Circuits with Bounded Top Fan-in. *SIAM J. Comput.*, 42(6):2114–2131, 2013. Conference version appeared in the proceedings of STOC 2010.
- 40 Zohar Shay Karnin and Amir Shpilka. Black box polynomial identity testing of generalized depth-3 arithmetic circuits with bounded top fan-in. *Comb.*, 31(3):333–364, 2011. Conference version appeared in the proceedings of CCC 2008.
- 41 Richard M. Karp, Eli Upfal, and Avi Wigderson. Constructing a perfect matching is in random NC. *Comb.*, 6(1):35–48, 1986. Conference version appeared in the proceedings of STOC 1985.
- 42 Neeraj Kayal. Algorithms for arithmetic circuits. *Electron. Colloquium Comput. Complex.*, 17:73, 2010. URL: <http://eccc.hpi-web.de/report/2010/073>.
- 43 Neeraj Kayal and Shubhangi Saraf. Blackbox polynomial identity testing for depth 3 circuits. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 198–207. IEEE Computer Society, 2009.
- 44 Neeraj Kayal and Nitin Saxena. Polynomial identity testing for depth 3 circuits. *Comput. Complex.*, 16(2):115–138, 2007. Conference version appeared in the proceedings of CCC 2006.
- 45 Adam R. Klivans and Daniel A. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 216–223, 2001.
- 46 Pascal Koiran. Arithmetic circuits: The chasm at depth four gets wider. *Theor. Comput. Sci.*, 448:56–65, 2012.
- 47 Pascal Koiran and Mateusz Skomra. Derandomization and absolute reconstruction for sums of powers of linear forms. *CoRR*, abs/1912.02021, 2019. URL: <http://arxiv.org/abs/1912.02021>.
- 48 Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. Equivalence of polynomial identity testing and polynomial factorization. *Comput. Complex.*, 24(2):295–331, 2015. Conference version appeared in the proceedings of CCC 2014.
- 49 László Lovász. On determinants, matchings, and random algorithms. In Lothar Budach, editor, *Fundamentals of Computation Theory, FCT 1979, Proceedings of the Conference on Algebraic, Arithmetic, and Categorical Methods in Computation Theory, Berlin/Wendisch-Rietz, Germany, September 17-21, 1979*, pages 565–574. Akademie-Verlag, Berlin, 1979.
- 50 László Lovász. Singular spaces of matrices and their application in combinatorics. *Boletim da Sociedade Brasileira de Matemática - Bulletin/Brazilian Mathematical Society*, 20(1):87–99, 1989.
- 51 Dori Medini and Amir Shpilka. Hitting Sets and Reconstruction for Dense Orbits in  $VP_e$  and  $\Sigma\Pi\Sigma$  Circuits. *CoRR*, abs/2102.05632, 2021. URL: <https://arxiv.org/abs/2102.05632>.
- 52 Daniel Minahan and Ilya Volkovich. Complete derandomization of identity testing and reconstruction of read-once formulas. *ACM Trans. Comput. Theory*, 10(3):10:1–10:11, 2018. Conference version appeared in the proceedings of CCC 2017.
- 53 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Comb.*, 7(1):105–113, 1987. Conference version appeared in the proceedings of STOC 1987.
- 54 K. Murota. Mixed matrices: Irreducibility and decomposition. In R. A. Brualdi, S. Friedland, and V. Klee, editors, *Combinatorial and Graph-Theoretical Problems in Linear Algebra. The IMA Volumes in Mathematics and its Applications, vol 50.*, pages 39–71. Springer, New York, NY, 1993.

- 55 H. Narayanan, Huzur Saran, and Vijay V. Vazirani. Randomized Parallel Algorithms for Matroid Union and Intersection, With Applications to Arborescences and Edge-Disjoint Spanning Trees. *SIAM J. Comput.*, 23(2):387–397, 1994. Conference version appeared in the proceedings of SODA 1992.
- 56 Noam Nisan. Lower Bounds for Non-Commutative Computation (Extended Abstract). 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 410–418. ACM, 1991.
- 57 Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994. Conference version appeared in the proceedings of FOCS 1988.
- 58 Noam Nisan and Avi Wigderson. Lower Bounds on Arithmetic Circuits Via Partial Derivatives. *Computational Complexity*, 6(3):217–234, 1997. Conference version appeared in the proceedings of FOCS 1995.
- 59 Shir Peleg and Amir Shpilka. A generalized sylvester-gallai type theorem for quadratic polynomials. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPICs*, pages 8:1–8:33. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
- 60 Shir Peleg and Amir Shpilka. Polynomial time deterministic identity testing algorithm for  $\Sigma^{[3]}\Pi\Sigma\Pi^{[2]}$  circuits via Edelstein-Kelly type theorem for quadratic polynomials. *CoRR*, abs/2006.08263, 2020.
- 61 Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Comput. Complex.*, 14(1):1–19, 2005. Conference version appeared in the proceedings of CCC 2004.
- 62 Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena. The power of depth 2 circuits over algebras. In Ravi Kannan and K. Narayan Kumar, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2009, December 15-17, 2009, IIT Kanpur, India*, volume 4 of *LIPICs*, pages 371–382. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2009.
- 63 Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena. A case of depth-3 identity testing, sparse factorization and duality. *Comput. Complex.*, 22(1):39–69, 2013.
- 64 Chandan Saha and Bhargav Thankey. Hitting Sets for Orbits of Circuit Classes and Polynomial Families. *Electron. Colloquium Comput. Complex.*, 28:15, 2021. URL: <https://eccc.weizmann.ac.il/report/2021/015>.
- 65 Shubhangi Saraf and Ilya Volkovich. Black-Box Identity Testing of Depth-4 Multilinear Circuits. *Comb.*, 38(5):1205–1238, 2018. Conference version appeared in the proceedings of STOC 2011.
- 66 Nitin Saxena. Diagonal circuit identity testing and lower bounds. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfssdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, volume 5125 of *Lecture Notes in Computer Science*, pages 60–71. Springer, 2008.
- 67 Nitin Saxena. Progress on polynomial identity testing. *Bull. EATCS*, 99:49–79, 2009.
- 68 Nitin Saxena. Progress on polynomial identity testing-ii. In M. Agrawal and V. Arvind, editors, *Perspectives in Computational Complexity*, volume 26 of *Progress in Computer Science and Applied Logic*, pages 131–146. Birkhäuser, Cham, 2014.
- 69 Nitin Saxena and C. Seshadhri. Blackbox Identity Testing for Bounded Top-Fanin Depth-3 Circuits: The Field Doesn’t Matter. *SIAM J. Comput.*, 41(5):1285–1298, 2012. Conference version appeared in the proceedings of STOC 2011.
- 70 Nitin Saxena and C. Seshadhri. From sylvester-gallai configurations to rank bounds: Improved blackbox identity test for depth-3 circuits. *J. ACM*, 60(5):33:1–33:33, 2013. Conference version appeared in the proceedings of FOCS 2010.

- 71 Jacob T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *J. ACM*, 27(4):701–717, 1980.
- 72 Amir Shpilka. Sylvester-gallai type theorems for quadratic polynomials. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1203–1214. ACM, 2019.
- 73 Amir Shpilka and Ilya Volkovich. Read-once polynomial identity testing. *Comput. Complex.*, 24(3):477–532, 2015. Conference versions appeared in the proceedings of STOC 2008 and APPROX-RANDOM 2009.
- 74 Amir Shpilka and Amir Yehudayoff. Arithmetic Circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010.
- 75 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.
- 76 Sébastien Tavenas. Improved bounds for reduction to depth 4 and depth 3. *Inf. Comput.*, 240:2–11, 2015. Conference version appeared in the proceedings of MFCS 2013.
- 77 Leslie G. Valiant, Sven Skyum, S. Berkowitz, and Charles Rackoff. Fast Parallel Computation of Polynomials Using Few Processors. *SIAM J. Comput.*, 12(4):641–644, 1983.
- 78 Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposium on Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings*, pages 216–226, 1979.

## A Missing proofs from Section 3

### A.1 Proof of Lemma 25

The entries of  $U$ , the columns of  $M$ , the rows and columns of  $D$ , and the rows of  $N$  are indexed by  $\mathbf{e} \in \{0, \dots, d\}^m$ . Impose an order  $\prec$ , say the lexicographical order, on the indices  $\mathbf{e} \in \{0, \dots, d\}^m$  of  $U$  and the other three matrices. Pick the *minimal* basis of the space spanned by the entries of  $U$  according to this order, i.e., consider the entries of  $U$  in the order dictated by  $\prec$  while forming the basis. Let  $\mathcal{B} := \{\mathbf{e} \in \{0, \dots, d\}^m : u_{\mathbf{e}} \text{ is in the minimal basis of } U \text{ w.r.t. } \prec\}$ .

**Construction of the matrix  $N$ .** The columns of  $N$  are indexed by  $\mathbf{b} \in F$ . We will now specify a set of column vectors  $\{\mathbf{n}_{\mathbf{b}} : \mathbf{b} \in F\}$  in the null space of  $U$  such that the column of  $N$  indexed by  $\mathbf{b} \in F$  is  $\mathbf{n}_{\mathbf{b}}$ . There are two cases for  $\mathbf{b} \in F$ :

**Case 1:**  $\mathbf{b} \in F \setminus \mathcal{B}$ . In this case,  $u_{\mathbf{b}}$  is dependent on  $\{u_{\mathbf{e}} : \mathbf{e} \in \mathcal{B} \text{ and } \mathbf{e} \prec \mathbf{b}\}$ . Pick this dependence vector as  $\mathbf{n}_{\mathbf{b}}$ .

**Case 2:**  $\mathbf{b} \in F \cap \mathcal{B}$ . Let there be  $p$  such  $\mathbf{b}$ , where  $p \leq |\mathcal{B}| \leq w^2$ . For a set  $E \subseteq [m]$  and  $\mathbf{b} \in \{0, \dots, d\}^m$ , let  $(\mathbf{b})_E$  denote the vector obtained by projecting  $\mathbf{b}$  to the coordinates in  $E$ . Roughly speaking, the following claim which is proved in the full version [64] says that each of these  $p$  vectors has a “small signature” that differentiates it from the other  $p - 1$  vectors.

▷ **Claim 35.** There exists a way of numbering all  $\mathbf{b} \in F \cap \mathcal{B}$  as  $\mathbf{b}_1, \dots, \mathbf{b}_p$  and there exist non-empty sets  $E_1, \dots, E_p \subseteq [m]$ , each of size at most  $\log p \leq \log w^2$  such that for all  $k \in [p - 1]$ ,

$$(\mathbf{b}_k)_{E_k} \neq (\mathbf{b}_\ell)_{E_k} \quad \forall \ell \in \{k + 1, \dots, p\} \quad (2)$$

We will call  $E_k$  the *signature* of  $\mathbf{b}_k$  for  $k \in [p]$ . The following claim tells us that for each vector  $\mathbf{b}_k$ , there is a vector that is not in  $\mathcal{B}$  and has support at most  $m - 1$ , but agrees with  $\mathbf{b}_k$  on its signature and so in some sense can be used as a proxy for  $\mathbf{b}_k$ .

▷ **Claim 36.** For every  $k \in [p]$ , there exists a vector  $\mathbf{b}'_k \in \{0, \dots, d\}^m \setminus (F \cup \mathcal{B})$  such that  $(\mathbf{b}'_k)_{E_k} = (\mathbf{b}_k)_{E_k}$  and also  $\mathbf{b}'_k$  and  $\mathbf{b}_k$  agree on all locations where  $\mathbf{b}'_k$  is non-zero.

A proof of the above claim is provided in the full version [64]. We will now use the above two claims to construct  $\mathbf{n}_{\mathbf{b}_k}$  for all  $k \in [p]$ . We will use  $\mathbf{b}'_k$  from Claim 36 as a proxy for  $\mathbf{b}_k$ . Notice that  $u_{\mathbf{b}'_k}$  is dependent on  $\{u_{\mathbf{e}} : \mathbf{e} \in \mathcal{B} \text{ and } \mathbf{e} \prec \mathbf{b}'_k\}$ . Let this dependence vector be  $\mathbf{n}_{\mathbf{b}_k}$ . This completes the construction of  $N$ . We will now show that  $[CMDN]_F$  is an invertible matrix.

**$[CMDN]_F$  is invertible.** As  $C$  is a diagonal matrix with non-zero entries, it is sufficient to show that  $[MDN]_F = [M]_F DN$  is an invertible matrix, where  $[M]_F$  is the sub-matrix of  $M$  consisting of only those rows of  $M$  that are indexed by  $\mathbf{b} \in F$ . The following claim lets us simplify the structure of  $[M]_F$  so that it becomes easier to argue that  $[M]_F DN$  is invertible.

▷ **Claim 37.** There is a row operation matrix  $R \in \text{GL}(d^m, \mathbb{F})$  with  $\det(R) = 1$  such that  $R[M]_F$  has the following structure: The rows of  $R[M]_F$  are indexed by  $\mathbf{b} = (b_1, \dots, b_m) \in F$  and its columns by  $\mathbf{e} = (e_1, \dots, e_m) \in \{0, \dots, d\}^m$ . Its entry indexed by  $(\mathbf{b}, \mathbf{e})$  is non-zero if and only if for all  $i \in [m]$ ,  $b_i = e_i$  if  $e_i \neq 0$ . All the non-zero entries of  $R[M]_F$  are  $\pm 1$ .

The above claim is proved in the full version [64]. Because of this claim, showing that  $R[M]_F DN$  is invertible would suffice. Just like we did with  $M$ , we also impose the order  $\prec$  on the columns of  $R[M]_F$  that are indexed by  $\mathbf{e} \in \{0, \dots, d\}^m$ . Recall that the rows of  $R[M]_F$  and the columns of  $N$  are indexed by  $\mathbf{b} \in F$ . We order these indices as follows: we keep the indices  $\mathbf{b} \in F \setminus \mathcal{B}$  before  $\mathbf{b}_1, \dots, \mathbf{b}_p$ . We will treat  $\mathbf{r}^{-\mathbf{e}}$  as a monomial in  $(-r_1)^{-1}, \dots, (-r_m)^{-1}$  “variables” and impose the order  $\prec$  on the monomials in these variables. Let  $A := \{\mathbf{b} : \mathbf{b} \in F \setminus \mathcal{B}\} \cup \{\mathbf{b}'_1, \dots, \mathbf{b}'_p\}$ ; notice that  $|A| = |F|$ . Also, the elements of  $A$  are ordered as the elements of  $F$  but with  $\mathbf{b}'_k$  replacing  $\mathbf{b}_k$  for  $k \in [p]$ . Then, from the Cauchy-Binet formula and the construction of the matrix  $N$ ,  $\det(R[M]_F DN)$  equals

$$\det([R[M]_F]_{\bullet, A}) [N]_A \cdot \prod_{\mathbf{e} \in A} \mathbf{r}^{-\mathbf{e}} + \text{lower order monomials in the } (-r_1)^{-1}, \dots, (-r_m)^{-1}.$$

Here  $[R[M]_F]_{\bullet, A}$  denotes the restriction of  $R[M]_F$  to the columns indexed by  $\mathbf{e} \in A$ , and  $[N]_A$  denotes the restriction of  $N$  to the rows indexed by  $\mathbf{e} \in A$ . Thus to show that  $R[M]_F DN$  (and therefore  $[CMDN]_F$ ) is invertible, the following two claims, both of which are proved in the full version [64], suffice.

▷ **Claim 38.**  $[N]_A$  is an identity matrix.

▷ **Claim 39.** The matrix  $[R[M]_F]_{\bullet, A}$  is an upper triangular matrix with 1 or  $-1$  entries on the diagonal.

## A.2 Proof of Theorem 21

Let  $f = \mathbf{1}^T \cdot M_1(x_1)M_2(x_2) \cdots M_n(x_n) \cdot \mathbf{1}$  be a width- $w$  commutative ROABP having individual degree at most  $d$ ; here  $M_i \in \mathbb{F}^{w \times w}[x_i]$  for all  $i \in [n]$ . Also, let  $F = M_1(x_1)M_2(x_2) \cdots M_n(x_n)$ . For any  $A \in \text{GL}(n, \mathbb{F})$ , let  $g = f(\mathbf{Ax})$  and  $G = F(\mathbf{Ax})$ . Suppose that  $A$  maps  $x_i \mapsto \ell_i(\mathbf{x})$  and let  $y_i = \ell_i(\mathbf{x})$  for all  $i \in [n]$ . Then,  $g = \mathbf{1}^T \cdot M_1(y_1)M_2(y_2) \cdots M_n(y_n) \cdot \mathbf{1}$  and



$G = M_1(y_1)M_2(y_2) \cdots M_n(y_n)$ . In Sections 3.1 and 3.2, we have shown that  $G(\mathbf{x} + \mathcal{G}_m^{SV})$  has support- $(m-1)$  rank concentration (for  $m = 2 \lceil \log w^2 \rceil + 1$ ) over  $\mathbb{F}(\mathbf{z})$  in the  $\mathbf{y}$ -variables; the  $\mathbf{z}$ -variables are the variables introduced by the  $\mathcal{G}_m^{SV}$  generator. From Observation 15, if  $g(\mathbf{x}) \neq 0$ , then  $g(\mathbf{x} + \mathcal{G}_m^{SV})$ , when viewed as a polynomial over  $\mathbb{F}[\mathbf{z}]$  in the  $\mathbf{y}$ -variables (this we can do as  $g(\mathbf{x} + \mathcal{G}_m^{SV}) = \mathbf{1}^T \cdot G(\mathbf{x} + \mathcal{G}_m^{SV}) \cdot \mathbf{1}$ , and  $G(\mathbf{x} + \mathcal{G}_m^{SV})$  can be viewed as a polynomial over  $\mathbb{A}[\mathbf{z}]$  in the  $\mathbf{y}$ -variables), has a  $\mathbf{y}$ -monomial of support at most  $m-1$ . Let the  $\mathbf{y}$ -degree of this monomial be  $D'$ . As the individual degree of every  $\mathbf{x}$ -variable in  $f$  is at most  $d$ , the individual degree of every  $\mathbf{y}$ -variable in  $g$  is also at most  $d$ . Thus,  $D' \leq (m-1)d$ . As the homogeneous component of  $g(\mathbf{x} + \mathcal{G}_m^{SV})$  of  $\mathbf{y}$ -degree  $D'$  is non-zero, the homogeneous component of  $g(\mathbf{x} + \mathcal{G}_m^{SV})$  (now viewed as polynomial over  $\mathbb{F}[\mathbf{z}]$  in the  $\mathbf{x}$ -variables) of  $\mathbf{x}$ -degree  $D'$  must also be non-zero, since  $\ell_1, \dots, \ell_n$  are linearly independent. This means that  $g(\mathbf{x} + \mathcal{G}_m^{SV})$ , when viewed as a polynomial over  $\mathbb{F}[\mathbf{z}]$  in the  $\mathbf{x}$ -variables, has an  $\mathbf{x}$ -monomial of support (in fact, degree) at most  $D' \leq (m-1)d$ . Thus,  $g(\mathcal{G}_{(m-1)d}^{SV} + \mathcal{G}_m^{SV}) \neq 0$ . Now, it follows directly from the definition of the SV generator that  $\mathcal{G}_{(m-1)d}^{SV} + \mathcal{G}_m^{SV} = \mathcal{G}_{m+(m-1)d}^{SV}$  and so  $g(\mathcal{G}_{m+(m-1)d}^{SV}) \neq 0$ . Replacing  $m$  by its value  $2 \lceil \log w^2 \rceil + 1$  proves the theorem. Note that the SV generator needs  $|\mathbb{F}| \geq n$ .

### A.3 Proof of Theorem 6

Let  $f$  be an  $n$ -variate polynomial computed by a width- $w$  commutative ROABP of individual degree at most  $d$ , and  $g \in \text{orb}(f)$ . Then, from Theorem 21,  $g(\mathcal{G}_{(2 \lceil \log w^2 \rceil (d+1)+1)}^{SV}) \neq 0$  whenever  $g \neq 0$ . Now,  $\mathcal{G}_{(2 \lceil \log w^2 \rceil (d+1)+1)}^{SV}$  has  $2(2 \lceil \log w^2 \rceil (d+1) + 1)$  variables, and is of degree  $n$ . So  $g(\mathcal{G}_{(2 \lceil \log w^2 \rceil (d+1)+1)}^{SV})$  also has  $2(2 \lceil \log w^2 \rceil (d+1) + 1)$  variables. Since the individual degree of  $f$  is at most  $d$ , the  $\deg(f) = \deg(g) \leq nd$ . So the degree of  $g(\mathcal{G}_{(2 \lceil \log w^2 \rceil (d+1)+1)}^{SV})$  is at most  $n^2d$ . Thus, as  $|\mathbb{F}| > n^2d$ , a hitting set for  $g$  can be computed in time  $(n^2d + 1)^{(2 \lceil \log w^2 \rceil (d+1)+1)} = (nd)^{O(d \log w)}$ .

## B Missing proofs from Section 4

### B.1 Proof of Lemma 31

The entries of  $U$ , the columns of  $M$ , the rows and columns of  $D$ , and the rows of  $N$  are indexed by  $\mathbf{e} \in \{0, 1\}^m$ . Impose the degree lexicographic order, denoted by  $\prec_{\text{dlex}}$ , on the indices  $\mathbf{e} \in \{0, 1\}^m$  of  $U$  and the other three matrices (by identifying  $\mathbf{e}$  with an  $m$ -variate monomial). Pick the *minimal* basis of the space spanned by the entries of  $U$  according to this order, i.e., consider the entries of  $U$  in the order dictated by  $\prec_{\text{dlex}}$  while forming the basis. Let  $\mathcal{B} := \{\mathbf{e} \in \{0, 1\}^m : u_{\mathbf{e}} \text{ is in the minimal basis of } U \text{ w.r.t. } \prec_{\text{dlex}}\}$ .

► **Observation 40.** *By the induction hypothesis, for every  $\mathbf{e} \in F \cap \mathcal{B}$ ,  $\text{Supp}(\mathbf{e}) = 2\mu - (q^* - 1)$ .*

**Construction of the matrix  $N$ .** The columns of  $N$  are indexed by  $\mathbf{b} \in F$ . We will now specify a set of column vectors  $\{\mathbf{n}_{\mathbf{b}} : \mathbf{b} \in F\}$  in the null space of  $U$  such that the column of  $N$  indexed by  $\mathbf{b} \in F$  is  $\mathbf{n}_{\mathbf{b}}$ . There are two cases for  $\mathbf{b} \in F$ :

**Case 1:**  $\mathbf{b} \in F \setminus \mathcal{B}$ . In this case,  $u_{\mathbf{b}}$  is dependent on  $\{u_{\mathbf{e}} : \mathbf{e} \in \mathcal{B} \text{ and } \mathbf{e} \prec_{\text{dlex}} \mathbf{b}\}$ . Pick this dependence vector as  $\mathbf{n}_{\mathbf{b}}$ .

**Case 2:**  $\mathbf{b} \in F \cap \mathcal{B}$ . Let there be  $p$  such  $\mathbf{b}, \mathbf{b}_1, \dots, \mathbf{b}_p$ , where  $p \leq |\mathcal{B}| \leq w^2$ . For a set  $E \subseteq [m]$  and  $\mathbf{b} \in \{0, 1\}^m$ , let  $(\mathbf{b})_E$  denote the vector obtained by projecting  $\mathbf{b}$  to the coordinates

in  $E$ . Roughly speaking, the following claim, which is proved in the full version [64], says that each of these  $p$  vectors has a “small signature” that differentiates it from the other  $p - 1$  vectors.

- ▷ **Claim 41.** There exist sets  $E_1, \dots, E_p \subseteq [m]$ , each of size  $w^2 - 1$  such that for all  $k \in [p]$ ,
1.  $\text{Supp}((\mathbf{b}_k)_{E_k}) = w^2 - 1$ ,
  2.  $(\mathbf{b}_k)_{E_k} \neq (\mathbf{b}_\ell)_{E_k} \forall \ell \neq k$ .

As before, we will call  $E_k$  the signature of  $\mathbf{b}_k$ . The following claim tells us that for each vector  $\mathbf{b}_k$ , there is a vector that is not in  $\mathcal{B}$  and has support less than  $2\mu - (q^* - 1)$ , but agrees with  $\mathbf{b}_k$  on its signature and so in some sense can be used as a proxy for  $\mathbf{b}_k$ .

- ▷ **Claim 42.** For every  $k \in [p]$ , there exists a vector  $\mathbf{b}'_k \in \{0, 1\}^m \setminus (F \cup \mathcal{B})$  such that  $(\mathbf{b}'_k)_{E_k} = (\mathbf{b}_k)_{E_k}$  and also  $\mathbf{b}'_k$  and  $\mathbf{b}_k$  agree on all locations where  $\mathbf{b}'_k$  is non-zero.

Proof. Similar to the proof of Claim 36. ◁

We will now use the above two claims to construct  $\mathbf{n}_{\mathbf{b}_k}$  for all  $k \in [p]$ . We will use  $\mathbf{b}'_k$  from Claim 42 as a proxy for  $\mathbf{b}_k$ . Notice that  $u_{\mathbf{b}'_k}$  is dependent on  $\{u_{\mathbf{e}} : \mathbf{e} \in \mathcal{B} \text{ and } \mathbf{e} \prec_{\text{dlex}} \mathbf{b}'_k\}$ . Let this dependence vector be  $\mathbf{n}_{\mathbf{b}_k}$ . This completes the construction of  $N$ . We will now show that  $[CMDN]_F$  is invertible. In fact, we will show that  $\det([CMDN]_F)$  is the ratio of a polynomial in  $\mathbb{F}[\mathbf{t}]$  which contains a monomial of degree at most  $2w^2\mu$  and a product of a bunch of non-zero linear forms in  $\mathbb{F}[\mathbf{t}]$ .

**$[CMDN]_F$  is invertible.** Let  $[M]_F$  be the restriction of  $M$  to the rows indexed by  $F$ , and  $[C]_F$  the restriction of  $C$  to the rows and columns indexed by  $F$ .

► **Observation 43.** *The matrix  $[M]_F$  has the following structure: The rows of  $[M]_F$  are indexed by  $\mathbf{b} = (b_1, \dots, b_m) \in F$  and its columns by  $\mathbf{e} = (e_1, \dots, e_m) \in \{0, 1\}^m$ . Its entry indexed by  $(\mathbf{b}, \mathbf{e})$  is non-zero if and only if for all  $i \in [m]$ ,  $b_i = e_i$  if  $e_i \neq 0$ . All non-zero entries are 1.*

We order the indices  $\mathbf{b} \in F$  as follows: Let  $F_0 := \{\mathbf{b} \in F : \text{Supp}(\mathbf{b}) > 2\mu - (q^* - 1)\}$  and  $F_1 := \{\mathbf{b} \in F : \text{Supp}(\mathbf{b}) = 2\mu - (q^* - 1)\}$ . We first keep the  $\mathbf{b} \in F_0$  in (descending) degree lexicographic order<sup>3</sup>, followed by  $\mathbf{b} \in F_1 \setminus \mathcal{B}$  in (reverse) lexicographic order<sup>4</sup>, and then  $\mathbf{b}_1, \dots, \mathbf{b}_p$ . Also, let  $A := (F \setminus \mathcal{B}) \uplus \{\mathbf{b}'_1, \dots, \mathbf{b}'_p\}$ . Notice that  $|A| = |F|$ . Also, the elements of  $A$  are ordered as the elements of  $F$  but with  $\mathbf{b}'_k$  replacing  $\mathbf{b}_k$  for  $k \in [p]$ . For any  $S \subseteq \{0, 1\}^m$  of size  $|S| = |F|$ , let  $[M]_{F,S}$  denote the restriction of  $[M]_F$  to the columns indexed by  $\mathbf{e} \in S$ , and  $[N]_S$  denote the restriction of  $N$  to the rows indexed by  $\mathbf{e} \in S$ . Now,

$$\begin{aligned} \det([CMDN]_F) &= \det([C]_F) \det([M]_F D N) \\ &= \prod_{\mathbf{b} \in F} \mathbf{r}^{\mathbf{b}} \cdot \left( \sum_{\substack{S \subseteq A \uplus \mathcal{B} \\ |S| = |F|}} \det([M]_{F,S}) \cdot \det([N]_S) \cdot \prod_{\mathbf{e} \in S} \mathbf{r}^{-\mathbf{e}} \right) \\ &= \prod_{\mathbf{b} \in F} \mathbf{r}^{\mathbf{b}} \cdot \left( \sum_{\substack{S \subseteq A \uplus \mathcal{B} \\ |S| = |F|}} \det([M]_{F,S}) \cdot \det([N]_S) \cdot \prod_{\mathbf{e} \in S \cap A} \mathbf{r}^{-\mathbf{e}} \cdot \prod_{\mathbf{e} \in S \cap \mathcal{B}} \mathbf{r}^{-\mathbf{e}} \right) \end{aligned}$$

<sup>3</sup> i.e.,  $\mathbf{b}$  comes before  $\hat{\mathbf{b}}$  if  $\text{Supp}(\mathbf{b}) > \text{Supp}(\hat{\mathbf{b}})$ , or if  $\text{Supp}(\mathbf{b}) = \text{Supp}(\hat{\mathbf{b}})$  and  $\hat{\mathbf{b}} \prec_{\text{lex}} \mathbf{b}$ .

<sup>4</sup> i.e.,  $\mathbf{b}$  comes before  $\hat{\mathbf{b}}$  if  $\hat{\mathbf{b}} \prec_{\text{lex}} \mathbf{b}$ .

$$= \prod_{\mathbf{b} \in F} \mathbf{r}^{\mathbf{b}} \cdot \prod_{\mathbf{e} \in A \uplus \mathcal{B}} \mathbf{r}^{-\mathbf{e}} \cdot \left( \sum_{\substack{S \subseteq A \uplus \mathcal{B} \\ |S|=|F|}} \det([M]_{F,S}) \cdot \det([N]_S) \cdot \prod_{\mathbf{e} \in A \setminus S} \mathbf{r}^{\mathbf{e}} \cdot \prod_{\mathbf{e} \in \mathcal{B} \setminus S} \mathbf{r}^{\mathbf{e}} \right),$$

where the second equality follows from the Cauchy-Binet formula and the third equality from the fact that for any  $S \not\subseteq A \uplus \mathcal{B}$ ,  $\det([N]_S) = 0$ . Now, notice that  $\prod_{\mathbf{b} \in F} \mathbf{r}^{\mathbf{b}} \cdot \prod_{\mathbf{e} \in A \uplus \mathcal{B}} \mathbf{r}^{-\mathbf{e}}$  is the reciprocal of a product of non-zero linear forms in  $\mathbf{t}$ -variables, as  $F \subseteq A \uplus \mathcal{B}$ . We shall now prove that

$$\sum_{\substack{S \subseteq A \uplus \mathcal{B} \\ |S|=|F|}} \det([M]_{F,S}) \cdot \det([N]_S) \cdot \prod_{\mathbf{e} \in A \setminus S} \mathbf{r}^{\mathbf{e}} \cdot \prod_{\mathbf{e} \in \mathcal{B} \setminus S} \mathbf{r}^{\mathbf{e}} \quad (3)$$

has a  $\mathbf{t}$ -monomial of degree at most  $w^2(2\mu - (q^* - 1))$ .

▷ **Claim 44.**  $[N]_A$  is an identity matrix.

Proof. Same as that of Claim 38. ◁

▷ **Claim 45.** The matrix  $[M]_{F,A}$  is an upper triangular matrix with ones on the diagonal.

The proof of the above claim is provided in the full version [64].

▷ **Claim 46.**  $\det([M]_{F,A}) \cdot \det([N]_A) \cdot \prod_{\mathbf{e} \in \mathcal{B} \setminus A} \mathbf{r}^{\mathbf{e}} = \prod_{\mathbf{e} \in \mathcal{B}} \mathbf{r}^{\mathbf{e}} \neq 0$  and has  $\mathbf{t}$ -degree at most  $2w^2\mu$ .

Proof.  $\det([M]_{F,A}) \cdot \det([N]_A) \cdot \prod_{\mathbf{e} \in \mathcal{B} \setminus A} \mathbf{r}^{\mathbf{e}} = \prod_{\mathbf{e} \in \mathcal{B}} \mathbf{r}^{\mathbf{e}} \neq 0$  follows from Claims 44 and 45 and the fact that  $A \cap \mathcal{B}$  is empty. For every  $\mathbf{e} \in \mathcal{B}$ ,  $\deg_{\mathbf{t}}(\mathbf{r}^{\mathbf{e}}) \leq 2\mu - (q^* - 1)$ . So,  $\deg_{\mathbf{t}}(\prod_{\mathbf{e} \in \mathcal{B}} \mathbf{r}^{\mathbf{e}}) \leq w^2 \cdot (2\mu - (q^* - 1)) \leq 2w^2\mu$ , as  $|\mathcal{B}| \leq w^2$ . ◁

▷ **Claim 47.** For any  $S \subseteq A \uplus \mathcal{B}$  such that  $|S| = |F|$  and  $\det([N]_S)$  is non-zero, there is a one-to-one correspondence between  $A \setminus S$  and  $S \cap \mathcal{B}$  such that if  $\mathbf{e} \in A \setminus S$  corresponds to  $\mathbf{e}' \in S \cap \mathcal{B}$ , then  $\mathbf{e}' \prec_{\text{dlex}} \mathbf{e}$ .

The above claim, which is proved in the full version [64], implies that for every  $S \in A \uplus \mathcal{B}$  of size  $|F|$ , either  $\det([M]_{F,S}) \cdot \det([N]_S) \cdot \prod_{\mathbf{e} \in A \setminus S} \mathbf{r}^{\mathbf{e}} \cdot \prod_{\mathbf{e} \in \mathcal{B} \setminus S} \mathbf{r}^{\mathbf{e}}$  is 0, or  $\prod_{\mathbf{e} \in \mathcal{B}} \mathbf{r}^{\mathbf{e}} \prec_{\text{dlex}} \prod_{\mathbf{e} \in A \setminus S} \mathbf{r}^{\mathbf{e}} \cdot \prod_{\mathbf{e} \in \mathcal{B} \setminus S} \mathbf{r}^{\mathbf{e}}$ . Hence,  $\prod_{\mathbf{e} \in \mathcal{B}} \mathbf{r}^{\mathbf{e}}$  is the smallest  $\mathbf{r}$ -monomial in the polynomial given in (3) w.r.t.  $\prec_{\text{dlex}}$  order, and so, the homogeneous component of this polynomial that has the same  $\mathbf{r}$ -degree as that of  $\prod_{\mathbf{e} \in \mathcal{B}} \mathbf{r}^{\mathbf{e}}$  survives. Now, from Claim 46 and the fact that  $\ell_1, \dots, \ell_n$  are linearly independent, the polynomial in (3) has a  $\mathbf{t}$ -monomial of degree  $\leq 2w^2\mu$ .

## B.2 Proof of Lemma 28

So far we have proved that there exist  $\{\beta_{p,q}(i) : p \in [\lceil \log n \rceil], q \in [\mu], i \in [n]\}$ , such that  $G \left( x_1 + \sum_{p \in [\lceil \log n \rceil], q \in [\mu]} s_{p,q} \cdot z_{p,q}^{\beta_{p,q}(1)}, \dots, x_n + \sum_{p \in [\lceil \log n \rceil], q \in [\mu]} s_{p,q} \cdot z_{p,q}^{\beta_{p,q}(n)} \right)$  has support- $\mu$  rank concentration in the  $\mathbf{y}$ -variables over  $\mathbb{F}(s_{p,q}, z_{p,q} : p \in [\lceil \log n \rceil], q \in [\mu])$ . Moreover, for each  $(p, q)$ , we can find all  $\beta_{p,q}(i)$  in time  $n^{O(w^4)}$  and each  $\beta_{p,q}(i) \leq n^{O(w^4)}$ . However, since the algorithm that follows from [45] is oblivious, the  $\beta_{p,q}(i)$  found for some fixed  $(p, q)$  can be used for all values of  $(p, q)$ . This proves the lemma.

### B.3 Proof of Theorem 8

Let  $f = \mathbf{1}^T \cdot M_1(x_1)M_2(x_2) \cdots M_n(x_n) \cdot \mathbf{1}$  be a multilinear width- $w$  ROABP; here  $M_i(x_i) \in \mathbb{F}^{w \times w}[x_i]$  for all  $i \in [n]$ . Also, let  $F = M_1(x_1)M_2(x_2) \cdots M_n(x_n)$ . For any  $A \in \text{GL}(n, \mathbb{F})$ , let  $g = f(A\mathbf{x})$  and  $G = F(A\mathbf{x})$ . For  $i \in [n]$ , suppose that  $A$  maps  $x_i \mapsto \ell_i(\mathbf{x})$ , where  $\ell_i$  is a linear form, and let  $y_i = \ell_i(\mathbf{x})$  and  $\mathbf{y} = \{y_1, \dots, y_n\}$ . Then,  $g = \mathbf{1}^T \cdot M_1(y_1)M_2(y_2) \cdots M_n(y_n) \cdot \mathbf{1}$  and  $G = M_1(y_1)M_2(y_2) \cdots M_n(y_n)$ . Let  $\mu = w^2 + \lceil \log w^2 \rceil$ . From Lemma 28, there exist polynomials, say  $t_1, \dots, t_n$ , in  $\mathbb{F}[s_{p,q}, z_{p,q} : p \in [\lceil \log n \rceil], q \in [\mu]]$  of degree at most  $n^{O(w^4)}$  such that  $G(x_1 + t_1, \dots, x_n + t_n)$  has support- $\mu$  rank concentration in the  $\mathbf{y}$ -variables over  $\mathbb{F}(\{s_{p,q}, z_{p,q}\}_{p,q})$ . Moreover, these polynomials can be computed in time  $n^{O(w^4)}$ . Suppose that  $g \neq 0$ . Then, from Observation 15,  $g(x_1 + t_1, \dots, x_n + t_n)$  has a support- $\mu$ ,  $\mathbf{y}$ -monomial when viewed as a polynomial over  $\mathbb{F}[\{s_{p,q}, z_{p,q}\}_{p,q}]$  in the  $\mathbf{y}$ -variables. Since  $f$  is multilinear, as seen in the proof of Theorem 21,  $g(x_1 + t_1, \dots, x_n + t_n)$  has a support- $\mu$ ,  $\mathbf{x}$ -monomial. Thus,  $g(\mathcal{G}_\mu^{SV} + (t_1, \dots, t_n)) \neq 0$ . Now,  $g(\mathcal{G}_\mu^{SV} + (t_1, \dots, t_n))$  is a polynomial in  $2\mu + \mu \cdot \lceil \log n \rceil$  variables over  $\mathbb{F}$ . Also, its degree is at most  $n^{O(w^4)}$ . So, if  $|\mathbb{F}| > n^{O(w^4)}$ , a hitting set for  $g$  can be computed in time  $n^{O(w^4 \cdot \mu \cdot \log n)} = n^{O(w^6 \cdot \log n)}$ . This, along with the time required to compute  $t_1, \dots, t_n$ , still gives a  $n^{O(w^6 \cdot \log n)}$ -time hitting set for  $g$ .

# Sampling Multiple Edges Efficiently

Talya Eden ✉ 🏠 

CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA

Saleet Mossel ✉

CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA

Ronitt Rubinfeld ✉ 🏠

CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA

---

## Abstract

We present a sublinear time algorithm that allows one to sample multiple edges from a distribution that is pointwise  $\epsilon$ -close to the uniform distribution, in an *amortized-efficient* fashion. We consider the adjacency list query model, where access to a graph  $G$  is given via degree and neighbor queries.

The problem of sampling a single edge in this model has been raised by Eden and Rosenbaum (SOSA 18). Let  $n$  and  $m$  denote the number of vertices and edges of  $G$ , respectively. Eden and Rosenbaum provided upper and lower bounds of  $\Theta^*(n/\sqrt{m})$  for sampling a single edge in general graphs (where  $O^*(\cdot)$  suppresses  $\text{poly}(1/\epsilon)$  and  $\text{poly}(\log n)$  dependencies). We ask whether the query complexity lower bound for sampling a single edge can be circumvented when multiple samples are required. That is, can we get an improved amortized per-sample cost if we allow a preprocessing phase? We answer in the affirmative.

We present an algorithm that, if one knows the number of required samples  $q$  in advance, has an overall cost that is sublinear in  $q$ , namely,  $O^*(\sqrt{q} \cdot (n/\sqrt{m}))$ , which is strictly preferable to  $O^*(q \cdot (n/\sqrt{m}))$  cost resulting from  $q$  invocations of the algorithm by Eden and Rosenbaum.

Subsequent to a preliminary version of this work, Tětek and Thorup (arXiv, preprint) proved that this bound is essentially optimal.

**2012 ACM Subject Classification** Theory of computation → Sketching and sampling

**Keywords and phrases** Sampling edges, graph algorithm, sublinear algorithms

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.51

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2008.08032>

**Funding** *Talya Eden*: This work was supported by the NSF Grant CCF-1740751, the Eric and Wendy Schmidt Fund, and Ben-Gurion University.

*Ronitt Rubinfeld*: This work was supported the NSF TRIPODS program (awards CCF-1740751 and DMS 2022448), NSF award CCF-2006664 and by the Fintech@CSAIL Initiative.

## 1 Introduction

The ability to select edges uniformly at random in a large graph or network, namely *edge sampling*, is an important primitive, interesting both from a theoretical perspective in various models of computation (e.g., [19, 2, 3, 1, 13, 12, 7, 4, 15]), and from a practical perspective in the study of real-world networks (e.g., [20, 22, 31, 6, 27]). We consider the task of outputting edges from a distribution that is close to uniform; more precisely, the output distribution on edges will be *pointwise*  $\epsilon$ -close to the uniform distribution, so that each edge will be returned with probability in  $[\frac{1-\epsilon}{m}, \frac{1+\epsilon}{m}]$ . Note that this is a stronger notion than the more standard



© Talya Eden, Saleet Mossel, and Ronitt Rubinfeld;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 51; pp. 51:1–51:15



Leibniz International Proceedings in Informatics

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

notion of  $\epsilon$ -close to uniform in *total variation distance* (TVD).<sup>1</sup> We consider this task in the sublinear setting, specifically, in the adjacency list query model, where the algorithm can perform uniform vertex queries, as well as degree and neighbor queries.

Three recent algorithms have been presented for this problem in the adjacency list model. The first, by Eden and Rosenbaum [13], is an  $O^*(n/\sqrt{m})$  query complexity<sup>2</sup> algorithm that works in general graphs.<sup>3</sup> This was later refined by Eden, Ron, and Rosenbaum [7] to an  $O^*(m\alpha/n)$  algorithm for graphs that have arboricity<sup>4</sup> at most  $\alpha$  (where it is assumed that  $\alpha$  is given as input to the algorithm). Finally, in [26], Tětek and Thorup combined techniques from the previous two works and presented the state of the art algorithm for sampling a single edge. This algorithm exponentially improves on the dependency in  $1/\epsilon$  compared to the algorithm by [13]. All of these algorithms were also shown to be essentially optimal if one is interested in outputting a *single* edge sample. Naively, to sample  $q$  edges in general graphs, one can invoke the [26] algorithm  $q$  times, with expected complexity  $O^*(q \cdot (n/\sqrt{m}))$ . In this paper, we prove that this query complexity can be improved to  $O^*(\sqrt{q} \cdot (n/\sqrt{m}))$ . That is, we prove that there exists an algorithm with a better *amortized* query complexity.

## 1.1 Results

We present an algorithm that returns an edge from a distribution that is pointwise  $\epsilon$ -close to uniform, and efficiently supports many edge sample invocations. Assuming one knows in advance the number of required edge samples  $q$ , the overall cost of  $q$  edge samples is  $O^*(q \cdot (n/\sqrt{m}) + q) = O^*(q \cdot (n/\sqrt{m}))$ , where the equality is since we can assume that  $q = O(n^2/m)$ .<sup>5</sup> Subsequent to a preliminary version of this work, Tětek and Thorup [26, Theorem 15] proved that the above result is essentially optimal.

Our algorithm is based on two procedures: a preprocessing procedure that is invoked once, and a sampling procedure which is invoked whenever an edge sample is requested. There is a trade-off between the preprocessing cost and per-sample cost of the sampling procedure. Namely, for a trade-off parameter  $x \geq 1$ , which can be given as input to the algorithm, the preprocessing query complexity is  $O^*(n^2/(m \cdot x))$  and the per-sample cost of the sampling procedure is  $O(x/\epsilon)$ .

► **Theorem 1.1 (Informal).** *Let  $G$  be a graph over  $n$  vertices and  $m$  edges. Assume access to  $G$  is given via the adjacency list query model. There exists an algorithm that, given an approximation parameter  $\epsilon$  and a trade-off parameter  $x$ , has two procedures: a preprocessing procedure, and a sampling procedure. The sampling procedure outputs an edge from a distribution that is pointwise  $\epsilon$ -close to uniform. The preprocessing procedure has  $O^*(n^2/(m \cdot x))$  expected query complexity, and the expected per-sample query complexity of the sampling procedure is  $O(x/\epsilon)$ .*

As mentioned previously, this result is essentially optimal, due to a lower bound by Tětek and Thorup [26].

<sup>1</sup> See Section 1.1 for a detailed discussion comparing TVD-closeness to pointwise closeness.

<sup>2</sup> We note that in all the mentioned algorithms the running time is asymptotically equal to the query complexity, and therefore we limit the discussion to query complexity.

<sup>3</sup> Throughout the paper  $O^*(\cdot)$  is used to suppresses  $\text{poly}(\log n/\epsilon)$  dependencies.

<sup>4</sup> The arboricity of a graph is the minimal number of forests required to cover its edge set.

<sup>5</sup> Observe that if the number of required samples  $q$  exceeds  $n^2/m$ , then one can simply perform  $O(n^2 \log n/m)$  uniform pair queries and with high probability recover all edges in the graph. Hence, we can assume that  $q \leq n^2/m$ , and so the term  $q$  does not asymptotically affect the complexity.

► **Theorem 1.2** (Theorem 15 in [26], restated). *Let  $\epsilon$  be some small constant  $0 < \epsilon < 1$ . Any algorithm that samples  $q$  edges from a distribution that is pointwise  $\epsilon$ -close to uniform in the adjacency list query model must perform  $\Omega(\sqrt{q} \cdot (n/\sqrt{m}))$  queries.*

To better understand how the complexity of our upper bound compares to what was previously known, we give some possible instantiations. First, setting  $x = n/\sqrt{m}$  implies a preprocessing phase with  $O^*(n/\sqrt{m})$  queries and a cost of  $O(n/\sqrt{m})$  per sample, thus recovering the bounds of [13]. Second, setting  $x = 1$  implies a preprocessing phase with  $O(n^2/m)$  queries and a cost of  $O(1/\epsilon)$  per sample. This can be compared to the naive approach of querying the degrees of all the vertices in the graph, and then sampling each vertex with probability proportional to its degree and returning an edge incident to the sampled vertex.<sup>6</sup> Hence, the naive approach yields an  $O(n)$  preprocessing cost and  $O(1)$  per-sample cost while our algorithm with  $x = 1$  yields an  $O^*(n^2/m) = O^*(n/d_{\text{avg}})$  preprocessing and  $O(1/\epsilon)$  per-sample cost, where  $d_{\text{avg}}$  denotes the average degree of the graph.

For a concrete example, consider the case where  $m = \Theta(n)$  and  $q = O(\sqrt{n})$  edge samples are required. Setting  $x = n^{1/4}$  gives an overall cost of  $n^{3/4}$  for sampling  $q$  edges, where previously this would have required  $O(n)$  queries (by either the naive approach, or performing  $O(\sqrt{n})$  invocations of the  $O^*(n/\sqrt{m}) = O^*(\sqrt{n})$  algorithm of [26]). In general, if the number of queries  $q$  is known in advance, then setting  $x = \frac{n/\sqrt{m}}{\sqrt{q}}$ , yields that sampling  $q$  edges has an overall cost of  $O^*(\sqrt{q} \cdot (n/\sqrt{m}))$ , where previously this would have required  $O^*(q \cdot (n/\sqrt{m}))$  queries resulting from  $q$  invocations of the algorithm by [26]. We discuss some more concrete applications in the following section.

### From the augmented model to the general query model

Recently, it has been suggested by Aliakbarpour et al. [3] to consider query models that also provide queries for uniform edge samples, and multiple algorithms have since been developed for this model, e.g., [4, 15, 5, 28].

Currently, for “transferring” results in models that allow uniform edge samples back to models that do not allow such queries in a black-box manner,<sup>7</sup> one must either (1) pay a multiplicative cost of  $O^*(n/\sqrt{m})$  per query (replacing each edge sample query in an invocation of the [13] algorithm for sampling edges), (2) pay an additive cost of  $O(n)$  (using the naive approach described above), or (3) pay an additive cost of  $O^*(n^2/m)$  if pair queries<sup>8</sup> are allowed.<sup>9</sup>

For example, the works by Assadi, Kapralov and Khanna [4], Fichtenberger, Gao and Peng [15], and Biswas, Eden and Rubinfeld [5] give algorithms that rely on edge samples for the tasks of approximately counting and uniformly sampling arbitrary subgraphs in sublinear time. Specifically, these works assume the *augmented* query model which allows for vertex, degree, neighbor, pair as well as uniform edge samples queries. When only vertex, degree, neighbor and pair queries (without uniform edge samples) are provided, this is referred to as the *general* query model [21]. Currently, there are no dedicated algorithms for these tasks in the general model, that does not allow edge samples. For approximating the number of 4-cycles, denoted  $\#C_4$ , the algorithms of [4, 15] have query complexity of  $O^*(m^2/\#C_4)$ .

<sup>6</sup> Indeed, the naive approach returns an edge from a distribution that is *exactly* uniform.

<sup>7</sup> This is true for results for which pointwise-close to uniform edge samples are sufficient, as in the case in all the current sublinear results that rely on edge samples (that we know of).

<sup>8</sup> Pair queries return whether there is an edge between two vertices in the graph.

<sup>9</sup> As one can sample *all* edges in the graph with high probability using  $O^*(n^2/m)$  uniform pair queries (by the coupon collector’s argument), and then return from the set of sampled edges.

## 51:4 Sampling Multiple Edges Efficiently

For a graph with  $m = O(n)$  edges and  $\#C_4 = \Theta(n^{3/2})$  4-cycles, this results in an  $O^*(\sqrt{n})$  query complexity in the augmented model. Using our algorithm, we can set  $q = O(\sqrt{n})$ , and approximately count the number of  $\#C_4$ 's in  $O^*(n^{3/4})$  queries in the general query model, where previously to our results this would have cost  $O(n)$  queries. We note that this “black-box” transformation from the augmented model to the general query model is not guaranteed to be optimal in terms of the resulting complexity in the general model. Indeed, dedicated algorithms for counting and sampling stars and cliques in the general model, prove that this is not the case [18, 9, 11, 10, 8, 28]. Nonetheless, to the best of our knowledge, no other results are currently known for subgraphs apart from stars or cliques, and so this approach provides the only known algorithms for arbitrary subgraph counting and sampling in the general model.

### Pointwise vs. TVD

A more standard measure of distance between two distributions  $P$  and  $Q$  is the total variation distance (TVD),  $d_{TV}(P, Q) = \frac{1}{2} \sum_{x \in \Omega} |P(x) - Q(x)|$ . Observe that this is a strictly weaker measure. That is, pointwise-closeness implies closeness in TVD. Thus our algorithm immediately produce a distribution that is TVD close to uniform. However, being close to a distribution in TVD, does not imply pointwise-closeness.<sup>10</sup> Furthermore, in various settings, this weaker definition is not sufficient, as is the case in some of the applications we mentioned previously. For instance, the uniform edge samples in the algorithms of [4, 15] cannot be replaced in a black-box manner by edge samples that are only guaranteed to be close to uniform in TVD. For a concrete example, consider the task of approximately counting the number of triangles. Let  $G = A \cup B$  be a graph, where  $A$  is a bipartite subgraph over  $(1 - \epsilon)m$  edges, and  $B$  is a clique over  $\epsilon m$  edges. An algorithm that returns a uniformly distributed edge in  $A$  is close in TVD to uniform over the entire edge set of  $G$ . However, it does not allow one to correctly approximate the number of triangles in  $G$ , as the algorithm will never return an edge from the clique, which is where all the triangles reside.

## 1.2 Technical Overview

Sampling (almost) uniformly distributed edges is equivalent to sampling vertices with probability (almost) proportional to their degree  $\frac{d(v)}{2m}$ .<sup>11</sup> Hence, from now on we focus on the latter task.

Consider first the following naive procedure for sampling vertices with probability proportional to their degree. Assume that  $d_{\max}$ , the maximum degree in the graph is known. Query a vertex uniformly at random and return it with probability  $\frac{d(v)}{d_{\max}}$ ; otherwise, return fail. Then each vertex is sampled with probability  $\frac{d(v)}{n \cdot d_{\max}}$ . Therefore, if we repeatedly invoke the above until a vertex is returned, then each vertex is returned with probability  $\frac{d(v)}{2m}$ , as desired. However, the expected number of attempts until a vertex is returned is  $O(\frac{n \cdot d_{\max}}{m})$  (since the overall success probability of a single attempt is  $\sum_{v \in V} \frac{d(v)}{n \cdot d_{\max}} = \frac{2m}{n \cdot d_{\max}}$ ), which could be as high as  $O(\frac{n^2}{m})$  when  $d_{\max} = \Theta(n)$ .

<sup>10</sup> E.g., a distribution that ignores  $\epsilon/2$ -fraction of the edges and is uniform on the rest is close in TVD to uniform, but clearly it is not pointwise close.

<sup>11</sup> Since if every  $v$  is sampled with probability in  $(1 \pm \epsilon) \frac{d(v)}{2m}$ , performing one more uniform neighbor query from  $v$  implies that each specific edge  $(v, w)$  in the graph is sampled with probability in  $(1 \pm \epsilon) \cdot \frac{1}{2m}$ .



Our idea is to partition the graph vertices into *light* and *heavy*, according to some degree threshold  $\tau$ , that will play a similar role to that of  $d_{\max}$  in the naive procedure above. Our algorithm has two procedures, a preprocessing procedure and a sampling procedure. The preprocessing procedure is invoked once in the beginning of the algorithm, and the sampling procedure is invoked every time an edge sample is requested. In the preprocessing procedure we construct a data structure that will later be used to sample heavy vertices. In the sampling procedure, we repeatedly try to sample a vertex, each time either a light or a heavy with equal probability, until a vertex is returned. To sample light vertices, we invoke the above simple procedure with  $\tau$  instead of  $d_{\max}$ . Namely, sample a uniform random vertex  $v$ , if  $d(v) \leq \tau$ , return it with probability  $\frac{d(v)}{\tau}$ . To sample heavy vertices, we use the data structure constructed by the preprocessing procedure as will be detailed shortly.

In the preprocessing procedure, we sample a set  $S$  of  $O\left(\frac{n}{\tau} \cdot \frac{\log n}{\epsilon^2}\right)$  vertices uniformly at random. We then construct a data structure that allows to sample edges incident<sup>12</sup> to  $S$  uniformly at random. It holds that with high probability for every heavy vertex  $v$ , its number of neighbors in  $S$ , denoted  $d_S(v)$ , is close to its expected value,  $d(v) \cdot \frac{|S|}{n}$ . Also, it holds that with high probability the sum of degrees of the vertices in  $S$ , denoted  $d(S)$ , is close to its expected value,  $2m \cdot \frac{|S|}{n}$ . Hence, to sample heavy vertices, we first sample an edge  $(u, v)$  incident to  $S$  uniformly at random (without loss of generality  $u \in S$ ) and then we check if the second endpoint  $v$  is heavy. If so, we return  $v$ , and otherwise we fail. By the previous discussion on the properties of  $S$ , it holds that every heavy vertex is sampled with probability approximately  $\frac{d_S(v)}{d(S)} \approx \frac{d(v)}{2m}$ .

### 1.3 Comparison to Previous Work

For the sake of this discussion assume that  $\epsilon$  is some small constant. Most closely related to our work, is the algorithm of [13]. Their algorithm also works by partitioning the graph's vertices to *light* and *heavy* vertices according to their some degree threshold  $\theta$ . Their method of sampling light edges is identical to ours: one simply samples a vertex uniformly at random, and keeps it with probability  $d(v)/\theta$ . In our algorithm,  $\tau$  is the degree threshold for light and heavy vertices, so that  $\tau$  and  $\theta$  plays the same role. The difference between our works is in the sampling of heavy vertices. To sample heavy vertices, the algorithm of [13] tries to reach heavy vertices by sampling light vertices, and then querying one of their neighbors uniformly at random. For this approach to output heavy vertices with almost equal probability to light vertices,  $\theta$  must be set to  $\Omega(\sqrt{m})$ . Our approach for sampling heavy vertices is different, and relies on the preprocessing phase, which later allows us to reach heavy vertices with  $O(1)$  queries. This allows us, in a sense, to decouple the dependence of the threshold  $\tau$  and the success probability of sampling light vertices. Hence, we can allow to set the degree threshold  $\tau$  to smaller values, which results in a more efficient per-sample complexity (at a cost of a preprocessing step).

The algorithm of [7] also outputs a uniformly distributed single edge, however in graphs with bounded arboricity  $\alpha$ . Here too the algorithm first defines light vertices, setting the threshold to  $\Theta(\alpha)$ . Sampling heavy edge is then performed by starting at light vertices as before, but taking longer random walks of length  $\ell$ , for  $\ell$  chosen uniformly in  $[\log n]$ . This method was later used by Tětek [26] to exponentially improve the dependence in  $\epsilon$  of sampling a single edge in the general setting. It is an interesting open question whether there exists an algorithm for sampling multiple edges in bounded arboricity graphs which has better complexity than the algorithm of this work.

<sup>12</sup>We say that an edge  $(u, v)$  is incident to  $S$  if either  $u$  or  $v$  are in  $S$ .

## 1.4 Further Related Work

We note that some of the related works were already mentioned, but we list them again for the sake of completeness.

### Sampling edges in the adjacency list model

As discussed previously, the most related work to ours is that of [13] for sampling a single edge from an almost uniform distribution in general graphs in  $O^*(n/\sqrt{m})$  expected time. This was later refined by Eden, Rosenbaum and Ron [7] to an  $O^*(n\alpha/m)$  expected time algorithm in bounded arboricity graphs, where a bound  $\alpha$  on the arboricity of the graph at question is also given as input to the algorithm.<sup>13</sup> Recently, Tětek and Thorup [26] proved that the dependency in  $\epsilon$  in the algorithm of [13] could be improved from  $1/\sqrt{\epsilon}$  to  $\log(1/\epsilon)$ . They further proved (subsequent to our work) that given additional access to what they refer to as hash-based neighbor queries, there exists an algorithm for sampling multiple edges (with and without replacement) from the exactly uniform distribution in  $O^*(\sqrt{q} \cdot (n/\sqrt{m}))$  time.

### The augmented edge samples model

In [3], Aliakbarpour et al. suggested a query model which allows access to uniform edge samples and degree queries. In this model they presented an algorithm for approximately counting the number of  $s$ -stars in expected time  $O^*(m/\#H^{1/s})$ , where  $\#H$  denotes the number of  $s$ -stars in the graph. In [4], Assadi, Kapralov and Khanna considered the combined power of neighbor, degree, pair and uniform vertex and edge samples. In this model, they presented an algorithm that approximates the number of occurrences of any arbitrary subgraph  $H$  in a graph  $G$  in expected time  $O^*(m^{\rho(H)}/\#H)$ , where  $\rho(H)$  is the fractional edge cover<sup>14</sup> of  $H$ , and  $\#H$  is the number of occurrences of  $H$  in  $G$ . In the same model, Fichtenberger, Gao, and Peng [15] simplified the above algorithm and proved the same complexity for the additional task of sampling a uniformly distributed copy of  $H$ . Recently, Biswas, Eden and Rubinfeld [5], parameterized the complexity of counting and sampling arbitrary subgraph by what they refer to as the decomposition cost of  $H$ , improving the above results for a large family of subgraphs  $H$ . In [28], Tětek considers this model in the context of approximately counting triangles in the super-linear regime.

### Sampling from networks

Sampling from networks is a very basic primitive that is used in a host of works for studying networks' parameters (e.g., [20, 22, 31, 6, 27]). Most approaches for efficiently sampling edges from networks are random walk based approaches, whose complexity is proportional to the mixing time of the network, e.g., [22, 16, 25, 24]. We note that our approach cannot be directly compared with that of the random walk based ones, as the query models are different: The adjacency list query model assumes access to uniform vertex queries and one can only query one neighbor at a time, while random walk based approaches usually only assume access to arbitrary seed vertices and querying a node reveals its set of neighbors. Furthermore, while in theory the mixing time of a graph can be of order  $O(n)$ , in practice,

<sup>13</sup>Note that since for all graphs  $\alpha \leq \sqrt{m}$ , this results is always at least as good as the previous one.

<sup>14</sup>The fractional edge cover of a graph is minimum weight assignment of weights to the graph's edges, so that the sum of weights over the edges incident to each vertex is at least 1.

social networks tend to have smaller mixing times [24], making random walk based approaches very efficient. Still, denoting the mixing time of the network by  $t_{mix}$ , such approaches require one to perform  $\Omega(t_{mix})$  queries in order to obtain *each* new sample, thus leaving the question of a more efficient amortized sampling procedure open.

## 2 Preliminaries

Let  $G = (V, E)$  be an undirected simple graph over  $n$  vertices. We consider the adjacency list query model, which assumes the following set of queries:

- **Uniform vertex queries:** which return a uniformly distributed vertex in  $V$ .
- **Degree queries:**  $deg(v)$ , which return the degree of the queried vertex.
- **Neighbor queries**  $nbr(v, i)$  which return the  $i^{\text{th}}$  neighbor of  $v$ , if one exists and  $\perp$  otherwise.

We sometimes say that we perform a “uniform neighbor query” from some vertex  $v$ . This can be simply implemented by choosing an index  $i \in [d(v)]$  uniformly at random, and querying  $nbr(v, i)$ .

Throughout the paper we consider each edge from both endpoints. That is, each edge  $\{u, v\}$  is considered as two oriented edges  $(u, v)$  and  $(v, u)$ . Abusing notation, let  $E$  denote the set of all oriented edges, so that  $m = |E| = \sum_{v \in V} d(v)$  and  $d_{\text{avg}} = m/n$ . Unless stated explicitly otherwise, when we say an “edge”, we refer to oriented edges.

For a vertex  $v \in V$  we denote by  $\Gamma(v)$  the set of  $v$ 's neighbors. For a set  $S \subseteq V$  we denote by  $E(S)$  the subset of edges  $(u, v)$  such that  $u \in S$ , and by  $m(S)$  the sum of degrees of all vertices in  $S$ , i.e.  $m(S) = |E(S)| = \sum_{v \in S} d(v)$ . For every vertex  $v \in V$  and set  $S \subseteq V$ , we denote by  $d_S(v)$  the degree of  $v$  in  $S$ ,  $d_S(v) = |\Gamma(v) \cap S|$ .

We consider the following definition of  $\epsilon$ -pointwise close distributions:

► **Definition 1** (Definition 1.1 in [13]). *Let  $Q$  be a fixed probability distribution on a finite set  $\Omega$ . We say that a probability distribution  $P$  is pointwise  $\epsilon$ -close to  $Q$  if for all  $x \in \Omega$ ,*

$$|P(x) - Q(x)| \leq \epsilon Q(x), \quad \text{or equivalently} \quad P(X) \in (1 \pm \epsilon)Q(X).$$

If  $Q = U$ , the uniform distribution on  $\Omega$ , then we say that  $P$  is pointwise  $\epsilon$ -close to uniform.

## 3 Multiple Edge Sampling

As discussed in the introduction, our algorithm consists of a preprocessing procedure that creates a data structure that enables one to sample heavy vertices, and a sampling procedure that samples an almost uniformly distributed edge. Also recall that our procedures are parameterized by a value  $x$  which allows for a trade-off between the preprocessing complexity and the per-sample complexity. Namely, allowing per-sample complexity of  $O(x/\epsilon)$ , our preprocessing procedure will run in time  $O^*(n/(d_{\text{avg}} \cdot x))$ . If one knows the number of queries,  $q$ , then setting  $x = \frac{n/\sqrt{m}}{\sqrt{q}}$  yields the optimal trade-off between the preprocessing and the sampling.

### 3.1 Preprocessing

In this section we present our preprocessing procedure that will later allow us to sample heavy vertices. The procedure and its analysis are similar to the procedure Sample-degrees-typical of Eden, Ron, and Seshadhri [11].

## 51:8 Sampling Multiple Edges Efficiently

The input parameters to the procedure are  $n$ , the number of vertices in the graph,  $x$ , the trade-off parameter,  $\delta$ , a failure probability parameter, and  $\epsilon$ , the approximation parameter. The output is a data structure that, with probability at least  $1 - \delta$ , allows one to sample heavy vertices with probability (roughly) proportional to their degree.

We note that we set  $\bar{x} = \min\{x, \sqrt{n/\bar{d}_{\text{avg}}}\}$  since for values  $x = \Omega(\sqrt{n/\bar{d}_{\text{avg}}})$  it is better to simply use the  $O^*(\sqrt{n/\bar{d}_{\text{avg}}})$  per-sample algorithm of [13]. We shall make use of the following theorems.

► **Theorem 3.1** (Theorem 1.1 of [17], restated.). *There exists an algorithm that, given query access to a graph  $G$  over  $n$  vertices and  $m$  edges, an approximation parameter  $\epsilon \in (0, \frac{1}{2})$ , and a failure parameter  $\delta \in (0, 1)$ , returns a value  $\bar{m}$  such that with probability at least  $1 - \delta$ ,  $\bar{m} \in [(1 - \epsilon)m, m]$ . The expected query complexity and running time of the algorithm are  $O(\frac{n}{\sqrt{m}} \cdot \frac{\log^2 n}{\epsilon^{2.5}})$ .*

► **Theorem 3.2** (Section 4.2 and Lemma 17 in [14], restated.). *For a set  $S$  of size at least  $\frac{n}{\sqrt{m}} \cdot \frac{34}{\epsilon}$ , it holds that with probability at least  $5/6$ ,  $m(S)/s > \frac{1}{2} \cdot (1 - \epsilon) \cdot d_{\text{avg}}$ .*

► **Theorem 3.3** (A data structure for a discrete distribution (e.g., [29, 30, 23])). *There exists an algorithm that receives as input a discrete probability distribution  $P$  over  $\ell$  elements, and constructs a data structure that allows one to sample from  $P$  in linear time  $O(\ell)$ .*

### Preprocessing $(n, \epsilon, \delta, x)$

1. Invoke the algorithm of [17]<sup>a</sup> to get an estimate  $\bar{d}_{\text{avg}}$  of the average degree  $d_{\text{avg}}$ .
2. Let  $\bar{x} = \min\left\{x, \sqrt{n/\bar{d}_{\text{avg}}}\right\}$ .
3. Let  $t = \lceil \log_3(\frac{3}{\delta}) \rceil$ , and let  $\tau = \frac{\bar{x} \cdot \bar{d}_{\text{avg}}}{\epsilon}$ .
4. For  $i = 1$  to  $t$  do:
  - a. Let  $S_i$  be a multiset of  $s = \frac{n}{\tau} \cdot \frac{35 \log(6nt/\delta)}{\epsilon^2}$  vertices chosen uniformly at random.
  - b. Query the degrees of all the vertices in  $S_i$  and compute  $m(S_i) = \sum_{v \in S_i} d(v)$ .
5. Let  $S$  be the first set  $S_i$  such that  $\frac{m(S_i)}{s} \in [\frac{1}{4} \cdot \bar{d}_{\text{avg}}, 12 \cdot \bar{d}_{\text{avg}}]$ .
  - a. If no such set exists, then **return fail**.
  - b. Else, set up a data structure<sup>b</sup>  $D(S)$  that supports sampling each vertex  $v \in S$  with probability  $\frac{d(v)}{m(S)}$ .
6. Let  $\bar{\gamma} = \frac{m(S)}{\bar{d}_{\text{avg}} \cdot |S|}$ .
7. **Return**  $(\bar{\gamma}, \tau, \bar{x}, D(S))$ .

<sup>a</sup> See Theorem 3.1

<sup>b</sup> See Theorem 3.3

The following definitions will be useful in order to prove the lemma regarding the performance of the **Preprocessing** procedure.

► **Definition 2.** *We say that a sampled set  $S \subseteq V$  is  $\epsilon$ -good if the following two conditions hold:*

- *For every heavy vertex  $v \in V_{>\tau}$ ,  $d_S(v) \in (1 \pm \epsilon)|S| \cdot \frac{d(v)}{n}$ .*
- *$\frac{m(S)}{s} \in [\frac{1}{4} \cdot d_{\text{avg}}, 12 \cdot d_{\text{avg}}]$ .*

► **Definition 3.** *We say that  $\bar{d}_{\text{avg}}$  is an  $\epsilon$ -good estimate of  $d_{\text{avg}}$  if  $\bar{d}_{\text{avg}} \in [(1 - \epsilon)d_{\text{avg}}, d_{\text{avg}}]$ .*

► **Lemma 4.** Assume query access to a graph  $G$  over  $n$  vertices,  $\epsilon \in (0, \frac{1}{2})$ ,  $\delta \in (0, 1)$ , and  $x \geq 1$ . The procedure **Preprocessing** $(n, \epsilon, \delta, x)$ , with probability at least  $1 - \delta$ , returns a tuple  $(\bar{\gamma}, \tau, \bar{x}, D(S))$  such that the following holds.

- $D(S)$  is a data structure that supports sampling a uniform edge in  $E(S)$ , for an  $\epsilon$ -good set  $S$ , as defined in Definition 2.
- $\bar{x} \in [1, \sqrt{n/\bar{d}_{\text{avg}}}]$ ,  $\tau = \frac{\bar{x} \cdot \bar{d}_{\text{avg}}}{\epsilon}$ , and  $\bar{\gamma} = \frac{m(S)}{\bar{d}_{\text{avg}} \cdot |S|}$ , where  $\bar{d}_{\text{avg}}$  is an  $\epsilon$ -good estimate of  $d_{\text{avg}}$ , as defined in Definition 3.

The expected query complexity and running time of the procedure are  $O\left(\max\left\{\frac{n}{d_{\text{avg}} \cdot x}, \sqrt{\frac{n}{\bar{d}_{\text{avg}}}}\right\} \cdot \frac{\log^2(n \log(1/\delta)/\delta)}{\epsilon}\right)$ .

**Proof.** We start with proving that with probability at least  $1 - \delta$  the set  $S$  chosen in Step 5 is a good set. Namely, that (1)  $\frac{m(S)}{|S|} \in [\frac{1}{4} \cdot \bar{d}_{\text{avg}}, 12 \cdot \bar{d}_{\text{avg}}]$ , and that (2) for all heavy vertices  $v \in V_{>\tau}$ ,  $d_S(v) \in (1 \pm \epsilon)s \cdot \frac{d(v)}{n}$ .

By Theorem 1.1 of [17] (see Theorem 3.1), with probability at least  $1 - \frac{\delta}{3}$ ,  $\bar{d}_{\text{avg}}$  is an  $\epsilon$ -good estimate of  $d_{\text{avg}}$ , that is

$$(1 - \epsilon)d_{\text{avg}} \leq \bar{d}_{\text{avg}} \leq d_{\text{avg}}. \quad (1)$$

We henceforth condition on this event, and continue to prove the latter property. Fix an iteration  $i \in [t]$ . Observe that  $\mathbb{E}\left[\frac{m(S_i)}{s}\right] = d_{\text{avg}}$ . By Markov's inequality,<sup>15</sup> equation (1), and the assumption that  $\epsilon \in (0, \frac{1}{2})$ ,

$$\Pr\left[\frac{m(S_i)}{s} > 12 \cdot \bar{d}_{\text{avg}}\right] \leq \frac{d_{\text{avg}}}{12 \cdot \bar{d}_{\text{avg}}} \leq \frac{1}{12(1 - \epsilon)} \leq \frac{1}{6}.$$

Recall that  $s = \frac{n}{\tau} \cdot \frac{35 \log(6nt/\delta)}{\epsilon^2}$ ,  $\tau = \frac{\bar{x} \cdot \bar{d}_{\text{avg}}}{\epsilon}$ , and  $\bar{x} \leq \sqrt{n/\bar{d}_{\text{avg}}}$  and that we condition on  $\bar{d}_{\text{avg}} \geq (1 - \epsilon)d_{\text{avg}}$ . Thus,  $\tau \leq \frac{\sqrt{m}}{\epsilon}$ , and  $s \geq \frac{34}{\epsilon} \cdot \frac{n}{\sqrt{m}}$ . Therefore, by Lemma 17 in [14] (see Theorem 3.2), for every  $i$ , it holds that

$$\Pr\left[\frac{m(S_i)}{s} \leq \frac{1}{2} \cdot (1 - \epsilon) d_{\text{avg}}\right] \leq \frac{1}{6}. \quad (2)$$

By equations (1), (2), and the assumption that  $\epsilon \in (0, \frac{1}{2})$ ,

$$\Pr\left[\frac{m(S_i)}{s} < \frac{1}{4} \cdot \bar{d}_{\text{avg}}\right] \leq \Pr\left[\frac{m(S_i)}{s} \leq \frac{1}{2} \cdot (1 - \epsilon) d_{\text{avg}}\right] \leq \frac{1}{6}$$

By the union bound, for every specific  $i$ ,

$$\Pr\left[\frac{m(S_i)}{s} < \frac{1}{4} \cdot \bar{d}_{\text{avg}} \quad \text{or} \quad \frac{m(S_i)}{s} > 12 \cdot \bar{d}_{\text{avg}}\right] \leq \frac{1}{3}.$$

Hence, the probability that for all the selected multisets  $\{S_i\}_{i \in [t]}$ , either  $\frac{m(S_i)}{s} < \frac{1}{4} \cdot \bar{d}_{\text{avg}}$  or  $\frac{m(S_i)}{s} > 12 \cdot \bar{d}_{\text{avg}}$  is bounded by  $\frac{1}{3^t} = \frac{\delta}{3}$  (recall  $t = \lceil \log_3(\frac{3}{\delta}) \rceil$ ). Therefore, with probability at least  $1 - \frac{2\delta}{3}$ , it holds that  $\frac{m(S)}{s} \in [\frac{1}{4} \cdot \bar{d}_{\text{avg}}, 12 \cdot \bar{d}_{\text{avg}}]$ , and the procedure does not return *fail* in Step 5a.

<sup>15</sup> Markov's inequality: if  $X$  is a non-negative random variable and  $a > 0$ ,  $P(X \geq a) \leq \frac{E(X)}{a}$ .

## 51:10 Sampling Multiple Edges Efficiently

Next, we prove that there exists a high-degree vertex  $v \in V_{>\tau}$  such that  $d_S(v) \notin (1 \pm \epsilon)s \cdot \frac{d(v)}{n}$  with probability at most  $\frac{\delta}{3}$ . Fix an iteration  $i \in [t]$ , and let  $S_i = \{u_1, \dots, u_s\}$  be the sampled set. For any fixed high-degree vertex  $v \in V_{>\tau}$  and for some vertex  $u \in V$ , let

$$\chi^v(u) = \begin{cases} 1 & u \text{ is a neighbor of } v \\ 0 & \text{otherwise} \end{cases}.$$

Observe that  $\mathbb{E}_{u \in V} [\chi^v(u)] = \frac{d(v)}{n}$ , and that  $d_{S_i}(v) = \sum_{j \in [s]} \chi^v(u_j)$ . Thus,  $\mathbb{E}[d_{S_i}(v)] = s \cdot \frac{d(v)}{n}$ . Since the  $\chi^v(u)$  variables are independent  $\{0, 1\}$  random variables, by the multiplicative Chernoff bound,<sup>16</sup>

$$\Pr \left[ \left| d_{S_i}(v) - \frac{s \cdot d(v)}{n} \right| \geq \epsilon \cdot \frac{s \cdot d(v)}{n} \right] \leq 2 \exp \left( -\frac{\epsilon^2 \cdot s \cdot d(v)}{3n} \right) \leq \frac{\delta}{3nt}, \quad (3)$$

where the last inequality is by the assumption that  $\epsilon \in (0, \frac{1}{2})$ , the setting of  $s = \frac{n}{\tau} \cdot \frac{35 \log(6nt/\delta)}{\epsilon^2}$ , and since we fixed a heavy vertex  $v$  so that  $d(v) \geq \tau$ . By taking a union bound over all high-degree vertices, it holds that there exists  $v \in V_{>\tau}$  such that  $d_{S_i}(v) \notin (1 \pm \epsilon) \frac{s \cdot d(v)}{n}$  with probability at most  $\frac{\delta}{3t}$ .

Hence, with probability at least  $1 - \delta$ ,  $D(S)$  is a data structure of a good set  $S$ . Moreover, by steps 2, 6, and 3 in the procedure **Preprocessing**( $n, \epsilon, \delta, x$ ) it holds that  $\bar{x} \in \left[ 1, \sqrt{n/\bar{d}_{\text{avg}}} \right]$ ,

$\bar{\gamma} = \frac{m(S)}{\bar{d}_{\text{avg}} \cdot |S|}$ , and  $\tau = \frac{\bar{x} \cdot \bar{d}_{\text{avg}}}{\epsilon}$  respectively. By equation (1),  $\bar{d}_{\text{avg}}$  is an  $\epsilon$ -good estimate for  $d_{\text{avg}}$ .

We now turn to analyze the complexity. By [17] (see Theorem 3.1), the query complexity and running time of step 1 is  $O\left(\frac{n}{\sqrt{m}} \cdot \frac{\log^2(n)}{\epsilon^{2.5}}\right)$ . The expected query complexity and running time of the for loop are  $O(t \cdot s) = O\left(\frac{n}{\bar{d}_{\text{avg}} \cdot \bar{x}} \cdot \frac{\log^2(n \log(1/\delta)/\delta)}{\epsilon}\right)$ , where the equality holds by the setting of  $s, t$  and since the expected value of  $\bar{d}_{\text{avg}}$  is  $d_{\text{avg}}$ . Step 5 takes  $O(t)$  time. By [29, 30, 23] (see Theorem 3.3), the running time of step 5b is  $O(s)$ . All other steps takes  $O(1)$  time. Hence, the expected query complexity and running time are dominated by the for loop. By the setting of  $\bar{x} = \min\{x, \sqrt{n/\bar{d}_{\text{avg}}}\}$  we have  $O(s \cdot t) = O\left(\frac{n}{\bar{d}_{\text{avg}} \cdot \bar{x}} \cdot \frac{\log^2(n \log(1/\delta)/\delta)}{\epsilon}\right) = O\left(\max\left\{\frac{n}{\bar{d}_{\text{avg}} \cdot x}, \sqrt{\frac{n}{\bar{d}_{\text{avg}}}}\right\} \cdot \frac{\log^2(n \log(1/\delta)/\delta)}{\epsilon}\right)$  which proves the claim.  $\blacktriangleleft$

### 3.2 Sampling an edge

In this section we present our sampling procedures. The following definition and claim will be useful in our analysis.

► **Definition 5.** Let  $\tau$  be a degree threshold. Let  $V_{\leq\tau} = \{v \in V \mid d(v) \leq \tau\}$ , and let  $V_{>\tau} = V \setminus V_{\leq\tau}$ . We refer to  $V_{\leq\tau}$  and  $V_{>\tau}$  as the sets of light vertices and heavy vertices, respectively. Let  $E_{\leq\tau} = \{(u, v) \mid u \in V_{\leq\tau}\}$  and  $E_{>\tau} = \{(u, v) \mid u \in V_{>\tau}\}$ .

► **Definition 6.** If the procedure **Preprocessing**( $n, \epsilon, \delta, x$ ) returns a tuple  $(\bar{\gamma}, \tau, \bar{x}, D(S))$  such that the following items of Lemma 4 hold, then we say that this invocation is successful.

- $D(S)$  is a data structure that supports sampling a uniform edge in  $E(S)$ , for an  $\epsilon$ -good set  $S$ , as defined in Definition 2.
- $\bar{x} \in [1, \sqrt{n/\bar{d}_{\text{avg}}}]$ ,  $\tau = \frac{\bar{x} \cdot \bar{d}_{\text{avg}}}{\epsilon}$ , and  $\bar{\gamma} = \frac{m(S)}{\bar{d}_{\text{avg}} \cdot |S|}$ , where  $\bar{d}_{\text{avg}}$  is an  $\epsilon$ -good estimate of  $d_{\text{avg}}$ , as defined in Definition 3.

<sup>16</sup> Multiplicative Chernoff bound: if  $X_1, \dots, X_n$  are independent random variables taking values in  $\{0, 1\}$ , then for any  $0 \leq \delta \leq 1$ ,  $\Pr \left[ \left| \sum_{i \in [n]} X_i - \mu \right| \geq \delta \mu \right] \leq 2e^{-\frac{\delta^2 \mu}{3}}$  where  $\mu = \mathbb{E} \left[ \sum_{i \in [n]} X_i \right]$ .

▷ **Claim 7.** Let  $\gamma = \frac{m(S)}{d_{\text{avg}} \cdot |S|}$  and  $\bar{\gamma} = \frac{m(S)}{d_{\text{avg}} \cdot |S|}$ . If  $S$  is an  $\epsilon$ -good set, as in Definition 2, and  $\bar{d}_{\text{avg}}$  is an  $\epsilon$ -good estimate of  $d_{\text{avg}}$ , as in Definition 3, then it holds that  $\bar{\gamma} \in [1/4, 12]$  and that  $\gamma \in [(1 - \epsilon)\bar{\gamma}, \bar{\gamma}]$ .

*Proof.* By the assumption that  $S$  is an  $\epsilon$ -good set, it holds that  $\frac{m(S)}{|S|} \in [\frac{1}{4} \cdot \bar{d}_{\text{avg}}, 12 \cdot \bar{d}_{\text{avg}}]$ . Therefore,  $\bar{\gamma} \in [\frac{1}{4}, 12]$ . By the assumption that  $\bar{d}_{\text{avg}}$  is an  $\epsilon$ -good estimate of  $d_{\text{avg}}$ , namely  $\bar{d}_{\text{avg}} \in [(1 - \epsilon)d_{\text{avg}}, d_{\text{avg}}]$ , it holds that  $\gamma \in [(1 - \epsilon)\bar{\gamma}, \bar{\gamma}]$ . ◁

### 3.2.1 The sampling procedures

We now present the two procedures for sampling light edges and heavy edges.

**Sample-Uniform-Edge** ( $\bar{\gamma}, \tau, \bar{x}, D(S), \epsilon$ )

1. While **True** do:
  - a. Sample uniformly at random a bit  $b \leftarrow \{0, 1\}$ .
  - b. If  $b = 0$  invoke **Sample-Light**( $\bar{\gamma}, \tau$ ).
  - c. Otherwise, invoke **Sample-Heavy**( $\tau, D(S), \bar{x}, \epsilon$ ).
  - d. If an edge  $(v, u)$  was returned, then **return**  $(v, u)$ .

**Sample-Light** ( $\bar{\gamma}, \tau$ )

1. Sample a vertex  $v \in V$  uniformly at random and query for its degree.
2. If  $d(v) > \tau$  **return fail**.
3. Query a uniform neighbor of  $v$ . Let  $u$  be the returned vertex.
4. **Return**  $(v, u)$  with probability  $\frac{d(v)}{\tau} \cdot \frac{1}{4\bar{\gamma}}$ , otherwise **return fail**.

**Sample-Heavy** ( $\tau, D(S), \bar{x}, \epsilon$ )

1. Sample from the data structure  $D(S)$  a vertex  $v \in S$  with probability  $\frac{d(v)}{m(S)}$ .
2. Sample uniform neighbor of  $v$ . Let  $u$  be the returned vertex.
3. If  $d(u) \leq \tau$  **return fail**.
4. Sample uniform neighbor of  $u$ . Let  $w$  be the returned vertex.
5. **Return**  $(u, w)$  with probability  $\epsilon/4\bar{x}$ , otherwise **return fail**.

Our procedure for sampling an edge **Sample-Uniform-Edge** gets as input a tuple  $(\bar{\gamma}, \tau, \bar{x}, D(S))$  which is the output of the procedure **Preprocessing**. Our guarantees on the resulting distribution of edge samples rely on the preprocessing being successful (see Definition 6), which happens with probability at least  $1 - \delta$ .

► **Lemma 8.** *Assume that **Preprocessing** has been invoked successfully, as defined in Definition 6. The procedure **Sample-Light**( $\bar{\gamma}, \tau$ ) returns an edge in  $E_{\leq \tau}$  such that each edge is returned with probability  $\frac{\epsilon|S|}{4n \cdot \bar{x} \cdot m(S)}$ . The query complexity and running time of the procedure are  $O(1)$ .*

**Proof.** Let  $(v, u)$  be a fixed edge in  $E_{\leq \tau}$ .

$$\begin{aligned} \Pr[(v, u) \text{ returned}] &= \Pr[(v \text{ is sampled in Step 1) and } (u \text{ sampled in Step 3}) \\ &\quad \text{and } ((v, u) \text{ returned in Step 4})] \\ &= \frac{1}{n} \cdot \frac{1}{d(v)} \cdot \frac{d(v)}{\tau \cdot 4\bar{\gamma}}. \end{aligned}$$

## 51:12 Sampling Multiple Edges Efficiently

Note that by Claim 7,  $1/4\bar{\gamma} \leq 1$  and therefore, Step 4 is valid and the above holds. Hence, by the setting of  $\tau = \frac{\bar{x} \cdot \bar{d}_{\text{avg}}}{\epsilon}$  and  $\bar{\gamma} = \frac{m(S)}{d_{\text{avg}} \cdot |S|}$ ,

$$\Pr[(v, u) \text{ is returned}] = \frac{1}{n \cdot \tau \cdot 4\bar{\gamma}} = \frac{\epsilon \cdot |S|}{4n \cdot \bar{x} \cdot m(S)}.$$

The procedure performs at most one degree query and one uniform neighbor query. All other operations take constant time. Therefore, the query complexity and running time of the procedure are constant.  $\blacktriangleleft$

► **Lemma 9.** *Assume that **Preprocessing** has been invoked successfully, as defined in Definition 6. The procedure **Sample-Heavy**( $\tau, D(S), \bar{x}, \epsilon$ ) returns an edge in  $E_{>\tau}$  such that each edge is returned with probability  $\frac{(1 \pm \epsilon)\epsilon|S|}{4n \cdot \bar{x} \cdot m(S)}$ . The query complexity and running time of the procedure are  $O(1)$ .*

**Proof.** Let  $(u, w)$  be an edge in  $E_{>\tau}$ . We first compute the probability that  $u$  is sampled in Step 2. Recall, the data structure  $D(S)$  supports sampling a vertex  $v$  in  $S$  with probability  $\frac{d(v)}{m(S)}$ . The probability that  $u$  is sampled in Step 2 is equal to the probability that a vertex  $v \in S$  which is a neighbor of  $u$  is sampled in step 1, and  $u$  is the selected neighbor of  $v$  in Step 2. Namely,

$$\Pr[u \text{ is sampled in Step 2}] = \sum_{v \in S \cap \Gamma(u)} \frac{d(v)}{m(S)} \cdot \frac{1}{d(v)} = \sum_{v \in S \cap \Gamma(u)} \frac{1}{m(S)} = \frac{d_S(u)}{m(S)}.$$

By the assumption that **Preprocessing** has been invoked successfully, so that  $S$  is  $\epsilon$ -good, and because  $u \in V_{>\tau}$ ,

$$d_S(u) \in (1 \pm \epsilon) \cdot |S| \cdot \frac{d(u)}{n}.$$

Hence, the probability that  $(u, w)$  is returned by the procedure is

$$\begin{aligned} \Pr[(u, w) \text{ is returned}] &= \Pr[(u \text{ sampled in Step 2}) \text{ and } (w \text{ sampled in Step 5}) \\ &\quad \text{and } ((u, w) \text{ returned in Step 5})] \\ &= \frac{d_S(u)}{m(S)} \cdot \frac{1}{d(u)} \cdot \frac{\epsilon}{4\bar{x}} \in \frac{(1 \pm \epsilon)|S| \cdot \frac{d(u)}{n} \cdot \epsilon}{m(S) \cdot d(u) \cdot 4\bar{x}} = \frac{(1 \pm \epsilon)\epsilon|S|}{4n \cdot \bar{x} \cdot m(S)}. \end{aligned}$$

The procedure performs one degree query and two neighbor queries, and the rest of the operations take constant time. Hence the query complexity and running time are constant.  $\blacktriangleleft$

We are now ready to prove the formal version of Theorem 1.1.

► **Theorem 3.4.** *There exists an algorithm that gets as input query access to a graph  $G$ ,  $n$ , the number of vertices in the graph,  $\epsilon \in (0, \frac{1}{2})$ , an approximation parameter,  $\delta \in (0, 1)$ , a failure parameter, and  $x > 1$ , a trade-off parameter. The algorithm has a preprocessing procedure and a sampling procedure.*

*The preprocessing procedure has expected query complexity  $O\left(\max\left\{\frac{n}{d_{\text{avg}} \cdot x}, \sqrt{\frac{n}{d_{\text{avg}}}}\right\} \cdot \frac{\log^2(n \log(1/\delta)/\delta)}{\epsilon}\right)$ , and it succeeds with probability at least  $1 - \delta$ . If the preprocessing procedure succeeds, then each time the sampling procedure is invoked it returns an edge such that the distribution on returned edges is  $2\epsilon$ -point-wise close to uniform, as defined in Definition 1. Each invocation of the sampling procedure has expected  $O(\bar{x}/\epsilon)$  query and time complexity.*



**Proof.** By 9, the procedure **Preprocessing** procedure succeeds with probability at least  $1 - \delta$ . Furthermore, it has expected running time and query complexity as stated.

Condition on the event that the invocation of **Preprocessing** was successful. Let  $P$  denote the distribution over the returned edges by the procedure **Sample-Uniform-Edge**. By Lemma 2.3 in [13], in order to prove that  $P$  is pointwise  $2\epsilon$ -close to uniform, it suffices to prove that for every two edges  $e, e'$  in the graph,  $\frac{P(e)}{P(e')} \in (1 \pm 2\epsilon)$ . By Lemma 8, every light edge  $e$  is returned with probability  $\frac{\epsilon \cdot |S|}{4n \cdot \bar{x} \cdot m(S)}$ . By Lemma 9, every heavy edge  $e'$  is returned with probability  $\frac{(1 \pm \epsilon)\epsilon |S|}{4n \cdot \bar{x} \cdot m(S)}$ . Therefore, for every two edges  $e, e'$  in the graph,  $\frac{P(e)}{P(e')} \in (1 \pm 2\epsilon)$ .

Next, we prove a lower bound on the success probability of a single invocation of the while loop in Step 1 in **Sample-Uniform-Edge**.

$$\begin{aligned} \Pr[\text{an edge is returned}] &= \frac{1}{2} \Pr[\text{Sample-Light returns an edge}] \\ &\quad + \frac{1}{2} \Pr[\text{Sample-Heavy returns an edge}] \\ &\geq \frac{1}{2} |E_{\leq \tau}| \cdot \frac{\epsilon \cdot |S|}{4n \cdot \bar{x} \cdot m(S)} + \frac{1}{2} \cdot |E_{> \tau}| \cdot \frac{(1 - \epsilon)\epsilon \cdot |S|}{4n \cdot \bar{x} \cdot m(S)} \\ &\geq \frac{1}{2} \cdot \frac{(1 - \epsilon) \cdot \epsilon |S| \cdot m}{4n \cdot \bar{x} \cdot m(S)} = \frac{(1 - \epsilon)\epsilon}{8\gamma \bar{x}} \geq \frac{\epsilon}{192x}, \end{aligned}$$

where the second inequality is due to Claim 7, i.e.  $\gamma \leq 12$ . Hence, the expected number of invocations until an edge is returned is  $O(\bar{x}/\epsilon)$ . ◀

---

## References



- 1 Nesreen K Ahmed, Nick Duffield, Theodore L Willke, and Ryan A Rossi. On sampling from massive graph streams. *Proceedings of the VLDB Endowment*, 10(11), 2017.
- 2 Nesreen K Ahmed, Jennifer Neville, and Ramana Kompella. Network sampling: From static to streaming graphs. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(2):1–56, 2013.
- 3 Maryam Aliakbarpour, Amartya Shankha Biswas, Themis Gouleakis, John Peebles, Ronitt Rubinfeld, and Anak Yodpinyanee. Sublinear-time algorithms for counting star subgraphs via edge sampling. *Algorithmica*, 80(2):668–697, 2018.
- 4 Sepehr Assadi, Michael Kapralov, and Sanjeev Khanna. A simple sublinear-time algorithm for counting arbitrary subgraphs via edge sampling. In *Innovations in Theoretical Computer Science Conference ITCS*, volume 124 of *LIPICs*, pages 6:1–6:20. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019.
- 5 Amartya Shankha Biswas, Talya Eden, and Ronitt Rubinfeld. Towards a decomposition-optimal algorithm for counting and sampling arbitrary motifs in sublinear time. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021*, to appear, 2021.
- 6 Colin Cooper, Tomasz Radzik, and Yiannis Siantos. Estimating network parameters using random walks. *Social Network Analysis and Mining*, 4(1):168, 2014.
- 7 Talya Eden, Dana Ron, and Will Rosenbaum. The arboricity captures the complexity of sampling edges. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9–12, 2019, Patras, Greece.*, pages 52:1–52:14, 2019. doi:10.4230/LIPICs.ICALP.2019.52.
- 8 Talya Eden, Dana Ron, and Will Rosenbaum. Almost optimal bounds for sublinear-time sampling of  $k$ -cliques: Sampling cliques is harder than counting, 2020. arXiv:2012.04090.
- 9 Talya Eden, Dana Ron, and C Seshadhri. Sublinear time estimation of degree distribution moments: The arboricity connection. *SIAM Journal on Discrete Mathematics*, 33(4):2267–2285, 2019.

- 10 Talya Eden, Dana Ron, and C Seshadhri. Faster sublinear approximation of the number of  $k$ -cliques in low-arboricity graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1467–1478. SIAM, 2020.
- 11 Talya Eden, Dana Ron, and C Seshadhri. On approximating the number of  $k$ -cliques in sublinear time. *SIAM Journal on Computing*, 49(4):747–771, 2020.
- 12 Talya Eden and Will Rosenbaum. Lower bounds for approximating graph parameters via communication complexity. In Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20–22, 2018 - Princeton, NJ, USA*, volume 116 of *LIPICs*, pages 11:1–11:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.APPROX-RANDOM.2018.11.
- 13 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.
- 14 Uriel Feige. On sums of independent random variables with unbounded variance and estimating the average degree in a graph. *SIAM Journal on Computing*, 35(4):964–984, 2006.
- 15 Hendrik Fichtenberger, Mingze Gao, and Pan Peng. Sampling arbitrary subgraphs exactly uniformly in sublinear time. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8–11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPICs*, pages 45:1–45:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ICALP.2020.45.
- 16 Minas Gjoka, Maciej Kurant, Carter T. Butts, and Athina Markopoulou. Walking in facebook: A case study of unbiased sampling of osns. In *INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15–19 March 2010, San Diego, CA, USA*, pages 2498–2506. IEEE, 2010. doi:10.1109/INFOCOM.2010.5462078.
- 17 Oded Goldreich and Dana Ron. Approximating average parameters of graphs. *Random Structures & Algorithms*, 32(4):473–493, 2008. doi:10.1002/rsa.20203.
- 18 Mira Gonen, Dana Ron, and Yuval Shavitt. Counting stars and other small subgraphs in sublinear-time. *SIAM Journal on Discrete Mathematics*, 25(3):1365–1411, 2011.
- 19 Hossein Jowhari, Mert Sağlam, and Gábor Tardos. Tight bounds for  $l_p$  samplers, finding duplicates in streams, and related problems. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 49–58, 2011.
- 20 Nadav Kashtan, Shalev Itzkovitz, Ron Milo, and Uri Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–1758, 2004.
- 21 Tali Kaufman, Michael Krivelevich, and Dana Ron. Tight bounds for testing bipartiteness in general graphs. *SIAM Journal on Computing*, 33(6):1441–1483, 2004. doi:10.1137/S0097539703436424.
- 22 Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06*, pages 631–636, New York, NY, USA, 2006. ACM. doi:10.1145/1150402.1150479.
- 23 George Marsaglia, Wai Wan Tsang, Jingbo Wang, et al. Fast generation of discrete random variables. *Journal of Statistical Software*, 11(3):1–11, 2004.
- 24 Abedelaziz Mohaisen, Aaram Yun, and Yongdae Kim. Measuring the mixing time of social graphs. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 383–389, 2010.
- 25 Bruno Ribeiro and Don Towsley. Estimating and sampling graphs with multidimensional random walks. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 390–403, 2010.

- 26 Jakub Tětek and Mikkel Thorup. Sampling and counting edges via vertex accesses. *arXiv preprint arXiv:2107.03821*, 2021.
- 27 Duru Türkoglu and Ata Turk. Edge-based wedge sampling to estimate triangle counts in very large graphs. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 455–464. IEEE, 2017.
- 28 Jakub Tětek. Approximate triangle counting via sampling and fast matrix multiplication. *CoRR*, abs/2104.08501, 2021. [arXiv:2104.08501](https://arxiv.org/abs/2104.08501).
- 29 Alastair J. Walker. New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronics Letters*, 10(8):127–128, 1974.
- 30 Alastair J. Walker. An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software*, 3(3):253–256, 1977.
- 31 Tianyi Wang, Yang Chen, Zengbin Zhang, Tianyin Xu, Long Jin, Pan Hui, Beixing Deng, and Xing Li. Understanding graph sampling algorithms for social network analysis. In *2011 31st international conference on distributed computing systems workshops*, pages 123–128. IEEE, 2011.



# Lower Bounds for XOR of Forrelations

Uma Girish  

Department of Computer Science, Princeton University, NJ, USA

Ran Raz  

Department of Computer Science, Princeton University, NJ, USA

Wei Zhan  

Department of Computer Science, Princeton University, NJ, USA

---

## Abstract

The Forrelation problem, first introduced by Aaronson [1] and Aaronson and Ambainis [2], is a well studied computational problem in the context of separating quantum and classical computational models. Variants of this problem were used to give tight separations between quantum and classical query complexity [2]; the first separation between poly-logarithmic quantum query complexity and bounded-depth circuits of super-polynomial size, a result that also implied an oracle separation of the classes BQP and PH [15]; and improved separations between quantum and classical communication complexity [12]. In all these separations, the lower bound for the classical model only holds when the advantage of the protocol (over a random guess) is more than  $\approx 1/\sqrt{N}$ , that is, the success probability is larger than  $\approx 1/2 + 1/\sqrt{N}$ . This is unavoidable as  $\approx 1/\sqrt{N}$  is the correlation between two coordinates of an input that is sampled from the Forrelation distribution, and hence there are simple classical protocols that achieve advantage  $\approx 1/\sqrt{N}$ , in all these models.

To achieve separations when the classical protocol has smaller advantage, we study in this work the XOR of  $k$  independent copies of (a variant of) the Forrelation function (where  $k \ll N$ ). We prove a very general result that shows that any family of Boolean functions that is closed under restrictions, whose Fourier mass at level  $2k$  is bounded by  $\alpha^k$  (that is, the sum of the absolute values of all Fourier coefficients at level  $2k$  is bounded by  $\alpha^k$ ), cannot compute the XOR of  $k$  independent copies of the Forrelation function with advantage better than  $O\left(\frac{\alpha^k}{N^{k/2}}\right)$ . This is a strengthening of a result of [8], that gave a similar statement for  $k = 1$ , using the technique of [15]. We give several applications of our result. In particular, we obtain the following separations:

**Quantum versus Classical Communication Complexity.** We give the first example of a partial Boolean function that can be computed by a simultaneous-message quantum protocol with communication complexity  $\text{polylog}(N)$  (where Alice and Bob also share  $\text{polylog}(N)$  EPR pairs), and such that, any classical randomized protocol of communication complexity at most  $\tilde{o}(N^{1/4})$ , with any number of rounds, has quasipolynomially small advantage over a random guess. Previously, only separations where the classical protocol has polynomially small advantage were known between these models [10, 12].

**Quantum Query Complexity versus Bounded Depth Circuits.** We give the first example of a partial Boolean function that has a quantum query algorithm with query complexity  $\text{polylog}(N)$ , and such that, any constant-depth circuit of quasipolynomial size has quasipolynomially small advantage over a random guess. Previously, only separations where the constant-depth circuit has polynomially small advantage were known [15].

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Communication complexity; Theory of computation  $\rightarrow$  Pseudorandomness and derandomization; Theory of computation  $\rightarrow$  Oracles and decision trees

**Keywords and phrases** Forrelation, Quasipolynomial, Separation, Quantum versus Classical, Xor

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.52

**Category** RANDOM



© Uma Girish, Ran Raz, and Wei Zhan;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 52; pp. 52:1–52:14



Leibniz International Proceedings in Informatics

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

**Related Version** *Full Version*: <https://arxiv.org/abs/2007.03631>

*Full Version*: <https://eccc.weizmann.ac.il/report/2020/101/>

**Funding** *Uma Girish*: Simons Collaboration on Algorithms and Geometry, by a Simons Investigator Award and by the National Science Foundation grants No. CCF-1714779, CCF-2007462.

*Ran Raz*: Simons Collaboration on Algorithms and Geometry, by a Simons Investigator Award and by the National Science Foundation grants No. CCF-1714779, CCF-2007462.

*Wei Zhan*: Simons Collaboration on Algorithms and Geometry, by a Simons Investigator Award and by the National Science Foundation grants No. CCF-1714779, CCF-2007462.

**Acknowledgements** We would like to thank Avishay Tal for very helpful conversations. We would also like to thank Chin Ho Lee for pointing out a simpler proof of Lemma 28.

## 1 Introduction

Several recent works used Fourier analysis to prove lower bounds for computing (variants of) the Forrelation (partial) function of [1, 2], in various models of computation and communication [15, 8, 12]. These works show that for many computational models, when analyzing the success probability of computing the Forrelation function, it's sufficient to bound the contribution of Fourier coefficients at level 2, ignoring all other Fourier coefficients [15, 8]. This holds for any computational model that is closed under restrictions and is proved by analyzing the Forrelation distribution as a distribution resulting from a certain random walk, rather than analyzing it directly.

While this is a powerful technique, it could only be used to bound computations of the Forrelation function with advantage (over a random guess) larger than  $\approx 1/\sqrt{N}$ , that is, computations with success probability larger than  $\approx 1/2 + 1/\sqrt{N}$ . Roughly speaking, this is because the bound on the Fourier coefficients at level 2 of the Forrelation function is  $\approx O(1/\sqrt{N})$ . We note that while ruling out protocols with advantage larger than  $1/\sqrt{N}$  is satisfactory in many cases, an advantage of  $1/\sqrt{N}$  is often viewed as non-negligible and it is often desirable to rule out protocols with negligible (sub-polynomially small) advantage as well.

In this work, we study the XOR of  $k$  independent copies of the Forrelation function of [15] (where  $k < o(N^{1/50})$ ). We show that for many computational models, when analyzing the success probability of computing the XOR of  $k$  independent copies of the Forrelation function, it's sufficient to bound the contribution of Fourier coefficients at level  $2k$ , ignoring all other Fourier coefficients. Our proof builds on the techniques of [15], and followup works [8, 12], by analyzing a “product” of  $k$  random walks, one for each of the independent copies of the Forrelation function. This can be viewed as a random walk with a  $k$ -dimensional time variable.

Consequently, we obtain a very general lower bound that shows that any family of Boolean functions that is closed under restrictions, whose Fourier mass at level  $2k$  is bounded by  $\alpha^k$  (that is, for every function in the family, the sum of the absolute values of all Fourier coefficients at level  $2k$  is bounded by  $\alpha^k$ ), cannot compute the XOR of  $k$  independent copies of the Forrelation function with advantage better than  $O\left(\frac{\alpha^k}{N^{k/2}}\right)$ , that is, with success probability larger than  $\frac{1}{2} + O\left(\frac{\alpha^k}{N^{k/2}}\right)$ . This is a strengthening of a result of [8], that gave a similar statement for  $k = 1$ , using the technique of [15]. While bounding the advantage of protocols for the XOR of  $k$  independent copies of a problem is often non-trivial, our result gives a very general way to do that for the special case of Forrelation.

We note that the requirement that the family of Boolean functions is closed under restrictions is satisfied by essentially all non-uniform computational models. The requirement of having a good bound on the Fourier mass at level  $2k$  is satisfied by several central and well-studied computational models (see for example [7] for a recent discussion). In particular, we focus in this work on three such models: communication complexity, query complexity (decision trees) and bounded-depth circuits. We note that our result is valid for any  $k < N^c$ , for some constant  $c > 0$ , and hence it can be used to prove lower bounds for circuits/protocols with exponentially small advantage, in all these models. However, for the applications of separating quantum and classical computational models, we take  $k$  to be poly-logarithmic in  $N$ , so that we have quantum protocols of poly-logarithmic cost. We use our main theorem to give several separations between quantum and classical computational models.

## 1.1 Communication Complexity

Quantum versus classical separations in communication complexity have been studied for more than two decades in numerous works. We briefly summarize the history of quantum advantage in communication complexity of partial functions, that is most relevant for us: First, Buhrman, Cleve and Wigderson proved an exponential separation between zero-error simultaneous-message quantum communication complexity (without entanglement) and classical deterministic communication complexity [4]. For the bounded-error model, Raz showed an exponential separation between two-way quantum communication complexity and two-way randomized communication complexity [14]. Gavinsky et al (building on Bar-Yossef et al [3]) gave an exponential separation between one-way quantum communication complexity and one-way randomized communication complexity [11]. Klartag and Regev gave an exponential separation between one-way quantum communication complexity and two-way randomized communication complexity [16]. The state of the art separation, by Gavinsky, gave an exponential separation between simultaneous-message quantum communication complexity (with entanglement) and two-way randomized communication complexity [10]. An alternative proof for Gavinsky's result was recently given by [12], as a followup to [15, 8], and had the additional desired property that in the quantum protocol, the time complexity of all the players is poly-logarithmic.

## Our Result

In all these works, the lower bounds for classical communication complexity only hold when the advantage of the protocol (over a random guess) is more than  $\approx 1/\sqrt{N}$ , that is, the success probability is larger than  $\approx 1/2 + 1/\sqrt{N}$ .

In this work, we give a partial Boolean function that can be computed by a simultaneous-message quantum protocol with communication complexity  $\text{polylog}(N)$  (where Alice and Bob also share  $\text{polylog}(N)$  EPR pairs), and such that, any classical randomized protocol of communication complexity at most  $\tilde{o}(N^{1/4})$ , with any number of rounds, has quasipolynomially small advantage over a random guess. This qualitatively matches the results of [10, 12] and has the additional desired property that the lower bound for the classical communication protocol holds for quasipolynomially small advantage, rather than polynomially small advantage. Moreover, as in [12], the quantum protocol in our upper bound has the additional property of being *efficiently implementable*, in the sense that it can be described by quantum circuits of size  $\text{polylog}(N)$ , with oracle access to the inputs.

To prove this result we use the XOR of  $k$  independent copies of the Forrelation function, lifted to communication complexity using XOR as the gadget [13], as in [12]. The quantum upper bound is simple. For the classical lower bound, we use ideas from [12] to bound the

level- $2k$  Fourier mass. This, along with our main theorem implies the desired separation. Our bounds for the level- $2k$  Fourier mass may be interesting in their own right and are proved in Section 7.

## Related Work

We note that an exponential separation between **two-way** quantum communication complexity and two-way randomized communication complexity, with quasipolynomially small advantage, can be proved by a combination of several previous results, as follows:

Start with an existing separation between quantum and classical query complexity, such as the one of [2]. Use Drucker's XOR lemma for randomized decision tree [9] to get a separation between quantum and classical query complexity, where the classical protocol has quasipolynomially small advantage. Finally, use the recent lifting theorem of [5] to lift the result to communication complexity. To the best of our knowledge, this separation was not previously observed.

It follows from these works that there exists a function computable in the quantum two-way model in communication complexity  $\text{polylog}(N)$ , for which randomized protocols of cost  $\tilde{o}(\sqrt{N})$  have at most quasipolynomially small advantage. While the lower bound is for cost  $\tilde{o}(\sqrt{N})$  protocols, which is quantitatively stronger than our lower bound for cost  $\tilde{o}(N^{1/4})$  protocols, the quantum upper bound in this result seems to require two rounds of communication, while our function is computable in the simultaneous model when Alice and Bob share entanglement.

## 1.2 Bounded Depth Circuits

Separations of quantum query complexity and bounded-depth classical circuit complexity have been studied in the context of oracle separations of the classes BQP and PH. An example of a partial Boolean function (Forrelation) that has a quantum query algorithm with query complexity  $\text{polylog}(N)$ , and such that, any constant-depth circuit of quasipolynomial size has polynomially small advantage over a random guess, was given in [15]. This result implied an oracle separation of the classes BQP and PH.

Here, we give the first example of a partial Boolean function (XOR of  $k$  copies of Forrelation) that has a quantum query algorithm with query complexity  $\text{polylog}(N)$ , and such that, any constant-depth circuit of quasipolynomial size has **quasipolynomially** small advantage over a random guess.

For the proof, we use our main theorem, together with Tal's bounds on the level- $2k$  Fourier mass of bounded-depth circuits [17].

## 1.3 Decision Trees

The query complexity model (also known as black box model or decision-tree complexity) has played a central role in the study of quantum computational complexity. Quantum advantages in query complexity (decision trees) have been demonstrated for partial functions in various settings and numerous works. For example, Aaronson and Ambainis [2] showed that the Forrelation problem can be solved by one quantum query, while its randomized query complexity is  $\Omega(\sqrt{N}/\log N)$ .

For classical randomized query complexity, there is a known XOR lemma, proved by Drucker [9]. In particular, Theorem 1.3 of [9], along with the result of [2] gives a partial function (XOR of  $\text{polylog}(N)$  copies of Forrelation) that can be computed by a quantum query algorithm with  $\text{polylog}(N)$  queries, while every classical randomized algorithm that makes  $\tilde{o}(N^{1/2})$  queries, has quasipolynomially small advantage.



Our main theorem implies a different proof for this result, using Tal’s recent bounds on the level- $2k$  Fourier mass of decision trees [18].

### 1.4 The Main Theorem

Our functions are obtained by taking an XOR of several copies of a variant of the Forrelation problem, as defined in [15].

Let  $N = 2^n$  for sufficiently large  $n \in \mathbb{N}$ . Let  $k \in \mathbb{N}$  be a parameter. We assume that  $k = o(N^{1/50})$ . Let  $\epsilon = \frac{1}{60k^2 \ln N}$  be a parameter.

Let  $H_N$  denote the  $N \times N$  normalized Hadamard matrix whose entries are either  $-\frac{1}{\sqrt{N}}$  or  $\frac{1}{\sqrt{N}}$ . Let

$$f_{\text{orr}}(z) := \frac{1}{N} \langle z_2, H_N z_1 \rangle$$

denote the *Forrelation* of a vector  $z = (z_1, z_2)$ , where  $z_1, z_2 \in \mathbb{R}^N$ . The **Forrelation Decision Problem** is the partial Boolean function  $F : \{-1, 1\}^{2N} \rightarrow \{-1, 1\}$  defined at  $z \in \{-1, 1\}^{2N}$  by

$$F(z) := \begin{cases} -1 & \text{if } f_{\text{orr}}(z) \geq \epsilon/2 \\ 1 & \text{if } f_{\text{orr}}(z) \leq \epsilon/4 \\ \text{undefined} & \text{otherwise} \end{cases}$$

The  $\oplus^k$  **Forrelation Decision Problem**  $F^{(k)} : \{-1, 1\}^{2kN} \rightarrow \{-1, 1\}$  is defined as the XOR of  $k$  independent copies of  $F$ . More precisely, for every  $z_1, \dots, z_k \in \{-1, 1\}^{2N}$ , let

$$F^{(k)}(z_1, \dots, z_k) := \prod_{j=1}^k F(z_j).$$

For our separation results, we take the function  $F^{(k)}$ , where  $k = \lceil \log^2 N \rceil$ . For our communication complexity separation we take the lift of  $F^{(k)}$  with XOR as the gadget. The quantum upper bounds in all these separation results are quite simple. Moreover, all the quantum algorithms in our upper bounds have the additional advantage of being *efficiently implementable*, in the sense that they can be described by quantum circuits of size  $\text{polylog}(N)$ , with oracle access to the inputs.

Our main contribution is the classical lower bound. Towards this, our main theorem provides an upper bound on the maximum correlation of  $F^{(k)}$  with any family of Boolean functions, in terms of the maximum level- $2k$  Fourier mass of a function in the family.

► **Main Theorem (Informal).** *There exist two distributions,  $\sigma_0^{(k)}$  and  $\sigma_1^{(k)}$ , on the NO and YES instances of  $F^{(k)}$ , respectively, with the following property. Let  $\mathcal{H}$  be a family of Boolean functions, each of which maps  $\{-1, 1\}^{2kN}$  into  $[-1, 1]$ . Assume that  $\mathcal{H}$  is closed under restrictions. For  $H \in \mathcal{H}$ , let  $L_{2k}(H) := \sum_{|S|=2k} |\widehat{H}(S)|$ . Let  $\alpha \in \mathbb{R}$  be such that  $\alpha^k := \sup_{H \in \mathcal{H}} (L_{2k}(H), 1)$ . Then, for every  $H \in \mathcal{H}$ ,*

$$\left| \mathbb{E}_{z \sim \sigma_0^{(k)}} [H(z)] - \mathbb{E}_{z \sim \sigma_1^{(k)}} [H(z)] \right| \leq O\left(\frac{\alpha^k}{N^{k/2}}\right)$$

Our main theorem implies that functions in  $\mathcal{H}$  cannot correlate with  $F^{(k)}$  by more than  $\frac{1}{2} + O\left(\frac{\alpha^k}{N^{k/2}}\right)$ . For the applications, we instantiate  $\mathcal{H}$  with the class of functions computed by classical protocols of small cost.

## 1.5 Overview of Proof of the Main Theorem for $k = 2$

Our proof builds on the techniques of [15], and followup works [8, 12], which, in turn, used a key idea from [7]. We will now give an overview of the proof of the Main Theorem for the special case  $k = 2$ , where one can already see most of the key ideas.

We start by recalling the hard distributions for  $k = 1$ , as in [15]. The **distribution  $\mathcal{U}$  on no instances** of  $F$  is the uniform distribution  $U_{2N}$  on  $\{-1, 1\}^{2N}$ . It can be shown that a bit string drawn uniformly at random almost always has low Forrelation. The **distribution  $\mathcal{G}$  on yes instances** of  $F$  is the Gaussian distribution with mean 0 and covariance matrix  $\epsilon \begin{bmatrix} \mathbb{I}_N & H_N \\ H_N & \mathbb{I}_N \end{bmatrix}$ . It can be shown that a vector drawn from this distribution almost always has high Forrelation (at least  $\epsilon/2$ ). Although  $\mathcal{G}$  is not a distribution over  $\{-1, 1\}^{2N}$ , this can be fixed (by probabilistically rounding the values) and we ignore this issue in the proof overview.

Our hard distributions for  $k \geq 2$  are obtained by naturally lifting these distributions. The **distribution  $\mu_0$  on no instances** of  $F^{(2)}$  is  $\frac{1}{2}(\mathcal{U} \times \mathcal{U} + \mathcal{G} \times \mathcal{G})$ . The **distribution  $\mu_1$  on yes instances** is  $\frac{1}{2}(\mathcal{U} \times \mathcal{G} + \mathcal{G} \times \mathcal{U})$ . It can be shown that these distributions indeed have almost all their mass on the YES and NO instances of  $F^{(2)}$ , respectively.

Throughout this proof, we identify functions in  $\mathcal{H}$  with their unique multilinear extensions. Using this identification, it follows that for all  $H \in \mathcal{H}$  and  $z_0 \in \mathbb{R}^{4N}$ , we have  $\mathbb{E}_{z \sim \mathcal{U}}[H(z_0 + (z, 0))] = \mathbb{E}_{z \sim \mathcal{U}}[H(z_0 + (0, z))] = \mathbb{E}_{z \sim \mathcal{U}^2}[H(z_0 + z)] = H(z_0)$ .

### Bounding the Advantage of $H$ in Distinguishing $p \cdot \mu_0$ and $p \cdot \mu_1$ , for Small $p$

As in [15, 8], in order to show that functions in  $\mathcal{H}$  can't distinguish between  $\mu_0$  and  $\mu_1$ , we first show that they can't distinguish between  $p \cdot \mu_0$  and  $p \cdot \mu_1$ , for small  $p$ . We show that for every  $H \in \mathcal{H}$ , and  $p \leq \frac{1}{2N}$ ,

$$\begin{aligned} \left| \mathbb{E}_{z \sim p \cdot \mu_0} [H(z)] - \mathbb{E}_{z \sim p \cdot \mu_1} [H(z)] \right| &\triangleq \frac{1}{2} \left| \mathbb{E}_{\substack{z_1 \sim p \cdot \mathcal{G} \\ z_2 \sim p \cdot \mathcal{G}}} [H(z_1, z_2) - H(z_1, 0) - H(0, z_2) + H(0, 0)] \right| \\ &\leq p^4 \cdot O\left(\frac{L_4(H)}{N}\right) + O(p^6 N^{1.5}) \end{aligned}$$

This claim is analogous to Claim 20 from [8]. For sufficiently small  $p$ , the second term in the R.H.S. of the inequality is negligible, compared to the first term. To prove this inequality, we use the Fourier expansion of  $H$  in the L.H.S. and bound the difference between the moments of  $p \cdot \mu_0$  and  $p \cdot \mu_1$ . We show that  $p \cdot \mu_0$  and  $p \cdot \mu_1$  agree on moments of degree less than 4, so these moments don't contribute to the difference. We then show that the contribution of the moments of degree 4 is  $L_4(H) \cdot O\left(\frac{p^4}{N}\right)$  and the contribution of moments of higher degrees is  $O(p^6 N^{1.5})$ .

**Bounding the Advantage of  $H(z_0 + z)$  in Distinguishing  $p \cdot \mu_0$  and  $p \cdot \mu_1$ , for Small  $p$**

Next, as in [15, 8], we show a similar statement for the function  $H(z_0 + z)$  of  $z$ , where  $z_0$  is not too large. We show that for every  $H \in \mathcal{H}$ , and every  $z_0 \in [-1/2, 1/2]^{2kN}$  and  $p \leq \frac{1}{2N}$ ,

$$\begin{aligned} & \left| \frac{1}{2} \mathbb{E}_{\substack{z_1 \sim p \cdot \mathcal{G} \\ z_2 \sim p \cdot \mathcal{G}}} [H(z_0 + (z_1, z_2)) - H(z_0 + (z_1, 0)) - H(z_0 + (0, z_2)) + H(z_0)] \right| \\ & \leq p^4 \cdot O\left(\frac{L_4(H)}{N}\right) + O(p^6 N^{1.5}) \end{aligned} \tag{1}$$

The proof of this inequality is similar to the proof of Claim 19 of [8], using key ideas from [7], and relies on the multilinearity of functions in  $\mathcal{H}$  and the closure of  $\mathcal{H}$  under restrictions.

**A Random Walk with Two-Dimensional Time Variable**

This is the main place where our proof differs from the one of [15] and followup works [8, 12]. In all these works the Forrelation distribution was ultimately analyzed as the distribution obtained by a certain random walk. Here, we consider a product of two random walks, which can also be viewed as a random walk with two-dimensional time variable.

Let  $T = 16N^4$  and  $p = \frac{1}{\sqrt{T}}$ . Let  $z_1^{(1)}, z_2^{(1)}, \dots, z_1^{(T)}, z_2^{(T)} \sim p \cdot \mathcal{G}$  be independent samples. Let  $t = (t_1, t_2)$  for  $t_1, t_2 \in \{0, \dots, T\}$ . Let  $z^{\leq(t)} := \left(\sum_{i=1}^{t_1} z_1^{(i)}, \sum_{i=1}^{t_2} z_2^{(i)}\right)$ . Note that  $z^{\leq(t)}$  is distributed according to  $(p\sqrt{t_1} \cdot \mathcal{G}) \times (p\sqrt{t_2} \cdot \mathcal{G})$ . In particular,  $z^{\leq(T,T)}$  is distributed according to  $\mathcal{G} \times \mathcal{G}$ . This implies that

$$(*) := \mathbb{E}_{z \sim \mu_0} [H(z)] - \mathbb{E}_{z \sim \mu_1} [H(z)] \triangleq \frac{1}{2} \mathbb{E} \left[ H(z^{\leq(T,T)}) - H(z^{\leq(T,0)}) - H(z^{\leq(0,T)}) + H(0,0) \right]$$

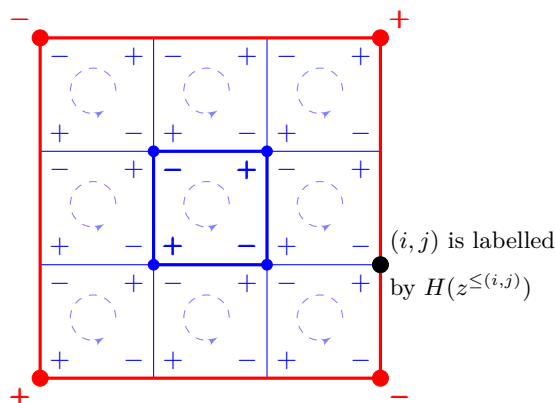
We now rewrite (\*) as follows.

$$(*) = \frac{1}{2} \sum_{\substack{t_1 \in [T] \\ t_2 \in [T]}} \mathbb{E} \left[ H(z^{\leq(t_1, t_2)}) - H(z^{\leq(t_1-1, t_2)}) - H(z^{\leq(t_1, t_2-1)}) + H(z^{\leq(t_1-1, t_2-1)}) \right] \tag{2}$$

The last equation follows by a two-dimensional telescopic cancellation, as depicted in Figure 1. This turns out to be a powerful observation. Note that for every fixed  $t = (t_1, t_2)$ , the random variable  $z^{\leq(t)} - z^{\leq(t-(1,1))} \triangleq (z_1^{(t_1)}, z_2^{(t_2)})$  is distributed according to  $p \cdot \mathcal{G}^2$ , by construction. We can thus apply Inequality(1), setting  $z_0 = z^{\leq(t-(1,1))}$ . This, along with the Triangle-Inequality implies that

$$\begin{aligned} |(*)| & \leq \frac{1}{2} \sum_{\substack{t_1 \in [T] \\ t_2 \in [T]}} \left| \mathbb{E} \left[ H(z^{\leq(t_1, t_2)}) - H(z^{\leq(t_1-1, t_2)}) - H(z^{\leq(t_1, t_2-1)}) + H(z^{\leq(t_1-1, t_2-1)}) \right] \right| \\ & \leq \frac{1}{2} \sum_{\substack{t_1 \in [T] \\ t_2 \in [T]}} \left( p^4 \cdot O\left(\frac{L_4(H)}{N}\right) + O(p^6 N^{1.5}) \right) \quad \text{by Inequality (1)} \\ & = O\left(\frac{L_4(H)}{N}\right) + o\left(\frac{1}{N}\right) \quad \text{since } T = 16N^4 = \frac{1}{p^2} \end{aligned}$$

This completes the proof overview for  $k = 2$ , albeit with many details left out.



■ **Figure 1** Consider the  $(T + 1) \times (T + 1)$  grid whose vertices are indexed by  $v \in (\{0\} \cup [T])^2$ . Each vertex  $v$  is labelled by  $H(z^{\leq (v)})$ . Each rectangle has a sign on its vertices as defined in Figure 1 and the label of a rectangle is the sum of signed labels of its vertices. The sum of labels of all  $1 \times 1$  rectangles equals the label of the larger  $T \times T$  rectangle. This is exactly the content of Equation (2).

## 1.6 Organization of the Paper

In the appendix, we present a formal description of our main results. The proofs can be found in the full version of the paper.

## 1.7 Related Work

Independently of our result, [6] demonstrated PRGs with polylogarithmic dependence on seed length, for a large class of boolean functions. Their result builds on the framework of [7, 8, 15] and constructs improved PRGs by leveraging level- $k$  Fourier bounds.

---

## References

- 1 Scott Aaronson. BQP and the Polynomial Hierarchy. In *STOC 2010*, 2010.
- 2 Scott Aaronson and Andris Ambainis. Forrelation: A Problem That Optimally Separates Quantum from Classical Computing. In *STOC 2015*, 2015.
- 3 Ziv Bar-Yossef, T. S. Jayram, and Iordanis Kerenidis. Exponential Separation of Quantum and Classical One-Way Communication Complexity. In *STOC 2004*, 2004.
- 4 Harry Buhrman, Richard Cleve, and Avi Wigderson. Quantum vs. Classical Communication and Computation. In *STOC 1998*, 1998.
- 5 Arkadev Chattopadhyay, Yuval Filmus, Sajin Koroth, Or Meir, and Toniann Pitassi. Query-To-Communication Lifting for BPP Using Inner Product. In *ICALP 2019*, 2019.
- 6 Eshan Chattopadhyay, Jason Gaitonde, Chin Ho Lee, Shachar Lovett, and Abhishek Shetty. Fractional Pseudorandom Generators from Any Fourier Level. *CoRR*, abs/2008.01316, 2020.
- 7 Eshan Chattopadhyay, Pooya Hatami, Kaave Hosseini, and Shachar Lovett. Pseudorandom Generators from Polarizing Random Walks. In *CCC 2018*, 2018.
- 8 Eshan Chattopadhyay, Pooya Hatami, Shachar Lovett, and Avishay Tal. Pseudorandom Generators from the Second Fourier Level and Applications to AC0 with Parity Gates. In *ITCS 2019*, 2019.
- 9 Andrew Drucker. Improved Direct Product Theorems for Randomized Query Complexity. In *CCC 2011*, 2011.
- 10 Dmitry Gavinsky. Entangled Simultaneity versus Classical Interactivity in Communication Complexity. In *STOC 2016*, 2016.

- 11 Dmitry Gavinsky, Julia Kempe, Iordanis Kerenidis, Ran Raz, and Ronald de Wolf. Exponential Separation for One-Way Quantum Communication Complexity, with Applications to Cryptography. In *STOC 2007*, 2007.
- 12 Uma Girish, Ran Raz, and Avishay Tal. Quantum versus Randomized Communication Complexity, with Efficient Players. In *ITCS 2021*, 2021.
- 13 Ran Raz. Fourier Analysis for Probabilistic Communication Complexity. In *Computational Complexity Journal 1995*, 1995.
- 14 Ran Raz. Exponential Separation of Quantum and Classical Communication Complexity. In *STOC 1999*, 1999.
- 15 Ran Raz and Avishay Tal. Oracle separation of BQP and PH . In *STOC 2019*, 2019.
- 16 Oded Regev and Boaz Klartag. Quantum One-Way Communication can be Exponentially Stronger than Classical Communication. In *STOC 2011*, 2011.
- 17 Avishay Tal. Tight Bounds on the Fourier Spectrum of AC0. In *CCC 2017*, 2017.
- 18 Avishay Tal. Towards Optimal Separations between Quantum and Randomized Query Complexities. In *FOCS 2020*, 2020.

## A Formal Description of the Main Results

### Notation

For  $n \in \mathbb{N}$ , we use  $[n]$  to denote the set  $\{1, 2, \dots, n\}$ . We typically use  $N$  to refer to  $2^n$ . For a set  $S \subseteq [n]$ , let  $\bar{S} := [n] \setminus S$  denote the complement of  $S$ . For sets  $S \subseteq [n], T \subseteq [m]$ , we typically use  $S \times T := \{(s, t) : s \in S, t \in T\}$  denote the set product of  $S$  and  $T$ . Sometimes, we use the notation  $(S, T)$ . Note that the map  $(i, j) \rightarrow m(i-1) + j$  is a bijection between  $[n] \times [m]$  and  $[nm]$ . Using this identification,  $S \times T$  is a subset of  $[nm]$ . We identify subsets  $S \subseteq [n]$  with their  $\{0, 1\}$  indicator vector, that is, the vector  $S \in \{0, 1\}^n$  such that for each  $j \in [n]$ ,  $S_j = 1$  if and only if  $j \in S$ .

Let  $v \in \mathbb{R}^n$ . For  $i \in [n]$ , we refer to the  $i$ -th coordinate of  $v$  by  $v_i$  or  $v(i)$ . For  $x, y \in \mathbb{R}^n$ , let  $x \cdot y \in \mathbb{R}^n$  be the pointwise product between  $x$  and  $y$ . This is the vector whose  $i$ -th coordinate is  $x_i y_i$ , for every  $i \in [n]$ . Let  $\langle x, y \rangle$  denote the real inner product between  $x$  and  $y$ . For  $x, y \in \{0, 1\}^n$ , let  $\langle x, y \rangle_2 := \sum_{i=1}^n x_i y_i \pmod 2$  denote the mod 2 inner product between  $x$  and  $y$ . We use  $\mathbb{I}_n$  to denote the  $n \times n$  identity matrix. We use  $0$  to denote the zero vector in arbitrary dimensions.

### Distributions

For a probability distribution  $D$ , let  $x \sim D$  denote a random variable  $x$  sampled according to  $D$ . For distributions  $D_1$  and  $D_2$ , we use  $D_1 \times D_2$  to denote the product distribution defined by sampling  $(x, y)$  where  $x \sim D_1$  and  $y \sim D_2$  are sampled independently. For  $n \in \mathbb{N}$  and a distribution  $D$ , let  $D^n$  denote the product of  $n$  distributions, each of which is  $D$ . Let  $\mu \in \mathbb{R}^n$  be a vector and  $\Sigma \in \mathbb{R}^{n \times n}$  be a positive semi-definite matrix. We use  $\mathcal{N}(\mu, \Sigma)$  to refer to the  $n$ -dimensional Gaussian distribution with mean  $\mu$  and covariance matrix  $\Sigma$ . Let  $U_n$  denote the uniform distribution on  $\{-1, 1\}^n$ . For a distribution  $D$  over  $\mathbb{R}^n$  and  $a \in \mathbb{R}^n$ , let  $a + D$  refer to the distribution obtained by sampling  $z \sim D$  and returning  $z + a$ . For  $P \in \mathbb{R}^n$  and a distribution  $D$  over  $\mathbb{R}^n$ , let  $P \cdot D$  denote the distribution obtained by sampling  $x \sim D$  and returning  $P \cdot x$ . For  $p \in \mathbb{R}$ , we use  $p \cdot D$  to denote the distribution obtained by sampling  $x \sim D$  and returning  $px$ . For  $I \subseteq [n]$ , let  $\hat{D}(I) := \mathbb{E}_{z \sim D} [\prod_{i \in I} z_i]$  refer to the  $I$ -th moment of  $D$ .

### Fourier Analysis

We refer to  $\{-1, 1\}^n$  as the Boolean hypercube in  $n$  dimensions. Let  $\mathcal{F} := \{f : \{-1, 1\}^n \rightarrow \mathbb{R}\}$  denote the real vector space of all Boolean functions on  $n$  variables. There is an inner product on this space as follows. For  $f, g \in \mathcal{F}$ , let  $\langle f, g \rangle := \mathbb{E}_{x \sim U_n}[f(x)g(x)]$ . For every  $S \subseteq [n]$ , there is a character function  $\chi_S : \{-1, 1\}^n \rightarrow \{-1, 1\}$  defined at  $x \in \{-1, 1\}^n$  by  $\chi_S(x) := \prod_{i \in S} x_i$ . The set of character functions  $\{\chi_S\}_{S \subseteq [n]}$  forms an orthonormal basis for  $\mathcal{F}$ . For  $f \in \mathcal{F}$  and  $S \subseteq [n]$ , let  $\widehat{f}(S) := \langle f, \chi_S \rangle$  denote the  $S$ -th Fourier coefficient of  $f$ . Note that for all  $f \in \mathcal{F}$ , we have  $f = \sum_{S \subseteq [n]} \widehat{f}(S) \chi_S$ . For  $f \in \mathcal{F}$ , the multilinear extension of  $f$  is the unique multilinear polynomial  $\widehat{f} : \mathbb{R}^n \rightarrow \mathbb{R}$  which agrees with  $f$  on  $\{-1, 1\}^n$ . For every  $S \subseteq [n]$ , the multilinear extension of  $\chi_S$  is the monomial  $\prod_{i \in S} x_i$ . This implies that the multilinear extension of  $f \in \mathcal{F}$  is  $\sum_{S \subseteq [n]} \widehat{f}(S) \prod_{i \in S} x_i$ . Henceforth, we identify Boolean functions with their multilinear extensions. With this identification, it can be shown that functions in  $\mathcal{F}$  which map  $\{-1, 1\}^n$  into  $[-1, 1]$  also map  $[-1, 1]^n$  into  $[-1, 1]$ . For  $f, g \in \mathcal{F}$ , let  $f * g \in \mathcal{F}$  be defined at  $z \in \{-1, 1\}^n$  by  $(f * g)(z) := \mathbb{E}_{x \sim U_n}[f(x)g(x \cdot z)]$ . It can be shown that for all  $S \subseteq [n]$ , we have  $\widehat{f * g}(S) = \widehat{f}(S)\widehat{g}(S)$ .

### Level- $k$ Fourier Mass

For  $f \in \mathcal{F}$  and  $k \in \{0, \dots, n\}$ , let  $L_k(f) := \sum_{|S|=k} |\widehat{f}(S)|$  denote the level- $k$  Fourier mass of  $f$ . For a family  $\mathcal{H} \subseteq \mathcal{F}$  of Boolean functions, let  $L_k(\mathcal{H}) := \sup_{H \in \mathcal{H}} L_k(H)$ .

## A.1 The Forrelation Problem

Let  $k, N \in \mathbb{N}$  be parameters, where  $N = 2^n$  for some  $n \in \mathbb{N}$ . We assume that  $k = o(N^{1/50})$ . Fix a parameter  $\epsilon = \frac{1}{60k^2 \ln N}$ . Let  $\mathcal{U}$  refer to  $U_{2N}$ .

### Hadamard Matrix

The Hadamard matrix  $H_N$  of size  $N$  is an  $N \times N$  matrix. The rows and columns are indexed by strings  $a$  and  $b$  respectively where  $a, b \in \{0, 1\}^n$  and the  $(a, b)$ -th entry of  $H_N$  is defined to be  $\frac{1}{\sqrt{N}}(-1)^{\langle a, b \rangle_2}$ . Equivalently,

$$H_N(a, b) := \begin{cases} \frac{-1}{\sqrt{N}} & \text{if } \sum_{i=1}^n a_i b_i \equiv 1 \pmod{2} \\ \frac{+1}{\sqrt{N}} & \text{if } \sum_{i=1}^n a_i b_i \equiv 0 \pmod{2} \end{cases}$$

### The Forrelation Function

The Forrelation Function  $f_{\text{orr}} : \mathbb{R}^{2N} \rightarrow \mathbb{R}$  is defined as follows. Let  $z \in \mathbb{R}^{2N}$  and  $x, y \in \mathbb{R}^N$  be such that  $z = (x, y)$ . Then,

$$f_{\text{orr}}(z) := \frac{1}{N} \langle x, H_N y \rangle$$

### The $\oplus^k$ Forrelation Decision Problem

► **Definition 1** (The  $\oplus^k$  Forrelation Decision Problem). *The Forrelation Decision Problem is the partial Boolean function  $F : \{-1, 1\}^{2N} \rightarrow \{-1, 1\}$  defined as follows. For  $z \in \{-1, 1\}^{2N}$ ,*

let

$$F(z) := \begin{cases} -1 & \text{if } \text{forr}(z) \geq \epsilon/2 \\ 1 & \text{if } \text{forr}(z) \leq \epsilon/4 \\ \text{undefined} & \text{otherwise} \end{cases}$$

The  $\oplus^k$  Forrelation Decision Problem  $F^{(k)} : \{-1, 1\}^{2kN} \rightarrow \{-1, 1\}$  is defined as the XOR of  $k$  independent copies of  $F$ . To be precise, for every  $z_1, \dots, z_k \in \{-1, 1\}^{2N}$ , let

$$F^{(k)}(z_1, \dots, z_k) := \prod_{j=1}^k F(z_j)$$

### The Gaussian Forrelation Distribution $\mathcal{G}$

► **Definition 2.** Let  $\mathcal{G}$  denote the Gaussian distribution over  $\mathbb{R}^{2N}$  defined by the following process.

1. Sample  $x_1, \dots, x_N \sim \mathcal{N}(0, \epsilon)$  independently.
2. Let  $x = (x_1, \dots, x_N)$  and  $y = H_N x$ .
3. Output  $(x, y)$ .

The distribution  $\mathcal{G}$  can be equivalently expressed as  $\mathcal{N}\left(0, \epsilon \begin{bmatrix} \mathbb{I}_N & H_N \\ H_N & \mathbb{I}_N \end{bmatrix}\right)$ .

### A.2 Hard Distributions over $\mathbb{R}^{2kN}$

Let  $\mathcal{P}, \mathcal{Q}$  be two probability distributions on the domain  $\mathbb{D} := \mathbb{R}^{2N}$ . Let  $S \subseteq [k]$ . We define  $\mathcal{P}^S \mathcal{Q}^{\bar{S}}$  to be the distribution on  $\mathbb{D}^k$  defined by sampling  $x = (x_1, \dots, x_k)$  where  $x_1, \dots, x_k \in \mathbb{D}$  are sampled as follows.

$$\text{For each } j \in [k], \text{ independently sample } \begin{cases} x_j \sim \mathcal{P} & \text{if } j \in S \\ x_j \sim \mathcal{Q} & \text{if } j \in \bar{S} \end{cases}$$

► **Definition 3.** Let  $\mathcal{G}$  be the distribution in Definition 2 and  $\mathcal{U} = U_{2N}$ . Define a pair of distributions  $\mu_0^{(k)}, \mu_1^{(k)}$  on  $\mathbb{R}^{2kN}$  as follows.

$$\mu_0^{(k)} := \frac{1}{2^{k-1}} \sum_{\substack{S \subseteq [k] \\ |S| \text{ is even}}} \mathcal{G}^S \mathcal{U}^{\bar{S}} \quad \text{and} \quad \mu_1^{(k)} := \frac{1}{2^{k-1}} \sum_{\substack{S \subseteq [k] \\ |S| \text{ is odd}}} \mathcal{G}^S \mathcal{U}^{\bar{S}}$$

### A.3 Rounding Distributions to the Boolean Hypercube

Let  $\text{trnc} : \mathbb{R} \rightarrow [-1, 1]$  denote the truncation function, whose action on  $a \in \mathbb{R}$  is given by

$$\text{trnc}(a) = \begin{cases} \text{sign}(a) & \text{if } a \notin [-1, 1] \\ a & \text{otherwise} \end{cases}$$

For  $l \in \mathbb{R}$ , we also use  $\text{trnc} : \mathbb{R}^l \rightarrow [-1, 1]^l$  to refer to the function that applies the above truncation function coordinate-wise.

► **Definition 4.** Let  $\mu$  be any distribution on  $\mathbb{R}^M$ . We define the rounded distribution  $\tilde{\mu}$  on  $\{-1, 1\}^M$  as follows.

1. Sample  $z \sim \mu$ .

## 52:12 Lower Bounds for XOR of Forrelations

2. For each coordinate  $i \in [M]$ , independently, let  $z'_i = 1$  with probability  $\frac{1+\text{trnc}(z_i)}{2}$  and  $z'_i = -1$  with probability  $\frac{1-\text{trnc}(z_i)}{2}$ .
  3. Output  $z' = (z'_1, \dots, z'_M)$ .
- Let  $z_0 \in \mathbb{R}^M$  and  $\mu$  be the distribution whose support is  $\{z_0\}$ . We use  $\tilde{z}_0$  to refer to  $\tilde{\mu}$ .

### A.4 The Forrelation Distribution

Let  $k \in \mathbb{N}$ . Let  $\tilde{\mu}_0^{(k)}$  and  $\tilde{\mu}_1^{(k)}$  (respectively  $\tilde{\mathcal{G}}$ ) be distributions over  $\{-1, 1\}^{2kN}$  (respectively  $\{-1, 1\}^{2N}$ ) generated from rounding  $\mu_1^{(k)}$  and  $\mu_0^{(k)}$  (respectively  $\mathcal{G}$ ) according to Definition 4. Observe that we may alternatively define  $\tilde{\mu}_0^{(k)}$  and  $\tilde{\mu}_1^{(k)}$  as follows.

► **Definition 5.** Let  $\mathcal{G}$  be as in Definition 2 and  $\mathcal{U} = U_{2N}$ . Let

$$\tilde{\mu}_0^{(k)} := \frac{1}{2^{k-1}} \sum_{\substack{S \subseteq [k] \\ |S| \text{ is even}}} \tilde{\mathcal{G}}^S \mathcal{U}^{\bar{S}} \quad \text{and} \quad \tilde{\mu}_1^{(k)} := \frac{1}{2^{k-1}} \sum_{\substack{S \subseteq [k] \\ |S| \text{ is odd}}} \tilde{\mathcal{G}}^S \mathcal{U}^{\bar{S}}$$

We refer to  $\tilde{\mu}_1^{(1)} \triangleq \tilde{\mathcal{G}}$  as the Forrelation Distribution.

We show that the distributions  $\tilde{\mu}_1^{(k)}$  and  $\tilde{\mu}_0^{(k)}$  put considerable mass on the YES and NO instances of  $F^{(k)}$ , respectively, where  $F^{(k)}$  is the  $\oplus^k$  Forrelation Decision Problem as in Definition 1.

► **Lemma 6.** Let  $\tilde{\mu}_0^{(k)}$  and  $\tilde{\mu}_1^{(k)}$  be distributions as in Definition 5 and  $F^{(k)}$  be the  $\oplus^k$  Forrelation Decision Problem as in Definition 1. Then,

$$\mathbb{P}_{z \sim \tilde{\mu}_0^{(k)}} [F^{(k)}(z) = 1] \geq 1 - O\left(\frac{k}{N^{6k^2}}\right) \quad \text{and} \quad \mathbb{P}_{z \sim \tilde{\mu}_1^{(k)}} [F^{(k)}(z) = -1] \geq 1 - O\left(\frac{k}{N^{6k^2}}\right)$$

### A.5 Closure under Restrictions

► **Definition 7.** Let  $a \in \{-1, 1, 0\}^M$ . Let  $\rho_a : \mathbb{R}^M \rightarrow \mathbb{R}^M$  be a restriction defined as follows. For  $v \in \mathbb{R}^M$ , let  $\rho_a(v) \in \mathbb{R}^M$  be such that for all  $j \in [M]$ ,

$$(\rho_a(v))(j) := \begin{cases} v(j) & \text{if } a(j) = 0 \\ a(j) & \text{otherwise} \end{cases}$$

For a function  $F : \{-1, 1\}^M \rightarrow \mathbb{R}$ , the restricted function  $F \circ \rho_v : \{-1, 1\}^M \rightarrow \mathbb{R}$  is defined at  $z \in \{-1, 1\}^M$  by  $(F \circ \rho_v)(z) := F(\rho_v(z))$ .

We say that a family  $\mathcal{H}$  of Boolean functions in  $M$  variables is closed under restrictions if for all restrictions  $v \in \{-1, 1, 0\}^M$  and  $H \in \mathcal{H}$ , the restricted function  $H \circ \rho_v$  is in  $\mathcal{H}$ .

## B The Main Result

Let  $N \in \mathbb{N}$  be a parameter describing the input size. We will assume that  $N$  is a sufficiently large power of 2. Let  $k \in \mathbb{N}$ . We assume that  $k = o(N^{1/50})$ . Let  $\epsilon = \frac{1}{60k^2 \ln N}$  be the parameter defining  $\mathcal{G}$  as before.

► **Theorem 8.** Let  $\mathcal{H}$  be a family of Boolean functions on  $2kN$  variables, each of which maps  $\{-1, 1\}^{2kN}$  into  $[-1, 1]$ . Assume that  $\mathcal{H}$  is closed under restrictions. Let  $\tilde{\mu}_0^{(k)}, \tilde{\mu}_1^{(k)}$  be the distributions over  $\{-1, 1\}^{2kN}$  as in Definition 5. Then, for every  $H \in \mathcal{H}$ ,

$$\left| \mathbb{E}_{z \sim \tilde{\mu}_0^{(k)}} [H(z)] - \mathbb{E}_{z \sim \tilde{\mu}_1^{(k)}} [H(z)] \right| \leq O\left(\frac{L_{2k}(\mathcal{H})}{N^{k/2}}\right) + o\left(\frac{1}{N^{k/2}}\right)$$



► **Definition 9.** Let  $\tilde{\mu}_0^{(k)}, \tilde{\mu}_1^{(k)}$  be as in Definition 5. Let  $\sigma_0^{(k)}$  (respectively  $\sigma_1^{(k)}$ ) be obtained by conditioning  $\tilde{\mu}_0^{(k)}$  on being a NO (respectively YES) instance of  $F^{(k)}$ .

► **Corollary 10.** Under the same hypothesis as Theorem 8, for every  $H \in \mathcal{H}$

$$\left| \mathbb{E}_{z \sim \sigma_0^{(k)}} [H(z)] - \mathbb{E}_{z \sim \sigma_1^{(k)}} [H(z)] \right| \leq O\left(\frac{L_{2k}(\mathcal{H})}{N^{k/2}}\right) + o\left(\frac{1}{N^{k/2}}\right)$$

## B.1 Applications to Quantum versus Classical Separations

### Query Complexity Separations

► **Lemma 11.** Let  $D : \{-1, 1\}^{2kN} \rightarrow \{-1, 1\}$  be a deterministic decision tree of depth  $d \geq 1$ . Then,

$$\left| \mathbb{E}_{z \sim \sigma_0^{(k)}} [D(z)] - \mathbb{E}_{z \sim \sigma_1^{(k)}} [D(z)] \right| \leq \left( \frac{O(d \log(kN))}{N^{1/2}} \right)^k$$

► **Theorem 12.**  $F^{(k)}$  can be computed in the bounded-error quantum query model with  $O(k^5 \log^2 N \log k)$  queries. However, every randomized decision tree of depth  $\tilde{O}(\sqrt{N})$  has a worst-case success probability of at most  $\frac{1}{2} + \exp(-\Omega(k))$ .

Setting  $k = \lceil \log^c N \rceil$  for  $c \in \mathbb{N}$  in Theorem 12 gives us an explicit family of partial functions that are computable by quantum query algorithms of cost  $\tilde{O}(\log^{5c+2} N)$ , however every randomized query algorithm of cost  $\tilde{O}(N^{\frac{1}{2}})$  has at most  $\frac{1}{2^{\Omega(\log^c N)}}$  advantage over random guessing.

### Communication Complexity Separations

► **Definition 13** (The  $\oplus^k$  Forrelation Communication Problem  $F^{(k)} \circ \text{XOR}$ ). Alice is given  $x$  and Bob is given  $y$  where  $x, y \in \{-1, 1\}^{2kN}$ . Let  $F^{(k)}$  be as in Definition 1. Their goal is to compute the partial function  $F^{(k)}(x \cdot y)$ .

► **Lemma 14.** Let  $C : \{-1, 1\}^{2kN} \times \{-1, 1\}^{2kN} \rightarrow \{-1, 1\}$  be any deterministic protocol of communication complexity  $c$ . Then,

$$\left| \mathbb{E}_{\substack{x \sim U_{2kN} \\ z \sim \sigma_0^{(k)}}} [C(x, x \cdot z)] - \mathbb{E}_{\substack{x \sim U_{2kN} \\ z \sim \sigma_1^{(k)}}} [C(x, x \cdot z)] \right| \leq O\left(\frac{(c + 8k)^{2k}}{N^{k/2}}\right)$$

► **Theorem 15.**  $F^{(k)} \circ \text{XOR}$  can be solved in the quantum simultaneous with entanglement model with  $O(k^5 \log^3 N \log k)$  bits of communication, when Alice and Bob share  $O(k^5 \log^3 N \log k)$  EPR pairs. However, any randomized protocol of cost  $\tilde{O}(N^{1/4})$  has a worst-case success probability of at most  $\frac{1}{2} + \exp(-\Omega(k))$ .

Setting  $k = \lceil \log^c N \rceil$  for  $c \in \mathbb{N}$  in Theorem 15 gives us an explicit family of partial functions that are computable by quantum simultaneous protocols of cost  $\tilde{O}(\log^{5c+3} N)$  when Alice and Bob share  $\tilde{O}(\log^{5c+3} N)$  EPR pairs, however every interactive randomized protocol of cost  $\tilde{O}(N^{\frac{1}{4}})$  has at most  $\frac{1}{2^{\Omega(\log^c N)}}$  advantage over random guessing.

## Circuit Complexity Separations

► **Lemma 16.** *Let  $C : \{-1, 1\}^{2kN} \rightarrow \{-1, 1\}$  be an AC0 circuit of depth  $d \geq 1$  and size  $s$ . Then,*

$$\left| \mathbb{E}_{z \sim \sigma_0^{(k)}} [C(z)] - \mathbb{E}_{z \sim \sigma_1^{(k)}} [C(z)] \right| \leq \left( \frac{O(\log^{2d-2}(s))}{N^{1/2}} \right)^k$$

► **Theorem 17.** *The distributions  $\sigma_1^{(k)}$  and  $\sigma_0^{(k)}$  can be distinguished by a bounded-error quantum query protocol with  $O(k^5 \log^2 N \log k)$  queries with  $2/3$  advantage. However, every constant depth circuit of size  $o\left(\exp\left(N^{\frac{1}{4(d-1)}}\right)\right)$  can distinguish these distributions with at most  $\exp(-\Omega(k))$  advantage.*

Setting  $k = \lceil \log^c N \rceil$  for  $c \in \mathbb{N}$  in Theorem 17 gives us an explicit family of distributions that are distinguishable by cost  $\tilde{O}(\log^{5c+2} N)$  quantum query algorithms, however every constant depth circuit of quasipolynomial size can distinguish them with at most  $\frac{1}{2^{\Omega(\log^c N)}}$  advantage.

# Fourier Growth of Structured $\mathbb{F}_2$ -Polynomials and Applications

Jarosław Błasiok ✉

Columbia University, New York, NY, USA

Peter Ivanov ✉

Northeastern University, Boston, MA, USA

Yaonan Jin ✉

Columbia University, New York, NY, USA

Chin Ho Lee ✉

Columbia University, New York, NY, USA

Rocco A. Servedio ✉

Columbia University, New York, NY, USA

Emanuele Viola ✉

Northeastern University, Boston, MA, USA

---

## Abstract

We analyze the *Fourier growth*, i.e. the  $L_1$  Fourier weight at level  $k$  (denoted  $L_{1,k}$ ), of various well-studied classes of “structured”  $\mathbb{F}_2$ -polynomials. This study is motivated by applications in pseudorandomness, in particular recent results and conjectures due to [9, 10, 8] which show that upper bounds on Fourier growth (even at level  $k = 2$ ) give unconditional pseudorandom generators.

Our main structural results on Fourier growth are as follows:

- We show that any symmetric degree- $d$   $\mathbb{F}_2$ -polynomial  $p$  has  $L_{1,k}(p) \leq \Pr[p = 1] \cdot O(d)^k$ . This quadratically strengthens an earlier bound that was implicit in [33].
- We show that any read- $\Delta$  degree- $d$   $\mathbb{F}_2$ -polynomial  $p$  has  $L_{1,k}(p) \leq \Pr[p = 1] \cdot (k\Delta d)^{O(k)}$ .
- We establish a composition theorem which gives  $L_{1,k}$  bounds on disjoint compositions of functions that are closed under restrictions and admit  $L_{1,k}$  bounds.

Finally, we apply the above structural results to obtain new unconditional pseudorandom generators and new correlation bounds for various classes of  $\mathbb{F}_2$ -polynomials.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Pseudorandomness and derandomization; Theory of computation  $\rightarrow$  Circuit complexity

**Keywords and phrases** Fourier analysis, Pseudorandomness, Fourier growth

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.53

**Category** RANDOM

**Related Version** *Full Version*: <http://arxiv.org/abs/2107.10797>

**Funding** *Jarosław Błasiok*: Junior Fellowship from the Simons Society of Fellows.

*Peter Ivanov*: NSF awards CCF-1813930 and CCF-2114116.

*Yaonan Jin*: NSF IIS-1838154, NSF CCF-1703925, NSF CCF-1814873 and NSF CCF-1563155.

*Chin Ho Lee*: Croucher Foundation and Simons Collaboration on Algorithms and Geometry.

*Rocco A. Servedio*: NSF grants CCF-1814873, IIS-1838154, CCF-1563155, and Simons Collaboration on Algorithms and Geometry.

*Emanuele Viola*: NSF awards CCF-1813930 and CCF-2114116.

**Acknowledgements** We thank Shivam Nadimpalli for stimulating discussions at the early stage of the project.



© Jarosław Błasiok, Peter Ivanov, Yaonan Jin, Chin Ho Lee, Rocco A. Servedio, and Emanuele Viola; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 53; pp. 53:1–53:20



Leibniz International Proceedings in Informatics

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

## 1 Introduction

### 1.1 Background: $L_1$ Fourier norms and Fourier growth

Over the past several decades, Fourier analysis of Boolean functions has emerged as a fundamental tool of great utility across many different areas within theoretical computer science and mathematics. Areas of application include (but are not limited to) combinatorics, the theory of random graphs and statistical physics, social choice theory, Gaussian geometry and the study of metric spaces, cryptography, learning theory, property testing, and many branches of computational complexity such as hardness of approximation, circuit complexity, and pseudorandomness. The excellent book of O’Donnell [29] provides a broad introduction. In this paper we follow the notation of [29], and for a Boolean-valued function  $f$  on  $n$  Boolean variables and  $S \subseteq [n]$ , we write  $\widehat{f}(S)$  to denote the Fourier coefficient of  $f$  on  $S$ .

Given the wide range of different contexts within which the Fourier analysis of Boolean functions has been pursued, it is not surprising that many different quantitative parameters of Boolean functions have been analyzed in the literature. In this work we are chiefly interested in the  $L_1$  Fourier norm at level  $k$ :

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

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

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

As we explain below, strong motivation for studying the  $L_1$  Fourier norm at level  $k$  (even for specific small values of  $k$  such as  $k = 2$ ) is given by exciting recent results in unconditional pseudorandomness. More generally, the notion of *Fourier growth* is a convenient way of capturing the  $L_1$  Fourier norm at level  $k$  for every  $k$ :

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

The notion of Fourier growth was explicitly introduced by Reingold, Steinke, and Vadhan in [33] for the purpose of constructing pseudorandom generators for space-bounded computation (though we note that the Fourier growth of DNF formulas was already analyzed in [26], motivated by applications in learning theory). In recent years there has been a surge of research interest in understanding the Fourier growth of different types of functions [38, 19, 11, 22, 17, 39, 36, 16]. One strand of motivation for this study has come from the study of quantum computing; in particular, bounds on the Fourier growth of  $\text{AC}^0$  [38] were used in the breakthrough result of Raz and Tal [31] which gave an oracle separation between the classes BQP and PH. More recently, in order to achieve an optimal separation between quantum and randomized query complexity, several researchers [39, 1, 36] have studied the Fourier growth of decision trees, with the recent work of [36] obtaining optimal bounds. Analyzing the Fourier growth of other classes of functions has also led to separations between quantum and classical computation in other settings [16, 17, 18].

Our chief interest in the current paper arises from a different line of work which has established powerful applications of Fourier growth bounds in pseudorandomness. We describe the relevant background, which motivates a new conjecture that we propose on Fourier growth, in the next subsection.

## 1.2 Motivation for this work: Fourier growth, pseudorandomness, $\mathbb{F}_2$ -polynomials, and the CHLT conjecture

### 1.2.1 Pseudorandom generators from Fourier growth bounds

Constructing explicit, unconditional pseudorandom generators (PRGs) for various classes of Boolean functions is an important goal in complexity theory. In the recent work [9], Chattopadhyay, Hatami, Hosseini, and Lovett introduced a novel framework for the design of such PRGs. Their approach provides an explicit pseudorandom generator for any class of functions that is closed under restrictions and has bounded Fourier growth:

► **Theorem 3** (PRGs from Fourier growth: Theorem 23 of [9]). *Let  $\mathcal{F}$  be a family of  $n$ -variable Boolean functions that is closed under restrictions and has Fourier growth  $L_1(a, b)$ . Then there is an explicit pseudorandom generator that  $\epsilon$ -fools  $\mathcal{F}$  with seed length  $O(b^2 \log(n/\epsilon)(\log \log n + \log(a/\epsilon)))$ .*

Building on Theorem 3, in [10] Chattopadhyay, Hatami, Lovett, and Tal showed that in fact it suffices to have a bound just on  $L_{1,2}(\mathcal{F})$  in order to obtain an efficient PRG for  $\mathcal{F}$ :

► **Theorem 4** (PRGs from  $L_1$  Fourier norm bounds at level  $k = 2$ : Theorem 2.1 of [10]). *Let  $\mathcal{F}$  be a family of  $n$ -variable Boolean functions that is closed under restrictions and has  $L_{1,2}(\mathcal{F}) \leq t$ . Then there is an explicit pseudorandom generator that  $\epsilon$ -fools  $\mathcal{F}$  with seed length  $O((t/\epsilon)^{2+o(1)} \cdot \text{polylog}(n))$ .*

Observe that while Theorem 4 requires a weaker structural result than Theorem 3 (a bound only on  $L_{1,2}(\mathcal{F})$  as opposed to  $L_{1,k}(\mathcal{F})$  for all  $k \geq 1$ ), the resulting pseudorandom generator is quantitatively weaker since it has seed length polynomial rather than logarithmic in the error parameter  $1/\epsilon$ . Even more recently, in [8] Chattopadhyay, Gaitonde, Lee, Lovett, and Shetty further developed this framework by interpolating between the two results described above. They showed that a bound on  $L_{1,k}^1$  for any  $k \geq 3$  suffices to give a PRG, with a seed length whose  $\epsilon$ -dependence scales with  $k$ :

► **Theorem 5** (PRGs from  $L_1$  Fourier norm bounds up to level  $k$  for any  $k$ : Theorem 4.3 of [8]). *Let  $\mathcal{F}$  be a family of  $n$ -variable Boolean functions that is closed under restrictions and has  $L_{1,k}(\mathcal{F}) \leq b^k$  for some  $k \geq 3$ . Then there exists a pseudorandom generator that  $\epsilon$ -fools  $\mathcal{F}$  with seed length  $O\left(\frac{b^{2+\frac{4}{k-2}} \cdot k \cdot \text{polylog}(\frac{n}{\epsilon})}{\epsilon^{k-2}}\right)$ .*

### 1.2.2 $\mathbb{F}_2$ -polynomials and the CHLT conjecture

The works [9] and [10] highlighted the challenge of proving  $L_{1,k}$  bounds for the class of bounded-degree  $\mathbb{F}_2$ -polynomials as being of special interest. Let

$\text{Poly}_{n,d} :=$  the class of all  $n$ -variate  $\mathbb{F}_2$ -polynomials of degree  $d$ .

It follows from Theorem 4 that even proving

$$L_{1,2}(\text{Poly}_{n,\text{polylog}(n)}) \leq n^{0.49} \tag{1}$$

---

<sup>1</sup> In fact, they showed that a bound on the weaker quantity  $M_{1,k}(f) := \max_{x \in \{-1,1\}^n} |\sum_{|S|=k} \widehat{f}(S)x^S|$  suffices.

would give nontrivial PRGs for  $\mathbb{F}_2$ -polynomials of polylog( $n$ ) degree, improving on [5, 24, 41]. By the classic connection (due to Razborov [32]) between such polynomials and the class  $\text{AC}^0[\oplus]$  of constant-depth circuits with parity gates, this would also give nontrivial PRGs, of seed length  $n^{1-c}$ , for  $\text{AC}^0[\oplus]$ . This would be a breakthrough improvement on existing results, which are poor either in terms of seed length [15] or in terms of explicitness [12].

The authors of [10] in fact conjectured the following bound, which is much stronger than Equation (1):

► **Conjecture 6** ([10]). *For all  $d \geq 1$ , it holds that  $L_{1,2}(\text{Poly}_{n,d}) = O(d^2)$ .*

### 1.2.3 Extending the CHLT conjecture

Given Conjecture 6, and in light of Theorem 5, it is natural to speculate that an even stronger result than Conjecture 6 might hold. We consider the following natural generalization of the [10] conjecture, extending it from  $L_{1,2}(\text{Poly}_{n,d})$  to  $L_{1,k}(\text{Poly}_{n,d})$ :

► **Conjecture 7**. *For all  $d, k \geq 1$ , it holds that  $L_{1,k}(\text{Poly}_{n,d}) = O(d)^k$ .*

The work [10] proved that  $L_{1,1}(\text{Poly}_{n,d}) \leq 4d$ , and already in [9] it was shown that  $L_{1,k}(\text{Poly}_{n,d}) \leq (2^{3d} \cdot k)^k$ , but to the best of our knowledge no other results towards Conjecture 6 or Conjecture 7 are known.

Given the apparent difficulty of resolving Conjecture 6 and Conjecture 7 in the general forms stated above, it is natural to study  $L_{1,2}$  and  $L_{1,k}$  bounds for specific subclasses of degree- $d$   $\mathbb{F}_2$ -polynomials. This study is the subject of our main structural results, which we describe in the next subsection.

## 1.3 Our results: Fourier bounds for structured $\mathbb{F}_2$ -polynomials

Our main results show that  $L_{1,2}$  and  $L_{1,k}$  bounds of the flavor of Conjecture 6 and Conjecture 7 indeed hold for several well-studied classes of  $\mathbb{F}_2$ -polynomials, specifically *symmetric*  $\mathbb{F}_2$ -polynomials and *read- $\Delta$*   $\mathbb{F}_2$ -polynomials. We additionally prove a composition theorem that allows us to combine such polynomials (or, more generally, any polynomials that satisfy certain  $L_{1,k}$  bounds) in a natural way and obtain  $L_{1,k}$  bounds on the resulting combined polynomials.

Before describing our results in detail, we pause to briefly explain why (beyond the fact that they are natural mathematical objects) such “highly structured” polynomials are attractive targets of study given known results. It has been known for more than ten years [2, Lemma 2] that for any degree  $d < (1 - \epsilon)n$ , a *random*  $\mathbb{F}_2$ -polynomial of degree  $d$  (constructed by independently including each monomial of degree at most  $d$  with probability  $1/2$ ) is extremely unlikely to have bias larger than  $\exp(-n/d)$ . It follows that as long as  $d$  is not too large, a random degree- $d$  polynomial  $p$  is overwhelmingly likely to have  $L_{1,k}(p) = o_n(1)$ , which is much smaller than  $d^k$ . (To verify this, consider the polynomials  $p_S$  obtained by XORing  $p$  with the parity function  $\sum_{i \in S} x_i$ . Note that the bias of  $p_S$  is the Fourier coefficient of  $(-1)^p$  on  $S$ . Now apply [2, Lemma 2] to each polynomial  $p_S$ , and sum the terms.)

Since the conjectures hold true for random polynomials, it is natural to investigate highly structured polynomials.

### 1.3.1 Symmetric $\mathbb{F}_2$ -polynomials

A *symmetric*  $\mathbb{F}_2$ -polynomial over  $x_1, \dots, x_n$  is one whose output depends only on the Hamming weight of its input  $x$ . Such a polynomial of degree  $d$  can be written in the form

$$p(x) := \sum_{k=0}^d c_k \sum_{|S|=k, S \subseteq [n]} \prod_{i \in S} x_i,$$

where  $c_0, \dots, c_d \in \{0, 1\}$ . While symmetric polynomials may seem like simple objects, their study can sometimes lead to unexpected discoveries; for example, a symmetric, low-degree  $\mathbb{F}_2$ -polynomial provided a counterexample to the “Inverse conjecture for the Gowers norm” [25, 20].

We prove the following upper bound on the  $L_1$  Fourier norm at level  $k$  for any symmetric polynomial:

► **Theorem 8.** *Let  $p(x_1, \dots, x_n)$  be a symmetric  $\mathbb{F}_2$ -polynomial of degree  $d$ . Then  $L_{1,k}(p) \leq \Pr[p = 1] \cdot O(d)^k$  for every  $k$ .*

We note that if  $d = n$  and  $p$  is the AND function, then an easy computation shows that  $L_{1,k}(p) = \Pr[p = 1] \cdot \binom{d}{k}$ . Moreover, in Appendix A we show that this implies that the upper bounds conjectured in Conjecture 7 are best possible for any constant  $k$ . Theorem 8 verifies the [10] conjecture (Conjecture 6), and even the generalized version Conjecture 7, for the class of symmetric polynomials.

Theorem 8 provides a quadratic sharpening of an earlier bound that was implicit in [33] (as well as providing the “correct” dependence on  $\Pr[p = 1]$ ). In [33] Reingold, Steinke and Vadhan showed that any function  $f$  computed by an oblivious, read-once, regular branching program of width  $w$  has  $L_{1,k}(f) \leq (2w^2)^k$ . It follows directly from a result of [3] (Lemma 15 below) that any symmetric  $\mathbb{F}_2$ -polynomial  $p$  of degree  $d$  can be computed by an oblivious, read-once, regular branching program of width at most  $2d$ , and hence the [33] result implies that  $L_{1,k}(p) \leq 8^k d^{2k}$ .

### 1.3.2 Read- $\Delta$ $\mathbb{F}_2$ -polynomials

For  $\Delta \geq 1$ , a *read- $\Delta$*   $\mathbb{F}_2$ -polynomial is one in which each input variable appears in at most  $\Delta$  monomials. The case  $\Delta = 1$  corresponds to the class of *read-once polynomials*, which are simply sums of monomials over disjoint sets of variables; for example, the polynomial  $x_1x_2 + x_3x_4$  is read-once whereas  $x_1x_2 + x_1x_4$  is read-twice. Read-once polynomials have been studied from the perspective of pseudorandomness [23, 27, 22, 14] as they capture several difficulties in improving Nisan’s generators [28] for width-4 read-once branching programs.

We show that the  $L_{1,k}$  Fourier norm of read- $\Delta$  polynomials is polynomial in  $d$  and  $\Delta$ :

► **Theorem 9.** *Let  $p(x_1, \dots, x_n)$  be a read- $\Delta$  polynomial of degree  $d$ . Then  $L_{1,k}(p) \leq \Pr[p = 1] \cdot O(k)^k \cdot (d\Delta)^{8k}$ .*

[22] showed that read-once polynomials satisfy an  $L_{1,k}$  bound of  $O(d)^k$  for every  $k$ , but we are not aware of previous bounds on even the  $L_1$  Fourier norm at level  $k = 2$  for read- $\Delta$  polynomials, even for  $\Delta = 2$ .

As any monomial with degree  $\Omega(\log n)$  vanishes under a random restriction with high probability, we have the following corollary which applies to polynomials of any degree.

► **Corollary 10.** *Let  $p(x_1, \dots, x_n)$  be a read- $\Delta$  polynomial. Then  $L_{1,k}(p) \leq O(k)^{9k} \cdot (\Delta \log n)^{8k}$ .*

### 1.3.3 A composition theorem

The upper bounds of Theorem 8 and Theorem 9 both include a factor of  $\Pr[p = 1]$ . (We observe that negating  $p$ , i.e. adding 1 to it, does not change its  $L_{1,2}$  or  $L_{1,k}$  and keeps  $p$  symmetric (respectively, read- $\Delta$ ) if it was originally symmetric (respectively, read- $\Delta$ ), and hence in the context of those theorems we can assume that this  $\Pr[p = 1]$  factor is at most  $1/2$ .) Level- $k$  bounds that include this factor have appeared in earlier works for other classes of functions [30, 4, 11, 39, 18], and have been used to obtain high-level bounds for other classes of functions [11, 39, 18] and to extend level- $k$  bounds to more general classes of functions [22]. Having these  $\Pr[p = 1]$  factors in Theorem 8 and Theorem 9 is important for us in the context of our composition theorem, which we now describe. We begin by defining the notion of a *disjoint composition* of functions:

► **Definition 11.** Let  $\mathcal{F}$  be a class of functions from  $\{-1, 1\}^m$  to  $\{-1, 1\}$  and let  $\mathcal{G}$  be a class of functions from  $\{-1, 1\}^\ell$  to  $\{-1, 1\}$ . Define the class  $\mathcal{H} = \mathcal{F} \circ \mathcal{G}$  of disjoint compositions of  $\mathcal{F}$  and  $\mathcal{G}$  to be the class of all functions from  $\{-1, 1\}^{m\ell}$  to  $\{-1, 1\}$  of the form

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

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

As an example of this definition, the class of *block-symmetric* polynomials (i.e. polynomials whose variables are divided into blocks and are symmetric within each block but not overall) are a special case of disjoint compositions where  $\mathcal{G}$  is taken to be the class of symmetric polynomials. We remark that block-symmetric polynomials are known to correlate better with parities than symmetric polynomials in certain settings [21].

We prove a composition theorem for upper-bounding the  $L_1$  Fourier norm at level  $k$  of the disjoint composition of any classes of functions that are closed under restriction and admit a  $L_{1,k}$  bound of the form  $\Pr[f = 1] \cdot a \cdot b^k$ :

► **Theorem 12.** Let  $g_1, \dots, g_m \in \mathcal{G}$  and let  $f \in \mathcal{F}$ , where  $\mathcal{F}$  is closed under restrictions. Suppose that for every  $1 \leq k \leq K$ , we have

1.  $L_{1,k}(f) \leq \Pr[f = 1] \cdot a_{\text{out}} \cdot b_{\text{out}}^k$  for every  $f \in \mathcal{F}$ , and
2.  $L_{1,k}(g) \leq \Pr[g = 1] \cdot a_{\text{in}} \cdot b_{\text{in}}^k$  for every  $g \in \mathcal{G}$ .

Then for every  $\pm 1$ -valued function  $h \in \mathcal{H} = \mathcal{F} \circ \mathcal{G}$ , we have that

$$L_{1,K}(h) \leq \Pr[h = 1] \cdot a_{\text{out}} \cdot (a_{\text{in}} b_{\text{in}} b_{\text{out}})^K.$$

See the full version of this paper for a slightly sharper bound. We remark that Theorem 12 does not assume any  $\mathbb{F}_2$ -polynomial structure for the functions in  $\mathcal{F}$  or  $\mathcal{G}$  and thus may be of broader utility.

## 1.4 Applications of our results

Our structural results imply new pseudorandom generators and correlation bounds.

### 1.4.1 Pseudorandom generators

Combining our Fourier bounds with the polarizing framework, we obtain new PRGs for read-few polynomials. The following theorem follows from applying Theorem 5 with some  $k = \Theta(\log n)$  and the  $L_{1,k}$  bound in Corollary 10.



► **Theorem 13.** *There is an explicit pseudorandom generator that  $\epsilon$ -fools read- $\Delta$   $\mathbb{F}_2$ -polynomials with seed length  $\text{poly}(\Delta, \log n, \log(1/\epsilon))$ .*

For constant  $\epsilon$ , this improves on a PRG by Servedio and Tan [34], which has a seed length of  $2^{O(\sqrt{\log(\Delta n)})}$ . (Note that read- $\Delta$  polynomials are also  $(\Delta n)$ -sparse.) We are not aware of any previous PRG for read-2 polynomials with  $\text{polylog}(n)$  seed length.

Note that the OR function has  $L_1$  Fourier norm  $O(1)$ . By expressing a DNF in the Fourier expansion of OR in its terms, it is not hard to see that the same PRG also fools the class of read- $\Delta$  DNFs (and read- $\Delta$  CNFs similarly) [35].

### 1.4.2 Correlation bounds

Exhibiting explicit Boolean functions that do not *correlate* with low-degree polynomials is a fundamental challenge in complexity. Perhaps surprisingly, this challenge stands in the way of progress on a striking variety of frontiers in complexity, including circuits, rigidity, and multiparty communication complexity. For a survey of correlation bounds and discussions of these connections we refer the reader to [40, 42, 44].

For polynomials of degree larger than  $\log_2 n$ , the state-of-the-art remains the lower bound proved by Razborov and Smolensky in the 1980s' [32, 37], showing that for any degree- $d$  polynomial  $p$  and an explicit function  $h$  (in fact, majority) we have:

$$\Pr[p(x) = h(x)] \leq 1/2 + O(d/\sqrt{n}).$$

Viola [43] recently showed that upper bounds on  $L_{1,k}(\mathcal{F})$  imply correlation bounds between  $\mathcal{F}$  and an explicit function  $h_k$  that is related to majority and is defined as

$$h_k(x) := \text{sgn}\left(\sum_{|S|=k} x^S\right).$$

In particular, proving Conjecture 6 or related conjectures implies new correlation bounds beating Razborov–Smolensky. The formal statement of the connection is given by the following theorem.

► **Theorem 14** (Theorem 1 in [43]). *For every  $k \in [n]$  and  $\mathcal{F} \subseteq \{f: \{0, 1\}^n \rightarrow \{-1, 1\}\}$ , there is a distribution  $D_k$  on  $\{0, 1\}^n$  such that for any  $f \in \mathcal{F}$ ,*

$$\Pr_{x \sim D_k} [f(x) = h_k(x)] \leq \frac{1}{2} + \frac{e^k}{2\sqrt{\binom{n}{k}}} L_{1,k}(\mathcal{F}).$$

For example, if  $k = 2$  and we assume that the answer to Conjecture 6 is positive, then the right-hand side above becomes  $1/2 + O(d^2/n)$ , which is a quadratic improvement over the bound by Razborov and Smolensky.

Therefore, Theorems 8 and 9 imply correlation bounds between these polynomials and an explicit function that are better than  $O(d/\sqrt{n})$  given in [32, 37]. We note that via a connection in [41], existing PRGs for these polynomials already imply strong correlation bounds between these polynomials and the class of NP. Our results apply to more general classes via the composition theorem, where it is not clear if previous techniques applied. For a concrete example, consider the composition of a degree- $(n^\alpha)$  symmetric polynomial with degree- $(n^\alpha)$  read- $(n^\alpha)$  polynomials. Theorem 12 shows that such polynomial has  $L_{1,2} \leq n^{O(\alpha)}$ . For a sufficiently small  $\alpha = \Omega(1)$ , we again obtain correlation bounds improving on Razborov–Smolensky.

## 1.5 Related work

We close this introduction by discussing a recent work of Girish, Tal and Wu [18] on parity decision trees that is related to our results.

Parity decision trees are a generalization of decision trees in which each node queries a parity of some input bits rather than a single input bit. The class of depth- $d$  parity decision trees is a subclass of  $\mathbb{F}_2$  degree- $d$  polynomials, as such a parity decision tree can be expressed as a sum of products of sums over  $\mathbb{F}_2$ , where each product corresponds to a path in the tree (and hence gives rise to  $\mathbb{F}_2$ -monomials of degree at most  $d$ ). The Fourier spectrum of parity decision trees was first studied in [4], which obtained a level-1 bound of  $O(\sqrt{d})$ . This bound was recently extended to higher levels in [18], showing that any depth- $d$  parity decision tree  $T$  over  $n$  variables has  $L_{1,k}(T) \leq d^{k/2} \cdot O(k \log n)^k$ .

## 2 Our techniques

We now briefly explain the approaches used to prove our results. We note that each of these results is obtained using very different ingredients, and hence the results can be read independently of each other.

### 2.1 Symmetric polynomials (Theorem 8, Section 4)

The starting point of our proof is a result from [3], which says that degree- $d$  symmetric  $\mathbb{F}_2$ -polynomials only depend on the Hamming weight of their input modulo  $m$  for some  $m$  (a power of two) which is  $\Theta(d)$ . Given this, since  $p(x)$  takes the same value for all strings  $x$  with the same weights  $\ell \bmod m$ , to analyze  $L_{1,k}(p)$  it suffices to analyze  $\mathbf{E}[(-1)^{x_1+\dots+x_k}]$  conditioned on  $x$  having Hamming weight exactly  $\ell \bmod m$ .

We bound this conditional expectation by considering separately two cases depending on whether or not  $k \leq n/m^2$ . For the case that  $k \leq n/m^2$ , we use a (slight sharpening of a) result from [6], which gives a bound of  $m^{-k}e^{-\Omega(n/m^2)}$ . In the other case, that  $k \geq n/m^2$ , in Lemma 17 we prove a bound of  $O(km/n)^k$ . This is established via a careful argument that gives a new bound on the Kravchuk polynomial in certain ranges (see the full version of the paper for more details), extending and sharpening similar bounds that were recently established in [13] (the bounds of [13] would not suffice for our purposes).

In each of the above two cases, summing over all the  $\binom{n}{k}$  coefficients gives the desired bound of  $O(m)^k = O(d)^k$ .

### 2.2 Read- $\Delta$ polynomials (Theorem 9, Section 5)

Writing  $f := (-1)^p$  for an  $\mathbb{F}_2$ -polynomial  $p$ , we observe that the coefficient  $\hat{f}(S)$  is simply the bias of  $p_S(x) := p(x) + \sum_{i \in S} x_i$ . Our high-level approach is to decompose the read-few polynomial  $p_S$  into many disjoint components, then show that each component has small bias. Since the components are disjoint, the product of these biases gives an upper bound on the bias of  $p_S$ .

In more detail, we first partition the variables according to the minimum degree  $t_i$  of the monomials containing each variable  $x_i$ . Then we start decomposing  $p_S$  by collecting all the monomials in  $p$  containing  $x_i$  to form the polynomial  $p_i$ . We observe that the larger  $t_i$  is, the more likely  $p_i$  is to vanish on a random input, and therefore the closer  $p_i + x_i$  is to being unbiased. For most  $S$ , we can pick many such  $p_i$ 's ( $i \in S$ ) from  $p$  so that they are disjoint. For the remaining polynomial  $r$ , because  $\Delta$  and  $d$  are small, we can further decompose  $r$  into

many disjoint polynomials  $r_i$ . Finally, our upper bound on  $|\widehat{f}(S)|$  will be the magnitude of the product of the biases of the  $p_i$ 's and  $r_i$ 's. We note that our decomposition of  $p$  uses the structure of  $S$ ; and so the upper bound on  $\widehat{f}(S)$  depends on  $S$  (see Lemma 18). Summing over each  $|\widehat{f}(S)|$  gives our upper bound.

### 2.3 Composition theorem (Theorem 12, Section 6)

As a warmup, let us first consider directly computing a degree-1 Fourier coefficient  $\widehat{h}(\{(i, j)\})$  of the composition. Since the inner functions  $g_i$  depend on disjoint variables, by writing the outer function  $f$  in its Fourier expansion, it is not hard to see that

$$\widehat{h}(\{(i, j)\}) = \sum_{S \ni i} \widehat{f}(S) \prod_{\ell \in S \setminus \{i\}} \mathbf{E}[g_\ell] \cdot \widehat{g}_i(\{j\}).$$

When the  $g_i$ 's are balanced, i.e.  $\mathbf{E}[g_i] = 0$ , we have  $\widehat{h}(\{(i, j)\}) = \widehat{f}(\{i\})\widehat{g}_i(\{j\})$ , and it follows that  $L_{1,1}(h) \leq L_{1,1}(\mathcal{F})L_{1,1}(\mathcal{G})$ . To handle the unbalanced case, we apply an idea from [9] that lets us relate  $\sum_{S \ni i} \widehat{f}(S) \prod_{\ell \in S \setminus \{i\}} \mathbf{E}[g_\ell]$  to the average of  $\widehat{f}_R(\{i\})$ , for some suitably chosen random restriction  $R$  on  $f$  (see Claim 20). As  $\mathcal{F}$  is closed under restrictions, we can apply the  $L_{1,1}(\mathcal{F})$  bound on  $f_R$ , which in turns gives a bound on  $\sum_{S \ni i} \widehat{f}(S) \prod_{\ell \in S \setminus \{i\}} \mathbf{E}[g_\ell]$  in terms of  $L_{1,1}(\mathcal{F})$  and  $\mathbf{E}[g_i]$ .

Bounding  $L_{1,k}(h)$  for  $k \geq 2$  is more complicated, as each  $\widehat{h}(S)$  involves  $\widehat{f}(J)$  and  $\widehat{g}_i(T)$ 's, where the sets  $J$  and  $T$  have different sizes. We provide more details in Section 6.

## 3 Preliminaries

**Notation.** For a string  $x \in \{0, 1\}^n$  we write  $|x|$  to denote its Hamming weight  $\sum_{i=1}^n x_i$ . We use  $\mathcal{X}_w$  to denote  $\{x : |x| = w\}$ , the set of  $n$ -bit strings with Hamming weight  $w$ , and  $\mathcal{X}_{\ell \bmod m} = \bigcup_{w:w \equiv \ell \bmod m} \mathcal{X}_w = \{x : |x| \equiv \ell \bmod m\}$ .

We recall that for an  $n$ -variable Boolean function  $f$ , the *level- $k$  Fourier  $L_1$  norm* of  $f$  is

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

We note that a function  $f$  and its negation have the same  $L_{1,k}$  for  $k \geq 1$ . Hence we can often assume that  $\Pr[f = 1] \leq 1/2$ , or replace the occurrence of  $\Pr[f = 1]$  in a bound by  $\min\{\Pr[f = 1], \Pr[f = 0]\}$  for a  $\{0, 1\}$ -valued function  $f$  (or by  $\min\{\Pr[f = 1], \Pr[f = -1]\}$  for a  $\{-1, 1\}$ -valued function). If  $f$  is a  $\{-1, 1\}$ -valued function then  $\frac{1 - \mathbf{E}[f]}{2}$  is equal to  $\min\{\Pr[f = 1], \Pr[f = -1]\}$ , and we will often write  $\frac{1 - \mathbf{E}[f]}{2}$  for convenience.

Unless otherwise indicated, we will use the letters  $p, q, r$ , etc. to denote  $\mathbb{F}_2$ -polynomials (with inputs in  $\{0, 1\}^n$  and outputs in  $\{0, 1\}$ ) and the letters  $f, g, h$ , etc. to denote general Boolean functions (where the inputs may be  $\{0, 1\}^n$  or  $\{-1, 1\}^n$  and the outputs may be  $\{0, 1\}$  or  $\{-1, 1\}$  depending on convenience). We note that changing from  $\{0, 1\}$  outputs to  $\{-1, 1\}$  outputs only changes  $L_{1,k}$  by a factor of 2.

We use standard multilinear monomial notation as follows: given a vector  $\beta = (\beta_1, \dots, \beta_n)$  and a subset  $T \subseteq [n]$ , we write  $\beta^T$  to denote  $\prod_{j \in T} \beta_j$ .

#### 4 $L_{1,k}$ bounds for symmetric polynomials

In this section we prove Theorem 8, which gives an upper bound on  $L_{1,k}(p)$  for any symmetric  $\mathbb{F}_2$ -polynomial  $p$  of degree  $d$ , covering the entire range of parameters  $1 \leq k, d \leq n$ .

##### 4.1 Proof idea

As the polynomial  $p$  is symmetric, its Fourier coefficient  $\widehat{p}(S)$  only depends on  $|S|$ , the size of  $S$ . Hence to bound  $L_{1,k}$  it suffices to analyze the coefficient  $\widehat{p}(\{1, \dots, k\}) = \mathbf{E}_{x \sim \{0,1\}^n} [p(x)(-1)^{x_1 + \dots + x_k}]$ .

Our proof uses a result from [3] (Lemma 15 below), which says that degree- $d$  symmetric  $\mathbb{F}_2$ -polynomials only depend on the Hamming weight of their input modulo  $m$  for some  $m = O(d)$ . Given this, since  $p(x)$  takes the same value for strings  $x$  with the same weights  $\ell \bmod m$ , we can in turn bound each  $\mathbf{E}[(-1)^{x_1 + \dots + x_k}]$  conditioned on  $x$  having Hamming weight exactly  $\ell \bmod m$ , i.e.  $x \in \mathcal{X}_{\ell \bmod m}$ . We consider two cases depending on whether or not  $k \leq n/m^2$ . If  $k \leq n/m^2$ , we can apply a (slight sharpening of a) result from [6], which gives a bound of  $m^{-k} e^{-\Omega(n/m^2)}$ . If  $k \geq n/m^2$ , in Lemma 17 we prove a bound of  $O(km/n)^k$ . In each case, summing over all the  $\binom{n}{k}$  coefficients gives the desired bound of  $O(m)^k = O(d)^k$ .

We now give some intuition for Lemma 17, which upper bounds the magnitude of the ratio

$$\mathbf{E}_{x \sim \mathcal{X}_{\ell \bmod m}} [(-1)^{x_1 + \dots + x_k}] = \frac{\sum_{x \in \mathcal{X}_{\ell \bmod m}} (-1)^{x_1 + \dots + x_k}}{|\mathcal{X}_{\ell \bmod m}|} \quad (2)$$

by  $O(km/n)^k$ . Let us first consider  $k = 1$  and  $m = \Theta(\sqrt{n})$ . As most strings  $x$  have Hamming weight within  $[n/2 - \Theta(\sqrt{n}), n/2 + \Theta(\sqrt{n})]$ , it is natural to think about the weight  $|x|$  in the form of  $n/2 + m\mathbb{Z} + \ell'$ . It is easy to see that the denominator is at least  $\Omega(2^n/\sqrt{n})$ , so we focus on bounding the numerator. Consider the quantity  $\sum_{x \in \mathcal{X}_{n/2+s}} \mathbf{E}[(-1)^{x_1}]$  for some  $s$ . As we are summing over all strings of the same Hamming weight, we can instead consider  $\sum_{x \in \mathcal{X}_{n/2+s}} \mathbf{E}_{i \sim [n]} [(-1)^{x_i}]$ . For any string of weight  $n/2 + s$ , it is easy to see that

$$\mathbf{E}_{i \sim [n]} [(-1)^{x_i}] = (1/2 - s/n) - (1/2 + s/n) = -2s/n. \quad (3)$$

Therefore, in the  $k = 1$  case we get that

$$\left| \mathbf{E}_{x \sim \mathcal{X}_{\ell \bmod m}} [(-1)^{x_1 + \dots + x_k}] \right| \leq 2 \sum_c \binom{n}{n/2 + cm + \ell'} \frac{|cm + \ell'|}{n}.$$

Using the fact that  $\binom{n}{n/2 + cm + \ell'}$  is exponentially decreasing in  $|c|$ , in the full version of the paper we show that this is at most  $O(2^n/n)$ . So the ratio in (2) is at most  $O(1/\sqrt{n})$ , as desired, when  $k = 1$ .

However, already for  $k = 2$ , a direct (but tedious) calculation shows that

$$\mathbf{E}_{i < j} [(-1)^{x_i + x_j}] = \frac{4s^2 - 2ns + n}{n(n-1)}, \quad (4)$$

which no longer decreases in  $s$  like in (3). Nevertheless, we observe that this is bounded by  $O(1/n + (|s|/n)^2)$ , which is sufficient for bounding the ratio by  $O(1/n)$ . Building on this, for any  $k$  we obtain a bound of  $2^{O(k)}((k/n)^{k/2} + (|s|/n))^k$  in the full version of the paper, and by a more careful calculation we are able to obtain the desired bound of  $O(km/n)^k$  on Equation (2).

### 4.2 Proof of Theorem 8

We now prove the theorem. We will use the following result from [3], which says that degree- $d$  symmetric  $\mathbb{F}_2$ -polynomials only depend on their input's Hamming weight modulo  $O(d)$ .

► **Lemma 15** (Theorem 2.4 in [3],  $p = 2$ ). *Let  $p: \{0, 1\}^n \rightarrow \{0, 1\}$  be a symmetric  $\mathbb{F}_2$ -polynomial of degree  $d$ , where  $m/2 \leq d < m$  and  $m$  is a power of two. Then  $p(x)$  only depends on  $|x| \bmod m$ .*

We will also use two bounds on the biases of parities under the uniform distribution over  $\mathcal{X}_{\ell \bmod m}$ , one holds for  $k \leq n/(2d)^2 \leq n/m^2$  (Claim 16) and the other for  $k \geq n/(2d)^2 \geq n/(4m^2)$  (Lemma 17). Claim 16 is essentially taken from [6]. However, the statement in [6] has a slightly worse bound; so in the full version of the paper we explain the changes required to give the bound of Claim 16. The proof of Lemma 17 involves bounding the magnitude of Kravchuk polynomials. As it is somewhat technical we defer its proof to the full version of the paper.

▷ **Claim 16** (Lemma 10 in [6]). For every  $1 \leq k \leq n/m^2$  and every integer  $\ell$ ,

$$2^{-n} \left| \sum_{x \in \mathcal{X}_{\ell \bmod m}} (-1)^{x_1 + \dots + x_k} \right| \leq m^{-(k+1)} e^{-\Omega(n/m^2)},$$

while for  $k = 0$ ,

$$\left| 2^{-n} |\mathcal{X}_{\ell \bmod m}| - 1/m \right| \leq m^{-1} e^{-\Omega(n/m^2)}.$$

► **Lemma 17.** *For  $k \geq n/(4m^2)$ , we have*

$$\binom{n}{k} \cdot \max_{\ell} \left| \frac{\sum_{x \in \mathcal{X}_{\ell \bmod m}} (-1)^{x_1 + \dots + x_k}}{|\mathcal{X}_{\ell \bmod m}|} \right| \leq O(m)^k.$$

We now use Claim 16 and Lemma 17 to prove Theorem 8.

**Proof of Theorem 8.** As  $p$  is symmetric, all the level- $k$  coefficients are the same, so it suffices to give a bound on  $\widehat{p}(\{1, 2, \dots, k\})$ . Let  $\tilde{p}: \{0, \dots, n\} \rightarrow \{0, 1\}$  be the function defined by  $\tilde{p}(|x|) := p(x_1, \dots, x_n)$ . By Lemma 15, we have  $\tilde{p}(\ell) = \tilde{p}(\ell \bmod m)$  for some  $d < m \leq 2d$  where  $m$  is a power of 2. Using the definition of  $\widehat{p}(\{1, \dots, k\})$ , we have

$$\begin{aligned} |\widehat{p}(\{1, \dots, k\})| &= \left| \mathbf{E}_{x \sim \{0,1\}^n} [p(x) (-1)^{x_1 + \dots + x_k}] \right| \\ &= \left| \sum_{\ell=0}^{m-1} \tilde{p}(\ell) \frac{|\mathcal{X}_{\ell \bmod m}|}{2^n} \cdot \frac{\sum_{x \in \mathcal{X}_{\ell \bmod m}} (-1)^{x_1 + \dots + x_k}}{|\mathcal{X}_{\ell \bmod m}|} \right| \\ &\leq \mathbf{E}[p] \cdot \max_{0 \leq \ell \leq m-1} \left| \frac{\sum_{x \in \mathcal{X}_{\ell \bmod m}} (-1)^{x_1 + \dots + x_k}}{|\mathcal{X}_{\ell \bmod m}|} \right|, \end{aligned}$$

where we use the shorthand  $\mathbf{E}[p] = \mathbf{E}_{x \sim \{0,1\}^n} [p(x)]$  in the last step.

When  $k \leq n/(2d)^2 \leq n/m^2$ , by Claim 16 (using the first bound for the numerator and the second  $k = 0$  bound for the denominator) we have

$$\max_{0 \leq \ell \leq m-1} \left| \frac{\sum_{x \in \mathcal{X}_{\ell \bmod m}} (-1)^{x_1 + \dots + x_k}}{|\mathcal{X}_{\ell \bmod m}|} \right| \leq \frac{m^{-(k+1)} e^{-\Omega(n/m^2)}}{m^{-1} (1 - e^{-\Omega(n/m^2)})} \leq O(1) \cdot m^{-k} e^{-\Omega(n/m^2)},$$

## 53:12 Fourier Growth of Structured $\mathbb{F}_2$ -Polynomials and Applications

where the last inequality holds because  $1 \leq k \leq n/m^2$  and hence the  $(1 - e^{-\Omega(n/m^2)})$  factor in the denominator of the left-hand side is  $\Omega(1)$ . Hence, summing over all the  $\binom{n}{k}$  level- $k$  coefficients, we get that

$$L_{1,k}(p) \leq \mathbf{E}[p] \cdot \binom{n}{k} \cdot O(1) \cdot m^{-k} e^{-\Omega(n/m^2)} \leq \mathbf{E}[p] \cdot O(1) \cdot m^k \left(\frac{ne}{km^2}\right)^k e^{-\Omega(n/m^2)} \leq \mathbf{E}[p] \cdot O(m)^k,$$

where the last inequality is because for constant  $c$ , the function  $(x/k)^k e^{-cx}$  is maximized when  $x = k/c$ , and is  $O(1)^k$ .

When  $k \geq n/(2d)^2 \geq n/(4m^2)$ , by Lemma 17 we have

$$L_{1,k}(p) \leq \mathbf{E}[p] \cdot \binom{n}{k} \max_{0 \leq \ell \leq m-1} \left| \frac{\sum_{x \in \mathcal{X}_{\ell \bmod m}} (-1)^{x_1 + \dots + x_k}}{|\mathcal{X}_{\ell \bmod m}|} \right| \leq \mathbf{E}[p] \cdot O(m)^k. \quad \blacktriangleleft$$

### 5 $L_{1,k}$ bounds for read- $\Delta$ polynomials

In this section we prove our  $L_{1,k}$  bounds for read-few polynomials, proving Theorem 9.

#### 5.1 Proof idea

We first observe that for  $f = (-1)^p$ , the Fourier coefficient  $\widehat{f}(S)$  is simply the bias of the  $\mathbb{F}_2$ -polynomial  $p_S(x) := p(x) + \sum_{i \in S} x_i$ . Assuming that  $p_S$  depends on all  $n$  variables, by a simple greedy argument we can collect  $n/\text{poly}(\Delta, d)$  polynomials in  $p_S$  so that each of them depends on disjoint variables, and it is not hard to show that the product of the biases of these polynomials upper bounds the bias of  $p_S$ . From this is easy to see that any read- $\Delta$  degree- $d$  polynomial has bias  $\exp(2^{-d}n/\text{poly}(\Delta, d))$ . However, this quantity is too large to sum over  $\binom{n}{k}$  coefficients.

Our next idea (Lemma 18) is to give a more refined decomposition of the polynomial  $p$  by inspecting the variables  $x_i : i \in S$  more closely. Suppose the variables  $x_i : i \in S$  are far apart in their dependency graph (see the definition of  $G_p$  below), as must indeed be the case for most of the  $\binom{n}{k}$  size- $k$  sets  $S$ . Then we can collect all the monomials containing each  $x_i$  to form a polynomial  $p_i$ , and these  $p_i$ 's will depend on disjoint variables. Moreover, if every monomial in  $p_i$  has high degree (see the definition of  $V_t(p)$  below), then  $p_i = 0$  with high probability and therefore  $p_i + x_i$  is almost unbiased. Therefore, we can first collect these  $p_i$  and  $x_i$  from  $p_S$ ; then, for the remaining  $m \geq |S| \cdot \text{poly}(\Delta, d)$  monomials in  $p_S$ , as before we collect  $m/\text{poly}(\Delta, d)$  polynomials  $r_i$  so that they depend on disjoint variables, but this time we collect these monomials using the variables in  $V_t(p)$ , and give an upper bound in terms of the size  $|V_t(p)|$ . Multiplying the biases of the  $p_i + x_i$ 's and the bias of  $r$  gives our refined upper bound on  $\widehat{f}(S)$  in Lemma 18.

#### 5.2 Proof of Theorem 9

We now proceed to the actual proof. We first define some notions that will be used throughout our arguments. For a read- $\Delta$  degree- $d$  polynomial  $p$ , we define  $V_t(p) : t \in [d]$  and  $G_p$  as follows.

For every  $t \in [d]$ , define

$$V_t(p) := \{i \in [n] : \text{the minimum degree of the monomials in } p \text{ containing } x_i \text{ is } t\}.$$

Note that the sets  $V_1(p), \dots, V_d(p)$  form a partition of the input variables  $p$  depends on.

Define the undirected graph  $G_p$  on  $[n]$ , where  $i, j \in [n]$  are adjacent if  $x_i$  and  $x_j$  both appear in the same monomial in  $q$ . Note that  $G_p$  has degree at most  $\Delta d$ . For  $S \subseteq [n]$ , we use  $N_{=d}(S)$  to denote the indices that are at distance exactly  $d$  to  $S$  in  $G_p$ , and use  $N_{\leq d}(S)$  to denote  $\bigcup_{j=0}^d N_{=j}(S)$ .

We first state our key lemma, which gives a refined bound on each  $\widehat{f}(S)$  stronger than the naive bound sketched in the first paragraph of the ‘‘Proof Idea’’ above, and use it to prove Theorem 9. Due to lack of space, we defer its proof to the full version of the paper.

► **Lemma 18** (Main lemma for read- $\Delta$  polynomials). *Let  $p(x_1, \dots, x_n)$  be a read- $\Delta$  degree- $d$  polynomial. Let  $S \subseteq [n], |S| \geq \ell$  be a subset containing some  $\ell$  indices  $i_1, \dots, i_\ell \in S$  whose pairwise distances in  $G_p$  are at least 4, and let  $t_1, \dots, t_\ell \in [d]$  be such that each  $i_j \in V_{t_j}(p)$ . Let  $f = (-1)^p$ . Then*

$$|\widehat{f}(S)| \leq O(1)^{|S|} \cdot \Delta^\ell \prod_{j \in [\ell]} \left( 2^{-t_j} \exp \left( -\frac{2^{-t_j} |V_{t_j}(p)|}{\ell \cdot (\Delta d)^4} \right) \right).$$

**Proof of Theorem 9.** Using a reduction given in the proof of [7, Lemma 2.2], it suffices to prove the same bound without the acceptance probability factor, i.e. to prove that for every  $1 \leq k \leq n$ ,

$$L_{1,k}(p) \leq O(k)^k \cdot (\Delta d)^{8k}.$$

As [7] did not provide an explicit statement of the reduction, for completeness we provide a self-contained statement and proof in Lemma 22 in Appendix A.

For every subset  $S \subseteq [n]$  of size  $k$ , there exists an  $\ell \leq k$  and  $i_1, \dots, i_\ell \in S$  such that their pairwise distances in  $G_p$  are at least 4, each  $i_j \in V_{t_j}(p)$  for some  $t_j \in [d]$ , and each of the remaining  $k - \ell$  indices in  $S$  is within distance at most 3 to some  $i_j$ .

Fix any  $i_1, \dots, i_\ell$ , and let us bound the number of subsets  $S \subseteq [n]$  of size  $k$  that can contain  $i_1, \dots, i_\ell$ . Because  $|N_{\leq 3}(j)| \leq (\Delta d)^3 + (\Delta d)^2 + \Delta d + 1 \leq 4(\Delta d)^3$  for every  $j \in [n]$ , the remaining  $k - \ell$  indices of  $S$  can appear in at most

$$\begin{aligned} \sum_{j_1 + \dots + j_\ell = k - \ell} \prod_{b \in [\ell]} \binom{4(\Delta d)^3}{j_b} &= \binom{4\ell(\Delta d)^3}{k - \ell} \\ &\leq (4(\Delta d)^3)^k \cdot e^{k - \ell} \left( \frac{\ell}{k - \ell} \right)^{k - \ell} \\ &\leq (e\Delta d)^{3k} \end{aligned}$$

different ways, where the equality uses the Vandermonde identity, the first inequality uses  $\binom{n}{k} \leq (en/k)^k$ , and the last one uses  $\left(\frac{\ell}{k - \ell}\right)^{k - \ell} \leq \left(1 + \frac{\ell}{k - \ell}\right)^{k - \ell} \leq e^\ell$  and  $4e < e^3$ . Therefore, by Lemma 18,

$$\begin{aligned} \sum_{S: |S|=k} |\widehat{f}(S)| &\leq \sum_{\ell=1}^k \sum_{t \subseteq [d]^\ell} \left[ \left( \prod_{j \in [\ell]} |V_{t_j}(p)| \right) \cdot (e\Delta d)^{3k} \cdot O(1)^k \Delta^\ell \prod_{j' \in [\ell]} \left( 2^{-t_{j'}} \exp \left( -\frac{2^{-t_{j'}} |V_{t_{j'}}(p)|}{\ell(\Delta d)^4} \right) \right) \right] \\ &\leq O(1)^k \cdot (\Delta d)^{3k} \sum_{\ell=1}^k \Delta^\ell \sum_{t \subseteq [d]^\ell} \prod_{j \in [\ell]} \left( 2^{-t_j} |V_{t_j}(p)| \exp \left( -\frac{2^{-t_j} |V_{t_j}(p)|}{\ell(\Delta d)^4} \right) \right) \\ &\leq O(1)^k \cdot (\Delta d)^{3k} \sum_{\ell=1}^k \Delta^\ell \cdot d^\ell \cdot (\ell(\Delta d)^4)^\ell \\ &\leq O(k)^k \cdot (\Delta d)^{3k} \cdot (\Delta d)^{5k} \\ &= O(k)^k \cdot (\Delta d)^{8k}, \end{aligned}$$

where the third inequality is because the function  $x \mapsto xe^{-x/c}$  is maximized when  $x = c$ . This completes the proof. ◀

## 6 $L_{1,k}$ bounds for disjoint compositions

In this section we give  $L_{1,k}$  bounds on *disjoint compositions* of functions, proving Theorem 12.

### 6.1 Proof idea

Before proving Theorem 12, we briefly describe the main ideas of the proof. For a subset  $J \subseteq [m]$ , let  $\partial_J f$  denote the  $J$ -th derivative of  $f$ , which can be expressed as

$$\partial_J f(x_1, \dots, x_m) := \sum_{T \supseteq J} \widehat{f}(T) x^{T \setminus J}.$$

Note that  $\widehat{f}(J) = \partial_J f(\vec{0})$ .

Let us begin by considering the task of bounding  $L_{1,1}(h) = \sum_{(i,j) \in [m] \times [\ell]} |\widehat{h}\{(i,j)\}|$ . Let  $\beta = (\beta_1, \dots, \beta_m)$ , where  $\beta_i := \mathbf{E}[g_i]$ . Using the Fourier expansion of  $f$ , we have

$$\widehat{h}\{(i,j)\} = \sum_{S \subseteq [m]} \widehat{f}(S) \mathbf{E} \left[ \prod_{k \in S} g_k(x_k) \cdot x_{i,j} \right].$$

If  $S \not\ni i$ , then the expectation is zero, because  $\prod_{k \in S} g_k(x_k)$  and  $x_{i,j}$  are independent and  $\mathbf{E}[x_{i,j}] = 0$ . So, we have

$$\widehat{h}\{(i,j)\} = \sum_{S \ni i} \widehat{f}(S) \beta^{S \setminus \{i\}} \cdot \widehat{g}_i(\{j\}) = \partial_i f(\beta) \cdot \widehat{g}_i(\{j\}).$$

If the functions  $g_i$  are balanced, i.e.  $\mathbf{E}[g_i] = 0$  for all  $i$ , then we would have  $\beta = \vec{0}$ , and

$$\widehat{h}\{(i,j)\} = \partial_i f(\vec{0}) \cdot \widehat{g}_i(\{j\}) = \widehat{f}(\{i\}) \widehat{g}_i(\{j\}).$$

So in this case we have

$$L_{1,1}(h) = \sum_{i \in [m], j \in [\ell]} |\widehat{h}\{(i,j)\}| = \sum_{i \in [m]} \sum_{j \in [\ell]} |\widehat{f}(\{i\}) \widehat{g}_i(\{j\})| = \sum_{i \in [m]} |\widehat{f}(\{i\})| \sum_{j \in [\ell]} |\widehat{g}_i(\{j\})|$$

and we can apply our bounds on  $L_{1,1}(\mathcal{F})$  and  $L_{1,1}(\mathcal{G})$  to  $\sum_{i \in [m]} \widehat{f}(\{i\})$  and  $\sum_{j \in [\ell]} \widehat{g}_i(\{j\})$  respectively. Specializing to the case  $g_1 = \dots = g_m$ , we have

▷ **Claim 19.** Suppose  $g_1 = g_2 = \dots = g_m =: g$  and  $\mathbf{E}[g] = 0$ . Then  $L_{1,1}(h) = L_{1,1}(f)L_{1,1}(g)$ .

In general the  $g_i$ 's may not all be the same and may not be balanced, and so it seems unclear how we can apply our  $L_{1,1}(\mathcal{F})$  bound on  $\sum_{i \in [m]} \partial_i f(\beta_1, \dots, \beta_m)$  when  $\beta \neq \vec{0}$ . To deal with this, in Claim 20 below we apply a clever idea introduced in [9] that lets us relate  $f(\beta)$  at a nonzero point  $\beta$  to the average of  $f_{R_\beta}(\vec{0})$ , where  $f_{R_\beta}$  is  $f$  with some of its inputs fixed by a random restriction  $R_\beta$ . As  $\mathcal{F}$  is closed under restrictions, we have that  $f_{R_\beta} \in \mathcal{F}$  and we can apply the  $L_{1,1}(\mathcal{F})$  bound on  $\sum_i \partial_i f_{R_\beta}(\vec{0})$ , which in turn gives a bound on  $\sum_{i \in [m]} \partial_i f(\beta_1, \dots, \beta_m)$ .

Bounding  $L_{1,K}(h)$  for  $K \geq 2$  is more complicated, as now each  $\widehat{h}(S)$  involves many  $\widehat{f}(J)$  and  $\widehat{g}_i(T)$ 's, where the sets  $J$  and  $T$  have different sizes. So one has to group the coefficients carefully.



### 6.2 Useful notation

For a set  $S \subseteq [m] \times [\ell]$ , let  $S|_f := \{i \in [m] : (i, j) \in S \text{ for some } j \in [\ell]\}$  be the “set of first coordinates” that occur in  $S$ , and let  $S|i := \{j \in [\ell] : (i, j) \in S\}$ . Note that if  $(i, j) \in S$ , then  $i \in S|_f$  and  $j \in S|i$ . Let  $\beta$  denote the vector  $(\beta_1, \dots, \beta_m)$ , where  $\beta_i := \mathbf{E}[g_i]$  for each  $i \in [m]$ . For a set  $J = \{i_1, \dots, i_{|J|}\} \subseteq [m]$  and  $f = f(y_1, \dots, y_m)$ , we write  $\partial_J f$  to denote  $\frac{\partial^{|J|} f}{\partial y_{i_1} \dots \partial y_{i_{|J|}}}$ . Since  $\partial_J y^T = \mathbb{1}(T \supseteq J) y^{T \setminus J}$ , by the multilinearity of  $f$  we have that

$$\partial_J f(\beta) = \sum_{T \supseteq J} \widehat{f}(T) \beta^{T \setminus J}. \tag{5}$$

### 6.3 The random restriction $R_\beta$

Given  $\beta \in [-1, 1]^m$ , let  $R_\beta$  be the random restriction which is the randomized function from  $\{-1, 1\}^m$  to  $\{-1, 1\}^m$  whose  $i$ -th coordinate is (independently) defined by

$$R_\beta(y)_i := \begin{cases} \text{sgn}(\beta_i) & \text{with probability } |\beta_i| \\ y_i & \text{with probability } 1 - |\beta_i|. \end{cases}$$

Note that we have

$$\mathbf{E}_{R_\beta, y} [R_\beta(y)_i] = \mathbf{E}_{R_\beta} [R_\beta(\vec{0})_i] = \beta_i.$$

Define  $f_{R_\beta}(y)$  to be the (randomized) function  $f(R_\beta(y))$ . By the multilinearity of  $f$  and independence of the  $R_\beta(y)_i$  we have

$$\mathbf{E}_{R_\beta, y} [f_{R_\beta}(y)] = \mathbf{E}_{R_\beta} [f_{R_\beta}(\vec{0})] = f(\beta).$$

The following claim relates the two derivatives  $\partial_S f(\beta)$  and  $\partial_S f_{R_\beta}(\vec{0}) = \widehat{f_{R_\beta}}(S)$ .

▷ Claim 20.

$$\partial_S f(\beta) = \prod_{i \in S} \frac{1}{1 - |\beta_i|} \cdot \mathbf{E}_{R_\beta} [\partial_S f_{R_\beta}(\vec{0})] = \prod_{i \in S} \frac{1}{1 - |\beta_i|} \cdot \mathbf{E}_{R_\beta} [\widehat{f_{R_\beta}}(S)].$$

Proof. Due to lack of space, we defer the proof to the full version of the paper. ◁

We can use Claim 20 to express each coefficient of  $h$  in terms of the coefficients of  $f$  and  $g_i$ .

► **Lemma 21.** For  $S \subseteq [m] \times [\ell]$ , we have  $\widehat{h}(S) = \prod_{i \in S|_f} \widehat{g}_i(S|i) \cdot \prod_{i \in S|_f} \frac{1}{1 - |\beta_i|} \cdot \mathbf{E}_{R_\beta} [\widehat{f_{R_\beta}}(S|_f)]$ .

Proof. Due to lack of space, we defer the proof to the full version of the paper. ◀

### 6.4 Proof of Theorem 12

By Lemma 21,  $L_{1,K}(h)$  is equal to

$$\sum_{S \subseteq [m] \times [\ell] : |S|=K} |\widehat{h}(S)| = \sum_{S \subseteq [m] \times [\ell] : |S|=K} \left| \prod_{i \in S|_f} \widehat{g}_i(S|i) \cdot \prod_{i \in S|_f} \frac{1}{1 - |\beta_i|} \cdot \mathbf{E}_{R_\beta} [\widehat{f_{R_\beta}}(S|_f)] \right|.$$

We enumerate all the subsets  $S \subseteq [m] \times [\ell]$  of size  $K$  in the following order: For every  $|J| = k \in [K]$  out of the  $m$  blocks of  $\ell$  coordinates, we enumerate all possible combinations

### 53:16 Fourier Growth of Structured $\mathbb{F}_2$ -Polynomials and Applications

of the (disjoint) nonempty subsets  $\{S_i : i \in J\}$  in those  $k$  blocks whose sizes sum to  $K$ . Rewriting the summation above in this order, we obtain

$$\begin{aligned} \sum_{S \subseteq [m] \times [\ell] : |S|=K} |\widehat{h}(S)| &= \sum_{k=1}^K \sum_{\substack{J \subseteq [m] \\ |J|=k}} \sum_{\substack{w \subseteq [\ell]^J \\ |J|=k}} \sum_{\substack{\{S_i\}_{i \in J} \subseteq [\ell]^J \\ \forall i \in J : |S_i|=w_i}} \left| \prod_{i \in J} \widehat{g}_i(S_i) \prod_{i \in J} \frac{1}{1-|\beta_i|} \mathbf{E}_{R_\beta} [\widehat{f_{R_\beta}}(J)] \right| \\ &\leq \sum_{k=1}^K \sum_{\substack{J \subseteq [m] \\ |J|=k}} \sum_{\substack{w \subseteq [\ell]^J \\ |J|=k}} \sum_{\substack{\{S_i\}_{i \in J} \subseteq [\ell]^J \\ \forall i \in J : |S_i|=w_i}} \prod_{i \in J} |\widehat{g}_i(S_i)| \prod_{i \in J} \frac{1}{1-|\beta_i|} \left| \mathbf{E}_{R_\beta} [\widehat{f_{R_\beta}}(J)] \right|. \end{aligned} \quad (6)$$

Since  $L_{1,w_i}(g_i) \leq \frac{1-|\beta_i|}{2} \cdot a_{\text{in}} \cdot b_{\text{in}}^{w_i}$ , for every  $\{w_i\}_{i \in J}$  such that  $\sum_{i \in J} w_i = K$ , we have

$$\sum_{\substack{\{S_i\}_{i \in J} \subseteq [\ell]^J : i \in J \\ \forall i \in J : |S_i|=w_i}} \prod_{i \in J} |\widehat{g}_i(S_i)| = \prod_{i \in J} L_{1,w_i}(g_i) \leq \prod_{i \in J} \left( \frac{1-|\beta_i|}{2} a_{\text{in}} b_{\text{in}}^{w_i} \right) = b_{\text{in}}^K a_{\text{in}}^{|J|} \prod_{i \in J} \frac{1-|\beta_i|}{2}.$$

Plugging the above into (6), we get that

$$\begin{aligned} \sum_{S \subseteq [m] \times [\ell] : |S|=K} |\widehat{h}(S)| &\leq b_{\text{in}}^K \sum_{k=1}^K a_{\text{in}}^k \sum_{\substack{J \subseteq [m] \\ |J|=k}} \sum_{\substack{w \subseteq [\ell]^J \\ \sum_{i \in J} w_i = K}} \prod_{i \in J} \left( \frac{1-|\beta_i|}{2} \cdot \frac{1}{1-|\beta_i|} \cdot \left| \mathbf{E}_{R_\beta} [\widehat{f_{R_\beta}}(J)] \right| \right) \\ &= b_{\text{in}}^K \sum_{k=1}^K \left( \frac{a_{\text{in}}}{2} \right)^k \sum_{\substack{J \subseteq [m] \\ |J|=k}} \left| \mathbf{E}_{R_\beta} [\widehat{f_{R_\beta}}(J)] \right| \sum_{\substack{w \subseteq [\ell]^J \\ \sum_{i \in J} w_i = K}} 1 \\ &\leq b_{\text{in}}^K \sum_{k=1}^K \left( \frac{a_{\text{in}}}{2} \right)^k \binom{K-1}{k-1} \sum_{\substack{J \subseteq [m] \\ |J|=k}} \left| \mathbf{E}_{R_\beta} [\widehat{f_{R_\beta}}(J)] \right|, \end{aligned} \quad (7)$$

where the last inequality is because for every subset  $J \subseteq [m]$ , the set  $\{w \subseteq [\ell]^J : \sum_{i \in J} w_i = K\}$  has size at most  $\binom{K-1}{|J|-1}$ . We now bound  $|\mathbf{E}_{R_\beta} [\widehat{f_{R_\beta}}(J)]|$ . Since for every restriction  $R_\beta$ , we have  $f_{R_\beta} \in \mathcal{F}$  (by the assumption that  $\mathcal{F}$  is closed under restrictions), it follows that

$$L_{1,k}(f_{R_\beta}) \leq \frac{1 - |\mathbf{E}_y[f_{R_\beta}(y)]|}{2} a_{\text{out}} b_{\text{out}}^k \leq \frac{1 - \mathbf{E}_y[f_{R_\beta}(y)]}{2} a_{\text{out}} b_{\text{out}}^k.$$

So

$$\begin{aligned} \sum_{J \subseteq [m], |J|=k} \left| \mathbf{E}_{R_\beta} [\widehat{f_{R_\beta}}(J)] \right| &\leq \mathbf{E}_{R_\beta} [L_{1,k}(f_{R_\beta})] \\ &\leq \frac{1 - \mathbf{E}_{R_\beta, y} [f_{R_\beta}(y)]}{2} a_{\text{out}} b_{\text{out}}^k \\ &= \frac{1 - \mathbf{E}[h]}{2} a_{\text{out}} b_{\text{out}}^k. \end{aligned}$$

Continuing from (7), we get

$$\begin{aligned} \sum_{S \subseteq [m] \times [\ell]: |S|=K} |\widehat{h}(S)| &\leq \frac{1 - \mathbf{E}[h]}{2} \cdot b_{\text{in}}^K \cdot \sum_{k=1}^K \left(\frac{a_{\text{in}}}{2}\right)^k \cdot \binom{K-1}{k-1} \cdot a_{\text{out}} b_{\text{out}}^k \\ &= \frac{1 - \mathbf{E}[h]}{2} \cdot a_{\text{out}} \cdot b_{\text{in}}^K \cdot \frac{a_{\text{in}} b_{\text{out}}}{2} \left(1 + \frac{a_{\text{in}} b_{\text{out}}}{2}\right)^{K-1} \\ &\leq \frac{1 - \mathbf{E}[h]}{2} \cdot a_{\text{out}} \cdot (a_{\text{in}} b_{\text{in}} b_{\text{out}})^K. \end{aligned}$$

where the last equality used the binomial theorem. Applying the same argument to  $-h$  lets us replace  $\frac{1 - \mathbf{E}[h]}{2}$  with  $\frac{1 + \mathbf{E}[h]}{2}$ , concluding the proof of Theorem 12.  $\blacktriangleleft$

## References

- 1 Nikhil Bansal and Makrand Sinha. K-forrelation optimally separates quantum and classical query complexity. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, New York, NY, USA, 2021. doi:10.1145/3406325.3451040.
- 2 Ido Ben-Eliezer, Rani Hod, and Shachar Lovett. Random low-degree polynomials are hard to approximate. *Comput. Complexity*, 21(1):63–81, 2012. doi:10.1007/s00037-011-0020-6.
- 3 Nayantara Bhatnagar, Parikshit Gopalan, and Richard J. Lipton. Symmetric polynomials over  $\mathbb{Z}_m$  and simultaneous communication protocols. *J. Comput. System Sci.*, 72(2):252–285, 2006. doi:10.1016/j.jcss.2005.06.007.
- 4 Eric Blais, Li-Yang Tan, and Andrew Wan. An inequality for the fourier spectrum of parity decision trees, 2015. arXiv:1506.01055.
- 5 Andrej Bogdanov and Emanuele Viola. Pseudorandom bits for polynomials. *SIAM J. Comput.*, 39(6):2464–2486, 2010. doi:10.1137/070712109.
- 6 Ravi Boppana, Johan Håstad, Chin Ho Lee, and Emanuele Viola. Bounded independence versus symmetric tests. *ACM Trans. Comput. Theory*, 11(4):Art. 21, 27, 2019. doi:10.1145/3337783.
- 7 Sourav Chakraborty, Nikhil S. Mande, Rajat Mittal, Tulasimohan Molli, Manaswi Paraashar, and Swagato Sanyal. Tight chang’s-lemma-type bounds for boolean functions. *CoRR*, abs/2012.02335, 2020. arXiv:2012.02335.
- 8 Eshan Chattopadhyay, Jason Gaitonde, Chin Ho Lee, Shachar Lovett, and Abhishek Shetty. Fractional Pseudorandom Generators from Any Fourier Level, 2020. arXiv:2008.01316.
- 9 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.
- 10 Eshan Chattopadhyay, Pooya Hatami, Shachar Lovett, and Avishay Tal. Pseudorandom generators from the second Fourier level and applications to AC0 with parity gates. In *10th Innovations in Theoretical Computer Science*, volume 124 of *LIPICs*, 2019. doi:10.4230/LIPICs.ITCS.2019.22.
- 11 Eshan Chattopadhyay, Pooya Hatami, Omer Reingold, and Avishay Tal. Improved pseudorandomness for unordered branching programs through local monotonicity. In *STOC’18—Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 363–375. ACM, New York, 2018. doi:10.1145/3188745.3188800.
- 12 Lijie Chen, Xin Lyu, and R. Ryan Williams. Almost-everywhere circuit lower bounds from non-trivial derandomization. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1–12, 2020. doi:10.1109/FOCS46700.2020.00009.
- 13 Gil Cohen, Noam Peri, and Amnon Ta-Shma. Expander random walks: A fourier-analytic approach. *Electron. Colloquium Comput. Complex.*, 27:163, 2020. URL: <https://eccc.weizmann.ac.il/report/2020/163>.

- 14 Dean Doron, Pooya Hatami, and William M. Hoza. Log-seed pseudorandom generators via iterated restrictions. In *35th Computational Complexity Conference*, volume 169 of *LIPICs*, 2020. doi:10.4230/LIPICs.CCC.2020.6.
- 15 Bill Fefferman, Ronen Shaltiel, Christopher Umans, and Emanuele Viola. On beating the hybrid argument. *Theory Comput.*, 9:809–843, 2013. doi:10.4086/toc.2013.v009a026.
- 16 Uma Girish, Ran Raz, and Avishay Tal. Quantum Versus Randomized Communication Complexity, with Efficient Players. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, volume 185 of *LIPICs*, 2021. doi:10.4230/LIPICs.ITCS.2021.54.
- 17 Uma Girish, Ran Raz, and Wei Zhan. Lower bounds for XOR of forrelations, 2020. arXiv:2007.03631.
- 18 Uma Girish, Avishay Tal, and Kewen Wu. Fourier Growth of Parity Decision Trees. In Valentine Kabanets, editor, *36th Computational Complexity Conference (CCC 2021)*, volume 200 of *LIPICs*, 2021. doi:10.4230/LIPICs.CCC.2021.39.
- 19 Parikshit Gopalan, Rocco A. Servedio, Avishay Tal, and Avi Wigderson. Degree and sensitivity: tails of two distributions. *Electron. Colloquium Comput. Complex.*, 23:69, 2016. URL: <http://eccc.hpi-web.de/report/2016/069>.
- 20 Ben Green and Terence Tao. The distribution of polynomials over finite fields, with applications to the Gowers norms. *Contrib. Discrete Math.*, 4(2):1–36, 2009.
- 21 Frederic Green, Daniel Kreymer, and Emanuele Viola. Block-symmetric polynomials correlate with parity better than symmetric. *Comput. Complexity*, 26(2):323–364, 2017. doi:10.1007/s00037-017-0153-3.
- 22 Chin Ho Lee. Fourier bounds and pseudorandom generators for product tests. In *34th Computational Complexity Conference*, volume 137 of *LIPICs*, 2019. doi:10.4230/LIPICs.CCC.2019.7.
- 23 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.
- 24 Shachar Lovett. Unconditional pseudorandom generators for low-degree polynomials. *Theory Comput.*, 5:69–82, 2009. doi:10.4086/toc.2009.v005a003.
- 25 Shachar Lovett, Roy Meshulam, and Alex Samorodnitsky. Inverse conjecture for the Gowers norm is false. *Theory Comput.*, 7:131–145, 2011. doi:10.4086/toc.2011.v007a009.
- 26 Yishay Mansour. An  $O(n^{\log \log n})$  learning algorithm for DNF under the uniform distribution. *J. Comput. System Sci.*, 50(3, part 3):543–550, 1995. Fifth Annual Workshop on Computational Learning Theory (COLT) (Pittsburgh, PA, 1992). doi:10.1006/jcss.1995.1043.
- 27 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.
- 28 Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992. doi:10.1007/BF01305237.
- 29 Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014. doi:10.1017/CB09781139814782.
- 30 Ryan O'Donnell and Rocco A. Servedio. Learning monotone decision trees in polynomial time. *SIAM J. Comput.*, 37(3):827–844, 2007. doi:10.1137/060669309.
- 31 Ran Raz and Avishay Tal. Oracle separation of BQP and PH. In *STOC'19—Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 13–23. ACM, New York, 2019. doi:10.1145/3313276.3316315.
- 32 Alexander A. Razborov. Lower bounds on the dimension of schemes of bounded depth in a complete basis containing the logical addition function. *Mat. Zametki*, 41(4):598–607, 623, 1987.

- 33 Omer Reingold, Thomas Steinke, and Salil P. Vadhan. Pseudorandomness for regular branching programs via fourier analysis. In *RANDOM 2013*, volume 8096 of *Lecture Notes in Computer Science*, pages 655–670, 2013.
- 34 Rocco A. Servedio and Li-Yang Tan. Improved pseudorandom generators from pseudorandom multi-switching lemmas. In *Approximation, randomization, and combinatorial optimization. Algorithms and techniques*, volume 145 of *LIPICs*, 2019. doi:10.4230/LIPICs.APPROX-RANDOM.2019.45.
- 35 Rocco A. Servedio and Li-Yang Tan. Pseudorandomness for read- $k$  DNF formulas. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 621–638. SIAM, Philadelphia, PA, 2019. doi:10.1137/1.9781611975482.39.
- 36 Alexander A. Sherstov, Andrey A. Storozhenko, and Pei Wu. An Optimal Separation of Randomized and Quantum Query Complexity, 2020. arXiv:2008.10223.
- 37 Roman 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, page 77–82, New York, NY, USA, 1987. Association for Computing Machinery. doi:10.1145/28395.28404.
- 38 Avishay Tal. Tight bounds on the Fourier spectrum of  $AC^0$ . In *32nd Computational Complexity Conference*, volume 79 of *LIPICs*, 2017. doi:10.4230/LIPICs.CCC.2017.15.
- 39 Avishay Tal. Towards optimal separations between quantum and randomized query complexities. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 228–239, 2020. doi:10.1109/FOCS46700.2020.00030.
- 40 Emanuele Viola. On the power of small-depth computation. *Foundations and Trends in Theoretical Computer Science*, 5(1):1–72, 2009.
- 41 Emanuele Viola. The sum of  $d$  small-bias generators fools polynomials of degree  $d$ . *Comput. Complexity*, 18(2):209–217, 2009. doi:10.1007/s00037-009-0273-5.
- 42 Emanuele Viola. Challenges in computational lower bounds. *SIGACT News, Open Problems Column*, 48(1), 2017.
- 43 Emanuele Viola. Fourier conjectures, correlation bounds, and majority. *Electron. Colloquium Comput. Complex.*, 27:175, 2020. URL: <https://eccc.weizmann.ac.il/report/2020/175>.
- 44 Emanuele Viola. New lower bounds for probabilistic degree and  $AC^0$  with parity gates. *Theory of Computing*, 2021. Available at <http://www.ccs.neu.edu/home/viola/>.

## A Reduction to bound without acceptance probability

In this section, we show that given any  $L_{1,k}$  Fourier norm bound on a class of functions that is closed under XOR on disjoint variables, such a bound can be automatically “upgraded” to a refined bound that depends on the acceptance probability:

► **Lemma 22.** *Let  $\mathcal{F}$  be a class of  $\{-1, 1\}$ -valued functions such that for every  $f \in \mathcal{F}$ , the XOR of disjoint copies of  $f$  (over disjoint sets of variables) also belongs to  $\mathcal{F}$ . If  $L_{1,k}(\mathcal{F}) \leq b^k$ , then for every  $f \in \mathcal{F}$  it holds that  $L_{1,k}(f) \leq 2e \cdot \frac{1-|\mathbf{E}[f]|}{2} \cdot b^k$ .*

**Proof.** Suppose not, and let  $f \in \mathcal{F}$  be such that  $L_{1,k}(f) > 2e \cdot \frac{1-|\mathbf{E}[f]|}{2} \cdot b^k$ . We first observe that since  $L_{1,k}(\mathcal{F}) \leq b^k$ , it must be the case that  $1 - |\mathbf{E}[f]| \leq 1/e$ . Let  $\alpha := \frac{1-|\mathbf{E}[f]|}{2} \in [0, \frac{1}{2e}]$  so that  $|\mathbf{E}[f]| = 1 - 2\alpha \geq 1 - 1/e$ . Let  $f^{\oplus t}$  be the XOR of  $t$  disjoint copies of  $f$  on  $tn$  variables, where the integer  $t$  is to be determined below. By our assumption, we have  $f^{\oplus t} \in \mathcal{F}$  and thus

$$\begin{aligned}
 L_{1,k}(f^{\oplus t}) &\geq \binom{t}{1} \cdot L_{1,0}(f)^{t-1} \cdot L_{1,k}(f) && \text{(by disjointness)} \\
 &= t \cdot (1 - 2\alpha)^{t-1} \cdot L_{1,k}(f) && (L_{1,0}(f) = \mathbf{E}[f]) \\
 &> t \cdot (1 - 2\alpha)^{t-1} \cdot 2e \cdot \alpha \cdot b^k =: \Lambda(t).
 \end{aligned}$$

### 53:20 Fourier Growth of Structured $\mathbb{F}_2$ -Polynomials and Applications

We note that if  $\alpha = 0$  then  $|\mathbf{E}[f]| = 1$ , so all the Fourier weight of  $f$  is on the constant coefficient, and hence the claimed inequality holds trivially. So we subsequently assume that  $0 < \alpha \leq \frac{1}{2e}$ . Let  $t^* := \frac{1}{-\ln(1-2\alpha)} > 0$ . It is easy to verify that  $\Lambda(t)$  is increasing when  $t \leq t^*$ , and is decreasing when  $t \geq t^*$ .

We choose  $t = \lceil t^* \rceil$ . Since  $\alpha \leq \frac{1}{2e} < \frac{e-1}{2e} \approx 0.3161$ , we have  $t^* > 1$  and thus

$$\begin{aligned} L_{1,k}(f^{\oplus t}) &> \Lambda(\lceil t^* \rceil) \geq \Lambda(t^* + 1) = \left( \frac{1}{-\ln(1-2\alpha)} + 1 \right) \cdot (1-2\alpha)^{-\frac{1}{\ln(1-2\alpha)}} \cdot 2e \cdot \alpha \cdot b^k \\ &= \left( \frac{2\alpha}{-\ln(1-2\alpha)} + 2\alpha \right) \cdot b^k \geq b^k, \end{aligned}$$

where the last inequality holds for every  $\alpha \in (0, \frac{e-1}{2e}]$  and can be checked via elementary calculations. This contradicts  $L_{1,k}(\mathcal{F}) \leq b^k$ , and the lemma is proved.  $\blacktriangleleft$

# Candidate Tree Codes via Pascal Determinant Cubes

Inbar Ben Yaacov ✉

The Blavatnik School of Computer Science, Tel-Aviv University, Israel

Gil Cohen ✉ 🏠

The Blavatnik School of Computer Science, Tel-Aviv University, Israel

Anand Kumar Narayanan ✉

CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

---

## Abstract

Tree codes are combinatorial structures introduced by Schulman [23] as key ingredients in interactive coding schemes. Asymptotically-good tree codes are long known to exist, yet their explicit construction remains a notoriously hard open problem. Even proposing a plausible construction, without the burden of proof, is difficult and the defining tree code property requires structure that remains elusive. To the best of our knowledge, only one candidate appears in the literature, due to Moore and Schulman [19].

We put forth a new candidate for an explicit asymptotically-good tree code. Our construction is an extension of the vanishing rate tree code by Cohen-Haeupler-Schulman [7], and its correctness relies on a conjecture that we introduce on certain Pascal determinants indexed by the points of the Boolean hypercube. Furthermore, using the vanishing distance tree code by Gelles *et al.* [12] enables us to present a construction that relies on an even weaker assumption. We furnish evidence supporting our conjecture through numerical computation, combinatorial arguments from planar path graphs and based on well-studied heuristics from arithmetic geometry.

**2012 ACM Subject Classification** Theory of computation → Error-correcting codes

**Keywords and phrases** Tree codes, Sparse polynomials, Explicit constructions

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.54

**Category** RANDOM

**Related Version** *Extended Version*: <https://eccc.weizmann.ac.il/report/2020/141/>

**Funding** *Inbar Ben Yaacov*: Funded by the Israel Science Foundation (grant number 1569/18).

*Gil Cohen*: Funded by the Israel Science Foundation (grant number 1569/18) and by the Azrieli Faculty Fellowship.

*Anand Kumar Narayanan*: Supported by the European Union’s H2020 Programme (grant agreement #ERC-669891).

**Acknowledgements** The second author wishes to thank Roni Con, Shir Peleg-Schatzman, Noam Peri, Tal Roth, and Shahar Samocha for interesting discussions on tree codes.

## 1 Introduction

Coding theory addresses the problem of communication over an imperfect channel. In the classic setting studied in the seminal work of Shannon [26], Alice wishes to communicate a message to Bob over a channel that may induce errors. The question then is: how should Alice encode her message so that if the amount of errors is not excessive, Bob can recover her message? Around the same time, Hamming [14] introduced the notion of an error-correcting code. A function  $C: \Sigma^k \rightarrow \Sigma^n$  is an *error-correcting code* with distance  $\delta$  if for every distinct  $x, y \in \Sigma^k$ , the respective images  $C(x), C(y)$  have relative Hamming distance at least  $\delta$ . The



© Inbar Ben Yaacov, Gil Cohen, and Anand Kumar Narayanan;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 54; pp. 54:1–54:22



Leibniz International Proceedings in Informatics

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

rate of information transmission  $\rho = \frac{k}{n}$  and the fraction of errors corrected (roughly  $\delta/2$ ) are competing quantities with a tradeoff between them. Among the most basic questions in coding theory is to obtain explicit *asymptotically good codes*, that is, codes over fixed  $\Sigma$  with constant distance  $\delta > 0$  and constant rate  $\rho > 0$ . By “explicit” we mean that  $C$  can be evaluated in time  $\text{poly}(n)$ . Justensen [15] was the first to devise such an explicit construction. Since then, several explicit constructions have appeared, including using algebraic geometry codes [28] and expander graphs [27].

While error-correcting codes can be used to solve the problem of sending a single message from Alice to Bob over an imperfect channel, in some settings, the two parties interact with each other, sending multiple messages where a message depends on previous messages that were exchanged. Interactive coding addresses the subtler problem of enabling such dynamic interaction over an imperfect channel. In this far more challenging setting, standard codes do not offer a satisfactory solution.

Tree codes are powerful combinatorial structures, defined by Schulman [23, 25] as key ingredients for achieving interactive coding schemes. They play a role analogous to that error-correcting codes take in the single message setting. Tree codes, as their name suggests, are trees with certain distance properties. To give the formal definition, we set some notation. Let  $T$  be a rooted binary tree that is endowed with an edge coloring from some ambient color set (or alphabet)  $\Sigma$ . For vertices  $u, v$  of equal depth let  $w$  be their least common ancestor and denote the distance, in edges, from  $u$  to  $w$  by  $\ell$ . Let  $p_u, p_v \in \Sigma^\ell$  be the sequences of colors on the path from  $w$  to  $u$  and to  $v$ , respectively. We define  $h(u, v)$  to be the relative Hamming distance between  $p_u$  and  $p_v$ . Informally,  $h(u, v)$  measures the distance between the two color sequences obtained by following the paths from the root to each of  $u$  and  $v$ , excluding the “non-interesting” common prefix. A tree code is any coloring that has a lower bound on this quantity. Formally,

► **Definition 1** (Tree codes [23]). *Let  $T$  be the complete rooted binary tree of depth  $n$ . The tree  $T$ , together with an edge-coloring of  $T$  by a color set  $\Sigma$  is called a tree code with distance  $\delta$  if for every pair of vertices  $u, v$  with equal depth it holds that  $h(u, v) \geq \delta$ .*

It is not clear at all that there exists a universal constant  $\delta > 0$  such that for every  $n$  there exists a depth- $n$  tree code with distance  $\delta$ . Namely, it is not clear that there is a family of tree codes  $(T_n)_{n \in \mathbb{N}}$ , where  $T_n$  has depth  $n$ , such that the color set  $\Sigma$  is common to all trees in the family, and every  $T_n$  has distance  $\delta$ . We refer to such a family as a tree code with distance  $\delta$  over the color set  $\Sigma$ .

Three different proofs were provided by Schulman, showing that for any constant  $\delta < 1$  there exists a tree code with alphabet size  $|\Sigma| = O_\delta(1)$  achieving distance  $\delta$ . More recently, based on Schulman’s ideas, it was shown that there is a tree code with only 4 colors, having positive distance (in particular, distance  $\delta = 0.136$ ) [8] and, moreover, 3 colors do not suffice to guarantee any constant distance  $\delta > 0$ . All of these proofs rely on the probabilistic method and thus are not explicit. The problem of constructing asymptotically-good tree codes has drawn substantial attention [24, 6, 19, 21, 12, 7, 20], but has endured as a difficult challenge.

Given this difficulty, it is natural to construct, for a given distance parameter  $\delta > 0$ , a family of tree codes  $(T_n)_{n \in \mathbb{N}}$  for which  $T_n$  is allowed to use some  $c(n)$  number of colors. The goal is to obtain an asymptotically slowly-growing function  $c$ . Note that constructing a tree code family with  $c(n) = 2^n$  colors is trivial. Indeed, having so many colors at hand, one can encode the entire path leading to a vertex on the edge preceding it, yielding distance  $\delta = 1$ . In an unpublished manuscript, Evans, Klugerman and Schulman [24] constructed a tree code with  $c(n) = n^{O_\delta(1)}$  colors. The state-of-the-art construction [7] achieves  $c(n) = (\log n)^{O_\delta(1)}$ . See [20] for alternative constructions achieving the same parameters as well as decoding algorithms, and [4] for an account relating [7] and [21].



Despite this progress, constructing asymptotically-good tree codes is wide open. Curiously, even candidate constructions are rare. This is mostly because a tree code is *not* a pseudorandom object. Its defining property requires structure that remains elusive. For this reason, even proposing a plausible construction, without the burden of proof, requires further insight and is not an easy task. To the best of our knowledge, there is a single candidate in the literature, due to Moore and Schulman [19]. The construction’s distance property relies on an intriguing open conjecture about certain exponential sums that the authors introduce. The Moore-Schulman conjecture was verified computationally for small instances, and the hope is that these represent the general case.

## 1.1 Our Contribution

In this work we put forth a candidate construction of asymptotically-good tree codes. Namely, for some universal constant  $c \geq 1$  and for every integer  $n \geq 1$  we give an explicit construction of a depth- $n$  binary tree code with  $c$  colors. The distance of the tree code is bounded below by some constant  $\delta > 0$ , independent of  $n$ , provided a conjecture that we introduce on certain Pascal determinants associated with the points of the Boolean hypercube holds. We give independent supporting evidence for our conjecture: first through the combinatorics of planar path graphs underlying our construction and then based on well-studied heuristics from arithmetic geometry. Furthermore, we verify the conjecture computationally on small values.

Our candidate tree code is an extension of the [7] construction. We set the stage in Section 2 with a discussion of [7] followed by a description of our contributions in Section 3. Underlying the [7] construction is a key online uncertainty principle for the Newton basis: a consequence of non-vanishing of Pascal (binomial) sub-matrix determinants, proved by invoking the combinatorial Lindström-Gessel-Viennot lemma. These determinants are in fact positive numbers growing exponentially with the depth of the tree, forcing the [7] construction to require poly-logarithmic number of colors. With the intent of reducing the number of colors, one may try to work modulo a prime in hopes the non-vanishing is still preserved. In Section 3.1 we reason the contrary is true: it is unlikely to work for primes small enough to guarantee a constant number of colors. There are exponentially many Pascal sub-matrix determinants, at least one of which is likely to vanish “accidentally” modulo the chosen prime.

Our main technical contribution is an extension of the [7] construction, which we present as a candidate asymptotically-good tree code. The construction extends ideas of [7] and further makes use of the vanishing-distance tree code by Gelles *et al.* [12], which allows us to relax our assumption. An informal description of the main ideas is in Section 3.3 with a formal treatment of the more intricate aspects deferred to Section 5. In our construction, the role of each Pascal sub-matrix determinant is recast as a bundle of Pascal sub-matrix determinants, parametrized by points on the Boolean hypercube of high enough dimension (hence the term “Pascal determinant cube” in the title). We then work modulo a prime  $p$  of appropriate size. Instead of worrying about a determinant vanishing modulo  $p$ , we only have to worry about the whole associated cube of Pascal determinants vanishing modulo  $p$ . Informed by computation, combinatorics and arithmetic, we formulate the Conjecture 4 in Section 3.2 that the cube of determinants never vanishes modulo our chosen prime. We prove that if the conjecture (or even an asymptotic version of it, Conjecture 5) holds then our construction is indeed asymptotically good.

In Appendix A, we investigate our conjecture through a combinatorial lens. Each determinant bundle in the conjecture can be encoded as an integer polynomial whose evaluation at the points of the Boolean hypercube gives the bundle. Through the Lindström-

Gessel-Viennot lemma, in Appendix A.1 we prove that the polynomial never vanishes on any point of the Boolean hypercube. For the conjecture to fail, all these exponentially many evaluations must be divisible by our chosen prime number, which we reason is likely impossible for our chosen parameters. This very scenario is reformulated in terms of Boolean functions in Appendix A.2, by multi-linearizing the aforementioned polynomial. Conjecture 4 is then rephrased as the non vanishing of an  $\mathbb{F}_p$ -valued Boolean function, furthering our belief in the conjecture.

In Appendix B, we look to deep results from arithmetic geometry to claim the plausibility of our conjecture. If the hypersurface of zeroes of the aforementioned polynomial encoding the bundle of determinants intersects with the Boolean hypercube generically, our conjecture holds true. Following Fouvry [10], we investigate this intersection deploying Katz-Laumon exponential sums. The bounds on Katz-Laumon sums and Fouvry's point counting technique fall short of quantitatively proving our conjecture. Yet, we show they suffice to prove a nontrivial relaxation of our conjecture: with the Boolean hypercube extended to hypercubes of side length  $\approx p^{3/4}$ . Despite falling short of proving our conjecture, the methods are illuminating and suggest there are no arithmetic obstructions to our conjecture.

## 1.2 Recent Developments

Since posting our report online, there have been several exciting developments. Brakerski, Kalai and Saxena [5] published a major advancement on the use of tree codes in interactive coding. They demonstrate how a tree code with an efficient encoding algorithm suffices in obtaining efficient and deterministic interactive coding schemes against adversarial errors. In particular, they completely eliminate the necessity of the tree code possessing an efficient decoding algorithm. Our candidate tree code clearly has an efficient encoding algorithm and seamlessly fits their needs. Therefore, proving our tree codes are indeed asymptotically good would immediately imply efficient and deterministic interactive coding schemes.

Pudlák gleaned an abstraction of our construction and reduced the problem of constructing asymptotically good tree codes to constructing block matrices of the following form [22]. Consider an  $n$  by  $n$  block matrix whose entries are 2 by 1 column vector blocks. Say it is triangular, meaning all blocks above the diagonal are  $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$  vectors. For every  $k$  by  $k$  block sub matrix (where the sorted column indices are never ahead of the row indices), Pudlák demands that the  $2k$  by  $k$  matrix induced by forgetting the block structure is full rank. This rank criterion is a relaxation of our determinant bundle non vanishing. Pudlák further proves that random triangular block matrices with entries from a finite field of size quadratic in  $n$  satisfy the rank criterion with high probability. Explicit deterministic construction of such block matrices beckons, with our construction being the only currently proposed candidate.

## 2 Cohen-Haeupler-Schulman Tree Codes

For the sequel, it is convenient to think of a tree code as an online version of a regular error correcting code. Recall that a tree code consists of a complete rooted, depth- $n$  binary tree in which each edge is labeled by a symbol from an alphabet  $\Sigma$ . This naturally induces a one-to-one mapping assigning each binary string  $s$  to a path starting at the root, where  $s$  indicates which child is taken in each of the steps. Such a path maps to a string over  $\Sigma$ , namely, the concatenation of symbols along the path. This way, a tree code  $T$  encodes any binary string  $s$  into an equally long string  $T(s)$  over  $\Sigma$ . This encoding has the online property because the encoding of any prefix does not depend on later symbols. Thus, one

can view a binary tree code as an online function  $T : \{0, 1\}^n \rightarrow \Sigma^n$ . It is useful to consider input alphabets other than binary (which corresponds to a larger arity of the tree). In [7], the input symbols are elements of  $\mathbb{Z}$  rather than  $\{0, 1\}$ .

The distance property of a tree code can be phrased as follows when viewed as a function  $T : \Sigma_{\text{in}}^n \rightarrow \Sigma_{\text{out}}^n$ . For every pair of distinct strings  $m = (m_0, \dots, m_{n-1})$ ,  $m' = (m'_0, \dots, m'_{n-1})$ ,  $c$  being the least integer such that  $m_c \neq m'_c$ , the following holds. For every  $\ell \in [0, n - c]$  (for integers  $a < b$  we write  $[a, b]$  for  $\{a, a + 1, \dots, b - 1\}$ ) the strings  $(T(m)_c, \dots, T(m)_{c+\ell})$ ,  $(T(m')_c, \dots, T(m')_{c+\ell})$  are at Hamming distance at least  $\delta(\ell + 1)$ .

**The Newton basis.** [7] makes use of the Newton basis for real polynomials. This basis consists of polynomials of the form  $\binom{x}{k} \in \mathbb{R}[x]$  for  $k \in \mathbb{N}$ , where  $\binom{x}{k} = \frac{x(x-1)\dots(x-(k-1))}{k!}$ . It is easy to verify that for every  $d \in \mathbb{N}$ , the set  $\{\binom{x}{k} \mid k = 0, 1, \dots, d\}$  forms a basis for the space of univariate real polynomials of degree at most  $d$ . The feature which makes the Newton basis suitable for constructing tree codes unlike, say the standard basis, is its online nature with respect to  $\mathbb{N}$ . Let  $m_0, \dots, m_t \in \mathbb{R}$ . Let  $f(x) = \sum_{i=0}^t a_i x^i$  be the least degree polynomial that interpolates on the points  $(0, m_0), \dots, (t, m_t)$ . Then, generally, given a new point  $(t + 1, m_{t+1})$ , the least degree polynomial,  $g(x) = \sum_{i=0}^{t+1} b_i x^i$ , that interpolates on  $(0, m_0), \dots, (t + 1, m_{t+1})$  will have a completely different sequence of coefficients (i.e.,  $a_i \neq b_i$ ). By contrast, using the Newton basis, the coefficients that were already “recorded” stay intact given the new point  $(t + 1, m_{t+1})$ . More precisely, if  $f(x) = \sum_{i=0}^t \gamma_i \binom{x}{i}$  then  $g(x) = f(x) + \gamma_{t+1} \binom{x}{t+1}$  for some  $\gamma_{t+1} \in \mathbb{R}$ . Thus, for every  $t$ , the coefficient  $\gamma_t$  is determined by  $m_0, m_1, \dots, m_t$ . Another convenient property of the Newton basis, not shared by the standard basis, is that if  $m_0, \dots, m_t$  are all integers, so are the coefficients  $\gamma_0, \dots, \gamma_t$ .

**The [7] tree code over the integers.** In [7], for every integer  $n \geq 1$  a function  $\text{TC}_{\mathbb{Z}} : \mathbb{Z}^n \rightarrow (\mathbb{Z} \times \mathbb{Z})^n$  is constructed as follows. Given  $m = (m_0, \dots, m_{n-1}) \in \mathbb{Z}^n$ , let  $f \in \mathbb{R}[T]$  be the least degree real polynomial that interpolates on  $(0, m_0), \dots, (n - 1, m_{n-1})$ . Expand  $f$  in the Newton basis  $f(T) = \sum_{t=0}^{n-1} \gamma_t \binom{T}{t}$ . With this notation, for every  $t \in [0, n)$ , define  $\text{TC}_{\mathbb{Z}}(m)_t = (m_t, \gamma_t)$ . In words, at time  $t$ , both the  $t^{\text{th}}$  input symbol is outputted as well as the “new” coefficient  $\gamma_t$ .

**Analysis.** To argue about the distance of  $\text{TC}_{\mathbb{Z}}$ , using the fact that it is  $\mathbb{R}$ -linear, one has to prove that if  $c \in [0, n)$  is the least integer for which  $m_c \neq 0$  then for every  $\ell \in [0, n - c)$ , at least  $\delta$ -fraction of the indices in  $[c, c + \ell]$  satisfies that  $\text{TC}_{\mathbb{Z}}(m)_t$  is nonzero (as a pair). If we write, for  $d \in [0, n)$ ,  $f_d(T) = \sum_{t=0}^d \gamma_t \binom{T}{t}$  then the number of non-zeros in the sequence  $\gamma_c, \gamma_{c+1}, \dots, \gamma_{c+\ell}$  is precisely the sparsity of  $f_{c+\ell}$  in the Newton basis. This, together with the fact that for every  $i \leq t$ ,  $m_i = f_t(i)$ , implies that to “break” the construction  $\text{TC}_{\mathbb{Z}}$ , one must come up with a sparse polynomial  $f_{c+\ell}$ , with respect to the Newton basis, that has many roots in  $I = \{c, c + 1, \dots, c + \ell\}$ . Indeed, if  $f_{c+\ell}$  is not sparse, then many of the  $\gamma$ -entries of  $(\text{TC}_{\mathbb{Z}}(m)_t)_{t \in I}$  will be nonzero. On the other hand, if  $f_{c+\ell}$  has only few roots in  $I$  then many of the  $m$ -entries are nonzero. To this end, the main lemma proved in [7] is a bound on the numbers of distinct integral roots a real polynomial can have as a function of its sparsity in the Newton basis.

► **Lemma 2** ([7]). *Let  $f \in \mathbb{R}[T]$  be a nonzero polynomial of sparsity  $s \geq 1$  in the Newton basis. Let  $c \geq 0$  be the least integer such that  $f(c) \neq 0$ . Then,  $f$  has at most  $s - 1$  distinct roots in  $[c, \infty) \cap \mathbb{N}$ .*

Lemma 2 implies that if the sparsity of  $f_{c+\ell}$  is  $s$  then there can be at most  $s - 1$  zeros among the  $m$ -entries of  $\{\text{TC}_{\mathbb{Z}}(m)_t\}_{t \in I}$ , establishing  $\text{TC}_{\mathbb{Z}}$  has distance at least  $\frac{1}{2}$ .

**The Lindström-Gessel-Viennot Lemma.** Lemma 2 is proved using a corollary of the Lindström-Gessel-Viennot Lemma. Let  $\mathbf{t} = (t_1, \dots, t_s)$ ,  $\mathbf{c} = (c_1, \dots, c_s)$  be strictly increasing sequences of non-negative integers. Let  $M_{\mathbf{t}, \mathbf{c}}$  be the  $s \times s$  matrix whose  $(i, j)^{\text{th}}$  entry is given by  $\binom{t_i}{c_j}$ . We write  $\mathbf{c} \leq \mathbf{t}$  if  $c_i \leq t_i$  for every  $i \in [s]$ .

► **Lemma 3** ([13], Corollary 2).  $\mathbf{c} \leq \mathbf{t} \iff \det M_{\mathbf{t}, \mathbf{c}} \neq 0$ .

For more recent treatments of the LGV Lemma see [1], Chapter 5.4 or [2], Chapter 25. This lemma is in fact much older, and we invite the reader to look at the appendix of [7] for more information regarding the history of this lemma.

**The binary tree code.** To reduce the alphabet to binary, [7] proves that if for every  $t$ ,  $|m_t| \leq 2^k$  for some  $k$  then  $|\gamma_t| \leq 2^{t+k}$ . Given a binary string  $m = (m_0, \dots, m_{n-1})$ , partition  $m$  to  $\sqrt{n}$  consecutive blocks of length  $\sqrt{n}$ , and interpret each block as a non-negative integer  $M_i$  of size at most  $2^{\sqrt{n}}$ . At this point, the tree code over the integers  $\text{TC}_{\mathbb{Z}} : \mathbb{N}^{\sqrt{n}} \rightarrow (\mathbb{Z} \times \mathbb{Z})^{\sqrt{n}}$  can be applied to  $M_0, \dots, M_{\sqrt{n}-1}$ . By the above bound,  $|\gamma_t| \leq 2^{t+\sqrt{n}} \leq 2^{2\sqrt{n}}$ . Hence, an output symbol  $(m_t, \gamma_t)$  can be encoded using  $3\sqrt{n}$  bits. Of course, these bits cannot be output on the fly as one must write a symbol only after all of the  $\sqrt{n}$  bits of the corresponding input symbol have been read. This creates a “lag” of length  $\sqrt{n}$  that can be resolved by using a depth- $\sqrt{n}$  tree code which is obtained recursively. As the recursive depth is  $O(\log \log n)$  and since for every bit read one writes  $O(1)$  bits per recursive call, the number of bits written per input bit is  $O(\log \log n)$ . Hence, the poly( $\log n$ ) alphabet size.

### 3 Our Contribution

#### 3.1 The Unlikeliness of an LGV-Like Lemma Over Small Fields

The reason that the [7] construction is not asymptotically-good is that their tree code is constructed over the integers, and the alphabet reduction that is invoked has a cost that is exponential in the depth of the recursion. The recursion’s depth is directly affected by the magnitude of the  $\gamma_t$  symbols which, unfortunately, are exponential in  $t$ . Taking  $\sqrt{n}$ -length blocks yields the best trade-off, resulting in depth  $O(\log \log n)$ .

One can show that resorting to such recursion could have been avoided if the construction was carried over a prime field  $\mathbb{F}_p$  with  $p = \text{poly}(n)$ . That is, instead of outputting  $\gamma_t$ , output its reduction modulo  $p$ . To be precise, for the construction to work, one must take  $p \geq n$  due to other considerations. However, as long as  $p < n^e$  for some constant  $e$ , standard techniques can be used to obtain an asymptotically-good binary tree code, where the constant  $e$  will affect the rate of the resulted tree.

A very similar approach to this was raised by Pudlák [21]. On this, we quote a sentence from the conclusion part of [21]: “This seems to be a very difficult problem and we do not dare to conjecture that  $p$  may be of polynomial size”. At this point, Pudlák suggests studying restricted cases for which small fields suffice and try to base tree code constructions on such results, but we digress.

In consensus with Pudlák, we too believe that the approach of working over  $\mathbb{F}_p$  as suggested above is not likely to work. That is, it seems very plausible to us that the LGV Lemma does not have an analog over a field of size poly( $n$ ). More precisely, we suspect that for every constant  $e \geq 1$ , there exists  $n_0 = n_0(e)$  such that for every  $n \geq n_0$  and  $p \leq n^e$ , there exists a pair  $\mathbf{t}, \mathbf{c} \in [0, n]^s$ , for some  $s \in [n]$ , satisfying  $\mathbf{c} \leq \mathbf{t}$ , such that  $\det M_{\mathbf{t}, \mathbf{c}} \equiv_p 0$ .

To get some intuition as to why we believe this is the case, fix some prime  $p$  and  $s \in [n]$ . There are between  $\binom{n}{s}$  and  $\binom{n}{s}^2$  pairs of sequences  $\mathbf{t}, \mathbf{c}$  to consider. Unless some structure is present, one would expect that roughly  $\frac{1}{p}$ -fraction of pairs  $\mathbf{t}, \mathbf{c}$  would satisfy  $p \mid \det M_{\mathbf{t}, \mathbf{c}}$ .

By that heuristic, we do not expect that  $p$  can be taken much smaller than  $\binom{n}{s}$ . As we are interested in  $s$  that can be as large as  $\Omega(n)$ , this heuristic points against the existence of a “good” prime  $p = 2^{o(n)}$ , let alone  $p = \text{poly}(n)$ .

This heuristic is supported by a computational search that we carried. Let  $\mathcal{P}_1 : \mathbb{N} \rightarrow \mathbb{N}$  be the function that maps  $n \in \mathbb{N}$  to the least prime  $p$  that satisfies the following property. For every  $s \in [n]$  and strictly increasing sequences  $\mathbf{c} = (c_1, \dots, c_s), \mathbf{t} = (t_1, \dots, t_s) \in [0, n]^s$  with  $\mathbf{c} \leq \mathbf{t}$  it holds that  $\det M_{\mathbf{t}, \mathbf{c}} \not\equiv_p 0$ . Informally,  $\mathcal{P}_1$  maps  $n$  to the smallest prime  $p$  that is “good” for  $n$ . An exhaustive search we have conducted for hundreds of computer hours seems to suggest that  $\mathcal{P}_1(n)$  grows exponentially with  $n$ .

■ **Table 1** Values of  $\mathcal{P}_1(n)$  obtained using a computer search.

$n$	6	7	8	9	10	11	12	13	14	15	16
$\mathcal{P}_1(n)$	13	17	47	89	241	641	2,687	6,521	15,401	74,257	> 250,000

Since posting our preprint online, Karingula and Lovett [16] consider the non singularity of submatrices of triangular matrices modulo  $p$  in a different context. They too arrive at our conclusion, in fact, conjecturing a stronger claim ([16], Conjecture 1.5): for every triangular integer matrix, a fraction of the determinants (corresponding to index sequences  $\mathbf{t}, \mathbf{c}$ , as above) are likely to vanish modulo  $p$  unless the field size  $p$  grows exponentially.

### 3.2 A Conjecture

The informal heuristic presented above makes the point that no  $\text{poly}(n)$ -size prime is likely to work against all  $\exp(n)$  many pairs of sequences as we have no evidence for a structural phenomenon to support the seemingly unlikely alternative. The main contribution of this work is a tree code construction—a variant of [7]—whose distance analysis relies on what we believe is a plausible statement which we put forth as a conjecture. To formally state our conjecture some preparation is required.

As before, let  $\mathbf{c} = (c_1, \dots, c_s), \mathbf{t} = (t_1, \dots, t_s) \in [0, n]^s$  be a pair of strictly increasing sequences with  $\mathbf{c} \leq \mathbf{t}$ . For symbolic variables  $X_1, \dots, X_s$ , define the  $s \times s$  (symbolic) matrix  $M_{\mathbf{t}, \mathbf{c}}(X_1, \dots, X_s)$  whose  $(i, j)$ <sup>th</sup> entry is given by  $\binom{X_i + t_i}{c_j}$ . Define  $\Phi_{\mathbf{t}, \mathbf{c}}(X_1, \dots, X_s) \triangleq \det M_{\mathbf{t}, \mathbf{c}}(X_1, \dots, X_s) \in \mathbb{Z}[X_1, \dots, X_s]$ . For a prime  $p$ , let  $\Phi_{\mathbf{t}, \mathbf{c}}^p(X_1, \dots, X_s) \in \mathbb{F}_p[X_1, \dots, X_s]$  denote the reduction of  $\Phi_{\mathbf{t}, \mathbf{c}}$  at  $p$ . That is, every coefficient of  $\Phi_{\mathbf{t}, \mathbf{c}}$  is taken modulo  $p$  to form  $\Phi_{\mathbf{t}, \mathbf{c}}^p$ . With this notation, to ensure that the [7] tree code works over  $\mathbb{F}_p$ , one must establish that  $\Phi_{\mathbf{t}, \mathbf{c}}^p(0, \dots, 0) \neq 0$  for all  $\mathbf{t}, \mathbf{c}$  in question. Put differently, the [7] construction fails if for some pair  $\mathbf{t}, \mathbf{c}$  as above,  $\Phi_{\mathbf{t}, \mathbf{c}}^p$  evaluates to 0 at the origin. Our main contribution is an explicit construction which fails only if  $\Phi_{\mathbf{t}, \mathbf{c}}^p$  evaluates to 0 on the *entire* Boolean hypercube  $\{0, 1\}^s$ . Equivalently, our construction is asymptotically-good if

$$\exists(x_1, \dots, x_s) \in \{0, 1\}^s \quad \Phi_{\mathbf{t}, \mathbf{c}}(x_1, \dots, x_s) \not\equiv_p 0. \tag{3.1}$$

#### 3.2.1 Preliminary Informal Discussion on the Plausibility of Equation (3.1)

To start with, consider a very informal point of view on the plausibility of Equation (3.1), a discussion similar in spirit to the one conducted for arguing against the plausibility of taking the [7] construction over  $\mathbb{F}_p$ . Heuristically, and very informally, one may think of the  $2^s$  conditions in Equation (3.1) as  $2^s$  trials that are “generated by  $s$  independent

random variables”  $X_1, \dots, X_s$ . Unless some structural obstruction is in place, the “event” in Equation (3.1) is expected to have probability of about  $p^{-s}$ . Continuing this informal line of reasoning, by a union bound, one would expect that for a choice of  $p$  satisfying  $p^{-s} \binom{n}{s}^2 \ll \frac{1}{n}$ , Equation (3.1) holds for every pair  $\mathbf{t}, \mathbf{c} \in [0, n]^s$ , for every  $s \in [n]$ . The latter holds by taking  $p \gg n^3$ .

Another informal argument supporting the validity of Equation (3.1) is as follows. Note that  $\Phi_{\mathbf{t}, \mathbf{c}}$  has total degree  $d \leq sn \leq n^2$ . In fact, as we only care about  $\Phi_{\mathbf{t}, \mathbf{c}}$  restricted to  $\{0, 1\}^s$ , we may assume that  $\Phi_{\mathbf{t}, \mathbf{c}}$  is multi-linear and so  $d \leq s \leq n$ . One can show that for  $p > n$ ,  $\Phi_{\mathbf{t}, \mathbf{c}}^p$  is a nonzero polynomial; thus, by Schwartz-Zippel,  $\Phi_{\mathbf{t}, \mathbf{c}}^p$  has at most  $\frac{d}{p} \leq \frac{n}{p}$  fraction of roots in  $\mathbb{F}_p^s$ . By taking, say,  $p \geq n^2$ , the roots of  $\Phi_{\mathbf{t}, \mathbf{c}}$  occupy at most  $\frac{1}{\sqrt{p}}$ -fraction of  $\mathbb{F}_p^s$ . Now, for the heuristic part, one may conjecture that  $\{0, 1\}^s$  “looks random” to the zero set  $V_{\mathbf{t}, \mathbf{c}}$  of  $\Phi_{\mathbf{t}, \mathbf{c}}$ . As a weak consequence,  $\{0, 1\}^s$  is not contained in  $V_{\mathbf{t}, \mathbf{c}}$ , which is the content of Equation (3.1).

### 3.2.2 The Conjecture

There is one small technical issue we need to address before presenting our formal conjecture. Note that if  $t_{i+1} = t_i + 1$  for some  $i$  then  $\Phi_{\mathbf{t}, \mathbf{c}}(x_1, \dots, x_s) = 0$  whenever  $x_i = 1$  and  $x_{i+1} = 0$  for the simple reason that two of the rows of  $M_{\mathbf{t}, \mathbf{c}}(x_1, \dots, x_s)$  are identical. Informally, from the heuristic point of view discussed above, when  $t_{i+1} = t_i + 1$ , the events associated with the variables  $X_i, X_{i+1}$  are dependent. To exclude these trivial roots of  $\Phi_{\mathbf{t}, \mathbf{c}}(x_1, \dots, x_s)$  we assume in the conjecture (and guarantee in the construction) that  $\mathbf{t}, \mathbf{c}$  only have even entries. In Appendix A.1 we prove that, having done so,  $\Phi_{\mathbf{t}, \mathbf{c}}$  has no root in  $\{0, 1\}^s$ . That is, when considering  $\mathbf{t}, \mathbf{c}$  with even entries,  $\Phi_{\mathbf{t}, \mathbf{c}}(x_1, \dots, x_s) \neq 0$  for every  $(x_1, \dots, x_s) \in \{0, 1\}^s$ , and so it is only the reduction modulo  $p$  that may yield roots. With this, we are finally ready to state our conjecture.

► **Conjecture 4** (The Pascal determinant cubes (PDC) conjecture). *There exists a universal constant  $e_p \geq 1$  such that for every integer  $n \geq 1$  and prime  $p \geq n^{e_p}$  the following holds. For every  $s \in [n]$  and a pair of strictly increasing sequences  $\mathbf{t} = (t_1, \dots, t_s), \mathbf{c} = (c_1, \dots, c_s) \in ([0, n] \cap 2\mathbb{Z})^s$  satisfying  $\mathbf{c} \leq \mathbf{t}$ ,  $\exists (x_1, \dots, x_s) \in \{0, 1\}^s \quad \Phi_{\mathbf{t}, \mathbf{c}}^p(x_1, \dots, x_s) \neq 0$ .*

### 3.2.3 Experiments Supporting Conjecture 4

To support Conjecture 4 and, more fundamentally, to verify that there is no “structure” obstructing our heuristic arguments, we ran a computer search. Let  $\mathcal{P}_2 : \mathbb{N} \rightarrow \mathbb{N}$  be the function that maps  $n \in \mathbb{N}$  to the least prime  $p$  that satisfies the following property. For every  $s \in [n]$ , and every pair of strictly increasing sequences  $\mathbf{t} = (t_1, \dots, t_s), \mathbf{c} = (c_1, \dots, c_s) \in ([0, n] \cap 2\mathbb{Z})^s$  satisfying  $\mathbf{c} \leq \mathbf{t}$ , it holds that  $\Phi_{\mathbf{t}, \mathbf{c}}^p(x_1, \dots, x_s) \neq 0$  for some  $(x_1, \dots, x_s) \in \{0, 1\}^s$ . Informally,  $\mathcal{P}_2$  maps  $n$  to the least prime that is “good” for  $n$  in our conjecture. In comparison with  $\mathcal{P}_1$ , for every  $\mathbf{t}, \mathbf{c}$  in question,  $\mathcal{P}_1$  provides  $\Phi_{\mathbf{t}, \mathbf{c}}^p$  a single trial by evaluating it over the origin, while  $\mathcal{P}_2$  evaluates it over the entire Boolean hypercube of dimension  $s$ , and accepts the smallest prime that for every such  $\mathbf{t}, \mathbf{c}$ ,  $\Phi_{\mathbf{t}, \mathbf{c}}^p$  doesn’t vanish on at least one of its points.

An exhaustive search we have conducted, spanned over hundreds of computer hours, verifies at least for small numbers, that unlike  $\mathcal{P}_1(n)$ , the function  $\mathcal{P}_2(n)$  grows very slowly with  $n$ . In fact, the data collected in Table 2 shows that for  $7 \leq n \leq 30$ ,  $\mathcal{P}_2(n)$  equals the least prime number  $p \geq n - 1$ .<sup>1</sup>

<sup>1</sup> This is tight, namely, for every  $n \geq 7$ ,  $\mathcal{P}_2(n) \geq n - 1$ . Indeed, take  $p < n - 1$  a prime. If  $p \geq 5$ , consider

■ **Table 2** Values of  $\mathcal{P}_2(n)$  obtained using a computer search. Note that for an even  $n$ ,  $\mathcal{P}_2(n) = \mathcal{P}_2(n - 1)$  as  $\mathbf{t}, \mathbf{c}$  have even entries. Thus, only the data of odd  $n$ 's is collected.

$n$	5	7	9	11	13	15	17	19	21	23	25	27	29
$\mathcal{P}_2(n)$	3	7	11	11	13	17	17	19	23	23	29	29	29

We do not expect  $\mathcal{P}_2(n)$  to grow so slowly and we certainly do not expect it to have such a simple formula. While we could not compute  $\mathcal{P}_2(n)$  for  $n > 29$ , we were able to show that  $\mathcal{P}_2(127) > 131$  by eliminating the first two “potential” primes 127, 131. To see that, say,  $\mathcal{P}_2(127) \neq 131$  we invite the diligent reader to verify that  $\mathbf{c} = (0, 4, 10)$ ,  $\mathbf{t} = (64, 68, 74)$  yields a counterexample. That is,

$$\left| \begin{pmatrix} 64 + x_1 \\ 0 \\ 68 + x_2 \\ 0 \\ 74 + x_3 \\ 0 \end{pmatrix} \begin{pmatrix} 64 + x_1 \\ 4 \\ 68 + x_2 \\ 4 \\ 74 + x_3 \\ 4 \end{pmatrix} \begin{pmatrix} 64 + x_1 \\ 10 \\ 68 + x_2 \\ 10 \\ 74 + x_3 \\ 10 \end{pmatrix} \right| \equiv_{131} 0$$

for every  $(x_1, x_2, x_3) \in \{0, 1\}^3$ .

### 3.2.4 Asymptotic Version of Conjecture 4

For the informal heuristic argument used in Section 3.2.1 the point made is that while the number of “tests”  $(\mathbf{t}, \mathbf{c})$  grows exponentially with  $s$ , so does the number of “trials”  $(x_1, \dots, x_s)$ . Thus, when considering such a heuristic,  $s$  is thought of as an asymptotic parameter. However, Conjecture 4 is stated for every  $s \geq 1$ . While it may very well be the case that our conjecture holds as is, we prefer to base our construction on a more robust conjecture that avoids the possible “irregularities” that may be present for small values of  $s$ .

A natural relaxation is to bound  $s$  from below by some parameter  $s_0$  that is may even be allowed to grow with  $n$ . However, note that this should be done with some care. Indeed, if Conjecture 4 can be falsified for some value  $s$ , it is immediately false for larger values of  $s$ . To see this, take the counterexample  $\mathbf{c} = (c_1, \dots, c_s)$ ,  $\mathbf{t} = (t_1, \dots, t_s) \in [0, n]^s$  and consider  $\mathbf{c}' = (c_1, \dots, c_s, c_{s+1})$ ,  $\mathbf{t}' = (t_1, \dots, t_s, t_{s+1}) \in [0, n]^s$  where  $c_{s+1}, t_{s+1}$  are chosen so that  $t_s < c_{s+1} \leq t_{s+1}$ . Observe that this has the effect of “embedding”  $M_{\mathbf{t}, \mathbf{c}}(X_1, \dots, X_s)$  as the top-left sub matrix of  $M_{\mathbf{t}', \mathbf{c}'}(X_1, \dots, X_{s+1})$ . Furthermore, all but the lowest entry of the rightmost column are 0. In particular,

$$\Phi_{\mathbf{t}', \mathbf{c}'}(X_1, \dots, X_{s+1}) = \begin{pmatrix} t_{s+1} + X_{s+1} \\ c_{s+1} \end{pmatrix} \cdot \Phi_{\mathbf{t}, \mathbf{c}}(X_1, \dots, X_s).$$

Thus, if  $\Phi_{\mathbf{t}, \mathbf{c}}$  vanishes on  $\{0, 1\}^s$  then  $\Phi_{\mathbf{t}', \mathbf{c}'}$  vanishes on  $\{0, 1\}^{s+1}$ .

The “correct” way of formalizing a relaxation of Conjecture 4 in which only sufficiently large  $s$  are of interest is to restrict to pairs  $\mathbf{t}, \mathbf{c}$  for which not only  $s \geq s_0$  but also  $t_1 \geq c_{s_0}$ . Observe that under this condition, counterexamples of size less than  $s_0$  cannot be embedded as in the above discussion. We state below a variant of Conjecture 4 on which our candidate constructions rely. However, when discussing our conjecture we do not distinguish between Conjecture 4 and Conjecture 5 unless such a distinction is essential.

---

the sequences  $\mathbf{t} = (0, p + 1)$  and  $\mathbf{c} = (0, 4)$ . Note that  $\Phi_{\mathbf{t}, \mathbf{c}}^p(x_1, x_2) = \binom{p+1+x_2}{4}$ , and that  $p$  divides both  $\binom{p+1}{4}$  and  $\binom{p+2}{4}$ . For  $p = 2, 3$  one can use  $\mathbf{t} = (0, 2p)$ ,  $\mathbf{c} = (0, 2)$ . By Table 2, the assertion is false for  $n < 7$ .

► **Conjecture 5** (Asymptotic PDC conjecture). *There exist universal constants  $e_p, e_s \geq 1$  such that for every integer  $n \geq 1$  and prime  $p \geq n^{e_p}$  the following holds. For every  $s \geq s_0 \triangleq (\log n)^{e_s}$  and every pair of strictly increasing sequences  $\mathbf{t} = (t_1, \dots, t_s), \mathbf{c} = (c_1, \dots, c_s) \in ([0, n) \cap 2\mathbb{Z})^s$  satisfying  $\mathbf{t} \geq \mathbf{c}$  and  $t_1 \geq c_{s_0}$ , it holds that  $\exists (x_1, \dots, x_s) \in \{0, 1\}^s$   $\Phi_{\mathbf{t}, \mathbf{c}}^p(x_1, \dots, x_s) \neq 0$ .*

We will overcome this relaxation with the aid of the explicit tree code by Gelles *et al.* [12], which will provide some structure to the polynomials we need to analyze to prove the correctness of our construction.

### 3.2.5 Structural Factors of $\Phi_{\mathbf{t}, \mathbf{c}}$ and Its Linearization

Conjecture 4 only concerns with the evaluation of  $\Phi_{\mathbf{t}, \mathbf{c}}$  at the Boolean hypercube which, recall, we prove never vanishes in Appendix A.1. But, as defined,  $\Phi_{\mathbf{t}, \mathbf{c}}$  does not encode this in any way. In this section, we identify and remove certain factors of  $\Phi_{\mathbf{t}, \mathbf{c}}$  that are, in a sense, “outside” the Boolean hypercube, and so are of no interest to us.

For sequences  $\mathbf{t}, \mathbf{c}$  as in Conjecture 4, consider the matrix  $M_{\mathbf{t}, \mathbf{c}}(X_1, \dots, X_s)$ . Take distinct  $i, j \in [s]$  with  $i > j$ . The substitution  $X_i = X_j + t_j - t_i$  turns the  $i^{\text{th}}$  and  $j^{\text{th}}$  rows identical, resulting in an identically zero determinant. By Hilbert’s Nullstellensatz,  $X_i - X_j + t_i - t_j$  divides  $\Phi_{\mathbf{t}, \mathbf{c}}(X_1, \dots, X_s)$  in  $\mathbb{Q}[X_1, \dots, X_s]$ . Therefore the determinant polynomial is of the form

$$\Phi_{\mathbf{t}, \mathbf{c}}(X_1, \dots, X_s) = \Xi_{\mathbf{t}, \mathbf{c}}(X_1, \dots, X_s) \cdot \prod_{i>j} (X_i - X_j + t_i - t_j)$$

for some polynomial  $\Xi_{\mathbf{t}, \mathbf{c}}[X_1, \dots, X_s] \in \mathbb{Q}[X_1, \dots, X_s]$ . In fact, by Gauss’s lemma for GCD domains,  $\Xi_{\mathbf{t}, \mathbf{c}}[X_1, \dots, X_s]$  is in  $\mathbb{Z}[X_1, \dots, X_s]$  since  $\Phi_{\mathbf{t}, \mathbf{c}}$  and the structural factor are both primitive. Thus, we can consider reduction modulo a prime  $p$ . Since  $t_i, t_j$  are distinct even numbers in  $[0, n)$ , the structural factors do not vanish at any point of the Boolean hypercube, even when reduced modulo a prime  $p > n$ . Therefore, studying the zeros of  $\Phi_{\mathbf{t}, \mathbf{c}}^p$  in the Boolean hypercube is equivalent to studying those of  $\Xi_{\mathbf{t}, \mathbf{c}}$ , even modulo a prime  $p > n$ .

Observe that the linearization of the univariate polynomial  $\binom{X+t}{c}$ , for  $c \geq 1$  takes the nice form  $\binom{X+t}{c} = \binom{t}{c-1}X + \binom{t}{c}$  as can be seen using Pascal’s identity. In Appendix A.2 we take these ideas a step further and obtain a reformulation of Conjecture 4 which, informally, states that a certain polynomial  $\Psi_{\mathbf{t}, \mathbf{c}}^p$  is nonzero (as an element of the ring  $\mathbb{F}_p[X_1, \dots, X_s]$ ). That is to say, while the [7] tree code fails over  $\mathbb{F}_p$  if a certain polynomial has a root at the origin, via its reformulation, Conjecture 4 is false only if a certain polynomial is the zero polynomial. An asymptotic version, equivalent to Conjecture 5 is immediate.

In Appendix B we suggest a stronger variant of Conjecture 4 and further study the plausibility of Conjecture 4 and its stronger variant based on deep results in arithmetic geometry. In particular, we reason about the distribution of values attain by  $\Phi_{\mathbf{t}, \mathbf{c}}$  on the Boolean hypercube by considering the exponential sum  $\sum_{(x_1, \dots, x_s) \in \{0, 1\}^s} \zeta_p^{\Phi_{\mathbf{t}, \mathbf{c}}(x_1, \dots, x_s)}$ , where  $\zeta_p$  is a  $p^{\text{th}}$  root of unity in  $\mathbb{C}$ , and collect computational data that appear in a longer version of the paper [3]. However, we wrap up this preliminary discussion on our conjecture and its variants. In the next section we go back to the problem of constructing tree codes, and give an informal presentation of our construction and its analysis, based on Conjecture 4 or, more precisely, based on the asymptotic variant, Conjecture 5.



### 3.3 The Candidate Tree Code

Our candidate construction is a variant of the construction discussed in Section 2. In fact, for obtaining distance larger than  $\frac{1}{2}$ , [7] modified their original construction so that at time  $t$ , not one but some  $r \geq 1$  number of evaluations of the “current” polynomial  $f_t$  is recorded. This enabled them to achieve distance  $1 - \frac{1}{r+1}$ . Our candidate construction is closely related to that variant. We make use of this idea of multiple evaluations not for improving the distance, but rather for relaxing the analysis so that it is plausible that the reduction modulo a small prime  $p$  yields non-vanishing distance and, in particular, follows by Conjecture 5.

Recall, however, that Conjecture 5 holds only for pairs of length  $s \geq s_0$  for which  $t_1 \geq c_{s_0}$ . Therefore, we need to introduce some mechanism to the construction so that its correctness does not rely on the behaviour when applied with small values of  $s$  (nor on invalid pairs). To this end, we make use of an explicit tree code construction by [12]. For every  $n \geq 1$ , an explicit tree code  $\text{TC}' : [n^2]^n \rightarrow [2n^2]^n$  having distance  $\frac{1}{\log n}$  is given (see Corollary 11 for a precise statement). Although  $\text{TC}'$  has a vanishing distance, it suffices for our needs as we will not use  $\text{TC}'$  directly for arguing about the distance; rather, we invoke  $\text{TC}'$  to guarantee some structure on the polynomials we need to analyze.

Take  $p > 2n^2$  a prime, and think of  $\text{TC}' : [n^2]^n \rightarrow \mathbb{F}_p^n$  in the natural way. Our construction proceeds as follows. Given  $m = (m_0, m_1, \dots, m_{n-1}) \in [n^2]^n$  we first apply  $\text{TC}'$  to obtain  $(\gamma_0, \gamma_2, \gamma_4, \dots, \gamma_{2(n-1)}) = \text{TC}'(m)$ . For  $t \in [0, n)$ , we define  $f_t(T) \in \mathbb{F}_p[T]$  by  $f_t(T) = \sum_{i=0}^t \gamma_{2i} \binom{T}{2i}$ . At time  $t \in [0, n)$ , our tree code  $\text{TC} : [n^2]^n \rightarrow (\mathbb{F}_p^3)^n$  outputs  $\text{TC}(m)_t = (\gamma_{2t}, f_t(2t), f_t(2t+1))$ . As mentioned, as the alphabet is of size  $\text{poly}(n)$ , standard techniques can then be used to obtain an explicit binary tree code with comparable parameters. We thus have,

► **Theorem 6.** *Assume that Conjecture 5 holds with parameters  $e_p, e_s$ . Then, there exist  $c = c(e_p, e_s) \in \mathbb{N}$  and  $\delta = \delta(e_p, e_s) \in (0, 1)$  such that the following holds. For every  $n \in \mathbb{N}$  there exists an explicit tree code  $\text{TC} : \{0, 1\}^n \rightarrow [c]^n$  with distance  $\delta$ .*

#### 3.3.1 Sketch of the Analysis

As for the analysis, consider distinct  $m = (m_0, m_1, \dots, m_{n-1})$ ,  $m' = (m'_0, m'_1, \dots, m'_{n-1})$ , and let  $c \in [0, n)$  be the least integer for which  $m_c \neq m'_c$ . By the property of  $\text{TC}'$  we get that for every  $\ell \in [0, n - c)$ , when restricted to  $[c, c + \ell]$ , the strings  $\gamma = \text{TC}'(m)$ ,  $\gamma' = \text{TC}'(m')$  are of distance  $s \geq \frac{\ell}{\log n}$ . In particular, when considering  $\ell \geq (\log n)^e$  for some constant  $e > 1$ , we have that  $s \geq (\log n)^{e-1}$ . Let us assume this bound on  $\ell$  for the moment. Observe now that, by construction,  $s$  is precisely the sparsity of the polynomial  $g(T) = f_{c+\ell}(T) - f'_{c+\ell}(T)$  with respect to the Newton basis. Thus, we can write  $g(T) = \sum_{j=1}^s \gamma''_{2c_j} \binom{T}{2c_j}$ , where  $c = c_1 < c_2 < \dots < c_s \leq c + \ell < n$  and  $\gamma''_{2c_j} = \gamma_{2c_j} - \gamma'_{2c_j}$ .

We wish to bound the number of integers  $t \in [c, c + \ell]$  for which  $g(2t) = g(2t+1) = 0$  as indeed for every such  $t$ ,  $\text{TC}(m)_t$  and  $\text{TC}(m')_t$  agree when projected to the last two entries of the triplet. To get a bound of  $b$  on such indices  $t$ , the natural approach is to assume the existence of some  $t_1 < t_2 < \dots < t_b$  in  $[c, c + \ell]$  with  $g(2t_i) = g(2t_i + 1) = 0$  for every  $i \in [b]$ , and try to get a contradiction via Conjecture 5 for a sufficiently large value  $b$ . Recall, however, that for the conjecture it is required that  $\mathbf{t} \geq \mathbf{c}$  which is not necessarily the case. In [7] this technical issue is resolved by observing that one can restrict to the longest prefixes  $(c_1, c_2, \dots, c_{s_1})$ ,  $(t_1, t_2, \dots, t_{s_1})$  of the original sequences for which  $c_i \leq t_i$  for every  $i \in [s_1]$ . Such  $s_1$  exists as  $c_1 \leq t_1$ .

Our analysis is somewhat trickier as we can only invoke Conjecture 5 starting from some  $s_0$  (and under some restriction on the pair). In particular, in the notation of Conjecture 5, we have  $s_0 = (\log(2n))^{e_s}$ , and it may very well be the case that the longest prefix length

## 54:12 Candidate Tree Codes via Pascal Determinant Cubes

$s_1 < s_0$ . To overcome this, and to satisfy the hypothesis of Conjecture 5, we first prove a bound of  $s$  on the number of  $t_i$ 's in  $[c_{s_0}, c + \ell]$  rather than in  $[c, c + \ell]$ . This can be done based on Conjecture 5 using a similar argument to that of [7] who invoke the LGV Lemma.

To bound the number of the remaining  $t_i$ 's, namely, those in  $[c, c_{s_0}]$  we bound the length of this interval. Had  $c_1, \dots, c_{s_0}$  been arbitrary, the interval's length could have been unbounded. However, recall that by construction,  $c_1, \dots, c_{s_0}$  are the indices in  $[c, c_{s_0}]$  for which  $\text{TC}'(m), \text{TC}'(m')$  disagree. Since  $\text{TC}'$  has distance  $\frac{1}{\log n}$  it follows that  $s_0 \geq \frac{c_{s_0} - c}{\log n}$ , and so the interval's length is bounded by  $c_{s_0} - c \leq s_0 \log n \leq (\log(2n))^{e_s+1}$ . Hence, the total number of  $t_i$ 's is bounded by  $s + (\log(2n))^{e_s+1}$ , and so the distance between  $\text{TC}(m)$  and  $\text{TC}(m')$  when restricted to  $[c, c + \ell]$  is at least  $\max(s, \ell - (s + (\log(2n))^{e_s+1}))$ . By taking  $\ell$  sufficiently large, the latter approaches  $\frac{\ell}{3}$ .

In the discussion above, we assumed  $\ell$  is sufficiently large. In particular,  $\ell > \ell_0 = (\log n)^e$  for some constant  $e$ . To resolve this "lag", namely, to handle also smaller values of  $\ell$ , we use a standard technique in which an explicit tree code of length  $O(\ell_0)$  is concatenated with the construction above.

### 4 Preliminaries

Let  $n \geq 1$  be an integer and  $\Sigma$  some (finite or infinite) set. For a string  $x = (x_1, \dots, x_n) \in \Sigma^n$  and integers  $1 \leq a \leq b \leq n$ , we let  $x_{[a,b]}$  denote the substring  $(x_a, \dots, x_b)$ . Given  $x, y \in \Sigma^n$ , we write  $\text{dist}(x, y)$  for their Hamming distance. For an integer  $n \geq 1$  write  $[n]$  for  $\{1, 2, \dots, n\}$ . For integers  $a < b$  we denote  $[a, b) = \{a, a + 1, \dots, b - 1\}$ . We use the conventions that the natural numbers are  $\mathbb{N} = \{0, 1, 2, \dots\}$ , and that  $\binom{a}{b} = 0$  for integers  $0 \leq a < b$ .

Tree codes, as their name suggest, are trees with certain distance properties. However, as discussed in Section 2, we use an equivalent definition of tree codes that more explicitly specifies their online characteristic. Recall that a function  $f: \Sigma_{\text{in}}^n \rightarrow \Sigma_{\text{out}}^n$  is said to be *online* if for every  $i \in [n]$  and  $x \in \Sigma_{\text{in}}^n$ ,  $f(x)_i$  is determined by  $x_1, \dots, x_i$ . For a pair of distinct  $x, y \in \Sigma^n$ , we define  $\text{split}(x, y)$  as the least integer  $s \in [n]$  such that  $x_s \neq y_s$ .

► **Definition 7** ([23]). *An online function  $\text{TC}: \Sigma_{\text{in}}^n \rightarrow \Sigma_{\text{out}}^n$  is a tree code with distance  $\delta$  if for every distinct  $x, y \in \Sigma_{\text{in}}^n$ , with  $s = \text{split}(x, y)$ , and every  $\ell \in [0, n - s)$ ,*

$$\text{dist}(\text{TC}(x)_{[s, s+\ell]}, \text{TC}(y)_{[s, s+\ell]}) \geq \delta(\ell + 1).$$

*We refer to  $n$  as the depth of  $\text{TC}$ . We refer to  $\Sigma_{\text{in}}, \Sigma_{\text{out}}$  as the input alphabet and output alphabet, respectively.*

We are interested in some further properties of tree codes.

► **Definition 8.** *Let  $\text{TC}: \Sigma_{\text{in}}^n \rightarrow \Sigma_{\text{out}}^n$  be a tree code.*

- *We say that  $\text{TC}$  is a binary tree code if  $\Sigma_{\text{in}} = \{0, 1\}$ .*
- *We say that  $\text{TC}$  is explicit if it can be evaluated on every input  $m \in \Sigma_{\text{in}}^n$  in polynomial time in the bit complexity of  $m$ .*

### 5 Proof of Theorem 6

In this section we present our candidate tree code and prove Theorem 6. Our construction is obtained in several steps, where the main part is to construct a relaxation of tree codes, called a lagged tree code. Informally, this is a tree code whose distance property holds only after a certain time interval.

► **Definition 9** ([7]). *An online function  $\text{TC} : \Sigma_{\text{in}}^n \rightarrow \Sigma_{\text{out}}^n$  is a lagged tree code with lag  $\ell_0$  and distance  $\delta$  if for every distinct  $x, y \in \Sigma_{\text{in}}^n$ , with  $s = \text{split}(x, y)$ , and every  $\ell \in [\ell_0, n - s]$ ,*

$$\text{dist}(\text{TC}(x)_{[s, s+\ell]}, \text{TC}(y)_{[s, s+\ell]}) \geq \delta(\ell + 1).$$

Note that a tree code is a lagged tree code with lag parameter  $\ell_0 = 0$ . It is straightforward to transform any lag- $\ell_0$  tree code to a tree code using a second tree code of length  $O(\ell_0)$ . Our construction of lagged tree codes, given below by Proposition 12, has lag  $\ell_0 = \text{poly}(\log n)$ . A result by Braverman [6] provides, for every constant  $\varepsilon \in (0, 1)$  and integer  $m$  an asymptotically-good tree code of length  $m$  in time  $2^{O(m^\varepsilon)}$ . Thus, asymptotically-good tree codes of length  $\ell_0$  can be obtained in time  $\text{poly}(n)$ . The obtained tree code (as well as the lagged tree code that is given by Proposition 12) is over a  $\text{poly}(n)$ -size alphabet. It is well-known how to reduce the alphabet to binary, obtaining tree codes with comparable parameters (see, e.g., [21], Proposition 3.1).

In light of the discussion above, we turn to present our candidate construction of  $\text{poly}(\log n)$ -lagged tree codes over  $\text{poly}(n)$ -size alphabet. Our construction makes use of a tree code construction by [12].

► **Lemma 10** (Lemma 5.1 in [12]). *There exists an absolute constant  $k_0 \in \mathbb{N}$  such that the following hold for every  $\varepsilon > 0$  and integers  $k, n \in \mathbb{N}$  such that  $\frac{k_0 \cdot \log n}{\varepsilon} \leq k \leq n$ . There exists an explicit tree code  $C : \Sigma_{\text{in}}^k \rightarrow \Sigma_{\text{out}}^k$  with  $\Sigma_{\text{in}} = \{0, 1\}^{\frac{\log n}{\varepsilon}}$ ,  $\Sigma_{\text{out}} = \{0, 1\}^{\frac{\log n}{\varepsilon} + 1}$ , rate  $\rho' = \frac{1}{1 + \varepsilon / \log n}$  and relative distance at least  $\delta' = \frac{1}{1 + 2 \log(n) / \varepsilon}$ .*

The following is a straightforward corollary of Lemma 10 obtained by taking  $\varepsilon = \frac{1}{2}$ . Note that the factors of 4 and 8 in the alphabet size of  $\text{TC}'$  in Corollary 11 are for obtaining a tree code for every  $n$ , not just a power of two as in Lemma 10.

► **Corollary 11.** *There exists a universal constant  $n_0 \geq 1$  such that for every integer  $n \geq n_0$  there exists an explicit tree code  $\text{TC}' : [4n^2]^n \rightarrow [8n^2]^n$  with distance  $\delta = \frac{1}{5 \log n}$ .*

Given an integer  $n \geq n_0$  we proceed as follows. Let  $p$  be the least prime number larger than  $\max(8n^2, (2n)^{e_p})$ , where  $e_p$  is the constant from Conjecture 5. By Corollary 11, there exists an explicit tree code  $\text{TC}' : [4n^2]^n \rightarrow [8n^2]^n$  with distance  $\frac{1}{5 \log n}$ . As  $p > 8n^2$  we can embed the output symbols of  $\text{TC}'$  in  $\mathbb{F}_p$  by identifying them with the field elements  $1, \dots, 8n^2$  of  $\mathbb{F}_p$ . Hence, we may think of  $\text{TC}'$  as a function of the form  $\text{TC}' : [4n^2]^n \rightarrow \mathbb{F}_p^n$ .

Define the function  $\text{TC} : [4n^2]^n \rightarrow (\mathbb{F}_p^3)^n$  as follows. Let  $m = (m_0, m_1, \dots, m_{n-1}) \in [4n^2]^n$ . Compute  $\text{TC}'(m) = (\gamma_0, \gamma_2, \gamma_4, \dots, \gamma_{2(n-1)}) \in \mathbb{F}_p^n$ . For  $t = 0, 1, \dots, n-1$  define the polynomial  $f_t(T) \in \mathbb{F}_p[T]$  by

$$f_t(T) = \sum_{i=0}^t \gamma_{2i} \binom{T}{2i}. \tag{5.1}$$

Finally, for  $t = 0, 1, \dots, n-1$ , define

$$\text{TC}(m)_t = (\gamma_{2t}, f_t(2t), f_t(2t+1)). \tag{5.2}$$

► **Proposition 12.** *Assume that Conjecture 5 holds with parameters  $e_p, e_s$ . Then,  $\text{TC}$  as defined in Equation (5.2) is an  $\ell_0$ -lagged tree code, where  $\ell_0 = 15(\log(2n))^{e_s+1}$ , having distance  $\frac{1}{3}$  and rate at least  $\frac{1}{2 \max(2, e_p)}$ .*

**Proof.** That the rate is bounded below by  $\frac{1}{2 \max(2, e_p)}$  is a straightforward calculation. We turn to analyze the distance. Note that  $\text{TC}$  is not linear and so, for the distance analysis, we consider two distinct messages. Let  $m = (m_0, \dots, m_{n-1}) \in [4n^2]^n$  and  $m' = (m'_0, \dots, m'_{n-1}) \in [4n^2]^n$

## 54:14 Candidate Tree Codes via Pascal Determinant Cubes

distinct. Let  $0 \leq c \leq n-1$  be the least integer for which  $m_c \neq m'_c$ , and let  $\ell \in [\ell_0, n-c)$ . Denote

$$\begin{aligned}\gamma &= (\gamma_0, \gamma_2, \dots, \gamma_{2(n-1)}) = \text{TC}'(m), \\ \gamma' &= (\gamma'_0, \gamma'_2, \dots, \gamma'_{2(n-1)}) = \text{TC}'(m').\end{aligned}$$

Since  $\text{TC}'$  has distance  $\frac{1}{5 \log n}$  it holds that

$$\begin{aligned}s &\triangleq \text{dist} \left( \gamma_{[c, c+\ell]}, \gamma'_{[c, c+\ell]} \right) \\ &= \text{dist} \left( (\gamma_{2c}, \gamma_{2(c+1)}, \dots, \gamma_{2(c+\ell)}), (\gamma'_{2c}, \gamma'_{2(c+1)}, \dots, \gamma'_{2(c+\ell)}) \right) \\ &\geq \frac{\ell+1}{5 \log n}.\end{aligned}$$

As  $\ell \geq \ell_0$  we have that  $s > s_0$ , where  $s_0 \triangleq (\log(2n))^{e_s}$ . Similarly to Equation (5.1), we define for  $t = 0, 1, \dots, n-1$  the polynomial  $f'_t(T) \in \mathbb{F}_p[T]$  by

$$f'_t(T) = \sum_{i=0}^t \gamma'_{2i} \binom{T}{2i}.$$

Observe that  $s$  is precisely the sparsity of  $f_{c+\ell}(T) - f'_{c+\ell}(T)$  with respect to the Newton basis. Let  $c \leq c_1 < c_2 < \dots < c_s \leq c+\ell$  be all the integers such that  $\gamma_{2c_j} \neq \gamma'_{2c_j}$  for every  $j \in [s]$ . As  $\text{TC}'$  is a tree code (with nonzero distance)  $\gamma_{2c} = \text{TC}'(m)_c \neq \text{TC}'(m')_c = \gamma'_{2c}$ , and so  $c_1 = c$ . By denoting  $\gamma''_i = \gamma_i - \gamma'_i$ , one can write the polynomial  $f_{c+\ell}(T) - f'_{c+\ell}(T)$  as

$$g(T) = \sum_{j=1}^s \gamma''_{2c_j} \binom{T}{2c_j}.$$

Define  $Z = \{t \in [c, c+\ell] \mid g(2t) = g(2t+1) = 0\}$ .

▷ **Claim 13.** Assuming Conjecture 5,  $|Z \cap [c_{s_0}, c+\ell]| < s$ .

*Proof.* Assume by way of contradiction that there are distinct integers  $t_1, \dots, t_s \in [c_{s_0}, c+\ell]$  such that

$$\forall i \in [s] \quad g(2t_i) = g(2t_i+1) = 0. \quad (5.3)$$

Assume further that  $t_1 < \dots < t_s$ . Let  $s_1 \in \{s_0, s_0+1, \dots, s\}$  be the largest integer with the property that for every  $i \in \{s_0, s_0+1, \dots, s_1\}$ ,  $t_i \geq c_i$ . Note that  $s_1$  is well-defined as  $t_{s_0} \geq c_{s_0}$  (and so the maximum is taken over a non-empty, finite, set). Let  $M(X_1, \dots, X_{s_1})$  be the  $s_1 \times s_1$  matrix whose  $(i, j)^{\text{th}}$  entry is

$$M_{i,j}(X_1, \dots, X_{s_1}) = \begin{pmatrix} X_i + 2t_i \\ 2c_j \end{pmatrix},$$

where  $X_1, \dots, X_{s_1}$  are formal variables. Let  $\Phi \in \mathbb{F}_p[X_1, \dots, X_{s_1}]$  be the polynomial that is given by  $\Phi(X_1, \dots, X_{s_1}) = \det M(X_1, \dots, X_{s_1})$ . Denote  $\mathbf{t} = (2t_1, \dots, 2t_{s_1})$  and  $\mathbf{c} = (2c_1, \dots, 2c_{s_1})$ . Note that  $\Phi$  as defined above is precisely  $\Phi_{\mathbf{t}, \mathbf{c}}^p$  in the notation of Conjecture 5. Clearly,  $\mathbf{t}, \mathbf{c} \in ([0, 2n) \cap 2\mathbb{Z})^{s_1}$ . We turn to show that  $\mathbf{c} \leq \mathbf{t}$ . Indeed, for  $i \in \{s_0, s_0+1, \dots, s_1\}$  we have that  $t_i \geq c_i$  by the definition of  $s_1$ . Moreover, recall that for every  $i \in [s]$ ,  $t_i \geq c_{s_0}$ , and so, for  $i < s_0$  we have that  $t_i \geq c_{s_0} > c_i$ . Recall that  $p \geq (2n)^{e_p}$ ,  $s_1 \geq s_0 = (\log(2n))^{e_s}$ , and  $2t_1 \geq 2c_{s_0}$ . Thus, the hypothesis of Conjecture 5 is met with  $s, n$  in the notation

the conjecture taken to be  $s_1$  and  $2n$  in our notation, respectively. Therefore, assuming the validity of Conjecture 5 we conclude the existence of  $(x_1, \dots, x_{s_1}) \in \{0, 1\}^{s_1}$  such that  $\Phi(x_1, \dots, x_{s_1}) \neq 0$  in  $\mathbb{F}_p$ .

We now use  $(x_1, \dots, x_{s_1})$  to get a contradiction. Let  $\Gamma \in \mathbb{F}_p^{s_1}$  be the vector with  $i^{\text{th}}$  entry  $\Gamma_i = \gamma''_{2c_i}$ . Observe that  $\Gamma$  is a nonzero vector. To see this, consider its first entry  $\Gamma_1 = \gamma''_{2c_1} = \gamma'_{2c}$ . Recall that  $\gamma''_{2c} = \gamma_{2c} - \gamma'_{2c}$ . As  $\text{TC}'$  is a tree code (with distance larger than 0) and since  $m_c \neq m'_c$  we have that  $\gamma_{2c} = \text{TC}'(m)_c \neq \text{TC}'(m')_c = \gamma'_{2c}$ . Thus,  $\Gamma_1 \neq 0$ .

Since  $\Phi(x_1, \dots, x_{s_1}) \neq 0$  we have that  $M(x_1, \dots, x_{s_1})$  is nonsingular, and therefore  $M(x_1, \dots, x_{s_1})\Gamma$  is a nonzero vector. Let then  $i \in [s_1]$  be such that  $(M(x_1, \dots, x_{s_1})\Gamma)_i \neq 0$ . Note that

$$(M(x_1, \dots, x_{s_1})\Gamma)_i = \sum_{j=1}^{s_1} \gamma''_{2c_j} \binom{x_i + 2t_i}{2c_j}. \quad (5.4)$$

Assume for the moment that  $s_1 < s$ . As  $i \leq s_1$  we have that  $i < s$  and so we may refer to  $t_{i+1}$ . As  $x_i \in \{0, 1\}$ , we have that  $2t_i + x_i \leq 2t_i + 1 < 2t_{i+1}$ . Hence, as  $i \leq s_1$ ,  $2t_i + x_i < 2t_{s_1+1}$ . By the definition of  $s_1$  we have that  $t_{s_1+1} < c_{s_1+1}$ , and so  $2t_i + x_i < 2c_{s_1+1}$ . Hence,  $\binom{x_i + 2t_i}{2c_j} = 0$  for all  $j \in \{s_1 + 1, \dots, s\}$ . Thus,

$$\sum_{j=1}^{s_1} \gamma''_{2c_j} \binom{x_i + 2t_i}{2c_j} = \sum_{j=1}^s \gamma''_{2c_j} \binom{x_i + 2t_i}{2c_j} = g(2t_i + x_i). \quad (5.5)$$

Equation (5.5) trivially follows also when  $s_1 = s$ , and so it holds in general, namely, without any assumption on  $s_1$ . Equations (5.4) and (5.5) together imply that

$$g(2t_i + x_i) = (M(x_1, \dots, x_{s_1})\Gamma)_i \neq 0$$

which, as  $x_i \in \{0, 1\}$ , stands in contradiction to Equation (5.3), and thus proving the claim.  $\triangleleft$

$\triangleright$  **Claim 14.**  $|Z| \leq s + 5(\log(2n))^{e_s+1}$ .

*Proof.* As  $\text{TC}'$  is a tree code with distance  $\frac{1}{5 \log n}$ , we have that

$$\begin{aligned} s_0 &= \text{dist} \left( (\gamma_{2c_1}, \gamma_{2(c_1+1)}, \dots, \gamma_{2c_{s_0}}), (\gamma'_{2c_1}, \gamma'_{2(c_1+1)}, \dots, \gamma'_{2c_{s_0}}) \right) \\ &= \text{dist} \left( \text{TC}'(m)_{[c_1, c_{s_0}]}, \text{TC}'(m')_{[c_1, c_{s_0}]} \right) \\ &\geq \frac{c_{s_0} - c_1 + 1}{5 \log n} \\ &\geq \frac{c_{s_0} - c}{5 \log n}. \end{aligned}$$

Now,  $s_0 = (\log(2n))^{e_s}$ , and so

$$c_{s_0} - c \leq 5(\log n)(\log(2n))^{e_s} \leq 5(\log(2n))^{e_s+1}.$$

This, together with Claim 13, implies that

$$\begin{aligned} |Z| &\leq (c_{s_0} - c) + |Z \cap [c_{s_0}, c + \ell]| \\ &\leq s + 5(\log(2n))^{e_s+1}. \end{aligned} \quad \triangleleft$$

$\triangleright$  **Claim 15.** For every  $t \in [c, c + \ell]$  and  $x \in \{0, 1\}$ ,

$$g(2t + x) = f_t(2t + x) - f'_t(2t + x).$$

## 54:16 Candidate Tree Codes via Pascal Determinant Cubes

Proof. Recall that  $c_1, \dots, c_s$  are precisely the indices in  $[c, c + \ell]$  for which  $\gamma$  and  $\gamma'$  disagree. More precisely, for  $i \in [c, c + \ell]$ ,  $\gamma_{2i} \neq \gamma'_{2i}$  if and only if  $i \in \{c_1, \dots, c_s\}$ . Hence, for every  $t \in [c, c + \ell]$  and  $x \in \{0, 1\}$ ,

$$\begin{aligned} f_t(2t+x) - f'_t(2t+x) &= \sum_{i=0}^t (\gamma_{2i} - \gamma'_{2i}) \binom{2t+x}{2i} \\ &= \sum_{\substack{j \in [s] \\ c_j \leq t}} \gamma''_{2c_j} \binom{2t+x}{2c_j} \\ &= \sum_{j=1}^s \gamma''_{2c_j} \binom{2t+x}{2c_j} \\ &= g(2t+x), \end{aligned}$$

where the penultimate equality follows since  $\binom{2t+x}{2c_j} = 0$  for every  $j \in [s]$  for which  $c_j > t$ . Indeed, if  $c_j > t$  then  $2c_j \geq 2t + 2$  and so  $\binom{2t}{2c_j} = \binom{2t+1}{2c_j} = 0$ .  $\triangleleft$

By Claim 15,  $t \in Z$  if and only if the last two entries of  $\text{TC}(m)_t$ , namely,  $f_t(2t), f_t(2t+1)$ , agree with the corresponding entries,  $f'_t(2t), f'_t(2t+1)$ , of  $\text{TC}(m')_t$ . As the third entry of  $\text{TC}(m)$  and  $\text{TC}(m')$ , when restricted to  $[c, c + \ell]$ , disagree on exactly  $s$  indices, we have that the number of indices  $t \in [c, c + \ell]$  for which  $\text{TC}(m)_t \neq \text{TC}(m')_t$  (as a triplet) is bounded below by

$$\begin{aligned} \max(s, \ell + 1 - |Z|) &\geq \max(s, \ell + 1 - s - 5(\log(2n))^{e_s+1}) \\ &\geq \frac{\ell - 5(\log(2n))^{e_s+1} + 1}{2} \\ &\geq \frac{\ell + 1}{3}, \end{aligned}$$

where the last inequality follows since  $\ell \geq \ell_0 = 15(\log(2n))^{e_s+1}$ .  $\blacktriangleleft$

---

### References

- 1 Martin Aigner. *A course in enumeration*, volume 238 of *Graduate Texts in Mathematics*. Springer, Berlin, 2007. doi:10.1145/1814370.1814375.
- 2 Martin Aigner and Günter M. Ziegler. *Proofs from The Book*. Springer, Berlin, sixth edition, 2018. See corrected reprint of the 1998 original [MR1723092], Including illustrations by Karl H. Hofmann. doi:10.1007/978-3-662-57265-8.
- 3 Inbar Ben Yaacov, Gil Cohen, and Anand Kumar Narayanan. Candidate tree codes via Pascal determinant cubes. *ECCC*, 2020. URL: <https://eccc.weizmann.ac.il/report/2020/141/>.
- 4 Siddharth Bhandari and Prahladh Harsha. A note on the explicit constructions of tree codes over polylogarithmic-sized alphabet. *arXiv preprint arXiv:2002.08231*, 2020. URL: <https://arxiv.org/abs/2002.08231>.
- 5 Zvika Brakerski, Yael Tauman Kalai, and Raghuvansh R. Saxena. Deterministic and efficient interactive coding from hard-to-decode tree codes. In *61st Annual IEEE Symposium on Foundations of Computer Science—FOCS 2020*, pages 446–457. IEEE Computer Soc., Los Alamitos, CA, 2020. doi:10.1109/FOCS46700.2020.00049.
- 6 Mark Braverman. Towards deterministic tree code constructions. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 161–167. ACM, New York, 2012. doi:10.1145/2090236.2090250.

- 7 Gil Cohen, Bernhard Haeupler, and Leonard J. Schulman. Explicit binary tree codes with polylogarithmic size alphabet. In *STOC'18—Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 535–544. ACM, New York, 2018. doi:10.1145/3188745.3188928.
- 8 Gil Cohen and Shahar Samocha. Palette-alternating tree codes. In *The 35th Computational Complexity Conference (CCC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CCC.2020.11.
- 9 Pierre Deligne. La conjecture de weil : I. *Publications Mathématiques de l’IHÉS*, 43:273–307, 1974. doi:10.1007/bf02684373.
- 10 Étienne Fouvry. Consequences of a result of N. Katz and G. Laumon concerning trigonometric sums. *Israel Journal of Mathematics*, 120:81–96, 2000. doi:10.1007/s11856-000-1272-z.
- 11 Étienne Fouvry and Nicholas Katz. A general stratification theorem for exponential sums, and applications. *Journal Fur Die Reine Und Angewandte Mathematik - J REINE ANGEW MATH*, 2001:115–166, January 2001. doi:10.1515/crll.2001.082.
- 12 Ran Gelles, Bernhard Haeupler, Gillat Kol, Noga Ron-Zewi, and Avi Wigderson. Towards optimal deterministic coding for interactive communication. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1922–1936. ACM, New York, 2016. doi:10.1137/1.9781611974331.ch135.
- 13 Ira Gessel and Gérard Viennot. Binomial determinants, paths, and hook length formulae. *Adv. in Math.*, 58(3):300–321, 1985. doi:10.1016/0001-8708(85)90121-5.
- 14 Richard W. Hamming. Error detecting and error correcting codes. *Bell System Tech. J.*, 29:147–160, 1950. doi:10.1002/j.1538-7305.1950.tb00463.x.
- 15 Jørn Justesen. Class of constructive asymptotically good algebraic codes. *IEEE Transactions on Information Theory*, 18(5):652–656, 1972. doi:10.1109/tit.1972.1054893.
- 16 Sankeerth R. Karingula and Shachar Lovett. Codes over integers, and the singularity of random matrices with large entries. *CoRR*, abs/2010.12081, 2020. URL: <https://arxiv.org/abs/2010.12081>.
- 17 Nicholas Katz and Gérard Laumon. Transformation de fourier et majoration de sommes exponentielles. *Publications Mathématiques de l’IHÉS*, 62:145–202, 1985. doi:10.1007/bf02698808.
- 18 Serge Lang and Andre Weil. Number of points of varieties over finite fields. *American Journal of Mathematics*, 76(4):819–827, 1954. URL: <https://www.jstor.org/stable/2372655>.
- 19 Cristopher Moore and Leonard J. Schulman. Tree codes and a conjecture on exponential sums. In *ITCS'14—Proceedings of the 2014 Conference on Innovations in Theoretical Computer Science*, pages 145–153. ACM, New York, 2014. doi:10.1145/2554797.2554813.
- 20 Anand Kumar Narayanan and Matthew Weidner. On decoding Cohen-Haeupler-Schulman tree codes. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1337–1356. SIAM, 2020. doi:10.1137/1.9781611975994.81.
- 21 Pavel Pudlák. Linear tree codes and the problem of explicit constructions. *Linear Algebra Appl.*, 490:124–144, 2016. doi:10.1016/j.laa.2015.10.030.
- 22 Pavel Pudlák. On matrices potentially useful for tree codes. *CoRR*, abs/2012.03013, 2020. URL: <https://arxiv.org/abs/2012.03013>.
- 23 Leonard J. Schulman. Deterministic coding for interactive communication. In *Proceedings of the 25th annual ACM Symposium on Theory of Computing*, pages 747–756, 1993. doi:10.1145/167088.167279.
- 24 Leonard J. Schulman. Postscript of 21 september 2003 to coding for interactive communication. <http://users.cms.caltech.edu/~schulman/Papers/intercodingpostscript.txt>, 1994.
- 25 Leonard J. Schulman. Coding for interactive communication. *IEEE Trans. Inform. Theory*, 42(6, part 1):1745–1756, 1996. Codes and complexity. doi:10.1109/18.556671.
- 26 Claude E. Shannon. A mathematical theory of communication. *Bell System Tech. J.*, 27:379–423, 623–656, 1948. doi:10.1002/j.1538-7305.1948.tb01338.x.

- 27 Michael Sipser and Daniel A. Spielman. Expander codes. *IEEE Transactions on Information Theory*, 42(6):1710–1722, 1996. doi:10.1109/18.556667.
- 28 Michael A. Tsfasman, Serge G. Vlăduț, and Thomas Zink. Modular curves, Shimura curves, and Goppa codes, better than Varshamov-Gilbert bound. *Mathematische Nachrichten*, 109(1):21–28, 1982. doi:10.1002/mana.19821090103.

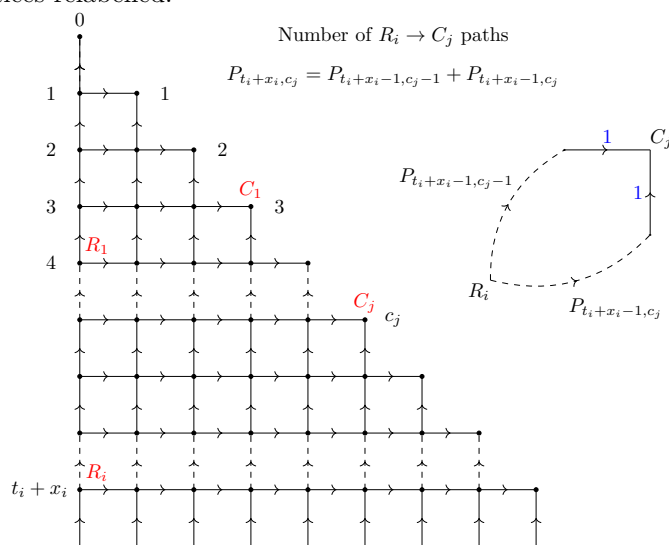
## A Combinatorics Corroborating Conjecture 4

### A.1 Non-Vanishing of $\Phi_{t,c}$ on the Boolean Hypercube

In this section we prove that the integer polynomial  $\Phi_{t,c}$  as in Conjecture 4 does not vanish on any point of the Boolean hypercube. To this end, we make use of ideas similar to those used by [7] to prove that  $\Phi_{t,c}$  has no root at the origin. Fix sequences  $t, c$  as in Conjecture 4 for the remainder of this section. Consider a directed acyclic graph  $G = (V, E)$  with edge weights  $\{w(e) \mid e \in E\}$  coming from a commutative ring with identity, along with two ordered vertex sets  $\mathcal{R} = \{R_1, R_2, \dots, R_d\}, \mathcal{C} = \{C_1, C_2, \dots, C_d\} \subseteq V$  of the same cardinality  $d$ . Associated to it is the path matrix  $M$ : the square matrix indexed by  $R, C$  with the  $R \in \mathcal{R}, C \in \mathcal{C}$  entry  $M_{R,C} \triangleq \prod_{P: R \rightarrow C} w(P)$  where the product is taken over all paths  $P$  from  $R$  to  $C$  and the weight  $w(P)$  is the product of edge weights in the path  $P$ . Paths of length 0 are included and given the weight 1. A path system  $\mathcal{P}$  from  $\mathcal{R}$  to  $\mathcal{C}$  consists of a permutation  $\sigma \in S_d$  and a set of paths  $\{P_i : R_i \rightarrow C_{\sigma(i)} \mid i \in [d]\}$ . Let  $\text{sgn}(\mathcal{P})$  denote the sign of  $\sigma$  and  $w(\mathcal{P})$  denote the product of the weights  $\prod_{i=1}^d w(P_i)$ . The path system is called vertex disjoint if its set of paths are vertex disjoint. The LGV Lemma is the expression for the determinant of the path matrix  $M$  in terms of the underlying path graph

$$\det(M) = \sum_{\substack{\text{vertex disjoint} \\ \text{path systems } \mathcal{P}}} \text{sgn}(\mathcal{P})w(\mathcal{P}).$$

Gessel and Viennot applied it to path graphs cut out from the square lattice and proved the non vanishing theorem for determinants of Pascal submatrices. We next show a non vanishing of determinants central to our construction using the same path graph but with vertices relabelled.





► **Lemma 16.** For all strictly increasing non negative integer sequences  $\mathbf{c} = (c_1, \dots, c_s), \mathbf{t} = (t_1, \dots, t_s)$  such that  $\mathbf{c} \leq \mathbf{t}$  and  $t_i$  is even for all  $i \in [s]$ , and  $\forall (x_1, x_2, \dots, x_s) \in \{0, 1\}^s$ ,

$$\Phi_{\mathbf{t}, \mathbf{c}}(x_1, \dots, x_s) \neq 0.$$

**Proof.** Fix numbers  $c_1, \dots, c_s, t_1, \dots, t_s, x_1, \dots, x_s$  as in the statement. Consider the directed acyclic graph below with unit edge weights and distinguished (in red) vertex subsets  $\{R_1, R_2, \dots, R_s\}$  and  $\{C_1, C_2, \dots, C_s\}$ . The  $(t_1 + x_1)^{th}$  vertex on the first column is labelled  $R_1$ , the  $(t_2 + x_2)^{th}$  vertex on the first column is labelled  $R_2$  and so on. The labels  $R_i$ s are well defined, for  $(t_i + x_i)$ s are distinct as  $t_i$ s are even and  $x_i$ s are in  $\{0, 1\}$ . The  $c_1^{th}$  vertex on the diagonal is labelled  $C_1$ , the  $c_2^{th}$  vertex on the diagonal is labelled  $C_2$  and so on. To illustrate,  $t_1 = 4, x_1 = 0, c_1 = 3$  in the diagram. The horizontal edges are directed from left to right and the vertical edges from bottom to top.

Since all the edge weights are 1, the  $(i, j)^{th}$  entry  $M_{i,j}$  of the path matrix is the number of paths  $P_{t_i+x_i, c_j}$  from  $R_i$  to  $C_j$ . This satisfies the two term recurrence  $P_{t_i+x_i, c_j} = P_{t_i+x_i-1, c_j-1} + P_{t_i+x_i-1, c_j}$  as evident from the picture on the right. This is Pascal's identity for binomials. The boundary conditions  $\binom{t_i+x_i}{0} = 1$  and  $\binom{t_i+x_i}{c_i} = 1$  for  $t_i + x_i = c_i$  are consistent with the path formulation. We conclude that the associated path matrix is  $M_{\mathbf{t}, \mathbf{c}} = \left\{ \binom{t_i+x_i}{c_j} \mid i, j \in [s] \right\}$ , whose determinant  $\Phi_{\mathbf{t}, \mathbf{c}}(x_1, \dots, x_s)$  is in question. The planar geometry forces all vertex disjoint path systems to have the identity permutation, which has sign 1. Hence the determinant is a positive number provided there is at least one vertex disjoint path system. By the condition  $t_i + x_i \geq c_i$  for all  $i$ , there is at least one, namely for each  $R_i \rightarrow C_i$ , traverse  $c_i$  edges right before turning up. ◀

## A.2 Reformulation of Conjecture 4

In this section we provide a reformulation of Conjecture 4. Fix sequences  $\mathbf{t}, \mathbf{c}$  as in Conjecture 4. Consider the variety  $X_{\mathbf{t}, \mathbf{c}}$  of intersection of the hypercube and the hypersurface generated by  $\Phi_{\mathbf{t}, \mathbf{c}}$ . The variety  $X_{\mathbf{t}, \mathbf{c}}$  is generated by the ideal

$$\mathcal{I}_{\mathbf{t}, \mathbf{c}} := \langle \Phi_{\mathbf{t}, \mathbf{c}}(X_1, X_2, \dots, X_s), X_1^2 - X_1, \dots, X_s^2 - X_s \rangle.$$

Clearly, the intersection variety is zero dimensional (or empty), since the hypercube is zero dimensional and  $\Phi_{\mathbf{t}, \mathbf{c}}$  is nonzero. The degree of the polynomial defining the hypersurface can be reduced through the relations carving out the hypercube as follows. Let  $\Psi_{\mathbf{t}, \mathbf{c}}(X_1, X_2, \dots, X_s) \in \mathbb{Z}[X_1, X_2, \dots, X_s]$  be the unique lift of

$$\Phi_{\mathbf{t}, \mathbf{c}}[X_1, X_2, \dots, X_s] \text{ mod } \langle X_1^2 - X_1, X_2^2 - X_2, \dots, X_s^2 - X_s \rangle$$

with degree in each variable at most 1. Informally,  $\Psi_{\mathbf{t}, \mathbf{c}}$  is merely  $\Phi_{\mathbf{t}, \mathbf{c}}$  with every indeterminate  $X_i^*$  replaced by  $X_i$ . The  $*$  in the superscript denotes some positive exponent. Since  $\Phi_{\mathbf{t}, \mathbf{c}}$  is nonzero, so is  $\Psi_{\mathbf{t}, \mathbf{c}}$ . The respective hypersurfaces generated by  $\Phi_{\mathbf{t}, \mathbf{c}}$  and  $\Psi_{\mathbf{t}, \mathbf{c}}$  have the same intersection with the Boolean hypercube and hence we can work with either. We will proceed with  $\Psi_{\mathbf{t}, \mathbf{c}}$  as it has the form

$$\Psi_{\mathbf{t}, \mathbf{c}}(X_1, X_2, \dots, X_s) = \sum_{b=(b_1, b_2, \dots, b_s) \in \{0, 1\}^s} a_b X_1^{b_1} X_2^{b_2} \dots X_s^{b_s}$$

familiar to Boolean functional analysts with possibly smaller degrees. Further, restricting to the Boolean cube removed the structural factors that concerned us in Section 3.2.5 from  $\Psi_{\mathbf{t}, \mathbf{c}}$ . Let  $\Psi_{\mathbf{t}, \mathbf{c}}^p \in \mathbb{F}_p[X_1, X_2, \dots, X_s]$  be the reduction of  $\Psi_{\mathbf{t}, \mathbf{c}}$  modulo the prime  $p$ .

Conjecture 4 amounts to  $\Psi_{\mathbf{t},\mathbf{c}}^p$  being a nonzero polynomial. This is, at least one of the coefficients  $a_b \bmod p, b \in \{0,1\}^s$  is nonzero. Equivalently, at least one of the evaluations  $\Psi_{\mathbf{t},\mathbf{c}}^p(e), e \in \{0,1\}^s \subset \mathbb{F}_p^s$  is nonzero. Below we choose to reformulate the asymptotic version, Conjecture 5.

► **Conjecture 17** (Conjecture 5 reformulated). *There exist universal constants  $e_p, e_s \geq 1$  such that for every integer  $n \geq 1$ , prime  $p \geq n^{e_p}$ , and  $s \geq (\log n)^{e_s}$  the following holds. For every pair of strictly increasing sequences  $\mathbf{t} = (t_1, \dots, t_s), \mathbf{c} = (c_1, \dots, c_s) \in ([0, n] \cap 2\mathbb{Z})^s$  satisfying  $\mathbf{c} \leq \mathbf{t}$ , it holds that  $\Psi_{\mathbf{t},\mathbf{c}}^p(X_1, X_2, \dots, X_s)$  is nonzero.*

## B Arithmetic Geometry Heuristics Supporting Conjecture 4

We laboured through the whole previous section trying to argue that the restriction  $\Psi_{\mathbf{t},\mathbf{c}}$  to the Boolean hypercube of  $\Phi_{\mathbf{t},\mathbf{c}}$  is not identically zero modulo our chosen prime  $p$ . Our starting observation in this section is that the reduction  $\Phi_{\mathbf{t},\mathbf{c}}^p$  of  $\Phi_{\mathbf{t},\mathbf{c}}$  is non zero, since  $\Phi_{\mathbf{t},\mathbf{c}}$  is primitive (it is apparent from the defining equation that the highest total degree term of  $\Phi_{\mathbf{t},\mathbf{c}}$  is monic). Therefore, the zeroes of  $\Phi_{\mathbf{t},\mathbf{c}}^p$  define a hypersurface (that is, of codimension 1). We study the intersection of the Boolean hypercube sitting inside  $\mathbb{F}_p^s$  with this hypersurface using arithmetic geometry. Our analysis falls short of proving Conjecture 4 owing the failure to control some error terms. But we will prove Conjecture 4 holds when relaxed to accommodate hypercubes of side length growing with  $p$ .

It is convenient to be ambitious and target stronger versions of Conjecture 4 (or its asymptotic variant, Conjecture 5) which, arguably, are even more natural. First, the distribution of values obtained by evaluating  $\Phi_{\mathbf{t},\mathbf{c}}^p$  on the Boolean hypercube  $\{0,1\}^s$ , for any  $\mathbf{t}, \mathbf{c}$  in question, is fairly balanced when  $p$  is taken sufficiently large compared to  $n$ . More precisely, we postulate the following conjecture.

► **Conjecture 18** (Strong form, value distribution). *There exist universal constants  $e_p, e_s \geq 1$  and  $\beta \in (0, 1)$  such that for every integer  $n \geq 1$ , prime  $p \geq n^{e_p}$ , and  $s \geq (\log n)^{e_s}$  the following holds. For every pair of strictly increasing sequences  $\mathbf{t} = (t_1, \dots, t_s), \mathbf{c} = (c_1, \dots, c_s) \in ([0, n] \cap 2\mathbb{Z})^s$  satisfying  $\mathbf{c} \leq \mathbf{t}$ , it holds that*

$$\left| \sum_{(x_1, \dots, x_s) \in \{0,1\}^s} \zeta_p^{\Phi_{\mathbf{t},\mathbf{c}}(x_1, \dots, x_s)} \right| \leq 2^{\beta s}, \quad (\text{B.1})$$

where  $\zeta_p$  is a  $p^{\text{th}}$  root of unity in  $\mathbb{C}$ .

When the prime  $p$  exceeds the height of  $\Phi_{\mathbf{t},\mathbf{c}}$ , the sum concentrates in a wedge above the positive real axis disturbing the equidistribution. Despite not stating explicitly, we are only interested in (and only claim the conjecture) when  $p$  is small compared to the height of  $\Phi_{\mathbf{t},\mathbf{c}}$ . What really concerns us is the distribution of zeroes

$$\Phi_{\mathbf{t},\mathbf{c}}(\mathbb{F}_p, 2) := \left\{ (x_1, x_2, \dots, x_s) \in \{0,1\}^s \subset \mathbb{F}_p^s \mid \Phi_{\mathbf{t},\mathbf{c}}^p(x_1, x_2, \dots, x_s) = 0 \right\}$$

of  $\Phi_{\mathbf{t},\mathbf{c}}^p$  on the Boolean hypercube; suggesting another strengthening of Conjecture 5.

► **Conjecture 19** (Strong form, point count). *There exist universal constants  $e_p, e_s \geq 1$  and  $\beta \in (0, 1)$  such that for every integer  $n \geq 1$ , prime  $p \geq n^{e_p}$ , and  $s \geq (\log n)^{e_s}$  the following holds. For every pair of strictly increasing sequences  $\mathbf{t} = (t_1, \dots, t_s), \mathbf{c} = (c_1, \dots, c_s) \in ([0, n] \cap 2\mathbb{Z})^s$  satisfying  $\mathbf{c} \leq \mathbf{t}$ , it holds that  $|\Phi_{\mathbf{t},\mathbf{c}}(\mathbb{F}_p, 2)| \leq 2^{\beta s}$ .*

We have gathered some data using a computer program to shed some more light on the exponential sum in Conjecture 18, presented in a longer version of the paper [3].

### B.1 Pascal Determinant Hypersurfaces

Using arithmetic geometry, we next argue for the rarity of zeroes as stated in Conjecture 19. We start with the most naive yet convincing argument. Before addressing the intersection with the Boolean hypercube, consider the  $\mathbb{F}_p$ -rational points  $\Phi_{\mathbf{t},\mathbf{c}}(\mathbb{F}_p) := \left\{w \in \mathbb{F}_p \mid \Phi_{\mathbf{t},\mathbf{c}}^p(w) = 0\right\}$  on the hypersurface of dimension  $s - 1$  and degree  $\leq ns$  in isolation. The Schwartz-Zippel Lemma implies  $|\Phi_{\mathbf{t},\mathbf{c}}(\mathbb{F}_p)| \leq nsp^{s-1}$ . If  $\Phi_{\mathbf{t},\mathbf{c}}^p$  is irreducible or if it has only a few (say  $N_{\mathbf{t},\mathbf{c}}^p$ ) irreducible components, the Lang-Weil bound gives the improved estimate [18]

$$|\Phi_{\mathbf{t},\mathbf{c}}(\mathbb{F}_p) - N_{\mathbf{t},\mathbf{c}}^p p^{s-1}| = (ns - 1)(ns - 2)p^{s-3/2} + O(nsp^{s-2}).$$

With the unimportant structured factors removed from  $\Phi_{\mathbf{t},\mathbf{c}}$ , the remaining  $\Xi_{\mathbf{t},\mathbf{c}}$  (which is also primitive, by Gauss’s lemma) also has non zero reduction  $\Xi_{\mathbf{t},\mathbf{c}}^p$ . It is not always irreducible. For instance, if the index sets  $\mathbf{t}, \mathbf{c}$  are such that  $c_j < t_{j+1}$  for some  $j$ , then the vertex disjoint paths connecting the first  $j$  vertices are decoupled from the rest: resulting in a factorization of  $\Xi_{\mathbf{t},\mathbf{c}}$ . But for the factorization induced by such decouplings, the reduction  $\Xi_{\mathbf{t},\mathbf{c}}^p$  is likely to be irreducible. Better still, if (the homogenization of)  $\Xi_{\mathbf{t},\mathbf{c}}^p$  is irreducible and defines a smooth projective variety, then deep results arising from Deligne’s proof of the Weil conjectures [9, Théorème 8.1] imply the full “square root cancellation”

$$|\Xi_{\mathbf{t},\mathbf{c}}(\mathbb{F}_p) - p^{s-1}| = O(b_{s-1} p^{\frac{s-1}{2}})$$

where  $b_{s-1} \leq \frac{1}{2}s(s+1)(sn)^s$  is the  $s - 1$ th Betti number. To derive our heuristic estimate, Schwartz-Zippel will suffice. For ease of exposition, we will use  $\Phi_{\mathbf{t},\mathbf{c}}$  in the ensuing analysis, even though  $\Xi_{\mathbf{t},\mathbf{c}}^p$  offers some minor gains degree wise. In spirit, the probability  $\Phi_{\mathbf{t},\mathbf{c}}^p$  is zero at a point in  $\mathbb{F}_p^s$  is centred at  $\frac{N_{\mathbf{t},\mathbf{c}}^p}{p}$  with an error term depending on the smoothness. Irrespective of the smoothness, the error term is negligible compared to the estimate for  $p$  a big enough polynomial in  $n$ . We hypothesise that the hypersurface intersects generically with the Boolean hypercube and the number of intersection points is bounded as

$$|\Phi_{\mathbf{t},\mathbf{c}}(\mathbb{F}_p, 2)| \approx |\Phi_{\mathbf{t},\mathbf{c}}(\mathbb{F}_p)| \left(\frac{2}{p}\right)^s. \tag{B.2}$$

By the Schwartz-Zippel lemma

$$|\Phi_{\mathbf{t},\mathbf{c}}(\mathbb{F}_p, 2)| \approx |\Phi_{\mathbf{t},\mathbf{c}}(\mathbb{F}_p)| \left(\frac{2}{p}\right)^s = O\left(\frac{ns2^s}{p}\right) \tag{B.3}$$

suggesting Conjecture 19 holds for  $p > n^2$ .

### B.2 Katz-Laumon Sums and Point Counting in Hypercubes

Through arithmetic geometric bounds on exponential sums, we argue our determinant hypersurfaces intersect generically with the Boolean hypercube. We show Conjecture 4 holds when relaxed to allow hypercubes of length (larger than 2) growing with the prime. Quantitatively, the bounds attained fall short of proving Conjecture 4. Yet, the methods are illuminating and suggest there are no arithmetic obstructions to our conjectures.

The key ingredient is the Katz-Laumon sum [17]. Building on Grothendieck’s foundational trace formula for  $\ell$ -adic cohomology and Deligne’s proof of the Weil conjectures, Katz and Laumon studied certain trigonometric sums over arbitrary high dimensional varieties over finite fields, parametrized by auxiliary points. They proved square root cancellation without any strong geometric assumption (such as smoothness) on the variety, for almost all choices

of the parameter. Fouvry [10] and Fouvry-Katz [11] extended Katz and Laumon's theorem to obtain a stratified theorem. Fouvry applied Katz-Laumon sums to count points of a variety on hypercubes (Boolean or more general). Fouvry and Katz extended this approach and proved better bounds provided more is assumed about the geometry of the variety. We adapt these techniques to bound the intersection of our hypersurfaces  $\Phi_{t,c}(\mathbb{F}_p)$  with Boolean hypercubes as (a proof of the bound is in a longer version of the paper [3])

$$|\Phi_{t,c}(\mathbb{F}_p, 2)| = \left(\frac{2}{p}\right)^s |\Phi_{t,c}(\mathbb{F}_p)| + O\left(p^{(s-1)/2}(\log p)^s + \frac{2^{s-1} \log p}{\sqrt{p}}\right). \quad (\text{B.4})$$

The constant hidden in the asymptotic  $O$  notation may depend on  $s$ , but this dependence will be subsumed and removed in Equation (B.5). Our ultimate goal is to claim the right hand side is strictly less than  $2^s$ , which would prove Conjecture 4. However,  $p^{(s-1)/2}$  is too large and muddies the estimate. The bounds are good enough if the hypercube side length is extended to  $b > 2$ , since Fouvry [10] shows for every  $\Phi_{t,c}$ , for large enough  $p$ ,

$$|\Phi_{t,c}(\mathbb{F}_p, b)| = \left(\frac{b}{p}\right)^s |\Phi_{t,c}(\mathbb{F}_p)| + O\left(p^{(s-1)/2}(\log p)^s + \frac{b^{s-1} \log p}{\sqrt{p}}\right). \quad (\text{B.5})$$

From the Schwartz-Zippel lemma bound Equation (B.3) on  $|\Phi_{t,c}(\mathbb{F}_p)|$ ,

$$|\Phi_{t,c}(\mathbb{F}_p, b)| \leq \frac{ns2^s}{p} + O\left(p^{(s-1)/2}(\log p)^s + \frac{b^{s-1} \log p}{\sqrt{p}}\right).$$

For  $b \gg p^{3/4}$ ,  $|\Phi_{t,c}(\mathbb{F}_p, b)| \ll b^s$ . Fouvry's theorem applies to arbitrary varieties and the "for large enough  $p$ " clause is primarily in place to ensure the defining polynomials do not identically vanish modulo  $p$ . To us,  $\Phi_{t,c}^p$  is non zero, so the bounds should hold uniformly for all  $p$ . Therefore, with some work to ensure uniformity of bounds, these methods prove Conjecture 19 when relaxed to Boolean cubes of length growing  $b \gg p^{3/4}$ . A proof is deferred to the full version of this paper.

We believe the large error term in Equation (B.4) and Fouvry's theorem Equation (B.5) to be artefacts of proof techniques and not intrinsic to the quantities. The primary lesson we advocate from these arithmetic geometric techniques is qualitative and not quantitative. There should be no arithmetic obstruction to equidistribution of the zeroes of the hypersurfaces defined by our determinant polynomials in the Boolean hypercube, as claimed in the strong form of our conjecture.

# Towards a Decomposition-Optimal Algorithm for Counting and Sampling Arbitrary Motifs in Sublinear Time

Amartya Shankha Biswas ✉

CSAIL, Massachusetts Institute of Technology, Cambridge MA, USA

Talya Eden ✉ 🏠 

CSAIL, Massachusetts Institute of Technology, Cambridge MA, USA

Ronitt Rubinfeld ✉ 🏠

CSAIL, Massachusetts Institute of Technology, Cambridge MA, USA

---

## Abstract

We consider the problem of sampling and approximately counting an arbitrary given motif  $H$  in a graph  $G$ , where access to  $G$  is given via queries: degree, neighbor, and pair, as well as uniform edge sample queries. Previous algorithms for these tasks were based on a decomposition of  $H$  into a collection of odd cycles and stars, denoted  $D^*(H) = \{O_{k_1}, \dots, O_{k_q}, S_{p_1}, \dots, S_{p_\ell}\}$ . These algorithms were shown to be optimal for the case where  $H$  is a clique or an odd-length cycle, but no other lower bounds were known.

We present a new algorithm for sampling arbitrary motifs which, up to  $\text{poly}(\log n)$  factors, is always at least as good, and for most graphs  $G$  is strictly better. The main ingredient leading to this improvement is an improved uniform algorithm for sampling stars, which might be of independent interest, as it allows to sample vertices according to the  $p$ -th moment of the degree distribution.

Finally, we prove that this algorithm is *decomposition-optimal* for decompositions that contain at least one odd cycle. These are the first lower bounds for motifs  $H$  with a nontrivial decomposition, i.e., motifs that have more than a single component in their decomposition.

**2012 ACM Subject Classification** Theory of computation → Streaming, sublinear and near linear time algorithms

**Keywords and phrases** Sublinear time algorithms, Graph algorithms, Sampling subgraphs, Approximate counting

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.55

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2107.06582>

**Funding** *Amartya Shankha Biswas*: Big George Ventures Fund, MIT-IBM Watson AI Lab and Research Collaboration Agreement No. W1771646, NSF awards CCF-173380, CCF-2006664 and IIS-1741137.

*Talya Eden*: This work was supported by the NSF grant CCF-1740751, Eric and Wendy Schmidt Fund, and Ben-Gurion University.

*Ronitt Rubinfeld*: This work was supported the NSF TRIPODS program (awards CCF-1740751 and DMS 2022448), NSF award CCF-2006664 and by the Fintech@CSAIL Initiative.

**Acknowledgements** Talya Eden is thankful to Dana Ron and Oded Goldreich for their valuable suggestions regarding the presentation of the lower bound results. The authors are thankful for the anonymous reviewers for their useful comments and observations.



© Amartya Shankha Biswas, Talya Eden, and Ronitt Rubinfeld; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 55; pp. 55:1–55:19



Leibniz International Proceedings in Informatics

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

## 1 Introduction

The problems of counting and sampling small motifs in graphs are fundamental algorithmic problems with many applications. Small motifs statistics are used for the study and characterization of graphs in multiple fields, including biology, chemistry, social networks and many others (see e.g., [36, 30, 21, 33, 32, 43, 28, 35, 38, 41, 31]). From a theoretical perspective, the complexity of the best known classical algorithms for exactly enumerating small motifs such as cliques and paths of length  $k$ , grows exponentially with  $k$  [42, 9]. On the more applied side, there is an extensive study of practical algorithms for approximate motif counting (e.g., [39, 5, 34, 1, 27, 12, 7, 24]). We study the problems of approximate motif counting and uniform sampling in the *sublinear-time* setting, where sublinear is with respect to the size of the graph. We consider the *augmented query model*, introduced by [2], where the allowed queries are degree, neighbor and pair queries as well as uniform edge sample queries.<sup>1</sup> We note that the model which only allows for the first three types of queries is referred to as the *general graph query model*, introduced by [29].

The problems of approximate counting and uniformly sampling of *arbitrary motifs* of constant size in sublinear-time have seen much progress recently, through the results of Assadi, Kapralov and Khanna [3], and Fichtenberger, Gao and Peng [23]. The algorithms of [3, 23] both start by computing an optimal (in a sense that will be clear shortly) decomposition of the motif  $H$  into vertex-disjoint odd cycles and stars, defined next.

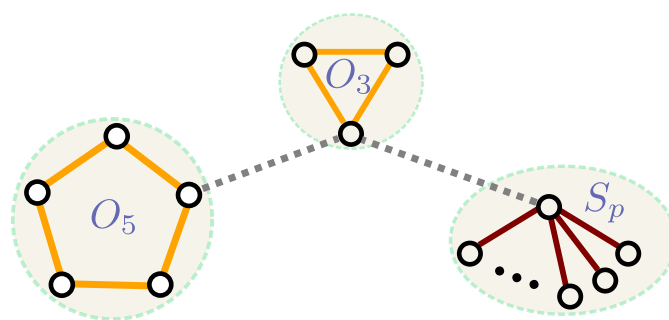
**A decomposition into odd cycles and stars.** A decomposition  $D$  of a motif (graph)  $H$  into a collection of vertex disjoint small cycles and stars  $\{O_{k_1}, \dots, O_{k_q}, S_{p_1}, \dots, S_{p_\ell}\}$  is valid if all vertices of  $H$  belong to either a star or an odd cycle in the collection. Each decomposition can be associated with a weight function  $f_D : E \rightarrow \{0, \frac{1}{2}, 1\}$  which assigns weight 1 to edges of its star components, weight  $1/2$  to edges of its odd cycle components and weight 0 to all other edges in  $H$ . See figure 1 for an illustration. Hence, each decomposition  $\{O_{k_1}, \dots, O_{k_q}, S_{p_1}, \dots, S_{p_\ell}\}$  has value  $\rho(D) = \sum_{e \in H} f_D(e) = \sum_{i=1}^q k_i/2 + \sum_{j=1}^{\ell} p_j$ , where throughout the paper  $k_i$  and  $p_j$  denote the length and number of petals in the  $i^{\text{th}}$  cycle and  $j^{\text{th}}$  star, respectively, in  $D^*(H)$ . For every  $H$ , its optimal decomposition value is  $\rho(H) = \min_D \{\rho(D)\}$ , and a decomposition  $D$  is said to be *optimal* for  $H$  if  $\rho(D) = \rho(H)$ . We fix (one of) the optimal decomposition of  $H$ , and denote it by  $D^*(H)$ . In [3], it is shown that an optimal decomposition of a motif  $H$  can be computed in polynomial time in  $|H|$ .<sup>2</sup>

The algorithm in [23] has expected running time <sup>3</sup>  $O\left(\frac{m^{\rho(H)}}{n}\right)$  for the task of uniformly

<sup>1</sup> Degree queries return the degree of the queried vertex, neighbor queries with index  $i \leq d(v)$  return the  $i^{\text{th}}$  neighbor of the queried vertex, pair queries return whether there is an edge between the queried pair of vertices, and uniform edge queries return a uniformly distributed edge in the graph.

<sup>2</sup> We note that  $\rho(H)$  is equal to the fractional edge cover value of  $H$ : the fractional edge cover value of a motif (graph)  $H$  is the solution to the following minimization problem. Minimize  $\sum_{e \in E} f(e)$  under the constraint that for every  $v \in H$ ,  $\sum_{e \ni v} f(e) \geq 1$ . In [3], the decomposition is computed by first computing an optimal fractional cover. However, as there exists a mapping between fractional edge covers to decompositions which preserves their value, we choose to define  $\rho(H)$  according to the minimal valid decomposition value.

<sup>3</sup> Throughout the paper, unless stated otherwise, the query complexity of the mentioned sublinear-time algorithms is the same as the minimum between their running time and  $\min\{n + m, m \log n\}$ . This is true since any algorithm can simply query the entire graph and continue computation locally. Querying the entire graph can either be performed by querying the neighbors of all vertices (which takes  $O(n + m)$  queries), or by performing  $m \log n$  uniform edge samples, which, with high probability, return all edges in the graph (note that we do not care about isolated vertices, as we assume the motif  $H$  is connected). Hence, we focus our attention on the running time complexity.



■ **Figure 1** An example of an optimal decomposition of a motif  $H$  into odd cycles and stars. The orange edges have weight  $1/2$ , the red edges have weight  $1$ , and the dotted edges have zero weight.

sampling a copy of  $H$ , where  $\bar{h}$  is the number of copies of  $H$  in  $G$ , and  $m$  is the number of oriented edges<sup>4</sup> in  $G$ . The algorithm in [3] for the  $(1 \pm \epsilon)$ -approximation task has the same complexity up to  $\text{poly}(\epsilon, |H|, \log n)$  factors, where  $n$  is the number of vertices in  $G$ .

## 1.1 Our results

We present improved upper and lower bounds for the tasks of estimating and sampling any arbitrary motif in a graph  $G$  in sublinear time (with respect to the size of  $G$ ). First, we give a new, essentially optimal, star-sampler for graphs. We also show that with few modifications, the star-sampler can be adapted to an optimal  $\ell_p$  sampler, which might be of independent interest. Based on this sampler, as well as an improved sampling approach, we present our main algorithm for sampling a uniformly distributed copy of any given motif  $H$  in a graph  $G$ . Our algorithm's complexity is parameterized by what we refer to as the *decomposition-cost* of  $H$  in  $G$ , denoted  $\text{DECOMP-COST}(G, H, D^*(H))$ . We further show that our motif sampling algorithm can be used to obtain a  $(1 \pm \epsilon)$ -estimate of the motif at question (with an overhead of an  $O(1/\epsilon^2)$  factor). As we shall see, our result is always at least as good as previous algorithms for these problems (up to a  $\log n \log \log n$  term), and greatly improves upon them for various interesting graph classes, such as random graphs and bounded arboricity graphs.

We then continue to prove that for any motif whose optimal decomposition contains at least one odd cycle, this bound is *decomposition-optimal*: we show that for every decomposition  $D$  that contains at least one odd cycle, there exists a motif  $H_D$  (with optimal decomposition  $D$ ) and a family of graphs  $\mathcal{G}$  so that in order to sample a uniformly distributed copy of  $H$  (or to approximate  $\bar{h}$ ) in a uniformly chosen graph in  $\mathcal{G}$ , the number of required queries is  $\Omega(\min\{\text{DECOMP-COST}(G, H, D^*(H)), m\})$  in expectation.

We start by describing the upper bound.

### 1.1.1 Optimal star/ $\ell_p$ -sampler

Our first contribution is an improved algorithm, *Sample-a-Star*, for sampling a (single) star uniformly at random, and its variant for sampling vertices according to the  $p^{\text{th}}$  moment. For a vertex  $v$ , we let  $\bar{s}_p(v) = \binom{d(v)}{p}$ , if  $d(v) \geq p$ , and otherwise,  $\bar{s}_p(v) = 0$ . We let  $\bar{s}_p = \sum_{v \in V} \bar{s}_p(v)$  denote the number of  $p$ -stars in the graph. We will also be interested in the closely related value of the  $p^{\text{th}}$  moment of the degree distribution,  $\bar{\mu}_p = \sum_{v \in V} d(v)^p$ .

<sup>4</sup> Throughout the paper we think of every edge  $\{u, v\}$  as two oriented edges  $(u, v)$  and  $(v, u)$ , and let  $m$  denote the number of oriented edges.

► **Theorem 1.** *There exists a procedure, **Sample-a-Star**, that given query access to a graph  $G$ , and a constant factor estimate of  $\bar{s}_p$ , returns a uniformly distributed  $p$ -star in  $G$ . The expected query complexity and running time of the procedure are  $O\left(\min\left\{\frac{m \cdot n^{p-1}}{\bar{s}_p}, \frac{m}{\bar{s}_p^{1/p}}\right\}\right)$  where  $\bar{s}_p$  denotes the number of  $p$ -stars in  $G$ .*

We note that a constant factor estimate of  $\bar{s}_p$  can be obtained by invoking one of the algorithms in [17, 2], in expected query complexity  $\tilde{O}\left(\min\left\{\frac{m \cdot n^{p-1}}{\bar{s}_p}, \frac{m}{\bar{s}_p^{1/p}}\right\}\right)$ . Therefore, if such an estimate is not known in advance, then it could be computed, with probability at least  $2/3$ , by only incurring a  $\log n$  factor to the expected time complexity.

We will also show a variant of **Sample-a-Star**, denoted **Sublinear- $\ell_p$ -Sampler**, that gives an optimal  $\ell_p$ -sampler for any integer  $p \geq 2$  in sublinear time. That is, **Sublinear- $\ell_p$ -Sampler** allows to sample according to the  $p^{\text{th}}$  moment of the degree distribution, so that every vertex  $v \in V$  is returned by it with probability  $d(v)^p / \bar{\mu}_p$ . The question of sampling according to the  $p^{\text{th}}$  moment for various values of  $p$  has been studied extensively in the streaming model where  $\ell_p$  samplers have found numerous applications, see, e.g., the recent survey by Cormode and Hossein [11] and the references therein. Therefore we hope it could find applications in the sublinear-time setting that go beyond subgraph sampling.

► **Theorem 2.** *There exists an algorithm, **Sublinear- $\ell_p$ -Sampler**, that returns a vertex  $v \in V$ , so that each  $v \in V$  is returned with probability  $d(v)^p / \bar{\mu}_p$ . The expected running time of the algorithm is  $O\left(\min\left\{\frac{m \cdot n^{p-1}}{\bar{\mu}_p}, \frac{m}{\bar{\mu}_p^{1/p}}\right\}\right)$ .*

Observe that for every value of  $p$ ,  $\bar{s}_p < \bar{\mu}_p$ . Furthermore, Since  $m$  and  $\bar{\mu}_p^{1/p}$  are simply the  $\ell_1$  and  $\ell_p$  norms of the degree distribution of  $G$ , it holds that  $\bar{\mu}_p^{1/p}$  is smaller than  $m$ , and could be as small as  $m/n^{1-1/p}$ . Therefore,  $\bar{\mu}_p^{1/p} < m \Leftrightarrow \bar{\mu}_p^{p-1/p} < m^{p-1}$ . and it follows that

$$m \cdot \min\left\{n^{p-1}, \bar{s}_p^{(p-1)/p}\right\} \leq m \cdot \bar{s}_p^{(p-1)/p} < m \cdot \bar{\mu}_p^{(p-1)/p} \leq m \cdot m^{p-1} = m^p. \quad (1)$$

Hence, not accounting for the  $O(\log n \log \log n)$  term, the expected complexity  $\tilde{O}(m \cdot \min\{n^{p-1}, \bar{s}_p^{(p-1)/p}\} / \bar{s}_p)$  of **Sample-a-Star** strictly improves upon the  $O(m^p / \bar{s}_p)$  expected complexity of the star-sampling algorithm by [23]. Accounting for that term, our algorithm is preferable when either  $d_{\text{avg}} = \omega(\log n \log \log n)$  or  $m / \bar{s}_p^{1/p} = \omega(\log n)$ .

Furthermore, the complexity of **Sample-a-Star** matches the complexities of the star approximation algorithms by [26, 2], thus proving that uniformly sampling and approximately counting stars in the augmented model have essentially the same complexity. Finally, the construction of the lower bound for the estimation variant by [26] proves that **Sample-a-Star** and **Sublinear- $\ell_p$ -Sampler** are essentially optimal.

### 1.1.2 An algorithm for sampling and estimating arbitrary motifs

Given the above star sampler, we continue to describe our main contribution: an algorithm, **Sample- $H$** , that for any graph  $G$  and given motif  $H$ , outputs a uniformly distributed copy of  $H$  in  $G$ .

To sample a copy of  $H$  we first sample copies of all basic components in its decomposition  $D^*(H)$ , and then check if they can be extended to a copy of  $H$  in  $G$ . Therefore, it will be useful to define the costs of these sampling operations.



► **Notation 3** (Basic components, counts and costs). Let  $H$  be a motif, and let  $D^*(H) = \{O_{k_1}, \dots, O_{k_q}, S_{p_1}, \dots, S_{p_\ell}\}$  be an optimal decomposition of  $H$ . We refer to the odd cycles and stars in  $D^*(H)$  as the basic components of the decomposition (or sometimes, abusing notation, of  $H$ ). We use the notation  $\{C_i\}_{i \in [r]}$ , to denote the set of all components in  $D^*(H)$ ,  $\{C_i\}_{i \in [r]} = D^*(H)$ , where  $r = q + \ell$ .

For every basic component  $C_i$  in  $D^*(H) = \{C_i\}_{i \in [r]}$ , we denote the number of copies of  $C_i$  in  $G$  as  $\bar{c}_i$  and refer to it as the count of  $C_i$ . Similarly,  $\bar{o}_k$  and  $\bar{s}_p$  denote the number of copies of length  $k$  odd cycles and  $p$ -stars in  $G$ , respectively.

We also define the sampling cost (or just cost in short) of  $C_i$  to be:

$$\text{cost}(C_i) = \begin{cases} m^{k/2}/\bar{o}_k & C_i = O_k \\ \min \left\{ \frac{m \cdot n^{p-1}}{\bar{s}_p}, \frac{m}{\bar{s}_p^{1/p}} \right\} & C_i = S_p \end{cases}.$$

Observe that indeed, by Theorem 1, sampling a single  $p$ -star in  $G$  takes  $\text{cost}(S_p) = \min \left\{ \frac{m \cdot n^{p-1}}{\bar{s}_p}, \frac{m}{\bar{s}_p^{1/p}} \right\}$  queries in expectation, and by [23, Lemma 3.1], sampling a single  $O_k$  odd cycle takes  $\text{cost}(O_k) = m^{k/2}/\bar{o}_k$  queries in expectation.

► **Notation 4** (Decomposition-cost). For a motif  $H$ , an optimal decomposition  $D^*(H)$  of  $H$ , and a graph  $G$ , the decomposition cost of  $H$  in  $G$ , denoted  $\text{DECOMP-COST}(G, H, D^*(H))$  is

$$\text{DECOMP-COST}(G, H, D^*(H)) = \max_{i \in [r]} \{\text{cost}(C_i)\} \cdot \frac{\prod \bar{c}_i}{\bar{\mathbf{h}}}.$$

Note that the motif  $H$  determines the counts of  $\bar{\mathbf{h}}$  and its decomposition  $D^*(H)$  determines what are the basic component counts in  $G$  that are relevant to the sampling cost.

► **Theorem 5.** Let  $G$  be a graph over  $n$  vertices and  $m$  edges, and let  $H$  be a motif such that  $D^*(H) = \{O_{k_1}, \dots, O_{k_q}, S_{p_1}, \dots, S_{p_\ell}\} = \{C_i\}_{i \in [r]}$ . There exists an algorithm, **Sample- $H$** , that returns a copy of  $H$  in  $G$ . With probability at least  $1 - 1/\text{poly}(n)$ , the returned copy is uniformly distributed in  $G$ . The expected query complexity of the algorithm is

$$O(\min \{\text{DECOMP-COST}(G, H, D^*(H)), m\}) \cdot \log n \log \log n.$$

In the full version of this paper ([8]), we prove that with slight modifications to the sampling algorithm we can obtain a  $(1 \pm \epsilon)$ -approximation algorithm for  $\bar{\mathbf{h}}$ , with the same expected query complexity and running time up to a multiplicative factor of  $O(1/\epsilon^2)$ .

**Comparison to previous bounds.** We would like to compare our algorithm's expected complexity stated in Theorem 5, to the expected complexity  $O\left(\frac{m^{\rho(H)}}{\bar{\mathbf{h}}}\right)$  of the counting and sampling algorithms by [3] and [23], respectively, where recall that for an optimal decomposition  $D^*(H) = \{O_{k_1}, \dots, O_{k_q}, S_{p_1}, \dots, S_{p_\ell}\}$  of  $H$ ,  $\rho(H) = \sum_{i \in [q]} k_i/2 + \sum_{i \in [\ell]} p_i$ .

Recalling Equation 1, and plugging in the costs of the basic components and the decomposition cost, defined in Notations 3 and 4, respectively, we get that for any graph  $G$  and motif  $H$ ,

$$\begin{aligned} \text{DECOMP-COST}(G, H, D^*(H)) &= \max_{i \in [r]} \{\text{cost}(C_i)\} \cdot \frac{\prod \bar{c}_i}{\bar{\mathbf{h}}} \\ &= \max_{i \in [r]} \{\text{cost}(C_i)\} \cdot \frac{\prod_{i \in [q]} \bar{o}_{k_i} \cdot \prod_{i \in [\ell]} \bar{s}_{p_i}}{\bar{\mathbf{h}}} \end{aligned}$$

$$\begin{aligned} &\leq \frac{\prod_{i \in [q]} m^{k_i/2} \cdot \prod_{i \in [\ell]} m \cdot (\min\{n^{p_i-1}, \bar{s}_{p_i}^{(p_i-1)/p_i}\})}{\bar{h}} \\ &< \frac{\prod_{i \in [q]} m^{k_i/2} \cdot \prod_{i \in [\ell]} m^p}{\bar{h}} = \frac{m^{\rho(H)}}{\bar{h}}. \end{aligned}$$

Therefore, as long as  $D^*(H)$  contains at least one star, and not accounting for the term  $O(\log n \log \log n)$ , our algorithm is preferable to the previous one, as we save a factor of at least  $d_{\text{avg}}^{p-1}$  for each  $p$ -star in  $D^*(H)$ .

Moreover, the complexity of our sampling algorithm is parameterized by the *actual* counts of the basic components  $O_{k_1}, \dots, O_{k_q}, S_{p_1}, \dots, S_{p_\ell}$  of the graph  $G$  at hand, rather than by the maximal possible counts of these components, respectively  $m^{k_1/2}, \dots, m^{k_q/2}, m^{p_1}, \dots, m^{p_\ell}$ , as is in previous algorithms. For example, if the max component cost is due to the odd cycle of length  $k_1$ , we get

$$O^* \left( \frac{m^{k_1/2} \cdot \bar{o}_{k_2} \cdot \dots \cdot \bar{o}_{k_q} \cdot \bar{s}_{p_1} \cdot \dots \cdot \bar{s}_{p_\ell}}{\bar{h}} \right) \text{ vs. } O^* \left( \frac{m^{k_1/2} \cdot m^{k_2/2} \cdot \dots \cdot m^{k_q/2} \cdot m^{p_1} \cdot \dots \cdot m^{p_\ell}}{\bar{h}} \right)$$

of the previous algorithms. Importantly, this parameterization arises *only* in the analysis, while the algorithm itself is very simple, and does not depend on prior knowledge of the actual values of these counts.

**Improved results for various graph classes.** Our parameterization immediately implies improved results in various interesting graph classes. For example, for sparse Erdős-Rényi random graphs  $\mathcal{G}(n, d/n)$ , the expected count of  $k$ -odd cycles is  $\Theta(d^k)$ , and of  $p$ -stars is  $\Theta(n \cdot d^p)$ . Hence, if we consider for example a motif  $H$  that is composed of a triangle connected to a 5-petals star, our algorithm has expected complexity  $O^* \left( \frac{m^{2.5} \cdot d^4}{\bar{h}} \right)$ , while the algorithms in [3, 23] have expected complexity  $O \left( \frac{m^{6.5}}{\bar{h}} \right)$ . In another example, for graphs of bounded arboricity<sup>5</sup>  $\alpha$ , the number of  $k$ -odd cycles is upper bounded<sup>6</sup> by  $\alpha \cdot m^{(k-1)/2}$ . Therefore, in the case that  $G$  has, e.g., constant arboricity, we save a multiplicative factor of  $\sqrt{m^q}$  or  $\sqrt{m^{q-1}}$ , depending on whether the max cost component is due to a star or an odd cycle, respectively (recall that  $q$  is the number of odd cycles in the decomposition).

### 1.1.3 Lower bound for estimating and sampling general motifs

In the full version, we prove the following lower bound, which states that for every decomposition  $D$  that contains at least one odd cycle component and every realizable value of DECOMP-COST, there exists a motif  $H_D$  such that  $D$  is an optimal decomposition of  $H_D$ , and for which our upper bound is optimal.

► **Theorem 6.** *For any decomposition  $D$  that contains at least one odd cycle, and for every  $n$  and  $m$  and realizable value  $DC$  of DECOMP-COST, there exists a motif  $H_D$ , with optimal decomposition  $D$ , and a family of graphs  $\mathcal{G}$  over  $n$  vertices and  $m$  edges, for which the following holds. For every  $G \in \mathcal{G}$ ,  $\text{DECOMP-COST}(G, H_D, D) = DC$ , and the expected query complexity of sampling (whp) a uniformly distributed copy of  $H_D$  in a uniformly chosen  $G \in \mathcal{G}$  is  $\Omega(DC)$ .*

<sup>5</sup> The arboricity of a graph  $G$  is the minimal number of forests required to cover the edge set of  $G$ .

<sup>6</sup> In a graph  $G$  with arboricity  $\alpha$  there exists an acyclic ordering of the graph's vertices, such that each vertex has  $O(\alpha)$  vertices exceeding it in the order. We can attribute each  $k$ -cycles in the graph to its first vertex in that ordering. It then holds that each vertex has at most  $(d^+(v))^2 \cdot m^{(k-3)/2}$  attributed cycles, and it follows that  $\bar{o}_k \leq \alpha \cdot m^{(k-1)/2}$ , where  $d^+(v)$  is the number of neighbors of  $v$  that exceed it in the aforementioned ordering.

Prior to this work, the only known lower bounds for the tasks of uniformly sampling or approximately counting motifs  $H$  that were either a clique [19], a single odd cycle [3], or a single star [26, 2, 19]. The above theorem provides the first lower bounds for motifs with non-trivial decompositions. Furthermore, even though our bounds are only *decomposition-optimal* (that is, they do not hold for *any* motif  $H$ ), each decomposition  $D$  corresponds to at least one motif  $H_D$  (generally, there are multiple valid ones), for which our bounds are tight.

In order to prove Theorem 6, we actually prove a stronger theorem, which relies on a technical notion of *good counts*, formally stated in Definition 17 in the full version.

► **Theorem 7.** *For any decomposition  $D = \{O_{k_1}, \dots, O_{k_q}, S_{p_1}, \dots, S_{p_\ell}\} = \{C_i\}_{i \in [r]}$  that contains at least one odd cycle component, for every  $n, m, \bar{\mathbf{h}}$  and a set of good counts,  $\{\bar{c}_i\}_{i \in [r]} = \{\bar{o}_{k_1}, \dots, \bar{o}_{k_q}, \bar{s}_{p_1}, \dots, \bar{s}_{p_\ell}\}$ , as defined in Definition 17 of the full version, the following holds. There exists a motif  $H_D$ , with an optimal decomposition  $D$ , and a family of graphs  $\mathcal{G}$  over  $n$  vertices and  $m$  edges, as follows. For every  $G \in \mathcal{G}$ , the basic component counts are as specified by  $\{\bar{c}_i\}_{i \in [r]}$ , the number of copies of  $H_D$  is  $\bar{\mathbf{h}}$ , and the expected query complexity of sampling (whp) a uniformly distributed copy of  $H_D$  in a uniformly chosen  $G \in \mathcal{G}$  is*

$$\Omega \left( \min \left\{ \max_{i \in [r]} \{ \text{cost}(C_i) \} \cdot \frac{\prod_i \bar{c}_i}{\bar{\mathbf{h}}}, m \right\} \right).$$

In the full version, we first prove that Theorem 6 follows from Theorem 7. Theorem 7 is essentially a substantial refinement of Theorem 6, in the following sense. Not only that for any decomposition cost we can match the lower bound (as stated in Theorem 6), but we can match it for a large variety of *specific* setting of the basic counts (as long as they are good, as stated in Theorem 7). While Theorem 7 does not state that the lower bound holds for *any* setting of the counts  $\{\bar{c}_i\}_{i \in [r]}$ , as we discuss in the full version, some of the constraints on these counts are unavoidable. It remains an open question whether this set of constraints can be weakened, or perhaps more interestingly, whether, given that a set of constraints that is *not good*, can a better upper bound be devised.

## 1.2 Organization of the paper

We give some preliminaries in Section 2. The discussion on additional related works on sublinear motif counting and sampling is deferred to Appendix A. In Section 3 we give a high level overview of our techniques. We present our algorithms for uniformly sampling stars and arbitrary motifs  $H$  in Section 4. Due to page limitation, the full details of the  $\ell_p$ -sampler, approximation algorithm, as well as the decomposition-optimal lower bounds are deferred to the full version of this paper [8].

## 2 Preliminaries and Notation

Let  $G = (V, E)$  be a simple undirected graph. We let  $n$  denote the number of vertices in the graph. We think of every edge  $\{u, v\}$  in the graph as two *oriented* edges  $(u, v)$  and  $(v, u)$ , and slightly abuse notation to let  $m$  denote the number of oriented edges, so that  $m = \sum_{v \in V} d(v) = 2|E|$ , and  $d_{\text{avg}} = m/n$ . Unless explicitly stated otherwise, when we say “edge” we mean an oriented edge. We let  $d(v)$  denote the degree of a given vertex. We let  $[r]$  denote the set of integers 1 through  $r$ .

**The augmented query model.** We consider the augmented query model which allows for the following queries. (1) A degree query,  $deg(v)$ , returns the degree of  $v$ ,  $d(v)$ ; (2) An  $i^{\text{th}}$  neighbor query,  $Nbr(v, i)$  returns the  $i^{\text{th}}$  neighbor of  $v$  if  $i \leq d(v)$ , and otherwise returns FAIL; (3) A pair query,  $pair(u, v)$ , returns whether  $(u, v) \in E$ ; and (4) Uniform edge query returns a uniformly distributed (oriented) edge in  $E$ .

**A decomposition into odd cycles and stars.** Given a motif  $H$ , the result in [3] is parameterized by the *fractional edge cover number*  $\rho(H)$ . The fractional edge cover number is the optimal solution to the *linear programming relaxation* of the integer linear program (ILP) for the minimum edge cover of  $H$ : The ILP allows each edge to take values in  $\{0, 1\}$ , under the constraint that the sum of edge values incident to any vertex  $v$  is at least 1. The LP relaxation allows values in  $[0, 1]$  instead, and  $\rho(H)$  is the minimum possible sum of all the (fractional) values. In [3], the authors strengthen an existing result by Atserias, Grohe and Marx [4], in order to prove that there always exists an optimal solution as follows. All of the weight (i.e., non zero edges) is supported on (the edges of) vertex-disjoint odd cycles and stars, where each odd cycle edge has weight  $1/2$ , and each star edge has weight 1. Consequently, the corresponding optimal solution of the LP for a given graph  $H$  is equivalent to a decomposition of  $H$  into a collection of vertex-disjoint odd cycles and stars, denoted  $D^*(H) = \{O_{k_1}, \dots, O_{k_q}, S_{p_1}, \dots, S_{p_\ell}\}$ . See Figure 1 for an illustration.

Generally, the motif we aim to sample (or approximate its counts) will be denoted by  $H$ , and the corresponding decomposition will be  $\mathcal{D}(H) = \{O_{k_1}, \dots, O_{k_q}, S_{p_1}, \dots, S_{p_\ell}\} = \{C_i\}_{i \in r}$  for  $r = q + \ell$ . We use a convention of using  $O_{k_i}$  to refer to the  $i^{\text{th}}$  decomposition component which is an odd cycle of size  $k_i$ , and  $S_{p_i}$  to refer to the  $i^{\text{th}}$  star component, which is a star with  $p_i$  petals. We use  $\bar{o}_k$  and  $\bar{s}_p$  denote the number of  $k$ -cycles and  $p$ -stars in  $G$  respectively, and we use  $\bar{h}$  to denote the number of copies of  $H$  in  $G$ .

Next, we formally define the fractional edge cover of a graph (or motif), and the resulting decomposition. We note that in this paper we will be interested in the decomposition of the motif  $H$ , and not the graph  $G$ .

► **Definition 8** (Fractional edge cover). *A fractional edge cover of a graph is a function  $f : E \rightarrow \mathbb{R}_{\geq 0}$  such that for every  $v \in V$ ,  $\sum_{e \ni v} f(e) \geq 1$ . We say that the cost of a given edge cover  $f$  is  $\sum_{e \in E} f(e)$ . For any graph (motif)  $H$ , its fractional edge cover value is the minimum cost over all of its fractional edge covers, and we denote this value by  $\rho(H)$ . An optimal edge-cover of  $H$  is any edge cover of  $H$  with cost  $\rho(H)$ .*

► **Lemma 9** (Lemma 4 in [3]). *Any graph (motif)  $H$  admits an optimal fractional edge cover  $x^*$ , whose support, denoted  $SUPP(x^*)$ , is a collection of vertex-disjoint odd cycles and stars, such that:*

- for every odd cycle  $C \in SUPP(x^*)$ , for every  $e \in C$ ,  $x^*(e) = 1/2$ .
- for every  $e \in SUPP(x^*)$  that does not belong to an odd cycle,  $x^*(e) = 1$ .

► **Definition 10** (Decomposition into odd-cycles and stars). *Given an optimal fractional edge-cover  $x^*$  as in Lemma 9, let  $\{O_{k_1}, \dots, O_{k_q}\}$  be the odd-cycles in the support of  $x^*$ , and let  $\{S_{p_1}, \dots, S_{p_\ell}\}$  be the stars. We refer to  $D^*(H) := \{O_{k_1}, \dots, O_{k_q}, S_{p_1}, \dots, S_{p_\ell}\}$  as an (optimal) decomposition of  $H$ .*

Given a graph (motif)  $H$ , its fractional edge cover value and an optimal decomposition can be computed efficiently:

► **Theorem 11** (Lemma 4 and Section 3 in [3]). *For any graph  $H$ , its fractional edge cover value  $\rho(H)$  and an optimal decomposition  $D^*(H)$  can be computed in polynomial time in  $|H|$ .*

### 3 Overview of Our Results and Techniques

We start with describing the ideas behind our upper bound result.

#### 3.1 An algorithm for sampling arbitrary motifs

We take the same approach as that of [23], of sampling towards estimating, but improve on the query complexity of their bound using two ingredients. The first is an improved star sampler, and the second is an improved sampling approach.

**Improved star sampler.** The algorithm of [23] tries to sample  $p$ -stars by sampling  $p$  edges uniformly at random, and checking if they form a star (by simply checking if all  $p$  edges agree on their first endpoint). Hence, each  $p$ -star is sampled with probability  $1/m^p$ . Our first observation is that it is more efficient to sample a *single* edge  $(u, v)$  and then sample  $p - 1$  neighbors of  $v$  uniformly at random, by drawing  $(p - 1)$  indices  $i_1, \dots, i_p$  in  $[d(v)]$  uniformly at random, and performing neighbor queries  $(v, i_j)$  for every  $j \in [p - 1]$ . However, this sampling procedure introduces biasing towards stars that are incident to lower degree endpoints. If we were also given an upper bound  $d_{ub}$  on the maximal degree in the graph, i.e., a value  $d_{ub}$  such that  $d_{max} \leq d_{ub}$ , where  $d_{max}$  is the maximum degree in  $G$ , then we could overcome the above biasing, by “unifying” all the degrees in the graph to  $d_{ub}$ . Specifically, this unification of degrees is achieved by querying the  $i^{\text{th}}$  neighbor of a vertex, where  $i$  is chosen uniformly at random in  $[d_{ub}]$ , rather than in  $[d(v)]$ .<sup>7</sup> By repeating this process  $p - 1$  times, we get that each specific copy of a  $p$ -star is sampled with equal probability  $\frac{1}{m \cdot (d_{ub})^{p-1}}$ . Observe that this is always preferable to  $1/m^p$ , i.e.  $\frac{1}{m \cdot (d_{ub})^{p-1}} > \frac{1}{m^p}$ , since for every graph  $G$ ,  $d_{ub} < m$ . While we are not given such a bound on the maximal degree, letting  $\bar{s}_p$  denote the number of  $p$ -stars in  $G$ , it always holds that  $d_{max} \leq \min\{n, \bar{s}_p^{1/p}\}$  (since every vertex with degree  $d > p$  contributes  $d^p$  to  $\bar{s}_p$ ). Hence, we can use the existing algorithms for star approximations by [26, 2, 17] in order to first get an estimate  $\hat{s}_p$  of  $\bar{s}_p$ , and then use this estimate to get an upper bound  $d_{ub}$  on  $d_{max}$  by setting  $d_{ub} = \min\{n, \hat{s}_p^{1/p}\}$ .

**An improved sampling approach.** In order to describe the second ingredient for improving over the bounds of [23], we first recall their algorithm. In the first step, their algorithm simultaneously attempts to sample a copy of each odd cycle and star in the decomposition of  $H$ . Then if all individual sampling attempt succeed, the algorithm proceeds to check if the sampled copies are connected in  $G$  in a way that is consistent with the non-decomposition edges of  $H$ . However, it is easy to see that this approach is wasteful. Even if all but one of the simultaneous sampling attempts of the first step succeed, the algorithm starts over. For example, if  $D^*(H)$  consists of a star and a triangle, then in the first step their algorithm attempts to sample simultaneously a star and a triangle, and in the case that, say, a triangle is sampled but the star sampling attempt fails, then the sampled triangle is discarded, and the algorithm goes back to the beginning of the first step.

To remedy this, in the first step our algorithm invokes the star- and odd-cycle samplers for every basic component in  $D^*(H)$ , until all samplers return an *actual* copy of of the requested component. This ensures that we proceed to the next step of verifying  $H$  only once we have actual copies of all the basic components. We then continue to check if these copies can be

<sup>7</sup> This is effectively equivalent to rejection sampling where first  $v$  is “kept” with probability  $d(v)/d_{ub}$ , and then a neighbor of  $v$  is sampled uniformly at random.

extended to a copy of  $H$  in  $G$ , as before. While this is a subtle change, it is exactly what allows us to replace the dependency in the maximum number of potential copies of the basic components, to a dependency in the actual number of copies in  $G$ .

We note that for motifs  $H$  whose decomposition has repeating smaller sub-motifs, our sampling approach can be used recursively, which can be more efficient. That is, instead of decomposing  $H$  to its most basic components, stars and odd-cycles, we can consider decomposing it to collections of more complex components. For example, if  $H$  has such a collection  $H_1 \subset H$  that is repeated more than once, then it is more beneficial to first try and sample all of the copies of  $H_1$  (as well as the other components of  $H$ ) and only then try to extend these copies to  $H$ . The sampling of the  $H_1$  copies can then be performed by a recursive call to the motif sampler. It can be shown that for any repeated motif  $H_1$  in the decomposition of  $H$ , applying the recursive sampling process results in an improved upper bound.

### From sampling to estimating

In order to obtain a  $(1 \pm \epsilon)$ -estimate of  $\bar{h}$ , we can use the sampling algorithm as follows. Consider a single sampling attempt in which we first sample all basic components of  $D^*(H)$  (at some cost  $Q$ ), and then perform all pair queries between the components to check if the sampled components induce a copy of  $H$  (at cost  $O(|H|^2)$ ). By the above description such an attempt succeeds with probability that depends on the counts of the basic components of  $D^*(H)$  and on the count  $\bar{h}$ . Hence we can think of the success probability of each attempt as a coin toss with bias  $p$ , where  $p$  depends only on the counts of the components and  $\bar{h}$ . By standard concentration bounds, using  $\Theta(1/(p\epsilon^2))$  sampling attempts, we can compute a  $(1 \pm \epsilon)$ -estimate  $\hat{p}$  of  $p$ . Since we can also get  $(1 \pm \epsilon)$ -multiplicative estimates of the counts of each basic component without asymptotically increasing the running time, we can deduce from  $\hat{p}$  a  $(1 \pm \Theta(\epsilon))$ -estimate of  $\bar{h}$ . See the full version for more details.

## 3.2 Decomposition-optimal lower bounds

**Theorem 6 follows from Theorem 7.** To prove Theorems 6 and 7, we first explain why given Theorem 7, Theorem 6 follows. We then describe at a high level a family of graphs  $\mathcal{G}$  in which sampling copies of a given motif is hard.

At a high level, Theorem 7 states that given (1) a decomposition  $D$  and (2) a set of good counts  $\{\bar{c}_i\}_{i \in [r]}$ , we can construct (3) a motif  $H_D$  (such that  $D$  is an optimal decomposition of  $H_D$ ) and (4) a family of graphs  $\mathcal{G}$  such that expected number of queries required to sampling copies of  $H_D$  in  $\mathcal{G}$  is

$$\max_{i \in [r]} \{cost(C_i)\} \cdot \frac{\prod \bar{c}_i}{\bar{h}}.$$

Theorem 6 states that given (a) a decomposition  $D$  and (b) a (realizable) decomposition cost DC, that there exists (c) a motif  $H_D$  and (d) a family of graphs for which the decomposition-cost of  $G, D$  and  $H_D$  is DC, and sampling copies of  $H_D$  in graphs of  $\mathcal{G}$  requires  $\Omega(\text{DC})$  queries.

To prove that Theorem 6 follows from Theorem 7, we then prove that given (a) and (b), we can specify a set of counts which both satisfies  $\text{DC} = \max_{i \in [r]} \{cost(C_i)\} \cdot \frac{\prod \bar{c}_i}{\bar{h}}$  and which is good. Since the set of counts is good, we can invoke Theorem 7, and get that there exists a motif  $H_D$  and a family of graphs in which it is hard to sample copies of  $H_D$ . Therefore, in the rest of the section we focus our attention on the proof of Theorem 7.

**Ideas behind the proof of Theorem 7.** Given a graph decomposition  $D$ , values  $n, m, \bar{h}$  and a set of counts  $\bar{c}_1, \dots, \bar{c}_r$  of its basic components, our lower bound proof starts by defining a motif  $H_D$ , and a family of graphs  $\mathcal{G}$  such that the following holds.

- The optimal decomposition of  $H_D$  is  $D$ ;
- For every  $G \in \mathcal{G}$  and  $O_{k_i}, S_{p_j} \in D$ , their number of copies in  $G$  is  $\Theta(\bar{o}_{k_i})$  and  $\Theta(\bar{s}_{p_j})$ , respectively;
- The number of copies of  $H$  in  $G$  is  $\Theta(\bar{h})$
- Sampling a uniformly distributed copy of  $H_D$  in a uniformly chosen  $G$  in  $\mathcal{G}$ , requires  $\Omega(\min\{m, DC\})$  queries in expectation.

There are several challenges in proving our lower bound. First, as they are very general and work for any given decomposition  $D$  that contains at least one odd cycle, there are many sub cases that need to be dealt with separately, depending on the mixture of components in  $D$ . Second, the lower bound term does not only depend on the different counts, but also on the relations between them, which determines the component that maximizes  $\text{cost}(C_i)$ . As mentioned previously, our lower bound only holds for the case that the max cost is due to an odd cycle component. It remains an open question whether a similar lower bound can be proven for the case that the max cost is due to a star, or whether in that case a better algorithm exists. The authors suspect the latter option. Third, as in most previous lower bounds for motif sampling and counting, we prove the hardness of the task by “hiding” a constant fraction of the copies of  $H_D$ , so that the existence of these copies depends on a small set of crucial edges. That is, we prove that we can construct the family of graphs  $\mathcal{G}$ , such that for every  $G \in \mathcal{G}$ , a specific set of  $t$  crucial edges, for some small  $t$  that depends on the basic counts and  $\bar{h}$ , contributes  $\Theta(\bar{h})$  copies of  $H_D$ . We then prove that detecting these edges requires many queries (this is formalized by a reduction from a variant of the SET-DISJOINTNESS communication complexity problem, based on the framework of [19]). This approach of constructing many copies of  $H_D$  which all depend on small set of crucial edges, leads the construction of the graphs  $\mathcal{G}$  to contain very dense components, which in turn causes correlations between the counts of the different components. A significant challenge is therefore to define the motif  $H_D$  and the graphs of  $\mathcal{G}$  in a way that satisfies all given counts simultaneously.

In each graph  $G$  in the hard family  $\mathcal{G}$ , we have a corresponding “gadget” to each of the components of  $D$ . Let  $k_1$  denote (one of) the maximum-cost odd-cycle components. For each odd-cycle component  $O_{k_i}$  for  $k_i \neq k_1$ , we define either a **few-cycles-gadget** or a **cycle-gadget** that induce  $\bar{o}_{k_i}$  odd cycles of length  $k_i$  according to the relation between  $k_i$  and  $k_1$ . For each star component  $S_{p_j}$  we define a **star-gadget** that induces  $\bar{s}_{p_j}$  many  $p_j$ -stars. The maximum-cost cycle component  $O_{k_1}$  has a different gadget, a **CC-gadget**. This gadget is used to hide the set of  $t$  crucial edges, and allows us to parameterize the complexity in terms of the cost  $\text{cost}\{O_{k_1}\}$ .

To formally prove the lower bound we make use the framework introduced in [19], which uses reductions from communication complexity problems to motif sampling and counting problems in order to prove hardness results of these latter tasks. This allows us to prove that one cannot, with high probability, witness an edge from the set of  $t$  hidden edges, unless  $\Omega(m/t)$  queries are performed. This in turn implies that one cannot, with high probability, witness a copy of  $H_D$  contributed by these edges. Hence, we obtain a lower of  $\Omega(m/t)$  for the task of outputting a uniformly sampling. Setting  $t$  appropriately gives the desired bound.

## 4 Upper Bounds for Sampling Arbitrary Motifs

In this section we present our improved sampling algorithm. Recall that our upper bound improvement has two ingredients, an improved star sampler, and an improved sampling approach. We start with presenting the improved star sampling algorithm.

### 4.1 An optimal ( $\ell_p$ ) star-sampler

Our star sampling procedure assumes that it gets as a parameter a value  $\hat{s}_p$  which is a constant-factor estimate of  $\bar{s}_p$ . This value can be obtained by invoking one of the star estimation algorithm of [2, 17].

► **Lemma 12** ([2], Theorem 1). *Given query access to a graph  $G$  and an approximation parameter  $\epsilon$ , there exists an algorithm, *Moment-Estimator*, that returns a value  $\hat{s}_p$ , such that with probability at least  $2/3$ ,  $\hat{s}_p \in [\bar{s}_p, 2\bar{s}_p]$ . The expected query complexity and running time  $O\left(\min\left\{m, \min\left\{\frac{m \cdot n^{p-1}}{\bar{s}_p}, \frac{m}{\bar{s}_p^{1/p}}\right\} \cdot \log \log n\right\}\right)$ .*

Given an estimate  $\hat{s}_p$  on  $\bar{s}_p$ , our algorithm sets an upper bound<sup>8</sup>  $d_{ub}$  on the maximal degree,  $d_{ub} = \min\{n, \hat{s}_p\}$ . It then tries to sample a copy of a  $p$ -star as follows. In each sampling attempt it samples a single edge  $(v_0, v_1)$ , and then performs  $p - 1$  neighbor queries  $nbr(v_0, i_j)$  for  $j = 2 \dots p$ , where each  $i_j$  is chosen independently and uniformly at random from  $[d_{ub}]$ . In order to ensure that the sampled neighbors are distinct, and to avoid multiplicity issues, a  $p$ -star is returned only if its petals are sampled in ascending order of ids. In every such sampling attempt, each specific  $p$ -star is therefore sampled with equal probability  $\frac{1}{m \cdot d_{ub}^{p-1}}$ .

Hence, invoking the above  $\frac{m \cdot d_{ub}^{p-1}}{\bar{s}_p}$  times, in expectation, returns a uniformly distributed copy of a  $p$ -star.

**Sample-a-Star**( $p, n, \hat{s}_p$ )

1. Let  $d_{ub} = \min\{n, (c_p \cdot \hat{s}_p)^{1/p}\}$  for a value  $c_p$  as specified in the proof of Theorem 1.
2. While **TRUE**:
  - a. Perform a uniform edge query, and denote the returned edge  $(v_0, v_1)$ .
  - b. Choose  $p-1$  indices  $i_2, \dots, i_p$  uniformly at random in  $[d_{ub}]$  (with replacement).
  - c. For every  $j \in [2..p]$ , query the  $i_j^{\text{th}}$  neighbor of  $v_0$ . Let  $v_2, \dots, v_p$  be the returned vertices, if all queries returned a neighbor. Otherwise break.
  - d. If  $id(v_2) < id(v_3) < \dots < id(v_p)$ , then **return**  $(v_0, v_1, \dots, v_p)$ .

► **Theorem 13.** *Assume that  $\hat{s}_p \in [\bar{s}_p, c \cdot \bar{s}_p]$  for some small constants  $c$ . The procedure **Sample-a-Star**( $p, \hat{s}_p$ ) returns a uniformly distributed  $p$ -star in  $G$ . The expected query complexity of the procedure is  $O\left(\min\left\{\frac{m \cdot n^{p-1}}{\bar{s}_p}, \frac{m}{\bar{s}_p^{1/p}}\right\}\right)$ .*

**Proof.** Let  $c_p$  denote the minimal value such that for every  $k \in [n]$ ,  $c_p \cdot \binom{k}{p} \geq k^p$  (note that  $c_p = \Theta(p!)$ ). Then  $\bar{s}_p = \sum_{v \in V} \binom{d(v)}{p} > \binom{d_{max}}{p} \geq d_{max}^p / c_p$ , and by the assumption on  $\hat{s}_p$ ,  $d_{max} < (c_p \cdot \bar{s}_p)^{1/p} \leq (c_p \cdot \hat{s}_p)^{1/p}$ . It follows by the setting of  $d_{ub} = \min\{n, (c_p \cdot \hat{s}_p)^{1/p}\}$  in Step 1, that  $d_{ub} \geq d_{max}$ .

<sup>8</sup> Observe that  $d_{max}$  is  $d_{max} = \max_v d(v)$ , while  $d_{ub}$  is simply a bound on  $d_{max}$ , so that  $d_{max} \leq d_{ub}$ .



Consider a specific copy  $\bar{S}_p = (a_0, a_1, \dots, a_p)$  of a  $p$ -star in  $G$ , where  $a_0$  is the star center and  $a_1$  through  $a_p$  are its petals in ascending id order. In each iteration of the while loop, the probability that  $\bar{S}_p$  is returned is

$$\begin{aligned} \Pr[\bar{S}_p \text{ is returned}] &= \Pr[(a_0, a_1) \text{ is sampled in Step 2a}] \\ &\quad \cdot \Pr[a_2, \dots, a_p \text{ are sampled in Step 2b}] \\ &= \frac{1}{m} \cdot \frac{1}{d_{ub}^{p-1}}. \end{aligned} \tag{2}$$

Note the the last equality crucially depends on  $d(v) \leq d_{max} \leq d_{ub}$  for all  $v \in V$ . (Indeed, if there exists a vertex  $v$  with degree  $d(v) > d_{ub}$ , then some of its incident stars will have zero probability of being sampled.) Hence, each copy is sampled with equal probability, implying that the procedure returns a uniformly distributed copy of a  $p$ -star.

We now turn to bound the expected query complexity. It follows from Equation 2 and the setting of  $d_{ub}$ , that the success probability of a single invocation of the while loop is  $\frac{\bar{s}_p}{m \cdot d_{ub}^{p-1}}$ . Hence, the expected number of invocations is  $\frac{m \cdot d_{ub}^{p-1}}{\bar{s}_p}$ . It follows that, for a constant  $p$ , the expected number of invocations is

$$O\left(\frac{m \cdot \min\{n, (c_p \cdot \bar{s}_p)^{1/p}\}^{p-1}}{\bar{s}_p}\right) = O\left(\min\left\{\frac{m \cdot n^{p-1}}{\bar{s}_p}, \frac{m}{\bar{s}_p^{1/p}}\right\}\right).$$

Since the query complexity and running time of a single invocation of the while loop are constant, the above is also a bound on the expected query complexity and running time of the while loop.  $\blacktriangleleft$

In the full version of this paper, we explain how algorithm **Sample-a-Star** can be slightly modified to produce an  $\ell_p$ -sampler, **Sublinear- $\ell_p$ -Sampler** as specified in Theorem 2.

## 4.2 General motif sampler

Our algorithm for sampling uniform copies of a motif  $H$  in a graph  $G$  relies on the above star sampler, and the odd cycle sampler of [23].

► **Lemma 14** (Lemma 3.3 in [23], restated). *There exists a procedure that, given a parameter  $k$  and an estimate  $\hat{m} \in [m, 2m]$ , samples each specific copy of an odd cycle of length  $k$  with probability  $1/m^{k/2}$ .*

It follows that by repeatedly invoking the procedure above until an odd cycle is returned we can get an odd cycle sampling algorithm.

► **Corollary 15.** *There exists a procedure, **Sample-Odd-Cycle**, that, given an estimate  $\hat{m} \in [m, 2m]$ , returns a uniformly distributed copy of an odd cycle of length  $k$ . The expected query complexity is  $O\left(\min\left\{m \log n, n + m, \frac{m^{k/2}}{\bar{o}_k}\right\}\right)$ , where  $\bar{o}_k$  denotes the number of odd cycles of length  $k$  in  $G$ .*

We also use the following algorithm from [25] to obtain an estimate of  $m$ .

► **Theorem 16** ([25], Theorem 1, restated). *There exists an algorithm that, given query access to a graph  $G$ , the number of vertices  $n$ , and a parameter  $\epsilon$ , returns a value  $\tilde{m}$ , such that with probability at least  $2/3$ ,  $\tilde{m} \in [m, (1 + \epsilon)m]$ . The expected query complexity and running time of the algorithm is  $O(n/\sqrt{m}) \cdot (\log \log n/\epsilon^2)$ .*

Our motif sampling algorithm invokes the star-sampler and odd-cycles-sampler for each of the star and odd-cycles components in  $D^*(H)$ , respectively. Once actual copies of all the components are sampled, it checks whether they form a copy of  $H$  in  $G$ , using  $O(|H|^2) = O(1)$  additional pair queries.

**Sample- $H$  ( $H, n$ )**

1. Compute a 2-factor estimate  $\hat{m}$  of  $m$  by invoking the algorithm of [25] with  $\epsilon = 1/2$  for  $10 \log n$  times, and letting  $\hat{m}$  be the median of the returned values.
2. Compute an optimal decomposition of  $H$ ,  $D^*(H) = \{O_{k_1}, \dots, O_{k_q}, S_{p_1}, \dots, S_{p_\ell}\}$ .
3. For every  $S_{p_i}$  in  $D$ , invoke algorithm **Moment-Estimator** with  $\epsilon = 1/2$  and  $r = p_i$  for  $t = 10 \log(n \cdot \ell)$  times to get  $t$  estimates of  $\bar{s}_{p_i}$ . Let  $\hat{s}_{p_i}$  be the median value among the  $t$  received estimates of each  $S_{p_i}$ .
4. While **True**:
  - a. For every  $i \in [q]$  do:
    - i. Invoke **Sample-Odd-Cycle**( $k_i, \hat{m}$ ), and let  $\bar{O}_i$  be the returned odd cycle.
  - b. For every  $i \in [\ell]$  do:
    - i. Invoke **Sample-a-Star**( $p_i, n, \hat{s}_{p_i}$ ), and let  $\bar{S}_i$  be the returned  $s_j$ -star.
  - c. Perform  $O(|H|^2)$  pair queries to verify whether the set of components  $\{\bar{O}_1, \dots, \bar{O}_q, \bar{S}_1, \dots, \bar{S}_\ell\}$  can be extended to a copy of  $H$  in  $G$ .
  - d. If a copy of  $H$  is discovered, then **return** it.
  - e. If the number of queries performed exceeds  $n + \hat{m}$ , then query all edges of the graph<sup>a</sup> and output a uniformly distributed copy of  $H$ .

<sup>a</sup> by either performing  $n + 2m$  degree and neighbor queries, or  $10m \log n$  uniform edge queries

We are now ready to prove our main upper bound theorem, which we recall here.

► **Theorem 17.** *Let  $G$  be a graph over  $n$  vertices and  $m$  edges, and let  $H$  be a motif such that  $D^*(H) = \{O_{k_1}, \dots, O_{k_q}, S_{p_1}, \dots, S_{p_\ell}\} = \{C_i\}_{i \in [r]}$ . There exists an algorithm, **Sample- $H$** , that returns a copy of  $H$  in  $G$ . With probability at least  $1 - 1/\text{poly}(n)$ , the returned copy is uniformly distributed in  $G$ . The expected query complexity of the algorithm is*

$$O(\min\{\text{DECOMP-COST}(G, H, D^*(H)), m\}) \cdot \log n \log \log n.$$

**Proof.** By Theorem 16, when invoked with a value  $\epsilon = 1/2$ , the edge estimation algorithm of [25] returns a value  $\tilde{m}$  such that, with probability at least  $2/3$ ,  $\tilde{m} \in [m, 1.5m]$ . Hence, with probability at least  $1 - 1/3n^2$ , the median value  $\hat{m}$  of the  $10 \log n$  invocations is such that  $\hat{m} \in [m, 1.5m]$ . We henceforth condition on this event.

We next prove that with probability at least  $1 - 1/3n^2$ , all the computed  $\hat{s}_{p_i}$  values are good estimates of  $\bar{s}_{p_i}$ . By Lemma 12, for a fixed  $p_i$ , with probability at least  $2/3$ , the value returned from **Moment-Estimator** is in  $[\hat{s}_{p_i}, 1.5 \cdot \hat{s}_{p_i}]$ . Therefore, the probability that the median value of the  $t = 10 \log(n\ell)$  invocations in Step 3 is outside this range is at most  $1/(3\ell n^2)$ . Hence, taking a union bound over all  $i \in [\ell]$ , with probability at least  $1 - 1/3n^2$ , for every  $i \in [\ell]$ ,  $\hat{s}_{p_i} \in [\bar{s}_{p_i}, 1.5 \cdot \bar{s}_{p_i}]$ . We henceforth condition on this event as well.

Fix a copy  $H'$  of  $H$  in  $G$ , and let  $O'_1, \dots, O'_q, S'_1, \dots, S'_\ell$  be its cycles and stars, corresponding to those of  $D^*(H)$ . By Corollary 15, for each  $O'_i$ , its probability of being returned in Step 4(a)i is  $1/\bar{o}_{k_i}$ . Similarly, by Lemma 1, for each  $S'_i$ , its probability of being returned in Step 4(b)i is  $1/\bar{s}_{p_i}$ . Therefore, in the case that the number of queries does not exceed  $\hat{m}$ , in every iteration

of the loop, each specific copy of  $H$  is returned with equal probability  $\frac{1}{\prod_{i=1}^q \bar{o}_{k_i} \cdot \prod_{i=1}^{\ell} \bar{s}_{p_i}}$ .<sup>9</sup> Hence, once a copy of  $H$  is returned, it is uniformly distributed in  $G$ . In the case that the number of queries exceeds  $\hat{m}$ , the algorithm either performs  $n + 2m$  queries to query all the neighbors of all vertices, or  $10m \log n$  queries, in order to discover all edges with high probability. In the former case, the entire graph  $G$  is known. In the latter case, by the coupon collector analysis, the probability that all edges are known at the end of the process is at least  $1 - 1/3n^2$ . Hence, with probability at least  $1 - 1/3n^2$ , at the end of this process, a uniformly distributed copy of  $H$  is returned.

It remains to bound the query complexity. By Lemma 12, Step 3 takes  $\sum_{p_i} t \cdot \min \left\{ \frac{m \cdot n^{p_i-1}}{\bar{s}_{p_i}}, \frac{m}{\bar{s}_{p_i}^{1/p_i}} \right\} \cdot \log n \log \log n$  queries in expectation. By the above discussion, it holds that the expected number of invocations of the while loop is  $\frac{\prod_{i=1}^q \bar{o}_{k_i} \cdot \prod_{i=1}^{\ell} \bar{s}_{p_i}}{\bar{h}}$ . Furthermore, by Lemma 1, the expected query complexity of sampling each  $S_{p_i}$  is  $\min \left\{ \frac{m \cdot n^{p_i-1}}{\bar{s}_{p_i}}, \frac{m}{\bar{s}_{p_i}^{1/p_i}} \right\}$ . By Lemma 15, the expected running time of each invocation of the  $k_i$ -cycle sampler is  $O \left( \frac{m^{k_i/2}}{\bar{o}_{k_i}} \right)$ . The complexity of Step 4c is  $O(|H|^2) = O(1)$  queries, and is subsumed by the complexity of the other steps. Hence, the expected cost of each invocation of the while loop is

$$\max_{i \in [q]} \left\{ \frac{m^{k_i/2}}{\bar{o}_{k_i}} \right\} + \max_{i \in [\ell]} \left\{ \min \left\{ \frac{m}{\bar{s}_{p_i}^{1/p_i}}, \frac{m \cdot n^{p_i-1}}{\bar{s}_{p_i}} \right\} \right\} = \max_{i \in [q]} \left\{ \frac{m^{k_i/2}}{\bar{o}_{k_i}} \right\} + \min \left\{ \frac{m}{\bar{s}_p^{1/p}}, \frac{m \cdot n^{p-1}}{\bar{s}_p} \right\},$$

where the equality holds since the maximum of the second term is always achieved by the largest star in the decomposition,  $S_p$ . Also, due to Step 4e and the assumption on  $\hat{m}$ , the query complexity of algorithm is always bounded by  $O(\min\{m \log n, n + m\})$ . Therefore, the overall expected query complexity is the minimum between  $O(\min\{m \log n, n + m\})$  and

$$\begin{aligned} & O \left( \left( \max_{i \in [q]} \left\{ \frac{m^{k_i/2}}{\bar{o}_{k_i}} \right\} + \min \left\{ \frac{m \cdot n^{p-1}}{\bar{s}_p}, \frac{m}{\bar{s}_p^{1/p}} \right\} \cdot \log n \log \log n \right) \cdot \frac{\prod_{i \in [r]} \bar{c}_i}{\bar{h}} \right) \\ &= O \left( \min \left\{ \max_{i \in [r]} \{ \text{cost}(C_i) \} \cdot \frac{\prod \bar{c}_i}{\bar{h}}, m \right\} \cdot \log n \log \log n \right) \\ &= O \left( \min \{ \text{DECOMP-COST}(G, H, D^*(H)), m, n \} \cdot \log n \log \log n \right), \end{aligned}$$

as claimed.  $\blacktriangleleft$

---

## References

- 1 Nesreen K Ahmed, Jennifer Neville, Ryan A Rossi, and Nick Duffield. Efficient graphlet counting for large networks. In *2015 IEEE International Conference on Data Mining*, pages 1–10. IEEE, 2015.
- 2 Maryam Aliakbarpour, Amartya Shankha Biswas, Themis Gouleakis, John Peebles, Ronitt Rubinfeld, and Anak Yodpinyanee. Sublinear-time algorithms for counting star subgraphs via edge sampling. *Algorithmica*, 80(2):668–697, 2018.
- 3 Sepehr Assadi, Michael Kapralov, and Sanjeev Khanna. A Simple Sublinear-Time Algorithm for Counting Arbitrary Subgraphs via Edge Sampling. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*, volume 124 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:20, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITCS.2019.6.

---

<sup>9</sup> To avoid multiplicity issues, if some components are repeated in the decomposition more than once, then we can assign ids to small components and verify they are sampled in ascending id order.

- 4 Albert Atserias, Martin Grohe, and Dániel Marx. Size bounds and query plans for relational joins. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 739–748. IEEE, 2008.
- 5 Haim Avron. Counting triangles in large graphs using randomized matrix trace estimation. In *Workshop on Large-scale Data Mining: Theory and Applications*, volume 10, pages 10–9, 2010.
- 6 Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge estimation with independent set oracles. *arXiv preprint arXiv:1711.07567*, 2017.
- 7 Suman K. Bera, Noujan Pashanasangi, and C. Seshadhri. Linear time subgraph counting, graph degeneracy, and the chasm at size six. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, pages 38:1–38:20, 2020. doi:10.4230/LIPIcs.ITCS.2020.38.
- 8 Amartya Shankha Biswas, Talya Eden, and Ronitt Rubinfeld. Towards a decomposition-optimal algorithm for counting and sampling arbitrary motifs in sublinear time, 2021. arXiv:2107.06582.
- 9 Andreas Bjöklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Counting paths and packings in halves. *Algorithms - ESA 2009*, page 578–586, 2009. doi:10.1007/978-3-642-04128-0\_52.
- 10 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 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2916–2935, 2020. doi:10.1137/1.9781611975994.177.
- 11 Graham Cormode and Hossein Jowhari. L p samplers and their applications: A survey. *ACM Computing Surveys (CSUR)*, 52(1):1–31, 2019.
- 12 Maximilien Danisch, Oana Balalau, and Mauro Sozio. Listing k-cliques in sparse real-world graphs. In *Proceedings of the 2018 World Wide Web Conference*, pages 589–598. International World Wide Web Conferences Steering Committee, 2018.
- 13 Talya Eden, Amit Levi, Dana Ron, and C Seshadhri. Approximately counting triangles in sublinear time. *SIAM Journal on Computing*, 46(5):1603–1646, 2017.
- 14 Talya Eden, Dana Ron, and Will Rosenbaum. The arboricity captures the complexity of sampling edges. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece.*, pages 52:1–52:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.52.
- 15 Talya Eden, Dana Ron, and Will Rosenbaum. Almost optimal bounds for sublinear-time sampling of k-cliques: Sampling cliques is harder than counting. *arXiv preprint arXiv:2012.04090*, 2020.
- 16 Talya Eden, Dana Ron, and C. Seshadhri. On approximating the number of k-cliques in sublinear time. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 722–734, 2018. doi:10.1145/3188745.3188810.
- 17 Talya Eden, Dana Ron, and C. Seshadhri. Sublinear time estimation of degree distribution moments: The arboricity connection. *SIAM J. Discrete Math.*, 33(4):2267–2285, 2019. doi:10.1137/17M1159014.
- 18 Talya Eden, Dana Ron, and C. Seshadhri. Faster sublinear approximation of the number of k-cliques in low-arboricity graphs. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1467–1478, 2020. doi:10.1137/1.9781611975994.89.
- 19 Talya Eden and Will Rosenbaum. Lower bounds for approximating graph parameters via communication complexity. In Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20-22, 2018 - Princeton, NJ, USA*, volume

- 116 of *LIPICs*, pages 11:1–11:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.APPROX-RANDOM.2018.11.
- 20 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.
  - 21 Patrick Eichenberger, Masaya Fujita, Shane T Jensen, Erin M Conlon, David Z Rudner, Stephanie T Wang, Caitlin Ferguson, Koki Haga, Tsutomu Sato, Jun S Liu, et al. The program of gene transcription for a single differentiating cell type during sporulation in bacillus subtilis. *PLoS biology*, 2(10):e328, 2004.
  - 22 Uriel Feige. On sums of independent random variables with unbounded variance and estimating the average degree in a graph. *SIAM Journal on Computing*, 35(4):964–984, 2006.
  - 23 Hendrik Fichtenberger, Mingze Gao, and Pan Peng. Sampling arbitrary subgraphs exactly uniformly in sublinear time. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, pages 45:1–45:13, 2020. doi:10.4230/LIPICs.ICALP.2020.45.
  - 24 Jacob Fox, Tim Roughgarden, C. Seshadhri, Fan Wei, and Nicole Wein. Finding cliques in social networks: A new distribution-free model. *SIAM J. Comput.*, 49(2):448–464, 2020. doi:10.1137/18M1210459.
  - 25 Oded Goldreich and Dana Ron. Approximating average parameters of graphs. *Random Structures & Algorithms*, 32(4):473–493, 2008. doi:10.1002/rsa.20203.
  - 26 Mira Gonen, Dana Ron, and Yuval Shavitt. Counting stars and other small subgraphs in sublinear-time. *SIAM Journal on Discrete Mathematics*, 25(3):1365–1411, 2011.
  - 27 Shweta Jain and C. Seshadhri. A fast and provable method for estimating clique counts using turán’s theorem. In *Conference on the World Wide Web*, pages 441–449, 2017.
  - 28 Krzysztof Juszczyszyn, Przemysław Kazienko, and Katarzyna Musiał. Local topology of social network based on motif analysis. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 97–105. Springer, 2008.
  - 29 Tali Kaufman, Michael Krivelevich, and Dana Ron. Tight bounds for testing bipartiteness in general graphs. *SIAM Journal on Computing*, 33(6):1441–1483, 2004. doi:10.1137/S0097539703436424.
  - 30 Tong Ihn Lee, Nicola J Rinaldi, François Robert, Duncan T Odom, Ziv Bar-Joseph, Georg K Gerber, Nancy M Hannett, Christopher T Harbison, Craig M Thompson, Itamar Simon, et al. Transcriptional regulatory networks in saccharomyces cerevisiae. *science*, 298(5594):799–804, 2002.
  - 31 Wenzhe Ma, Ala Trusina, Hana El-Samad, Wendell A Lim, and Chao Tang. Defining network topologies that can achieve biochemical adaptation. *Cell*, 138(4):760–773, 2009.
  - 32 DE Nelson, AEC Ihekweba, M Elliott, JR Johnson, CA Gibney, BE Foreman, G Nelson, V See, CA Horton, DG Spiller, et al. Oscillations in nf- $\kappa$ b signaling control the dynamics of gene expression. *Science*, 306(5696):704–708, 2004.
  - 33 Duncan T Odom, Nora Zizlsperger, D Benjamin Gordon, George W Bell, Nicola J Rinaldi, Heather L Murray, Tom L Volkert, Jörg Schreiber, P Alexander Rolfe, David K Gifford, et al. Control of pancreas and liver gene expression by hnf transcription factors. *Science*, 303(5662):1378–1381, 2004.
  - 34 Rasmus Pagh and Charalampos E Tsourakakis. Colorful triangle counting and a mapreduce implementation. *Information Processing Letters*, 112:277–281, 2012.
  - 35 Ashwin Paranjape, Austin R Benson, and Jure Leskovec. Motifs in temporal networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 601–610. ACM, 2017.
  - 36 Shai S Shen-Orr, Ron Milo, Shmoolik Mangan, and Uri Alon. Network motifs in the transcriptional regulation network of escherichia coli. *Nature genetics*, 31(1):64, 2002.
  - 37 Jakub Tětek and Mikkel Thorup. Sampling and counting edges via vertex accesses, 2021.

- 38 Alexandru Topirceanu, Alexandra Duma, and Mihai Udrescu. Uncovering the fingerprint of online social networks using a network motif based approach. *Computer Communications*, 73:167–175, 2016.
- 39 Charalampos E Tsourakakis. Fast counting of triangles in large real networks without counting: Algorithms and laws. In *International Conference on Data Mining*, pages 608–617, 2008.
- 40 Jakub Tětek. Approximate triangle counting via sampling and fast matrix multiplication. *CoRR*, abs/2104.08501, 2021. [arXiv:2104.08501](https://arxiv.org/abs/2104.08501).
- 41 John J Tyson and Béla Novák. Functional motifs in biochemical reaction networks. *Annual review of physical chemistry*, 61:219–240, 2010.
- 42 Virginia Vassilevska. Efficient algorithms for clique problems. *Information Processing Letters*, 109(4):254–257, 2009. doi:10.1016/j.ipl.2008.10.014.
- 43 Qiankun Zhao, Yuan Tian, Qi He, Nuria Oliver, Ruoming Jin, and Wang-Chien Lee. Communication motifs: a tool to characterize social communications. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1645–1648. ACM, 2010.

## A Related Work

We note that some of the works were mentioned before, but we repeat them here for the sake of completeness. Over the past decade, there has been a growing body of work investigating the questions of approximately counting and sampling motifs in sublinear time. These questions were considered for various motifs  $H$ , classes of  $G$ , and query models.

The study of sublinear time estimation of motif counts was initiated by the works of Feige [22] and of Goldreich and Ron [25] on approximating the average degree in general graphs. Feige [22] investigated the problem of estimating the average degree of a graph, denoted  $d_{\text{avg}}$ , when given query access to the degrees of the vertices. By performing a careful variance analysis, Feige proved that  $O\left(\sqrt{n/d_{\text{avg}}}/\epsilon\right)$  queries are sufficient in order to obtain a  $(\frac{1}{2} - \epsilon)$ -approximation of  $d_{\text{avg}}$ . He also proved that a better approximation ratio cannot be achieved in sublinear time using only degree queries. The same problem was then considered by Goldreich and Ron [25]. Goldreich and Ron proved that an  $(1 + \epsilon)$ -approximation can be achieved with  $O\left(\sqrt{n/d_{\text{avg}}}\right) \cdot \text{poly}(1/\epsilon, \log n)$  queries, if neighbor queries are also allowed. Building on these ideas, Gonen et al. [26] considered the problem of approximating the number of  $s$ -stars in a graph. Their algorithm only assumed neighbor and degree queries. In [2], Aliakbarpour, Biswas, Gouleakis, Peebles, and Rubinfeld and Yodpinyanee considered the same problem of estimating the number of  $s$ -stars in the augmented edge queries model, which allowed them to circumvent the lower bounds of [26] for this problem. In [17], Eden, Ron and Seshadhari again considered this problem, and presented improved bound for the case where the graph  $G$  has bounded arboricity. In [13, 16, 18], Eden, Ron and Seshadhari considered the problems of estimating the number of  $k$ -cliques in general and in bounded arboricity graphs, in the general graph query model, and gave matching upper and lower bounds. In [40], Tětek considers both the general and the augmented query models for approximately counting triangles in the super-linear regime. In [19], Eden and Rosenbaum presented a framework for proving motif counting lower bounds using reduction from communication complexity, which allowed them to reprove the lower bounds for all of the variants listed above.

In [20, 14], Eden and Rosenbaum and Ron has initiated the study of sampling motifs (almost) uniformly at random. They considered the general graph query model, and presented upper and matching lower bounds up to  $\text{poly}(\log n/1/\epsilon)$  factors, for the task of sampling edges almost uniformly at random, both for general graphs and bounded arboricity graphs. Recently, Tětek and Thorup [37] presented an improved analysis which reduced the dependency in

$\epsilon$  to  $\log(1/\epsilon)$ . This result implies that for all practical applications, the edge sampler is essentially as good as a truly uniform sampler. They also proved that given access to what they refer to as hash-based neighbor queries, there exists an algorithm that samples from the exact uniform distribution. The authors of [14] also raised the question of approximating vs. sampling complexity, and gave preliminary results that there exists motifs  $H$  (triangles) and classes of graphs  $G$  (bounded arboricity graphs) in which approximating the number of  $H$ 's is strictly easier than sampling an almost uniformly distributed copy of  $H$ . This question was very recently resolved by them, proving a separation for the tasks of counting and uniformly sampling cliques in bounded arboricity graphs [15].

A significant result was achieved recently, when Assadi, Kapralov and Khanna gave an algorithm for approximately counting the number of copies of any given general  $H$ , in the edge queries augmented query model. They also gave a matching lower bound for the case that  $H$  is an odd cycle. Fichtenberger, Gao and Peng presented a cleaner algorithm with a much simplified analysis for the same problem, that also returns a uniformly distributed copy of  $H$ .

Another query model was suggested recently by Beame et al. [6], which assumes access to only *independent set* (IS) queries or *bipartite independent set* (BIS) queries. Inspired by group testing, IS queries allow to ask whether a given set  $A$  is an independent set, and BIS queries allow to ask whether two sets  $A$  and  $B$  have at least one edge between them. In this model they considered the problem of estimating the average degree and gave an  $O(n^{2/3}) \cdot \text{poly}(\log n)$  algorithm using IS queries, and  $\text{poly}(\log n)$  algorithm using BIS queries. Chen, Levi and Waingarten [10] later improved the first bound to  $O(n/\sqrt{m}) \cdot \text{poly}(\log n)$  and also proved it to be optimal.





# Ideal-Theoretic Explanation of Capacity-Achieving Decoding

Siddharth Bhandari ✉

Tata Institute of Fundamental Research, Mumbai, India

Prahladh Harsha ✉ 

Tata Institute of Fundamental Research, Mumbai, India

Mrinal Kumar ✉

Department of Computer Science and Engineering, IIT Bombay, India

Madhu Sudan ✉

School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA

---

## Abstract

In this work, we present an abstract framework for some algebraic error-correcting codes with the aim of capturing codes that are list-decodable to capacity, along with their decoding algorithm. In the polynomial ideal framework, a code is specified by some ideals in a polynomial ring, messages are polynomials and their encoding is the residue modulo the ideals. We present an alternate way of viewing this class of codes in terms of linear operators, and show that this alternate view makes their algorithmic list-decodability amenable to analysis.

Our framework leads to a new class of codes that we call *affine Folded Reed-Solomon codes* (which are themselves a special case of the broader class we explore). These codes are common generalizations of the well-studied Folded Reed-Solomon codes and Univariate Multiplicity codes, while also capturing the less-studied Additive Folded Reed-Solomon codes as well as a large family of codes that were not previously known/studied.

More significantly our framework also captures the algorithmic list-decodability of the constituent codes. Specifically, we present a unified view of the decoding algorithm for ideal-theoretic codes and show that the decodability reduces to the analysis of the distance of some related codes. We show that good bounds on this distance lead to capacity-achieving performance of the underlying code, providing a unifying explanation of known capacity-achieving results. In the specific case of affine Folded Reed-Solomon codes, our framework shows that they are list-decodable up to capacity (for appropriate setting of the parameters), thereby unifying the previous results for Folded Reed-Solomon, Multiplicity and Additive Folded Reed-Solomon codes.

**2012 ACM Subject Classification** Mathematics of computing → Coding theory

**Keywords and phrases** List Decodability, List Decoding Capacity, Polynomial Ideal Codes, Multiplicity Codes, Folded Reed-Solomon Codes

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.56

**Category** RANDOM

**Related Version** *Full Version:* <https://arxiv.org/abs/2103.07930>

*Full Version:* <https://eccc.weizmann.ac.il/report/2021/036/>

**Funding** *Siddharth Bhandari & Prahladh Harsha:* Research supported by the Department of Atomic Energy, Government of India, under project 12-R&D-TFR-5.01-0500 and in part by the Google PhD and Swarnajayanti Fellowships.

*Madhu Sudan:* Supported in part by a Simons Investigator Award and NSF Award CCF 1715187.



© Siddharth Bhandari, Prahladh Harsha, Mrinal Kumar, and Madhu Sudan; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 56; pp. 56:1–56:21



Leibniz International Proceedings in Informatics

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

## 1 Introduction

Reed-Solomon codes are obtained by evaluations of polynomial of degree less than  $k$  at  $n$  distinct points in a finite field  $\mathbb{F}$ . Folded-Reed-Solomon (FRS) codes are obtained by evaluating a polynomial at  $sn$  (carefully chosen) points that are grouped into  $n$  bundles of size  $s$  each, and then viewing the resulting  $sn$  evaluations as  $n$  elements of  $\mathbb{F}^s$ . Multiplicity codes are obtained by evaluating the polynomial, and  $s - 1$  of its derivatives and again viewing the resulting  $sn$  evaluations as  $n$  elements of  $\mathbb{F}^s$ .

This “bundling” (or folding, as it is called for FRS codes) in FRS codes and Multiplicity codes may be viewed at best as a harmless operation – it does not hurt the rate and (relative) distance of a code which is already optimal in these parameters. But far from merely being harmless, in the context algorithmic list-decoding, bundling has led to remarkable improvements and to two of the very few explicit capacity achieving codes in the literature. Indeed the only other codes that achieve list-decoding capacity algorithmically and do not use one of the above codes as an ingredient are the Folded Algebraic-Geometric codes, which also use bundling. Despite this central role, the bundling operation is not well-understood algebraically. Indeed it seems like an “ad hoc” operation rather than a principled one. Unearthing what bundling is and understanding when and why it turns out to be so powerful is the primary goal of this work, and we make some progress towards this.

Turning to the algorithms for list-decoding the above codes close to capacity, there are two significantly different ones in the literature. A (later) algorithm due to Guruswami and Wang [6]<sup>1</sup> which seems more generalizable, and the original algorithm of Guruswami and Rudra [3] which is significantly more challenging to apply to multiplicity codes (see [9]). In both cases, while the algorithm for FRS works in all (reasonable) settings, the algorithms for multiplicity codes only work when the characteristic of the field is larger than the degrees of the polynomials in question. Looking more closely at FRS codes, part of the careful choice of bundling in FRS codes is to pick each bundle to be a geometric progression. If one were to switch this to an arithmetic progression, then one would get a less-studied family codes called the Additive-FRS. It turns out the Additive-FRS codes are also known to be list-decodable to capacity but only via the original algorithm. Thus, the short summary of algorithmic list-decoding is that there is no short summary! Algorithms tend to work but we need to choose carefully and read the fine print.

The goal of this paper is to provide a unifying algebraic framework that (a) captures bundling algebraically, (b) captures most of the algorithmic success also algebraically, leaving well-defined parts for combinatorial analysis and (c) leads to new codes that also achieve capacity. In this work we use basic notions from linear algebra and polynomial rings to present a unifying definition (see Definitions 3.1 and 4.4) that captures the codes very generally, and also the decoding ability (see Theorem 1.1). We elaborate on these below.

### Polynomial ideal codes

Our starting point is what we term “polynomial ideal codes”. A *polynomial ideal code* over a finite field  $\mathbb{F}$  and parameters  $k, s$  is specified by  $n$  pairwise relatively prime monic polynomials  $E_0(X), \dots, E_{n-1}(X) \in \mathbb{F}[X]$  of degree equal to  $s$ .<sup>2</sup> The encoding maps a message  $p \in \mathbb{F}^k$

<sup>1</sup> We note that the Guruswami-Wang algorithm is inspired by an idea due to Vadhan [12, Theorem 5.24] that shows that it suffices to interpolate a polynomial  $Q$  which is linear in the  $y$ -variables. However, the algorithm from [12] is not applicable to our setting since it uses polynomial factorization as well as analysis tools that are specific to Reed-Solomon codes. The further simplifications developed in [6] are key to the applicability in our setting.

<sup>2</sup> Here  $\mathbb{F}[X]$  refers to the ring of univariate polynomials in the variable  $X$  over the field  $\mathbb{F}$  while  $\mathbb{F}_{<k}[X]$  refers to the vector-space of polynomials in  $\mathbb{F}[X]$  of degree strictly less than  $k$ .

(interpreted as a polynomial of degree less than  $k$ ) to  $n$  symbols as follows:

$$\begin{aligned} \mathbb{F}_{<k}[X] &\longrightarrow (\mathbb{F}_{<s}[X])^n \\ p(X) &\longmapsto (p(X) \pmod{E_i(X)})_{i=0}^{n-1} \end{aligned}$$

The codes described above, Reed-Solomon, FRS, Multiplicity and Additive-FRS, are all examples of polynomial ideal codes. For Reed-Solomon codes, this is folklore knowledge: the evaluation point  $a_i$  corresponds to going mod  $E_i(X) = (X - a_i)$ . For any bundling of the Reed-Solomon codes this follows by taking product of the corresponding polynomials. For multiplicity codes of order  $s$ , the evaluation of a polynomial and its derivatives at  $a_i$  corresponds to going modulo  $E_i(X) = (X - a_i)^s$ .

The abstraction of polynomial ideal codes is not new to this work. Indeed Guruswami, Sahai and Sudan [4, Appendix A] already proposed these codes as a good abstraction of algebraic codes. Their framework is even more general, in particular they even consider non-polynomial ideals such as in  $\mathbb{Z}$ . They suggest algorithmic possibilities but do not flesh out the details. In this work we show that polynomial ideal codes, as we define them, are indeed list-decodable up to the Johnson radius. We note that the proof involves some steps not indicated in the previous work but for the most part this confirms the previous thinking.

The abstraction above also captures “bundling” (or folding) nicely - we get them by choosing  $E_i(X)$  to be a product of some  $E_{ij}(X)$ . But the above abstraction thus far fails to capture the capacity-achieving aspects of the codes (i.e., the benefits of this bundling) and the decoding algorithms. This leads us to the two main *novel* steps of this paper:

- We present an alternate viewpoint of polynomial ideal *codes* in terms of *linear operators*.
- We abstract the Guruswami-Wang linear-algebraic list-decoding *algorithm* in terms of *linear operators*.

The two sets of “linear operators”, in the codes and in the decoding algorithm, are not the same. But the linearity of both allows them to interact nicely with each other. We elaborate further below after introducing them.

### Linear operator codes

In this work, a linear operator is an  $\mathbb{F}$ -linear function  $L : \mathbb{F}[X] \rightarrow \mathbb{F}[X]$ . A *linear operator code* is characterized by a family of linear operators  $\mathcal{L} = (L_0, \dots, L_{s-1})$ , a set  $A = \{a_0, \dots, a_{n-1}\} \subseteq \mathbb{F}$  of evaluation points and  $k$  a degree parameter such that  $k \leq s \cdot n$ . The corresponding linear operator code, denoted by  $LO_k^A(\mathcal{L})$ , is given as follows:

$$\begin{aligned} \mathbb{F}_{<k}[X] &\longrightarrow (\mathbb{F}^s)^n \\ p(X) &\longmapsto (\mathcal{L}(p)(a_i))_{i=0}^{n-1} \end{aligned}$$

Linear operator codes easily capture polynomial ideal codes. For instance, the multiplicity codes are linear operator codes wherein the linear operators are the successive derivative operators. But they are also too general – even if we restrict the operators to map  $\mathbb{F}_{<k}[X]$  to itself, an operator allows  $k^2$  degrees of freedom.

We narrow this broad family by looking subfamilies of linear operators and codes. The specific subfamily we turn are what we call “ideal linear operators”. We say that linear operators  $L_0, \dots, L_{s-1}$  are *ideal linear operators* with respect to a set  $A$  of evaluation points if for every  $a \in A$ , the vector space

$$I^a(\mathcal{L}) = \{p \in \mathbb{F}[X] \mid \mathcal{L}(p)(a) = \bar{0}\}$$

is an ideal. (When the set of evaluation points is clear from context, we drop the phrase “with respect to  $A$ ”.) Linear operator codes corresponding to ideal linear operators are called *ideal linear operator codes* (see Definitions 4.1 and 4.4 for precise definitions).

It is not hard to see that a family of linear operators  $\mathcal{L} = (L_0, \dots, L_{s-1})$  has the ideal property if it satisfies the following *linearly-extendibility* property: There exists a matrix  $M(X) \in \mathbb{F}[X]^{s \times s}$  such that for all  $p \in \mathbb{F}[X]$  we have

$$\mathcal{L}(X \cdot p(X)) = M(X) \cdot \mathcal{L}(p(X)).$$

This motivates yet another class of linear operators and code: We say that an operator family  $\mathcal{L}$  is a *linearly-extendible linear operator* if such a matrix  $M(X)$  exists and the resulting code is said to be a *linearly-extendible linear operator code* (see Definitions 4.2 and 4.4 for precise definitions).

It turns out that these three definitions of codes – polynomial ideal codes, ideal linear operator codes and linearly-extendible linear operator codes – are equivalent (see Propositions 4.6 and 4.8 and Corollary 4.9). And while the notion of polynomial ideal codes captures the codes mentioned thus far naturally, the equivalent notion of linearly-extendible codes provides a path to understanding the applicability of the linear-algebraic list-decoding algorithm of Guruswami and Wang.

While it is not the case that every linearly-extendible linear operator code (and thus every polynomial ideal code) is amenable to this list-decoding algorithm, it turns out that one can extract a nice sufficient condition on the linear-extendibility for the algorithm to be well-defined. This allows us to turn the question of list-decodability into a quantitative one – how many errors can be corrected. And the linear operator framework now converts this question into analyzing the rank of an associated matrix.

The sufficient condition we extract is the following: we say that an operator  $L : \mathbb{F}[X] \rightarrow \mathbb{F}[X]$  is degree-preserving if  $\deg_X(Lf) \leq \deg_X(f)$  for all  $f \in \mathbb{F}[X]$ . Observe that any degree-preserving linear operator when restricted to  $\mathbb{F}_{<k}[X]$  can be represented by an upper-triangular matrix in  $\mathbb{F}^{k \times k}$ . A family of linear operators obtained by repeated iteration,  $\mathcal{L} = (L = L^0, L = L^1, L^2, \dots, L^{s-1})$  is called an *iterative* family. We associate with any degree-preserving family  $\mathcal{L} = (L_0, \dots, L_{s-1})$  of linear operators a simple matrix in  $\mathbb{F}^{s \times k}$  called  $\text{Diag}(\mathcal{L})$ , whose  $i$ th row is the diagonal of  $L_i$  and consider the code in  $\mathbb{F}^k$  generated by  $\text{Diag}(\mathcal{L})$ .

The following theorem now shows that for any degree-preserving iterative linearly-extendible operator codes, lower bound on the distance of  $\text{Diag}(\mathcal{L})$  yields an upper bound on the list size obtained by the Guruswami-Wang algorithm, even when the number of errors approaches  $(1 - \text{rate})$  of the code.

► **Theorem 1.1.** *Suppose  $L : \mathbb{F}[X] \rightarrow \mathbb{F}[X]$  is a degree-preserving linear operator and  $A$  a set of evaluation points such that for  $\mathcal{L} = (L^0, L^1, \dots, L^{s-1})$  the corresponding code  $\mathcal{C}$  is a linearly-extendible linear operator code. Furthermore, if the matrix  $\text{Diag}(\mathcal{L}) \in \mathbb{F}^{s \times k}$  formed by stacking the diagonals of the  $s$  linear operators as the rows is the generator matrix of a code with distance  $1 - \frac{\ell}{k}$ , then,  $\mathcal{C}$  is list-decodable up to the distance  $1 - \frac{k}{(s-m+1)n} - \frac{1}{m}$  with list size  $q^\ell$  for any  $1 \leq m \leq s$ .*

We remark that our actual theorem is more general (see Theorem 5.2) where we further separate the role of linear operators used to build the code, from those that seed the decoding algorithm. But it immediately implies Theorem 1.1 above, which in turn already suffices to capture the capacity achieving decodability of FRS, multiplicity and additive-FRS codes. Regarding the aspect of the need to lower bound the distance of the code generated by

$\text{Diag}(\mathcal{L})$ , to bound the list size of the codes, we stress that for each of these codes the lower bound on the distance follows from fairly simple arguments. Indeed the generality of the arguments allows us to capture broader families of codes uniformly, as described next.

### A Common Generalization

Our framework leads very naturally to a *new* class of codes that we call the *Affine Folded Reed-Solomon* (Affine-FRS) codes: these are codes defined by ideals of the form  $\prod_{i=0}^{s-1} (X - \ell^{(i)}(a))$  where  $\ell(z) = \alpha z + \beta$  is any linear form and  $\ell^{(i)}(z) = \underbrace{\ell(\ell \dots \ell(z) \dots)}_{i \text{ times}}$ . These codes generalize

all the previously considered codes: The case  $\ell(z) = \gamma z$  are the FRS codes, the case  $\ell(z) = z$  are the Multiplicity codes, and the case  $\ell(z) = z + \beta$  are the Additive FRS codes!

► **Theorem 1.2** (Informal statement – see Theorem A.8). *Let  $\ell$  be any linear form such that either  $\text{ord}(\ell) \geq k$  or  $(\text{char}(\mathbb{F}) \geq k \text{ and } \beta \neq 0)$ <sup>3</sup>. Then the Affine-FRS codes corresponding to the linear form  $\ell$  are list-decodable up to capacity.*

Previously, even for the special case of the Additive FRS codes, list-decodability close to capacity was only achieved by the more involved algorithm of Guruswami & Rudra [3] and Kopparty [9] (see paragraph on Additive Folding and Footnote 4 in [2, Section III]). (A similar approach can be extended to cover the case of  $\text{ord}(\ell) \geq k$  in Theorem 1.2: however, it seems difficult to do so for the case when  $\text{ord}(\ell)$  is small.)

Thus, our Affine-FRS codes lead to the first common abstraction of the three codes as well as the first common algorithm for solving the list-decoding problem for these codes. (Furthermore, this algorithm is linear-algebraic.)<sup>4</sup> Arguably thus, even if the Affine-FRS codes had been studied previously, it is not clear that the ability to decode them for every choice of  $\ell(z)$  would be obvious.

## Organization

The rest of the paper is organized as follows. We begin with some preliminaries in Section 2. We then formally define polynomial ideal codes and linear operator codes in Sections 3 and 4 respectively. In Section 5, we discuss list-decoding algorithms for polynomial ideal codes. We first present the list-decoding algorithm for *all* polynomial ideal codes up to the Johnson radius in Section 5.1 and then the list-decoding algorithm beyond the Johnson radius for special families of linear operator codes in Section 5.2. Finally, we conclude by demonstrating how these results can be used to show that several well-known families of codes (Folded Reed-Solomon, multiplicity, additive Folded Reed-Solomon codes) as well as their common generalization affine folded Reed-Solomon achieve list-decoding capacity in Appendix A.

Throughout the paper, we skip the proofs of various claims due to space constraints. We refer the interested reader to the full version of the paper [1] for the complete proofs.

## 2 Notations and Preliminaries

We start with some notations that we follow in the rest of this paper.

- For a natural number  $n$ ,  $[n]$  denotes the set  $\{0, 1, \dots, n-1\}$ .
- $\mathbb{F}$  denotes a field.

<sup>3</sup>  $\text{ord}(\ell)$  refers to the smallest positive integer  $u$  such that  $\ell^{(u)}(z) = z$ .

## 56:6 Ideal-Theoretic Explanation of Capacity-Achieving Decoding

- For  $a, b, i, j \in \mathbb{Z}$ , where  $a, b, i, j \geq 0$  the bivariate monomial  $X^i Y^j$  is said to have  $(a, b)$ -weighted degree at most  $d$  if  $ai + bj \leq d$ .  $N(a, b)$  denotes the number of bivariate monomials of  $(1, a)$ -weighted degree at most  $b$ .
- For  $a, b \in \mathbb{Z}$ , a bivariate polynomial  $Q(X, Y)$  is said to have  $(a, b)$ -weighted degree at most  $d$ , if it is supported on monomials of  $(a, b)$ -weighted degree at most  $d$ .
- We say that a function  $f(n) : \mathbb{N} \rightarrow \mathbb{N}$  is **poly**( $n$ ), if there are constants  $c, n_0 \in \mathbb{N}$  such that for all  $n \geq n_0$ ,  $f(n) \leq n^c$ .
- $\mathbb{F}[X]$  is the ring of univariate polynomials with coefficients in  $\mathbb{F}$ , and for every  $k \in \mathbb{N}$ ,  $\mathbb{F}_{<k}[X]$  denotes the set of polynomials in  $\mathbb{F}[X]$  of degree strictly less than  $k$ .
- For a multivariate polynomial  $f(X_0, X_1, \dots, X_{n-1}) \in \mathbb{F}[X_0, X_1, \dots, X_{n-1}]$ ,  $\deg_{X_i}(f)$  denotes the degree of  $f$ , when viewing it as a univariate in  $X_i$ , with coefficients in the polynomial ring on the remaining variables over the field  $\mathbb{F}$ .

### Estimates on $N(a, b)$

We rely on the following simple lemma to estimate the number of bivariate monomials with  $(1, a)$ -weighted degree at most  $b$ . See the full version [1] for the proof.

► **Lemma 2.1.** *For every  $a, b \in \mathbb{N}$ , let  $N(a, b)$  denote the number of bivariate monomials with  $(1, a)$ -weighted degree at most  $b$ . Then, the following are true.*

1.  $N(a - 1, b) \geq b^2/2a$ .
2. For every  $\eta \in \mathbb{N}$ , if  $a$  divides  $b$ , then

$$N(a, b) - N(a, b - a\eta) - \eta(b - a\eta + 1) = a\eta(\eta + 1)/2.$$

### Johnson radius

► **Theorem 2.2** (List decoding up to Johnson radius). *Let  $q \in \mathbb{N}$  be a natural number. Any code with block length  $n$  and relative distance  $\delta$  over an alphabet of size  $q$  is (combinatorially) list decodable from  $(1 - \sqrt{1 - \delta})$  fraction of errors with list size at most  $n^2 q \delta$ .*

We have the following bound for codes, referred to popularly as the *Singleton bound* [11], though the bound appears earlier in the works of Joshi [7] and Komamiya [8].

► **Theorem 2.3** (Komamiya-Joshi-Singleton bound). *The rate  $R$  and the relative distance  $\delta$  of a code satisfy  $R + \delta \leq 1 + o(1)$ .*

In particular, for codes which lie on the Komamiya-Joshi-Singleton bound, we have that they are combinatorially list decodable from  $1 - \sqrt{R} - o(1)$  fraction errors with polynomial list size.

### List-decoding upto capacity

► **Definition 2.4** (List-decoding Capacity). *Consider a family of codes  $\mathcal{C} = \{C_1, \dots, C_n, \dots\}$  where  $C_n$  has rate  $\rho_n$  and block length  $n$  with alphabet  $\Sigma_n$ . Then,  $\mathcal{C}$  is said to achieve list-decoding capacity if  $\forall \varepsilon > 0$  there exists an  $n_0$  such that  $\forall n \geq n_0$  and all received words  $w \in \Sigma_n$ , there exists at most a polynomial number of codewords  $c \in C_n$  such that  $\delta(c, w) \leq (1 - \rho_n(1 + \varepsilon))$ .*

*Further, if there exists an efficient algorithm for finding all these codewords, then,  $\mathcal{C}$  is said to achieve list-decoding capacity efficiently. Ideally, we want to keep  $\Sigma_n$  as small as possible.*

### Chinese remainder theorem

We also rely on the following version of the Chinese Remainder Theorem for the polynomial ring.

► **Theorem 2.5.** *Let  $E_0(X), E_1(X), \dots, E_{s-1}(X)$  be univariate polynomials of degree equal to  $d$  over a field  $\mathbb{F}$  such that for every distinct  $i, j \in [s]$ ,  $E_i$  and  $E_j$  are relatively prime. Then, for every  $s$ -tuple of polynomials  $(r_0(X), \dots, r_{s-1}(X)) \in \mathbb{F}[X]^s$  such that each  $r_i$  is of degree strictly less than  $d$  (or zero), there is a unique polynomial  $p(X) \in \mathbb{F}[X]$  of degree at most  $d^s - 1$  such that for all  $i \in [s]$ ,*

$$p(X) = r_i(X) \pmod{E_i(X)}.$$

### Polynomial ideals

► **Definition 2.6.** *A subset  $I \subseteq \mathbb{F}[X]$  of polynomials is said to be an ideal if the following are true.*

- $0 \in I$ .
- For all  $p(X), q(X) \in I$ ,  $p + q \in I$ .
- For every  $p(X) \in I$  and  $q(X) \in \mathbb{F}[X]$ ,  $p(X) \cdot q(X) \in I$ .

For the univariate polynomial ring  $\mathbb{F}[X]$ , we also know that every ideal  $I$  is principal, i.e. there exists a polynomial  $p(X) \in I$  such that

$$I = \{p(X)q(X) : q(X) \in \mathbb{F}[X]\}.$$

## 3 Polynomial ideal codes

In this section, we discuss polynomial ideal codes in more detail, and see how this framework captures some of the well studied families of algebraic error correcting codes.

We start with the formal definition of polynomial ideal codes.

► **Definition 3.1 (polynomial ideal codes).** *Given a field  $\mathbb{F}$ , parameters  $s, k$  and  $n$  satisfying  $k < s \cdot n$ , the polynomial ideal code is specified by a family of  $n$  polynomials  $E_0, \dots, E_{n-1}$  in the ring  $\mathbb{F}[X]$  of univariate polynomials over the field  $\mathbb{F}$  satisfying the following properties.*

1. For all  $i \in [n]$ , polynomial  $E_i$  has degree exactly  $s$ .
2. The  $E_i$ 's are monic polynomials.
3. The polynomials  $E_i$ 's are pairwise relatively prime.

The encoding of the polynomial ideal code maps is as follows:

$$\begin{aligned} \mathbb{F}_{<k}[X] &\longrightarrow (\mathbb{F}_{<s}[X])^n \\ p(X) &\longmapsto (p(X) \pmod{E_i(X)})_{i=0}^{n-1} \end{aligned}$$

As is clear from the definition, polynomial ideal codes are linear over  $\mathbb{F}$  and have rate  $k/sn$  and relative distance  $(1 - (k - 1)/sn)$ . Since the sum of rate and relative distance satisfy the Komamiya-Joshi-Singleton bound, these codes are maximal-distance separable (MDS) codes.

We note that in general,  $E_i$ 's need not have the same degree, but for notational convenience, we work in the setting when each of them is of degree equal to  $s$ . We also note that these codes continue to be well defined even if the  $E_i$ 's are not relatively prime. In this case, the condition,  $k < s \cdot n$  is replaced by  $k$  being less than the degree of the lowest common multiple of  $E_0, E_1, \dots, E_{n-1}$ . However, the distance of the code suffers in this case, and such codes need not approach the Komamiya-Joshi-Singleton bound. We now observe that some of the standard and well studied family of algebraic error correcting codes are in fact instances of polynomial ideal codes for appropriate choice of  $E_0, E_1, \dots, E_{n-1}$ .

### 3.1 Some well known codes via polynomial ideals

The message space for all these codes is identified with univariate polynomials of degree at most  $k - 1$  in  $\mathbb{F}[X]$ . We assume that the underlying field  $\mathbb{F}$  is of size at least  $n$  for this discussion, else, we work over a large enough extension of  $\mathbb{F}$ .

#### Reed-Solomon Codes

Let  $a_0, a_1, \dots, a_{n-1}$  be  $n$  distinct elements of  $\mathbb{F}$ . In a Reed-Solomon code, we encode a message polynomial  $p(X) \in \mathbb{F}[X]_{<k}$  by its evaluation on  $a_0, a_1, \dots, a_{n-1}$ . To view these as a polynomial ideal code, observe that  $p(a_i) = p(X) \pmod{(X - a_i)}$ . Thus, we can set the polynomials  $E_i(X)$  in Definition 3.1 to be equal to  $(X - a_i)$  for each  $i \in [n]$ . Thus,  $s = 1$ . Clearly, the  $E_i$ 's are relatively prime since  $a_0, a_1, \dots, a_{n-1}$  are distinct.

#### Folded Reed-Solomon Codes [3]

Let  $\gamma \in \mathbb{F}_q^*$  be an element of multiplicative order at least  $s$ , i.e.  $\gamma^0, \gamma, \dots, \gamma^{s-1}$  are all distinct field elements. Further, let the set of evaluation points be  $A = \{a_0, \dots, a_{n-1}\}$  such that for any two distinct  $i$  and  $j$  the sets  $\{a_i, a_i\gamma, \dots, a_i\gamma^{s-1}\}$  and  $\{a_j, a_j\gamma, \dots, a_j\gamma^{s-1}\}$  are disjoint. In a Folded Reed-Solomon code, with block length  $n$  and folding parameter  $s$  is defined by the following encoding function.

$$p(X) \mapsto \left( p(a_i), p(a_i\gamma^1), \dots, p(a_i\gamma^{s-1}) \right)_{i=0}^{n-1}$$

Thus, these are codes over the alphabet  $\mathbb{F}^s$ .

To view these as polynomial ideal codes, we set  $E_i(X) = \prod_{j=0}^{s-1} (X - a_i\gamma^j)$ . Clearly, each such  $E_i$  is a polynomial of degree equal to  $s$ , and since for any two distinct  $i$  and  $j$  the sets  $\{a_i, a_i\gamma, \dots, a_i\gamma^{s-1}\}$  and  $\{a_j, a_j\gamma, \dots, a_j\gamma^{s-1}\}$  are disjoint, the polynomials  $E_0, E_1, \dots, E_{n-1}$  are all relatively prime.

To see the equivalence between these two viewpoints observe that  $p(a_i\gamma^j) = p(X) \pmod{(X - a_i\gamma^j)}$ . Moreover,  $(X - a_i\gamma^j)$  are all relatively prime as  $j$  varies in  $[s]$  for every  $i \in [n]$ . Thus, by the Chinese Remainder Theorem over  $\mathbb{F}[X]$ , there is a bijection between remainders of a polynomial modulo  $\{(X - a_i\gamma^j) : j \in [s]\}$  and the remainder modulo the product  $E_i = \prod_{j \in [s]} (X - a_i\gamma^j)$  of these polynomials.

#### Additive Folded Reed-Solomon Codes [3]

Additive Folded Reed-Solomon codes are a variant of the Folded Reed-Solomon codes defined above. Let  $\mathbb{F}_q$  have characteristic at least  $s$  and let  $\beta \in \mathbb{F}_q^*$ . Further, let the set of evaluation points be  $A = \{a_0, \dots, a_{n-1}\}$  where  $a_i - a_j \notin \{0, \beta, 2\beta, \dots, (s-1)\beta\}$  for distinct  $i$  and  $j$ . Here,  $s$  denotes the folding parameter. The encoding is defined as follows.

$$p(X) \mapsto \left( p(a_i), p(a_i + \beta), \dots, p(a_i + \beta(s-1)) \right)_{i=0}^{n-1}$$

Thus, these are also codes over the alphabet  $\mathbb{F}^s$ .

To view these as polynomial ideal codes, we set  $E_i(X) = \prod_{j=0}^{s-1} (X - a_i + \beta j)$ . Clearly, each such  $E_i$  is a polynomial of degree equal to  $s$ , and since  $a_i - a_j \notin \{0, \beta, 2\beta, \dots, (s-1)\beta\}$  for distinct  $i$  and  $j$ , the polynomials  $E_0, E_1, \dots, E_{n-1}$  are all relatively prime.

To see the equivalence between the two definitions, the argument is again identical to that for Folded Reed-Solomon codes discussed earlier in this section. We just observe  $(X - a_i + \beta j)$  are all relatively prime  $j$  varies in  $[s]$  for every  $i \in [n]$ , and thus by the Chinese Remainder



Theorem over  $\mathbb{F}[X]$ , there is a bijection between remainders of a polynomial modulo  $\{(X - a_i + \beta j) : j \in [s]\}$  and the remainder modulo the product  $E_i = \prod_{j \in [s]} (X - a_i + \beta j)$  of these polynomials.

### Univariate Multiplicity Codes [10]

Univariate multiplicity codes, or simply multiplicity codes are a variant of Reed-Solomon, where in addition to the evaluation of the message polynomial at every  $a_i$ , we also give the evaluation of its derivatives of up to order  $s - 1$ . While they can be defined over all fields, for the exposition in this paper, we consider these codes over fields  $\mathbb{F}$  of characteristic at least  $sn$ . Moreover, we also work with the standard derivatives (from analysis), as opposed to Hasse derivatives which is typically the convention in coding theoretic context. Let  $a_0, a_1, \dots, a_{n-1} \in \mathbb{F}$  be distinct field elements.

The encoding is defined as follows.

$$p(X) \mapsto \left( p(a_i), \frac{\partial p}{\partial X}(a_i), \dots, \frac{\partial^{s-1} p}{\partial X^{s-1}}(a_i) \right)_{i=0}^{n-1}$$

Here,  $\frac{\partial^j p}{\partial X^j}$  denotes the (standard)  $j$ th order derivative of  $p$  with respect to  $X$ .

To view these as polynomial ideal codes, we set  $E_i(X) = (X - a_i)^s$ . Clearly, each such  $E_i$  is a polynomial of degree equal to  $s$ , and since  $a_i$ 's are all distinct, these polynomials  $E_0, E_1, \dots, E_{n-1}$  are all relatively prime.

The equivalence of these two definitions follows from an application of Taylor's theorem to univariate polynomials, which says the following.

$$\begin{aligned} p(X) &= p(a_i + X - a_i) \\ &= p(a_i) + (X - a_i) \frac{\partial p}{\partial X}(a_i) + \dots + \frac{1}{(s-1)!} (X - a_i)^{s-1} \frac{\partial^{s-1} p}{\partial X^{s-1}}(a_i) + (X - a_i)^s \cdot q(X) \end{aligned}$$

for some polynomial  $q(X) \in \mathbb{F}[X]$ . Thus,

$$p(X) \pmod{(X - a_i)^s} = p(a_i) + (X - a_i) \frac{\partial p}{\partial X}(a_i) + \dots + \frac{1}{(s-1)!} (X - a_i)^{s-1} \frac{\partial^{s-1} p}{\partial X^{s-1}}(a_i).$$

Therefore,  $p(X) \pmod{(X - a_i)^s}$  we can *read* off the evaluations of the derivatives of  $p$  of order up to  $s - 1$  at  $a_i$  by explicitly writing  $p(X) \pmod{(X - a_i)^s}$  as a polynomial in  $(X - a_i)$  (via interpolation for instance), and reading off the various coefficients. Similarly, using the above expression, given the evaluation of all the derivatives of order up to  $s - 1$  of  $p$  at  $a_i$ , we can also reconstruct  $p(X) \pmod{(X - a_i)^s}$ .

### Affine Folded Reed-Solomon Codes

We now describe a common generalization of the codes defined above, which we call Affine Folded Reed-Solomon Codes. Fix integers  $k, n, q$  with  $n \leq q$ . Let  $\alpha \in \mathbb{F}_q^*$  and  $\beta \in \mathbb{F}_q$  such that the multiplicative order of  $\alpha$  is  $u$ . Further, define  $\ell(X) = \alpha X + \beta$  and

$$\ell^{(i)}(X) = \underbrace{\ell(\ell \dots \ell(X))}_{i \text{ times}} = \alpha^i X + \beta \cdot \sum_{j=0}^{i-1} \alpha^j = \alpha_i X + \beta_i.$$

In fact, if  $\alpha \neq 1$ , i.e,  $u > 1$  then,  $\beta_u = \beta \cdot \sum_{j=0}^{u-1} \alpha^j = 0$  and hence  $\ell^{(u)}(X) = \ell^{(0)}(X)$ . Let  $\text{ord}(\ell)$  denote the smallest positive integer  $t$  such that  $\ell^{(t)}(X) = X$ . The message space of the Affine Folded Reed-Solomon code of degree  $k$  with block length  $n$  and folding

## 56:10 Ideal-Theoretic Explanation of Capacity-Achieving Decoding

parameter  $s$  is polynomials of degree at most  $k - 1$  over  $\mathbb{F}[X]$ , i.e.,  $\mathbb{F}_{<k}[X]$  where  $\mathbb{F} = \mathbb{F}_q$ . Let the set of evaluation points be  $A = \{a_0, \dots, a_{n-1}\}$  such that for distinct  $i, j$  the sets  $\{\ell^{(0)}(a_i), \dots, \ell^{(s-1)}(a_i)\}$  and  $\{\ell^{(0)}(a_j), \dots, \ell^{(s-1)}(a_j)\}$  are disjoint.

The encoding function of Affine Folded Reed-Solomon Codes is given as: (Recall that  $t = \text{ord}(\ell)$ ; let  $s = v \cdot t + r$  where  $r < t$ .)

$$p(X) \mapsto \left( \begin{array}{cccccc} p(\ell^{(0)}(a_i)) & \frac{\partial p}{\partial X}(\ell^{(0)}(a_i)) & \dots & \frac{\partial^{v-1} p}{\partial X^{v-1}}(\ell^{(0)}(a_i)) & \frac{\partial^v p}{\partial X^v}(\ell^{(0)}(a_i)) & \\ \vdots & \vdots & \dots & \vdots & \vdots & \\ \vdots & \vdots & \dots & \vdots & \vdots & \frac{\partial^v p}{\partial X^v}(\ell^{(r-1)}(a_i)) \\ p(\ell^{(t-1)}(a_i)) & \frac{\partial p}{\partial X}(\ell^{(t-1)}(a_i)) & \dots & \frac{\partial^{v-1} p}{\partial X^{v-1}}(\ell^{(t-1)}(a_i)) & & \end{array} \right)_{i=0}^{n-1}.$$

Thus, these are also codes over the alphabet  $\mathbb{F}^s$ .

To view these as polynomial ideal codes we set

$$E_i(X) = \prod_{j=0}^{s-1} (X - \alpha_j a_i - \beta_j) = \prod_{j=0}^{r-1} (X - \ell^{(j)}(a_i))^{v+1} \cdot \prod_{j=r}^{t-1} (X - \ell^{(j)}(a_i))^v.$$

For the choice of  $A$  as above, the polynomials  $E_i = E(X, a_i)$  are pairwise co-prime. Similar to the previous cases of Folded/Additive Reed-Solomon and Multiplicity codes we have a bijection between the remainders of a polynomial modulo  $E_i$  and the encoding of the polynomial at  $a_i$ .

### 3.2 An alternate definition

We now discuss an alternate definition of polynomial ideal codes; the advantage being that this definition ties together the polynomials  $E_0, E_1, \dots, E_{n-1}$  into a single bivariate polynomial. This would be useful later on when we discuss the connection between polynomial ideal codes and linear operator codes.

► **Definition 3.2** (polynomial ideal codes (in terms of bivariate polynomials)). *Given a field  $\mathbb{F}$ , parameters  $s, k$  and  $n$  satisfying  $k < s \cdot n$ , the polynomial ideal code is specified by a bivariate polynomial  $E(X, Y)$  over the field  $\mathbb{F}$  and a set of  $n$  field elements  $a_0, a_1, \dots, a_{n-1}$  in  $\mathbb{F}$  satisfying the following properties.*

1.  $\deg_X E(X, Y) = s$ .
2.  $E(X, Y)$  is a monic polynomial in the variable  $X$ .
3. The polynomials  $E(X, a_i)$ 's are pairwise relatively prime.

*Since  $E$  is monic and has (exact) degree  $s$  in the variable  $X$ , any polynomial  $p \in \mathbb{F}[X]$  has the following unique representation.*

$$p(X) = Q^{(p)}(X, Y) \cdot E(X, Y) + R^{(p)}(X, Y) \quad \text{where } \deg_X(R^{(p)}(X, Y)) < s.$$

*The encoding of the polynomial ideal code maps  $p$  as follows:*

$$\begin{aligned} \mathbb{F}_{<k}[X] &\longrightarrow (\mathbb{F}_{<s}[X])^n \\ p(X) &\longmapsto \left( R^{(p)}(X, a_i) \right)_{i=0}^{n-1}. \end{aligned}$$

The equivalence of Definitions 3.1 and 3.2 is not hard to see. We summarize this in the simple observation below.

► **Observation 3.3.** *Definitions 3.1 and 3.2 are equivalent.*

**Proof.** Given a code as per Definition 3.1, we can view this as a code according to Definition 3.2 by picking  $n$  distinct  $a_0, a_1, \dots, a_{n-1} \in \mathbb{F}$  (or in a large enough extension of  $\mathbb{F}$  of size at least  $n$ ) and use standard Lagrange interpolation to find a bivariate polynomial  $E(X, Y)$  such that for every  $i \in [n]$ ,

$$E(X, a_i) = E_i.$$

More precisely, we define  $E(X, Y)$  as follows.

$$E(X, Y) := \sum_{i \in [n]} \left( \prod_{j \in [n] \setminus \{i\}} \frac{(Y - a_j)}{(a_j - a_i)} \right) \cdot E_i(X).$$

Clearly,  $E(X, a_i)$ 's are relatively prime, and their degree in  $X$  equals  $s$  and  $E(X, Y)$  is monic in  $X$ . The equivalence of the encoding function also follows immediately from the definitions.

The other direction is even simpler. Given a code as per Definition 3.2, we can view this as a code as per Definition 3.1 by just setting  $E_i(X)$  to be equal to  $E(X, a_i)$  for every  $i \in [n]$ . The condition on the degree of  $E_i$  and their relative primality follows immediately from the fact that  $E(X, Y)$  is monic in  $X$  of degree  $s$ , and  $E(X, a_i)$ 's are relatively prime. Once again, the encoding map can be seen to be equivalent in both the cases. ◀

From Observation 3.3 and the discussion in Section 3.1, the Reed-Solomon codes, Folded Reed-Solomon codes, Additive Folded Reed-Solomon codes and Multiplicity codes can also be viewed as polynomial ideal codes as per Definition 3.2.

- **Reed-Solomon codes:** We take  $E(X, Y)$  to be equal to  $(X - Y)$ , the set of points  $a_0, \dots, a_{n-1}$  remain the same.
- **Folded Reed-Solomon codes:** We take  $E(X, Y) = \prod_{j \in [s]} (X - \gamma^j Y)$  and the set of evaluation points  $a_0, \dots, a_{n-1}$  are set as before, and  $\gamma \in \mathbb{F}^*$  is an element of high enough order.
- **Additive Folded Reed-Solomon codes:** We take  $E(X, Y) = \prod_{j \in [s]} (X - Y + \beta j)$  and the set of evaluation points  $a_0, \dots, a_{n-1}$  are set as before. Recall that  $\mathbb{F}$  is taken to be a field of characteristic at least  $s$  for these codes.
- **Multiplicity codes:** We take  $E(X, Y)$  to be equal to  $(X - Y)^s$ , the set of points  $a_0, \dots, a_{n-1}$  are distinct.
- **Affine Folded Reed-Solomon codes:** We take  $E(X, Y) = \prod_{i=0}^{s-1} (X - \ell^{(i)}(Y))$  where  $\ell(Y) = \alpha Y + \beta$  with  $\alpha \in \mathbb{F}_q^*$  and  $\beta \in \mathbb{F}_q$ . Recall that the set of evaluation points  $A = \{a_0, \dots, a_{n-1}\}$  is such that for distinct  $i, j$  the sets  $\{\ell^{(0)}(a_i), \dots, \ell^{(s-1)}(a_i)\}$  and  $\{\ell^{(0)}(a_j), \dots, \ell^{(s-1)}(a_j)\}$  are disjoint.

It follows immediately from these definitions that all the desired properties in Definition 3.2 are indeed satisfied. We skip the remaining details.

## 4 Linear operator codes

In this section, we give an alternate viewpoint of polynomial ideal codes in terms of codes defined based on linear operators on the ring of polynomials.

► **Definition 4.1** (linear operators). *Let  $\mathcal{L} = (L_0, \dots, L_{s-1})$  be a of  $s$  linear operators where each  $L_i : \mathbb{F}[X] \rightarrow \mathbb{F}[X]$  is a  $\mathbb{F}$ -linear operator over the ring  $\mathbb{F}$ . For any  $f \in \mathbb{F}[X]$ , it will be convenient to denote by  $\mathcal{L}(f)$  the (row) vector  $(L_0(f), \dots, L_{s-1}(f)) \in \mathbb{F}[X]^s$ .*

## 56:12 Ideal-Theoretic Explanation of Capacity-Achieving Decoding

Given any such family  $\mathcal{L}$  and element  $a \in \mathbb{F}$ , define

$$I^a(\mathcal{L}) = \{p(X) \in \mathbb{F}[X] \mid \mathcal{L}(p)(a) = \bar{0}\}.$$

If the family  $\mathcal{L}$  of linear operators family and the set of field elements  $A \subseteq \mathbb{F}$  further satisfy the property that  $I^a(\mathcal{L})$  is an ideal for each  $a \in A$ , we refer to the family  $\mathcal{L}$  as an ideal family of linear operators with respect to  $A$ .

In this case, since  $\mathbb{F}[X]$  is a principal ideal domain, for each  $a \in A$ ,  $I^a(\mathcal{L}) = \langle E^a(\mathcal{L})(X) \rangle$  for some monic polynomial  $E^a(\mathcal{L}) \in F[X]$ .

We now define a special condition on the family of linear operators  $\mathcal{L}$  which will help us capture when  $I^a(\mathcal{L})$  forms an ideal.

► **Definition 4.2** (linearly-extendible linear operators). *The family  $\mathcal{L}$  of linear operators is said to be linearly-extendible if there exists a matrix  $M(X) \in \mathbb{F}[X]^{s \times s}$  such that for all  $p \in F[X]$  we have*

$$\mathcal{L}(X \cdot p(X)) = M(X) \cdot \mathcal{L}(p(X)). \quad (1)$$

We give two examples to illustrate the definition:

- Let  $L_0(f(X)) = f(X)$  and  $L_1(f(X)) = f'(X)$  where  $f'$  is the formal derivative of  $f$ . Then, by the product rule  $L_1(Xf(X)) = X \cdot f'(X) + f(X)$ . Hence, in this case  $M(X) = \begin{pmatrix} X & 0 \\ 1 & X \end{pmatrix}$ .
- Let  $L_0(f(X)) = f(X)$  and  $L_1(f(X)) = f(\gamma X)$  where  $\gamma \in \mathbb{F}_q$  is non-zero. Then, we have  $L_1(Xf(X)) = \gamma X f(\gamma X)$ . Hence, in this case  $M(X) = \begin{pmatrix} 1 & 0 \\ 0 & \gamma X \end{pmatrix}$ .

► **Observation 4.3.** *Suppose  $\mathcal{L}$  is linearly-extendible and  $M(X)$  is the corresponding matrix from Equation (1).*

- *For any  $j \geq 0$  we have  $\mathcal{L}(X^j \cdot p(X)) = (M(X))^j \cdot \mathcal{L}(p(X))$ . Thus, by linearity we have that for any  $q \in \mathbb{F}[X]$ :*

$$\mathcal{L}(q(X) \cdot p(X)) = q(M(X)) \cdot \mathcal{L}(p(X)).$$

*For instance if  $q(X) = X^j$  then  $\mathcal{L}(X^j \cdot p(X)) = (M(X))^j \cdot \mathcal{L}(p(X))$ .*

- *The family  $\mathcal{L}$  is completely specified by  $\mathcal{L}(1)$  and  $M(X)$ . In other words,  $\mathcal{L}(p(X)) = p(M(X)) \cdot \mathcal{L}(1)$ .*
- *For every set  $A$  of evaluation points,  $\mathcal{L}$  is an ideal family of linear operators with respect to  $A$ . This is because if at a point  $a$  we have  $\mathcal{L}(p)(a) = 0$  then  $\mathcal{L}(Xp)(a) = (M(X) \cdot \mathcal{L}(p(X)))(a) = M(X = a) \cdot \mathcal{L}(p)(a) = 0$ . This means that if  $p(X) \in I^a(\mathcal{L})$  then  $Xp(X) \in I^a(\mathcal{L})$ , and hence by linearity for any  $q(X) \in \mathbb{F}[X]$  we have  $q(X) \cdot p(X) \in I^a(\mathcal{L})$ .*

► **Definition 4.4** (linear operator codes). *Let  $\mathcal{L} = (L_0, \dots, L_{s-1})$  be a family of linear operators,  $A = \{a_1, \dots, a_n\} \subseteq \mathbb{F}$  be a set of evaluation points and  $k$  a degree parameter such that  $k \leq s \cdot n$ . Then the linear operator code generated by  $\mathcal{L}$  and  $A$ , denoted by  $LO_k^A(\mathcal{L})$ , is given as follows:*

$$\begin{aligned} \mathbb{F}_{<k}[X] &\longrightarrow (\mathbb{F}^s)^n \\ p(X) &\longmapsto (\mathcal{L}(p)(a_i))_{i=1}^n. \end{aligned}$$

- *If  $\mathcal{L}$  is an ideal family of linear operators with respect to  $A$  where the polynomials  $E_i := E^{a_i}(\mathcal{L})$ , which are the monic generator polynomials for the ideals  $I^{a_i}(\mathcal{L})$ , further satisfy the following:*

1. *For all  $i \in [n]$ , polynomial  $E_i$  has degree exactly  $s$ .*
2. *The polynomials  $E_i$ 's are pairwise relatively prime.*

Then the linear operator code is said to be an ideal linear operator code and denoted by  $ILO_k^A(\mathcal{L})$ .

- If the ideal linear operator code  $ILO_k^A(\mathcal{L})$  further satisfies that  $\mathcal{L}$  is linearly-extendible, then the ideal linear operator code is said to be a linearly-extendible linear operator code, denoted by  $LELO_k^A(\mathcal{L})$ .

► **Remark 4.5.** The rate of the  $LO_k^A(\mathcal{L})$  code is  $k/sn$ . Further, if the code is an ideal linear operator code, i.e.,  $ILO_k^A(\mathcal{L})$ , then its distance is  $1 - \frac{k-1}{sn}$ . Hence,  $ILO_k^A(\mathcal{L})$  is an MDS code.

► **Proposition 4.6.** Any polynomial ideal code is a linearly-extendible linear operator code.

See the full version [1] for a proof.

► **Remark 4.7.** (degree preserving) If the bivariate polynomial  $E(X, Y)$  has total degree  $s$ , then, the linear operator in the  $LELO$  code obtained above has the property that  $\deg_X L_i(X^j) \leq j$ : in fact,  $\deg_X L_i(X^j) \leq j - i$ .

► **Proposition 4.8.** Any ideal linear operator code is a polynomial ideal code.

**Proof.** Consider an ideal linear operator code  $ILO_k^A(\mathcal{L})$ . For any polynomial  $p(X) \in \mathbb{F}[X]$  and a point  $a_i \in A$ , giving  $\mathcal{L}(p(X))(a)$  is equivalent to giving  $p(X) \bmod \langle E_i \rangle$  where  $\langle E_i \rangle = I^{a_i}(\mathcal{L})$ . However, the  $E_i$ s readily satisfy Definition 3.1. ◀

Now, we state a corollary which further corroborates the notion of linear-extendibility.

► **Corollary 4.9** (Equivalence of  $ILO$  and  $LELO$ ). From Propositions 4.6 and 4.8 it follows that every ideal linear operator code is also a linearly-extendible linear operator code.

Below we state some well known codes in their linear operator descriptions (a more formal treatment is given in *Appendix A*):

- **Reed-Solomon Codes:** Let  $A = \{a_0, \dots, a_{n-1}\}$  be distinct elements in  $\mathbb{F}_q$ . These are  $LELO_{\mathcal{L}, A}$  where  $\mathcal{L} = (I)$ . That is the encoding of the message polynomial  $p(X) \in \mathbb{F}_{<k}[X]$  at a point  $a$  is  $L(f(X))(a) = f(a)$ .
- **Folded Reed-Solomon Codes:** Let  $\gamma \in \mathbb{F}_q^*$  with multiplicative order at least  $s$ .  $FRS[k, n]$  with folding parameter  $s$  are linearly-extendible linear operator codes  $LELO_{\mathcal{L}, A}$  where:
  - $\mathcal{L} = (L_0, \dots, L_{s-1})$  with  $L_1(f(X)) = f(\gamma Y)$  for  $f(X) \in \mathbb{F}_q[X]$  and  $L_i = L_1^i$  for  $i \in \{0, 1, \dots, s-1\}$ .
  - For the above family of operators  $M(X)$  is given by  $M(X)_{ij} = \gamma^i X \cdot \mathbb{I}[i = j]$  for  $i, j \in [s]$ .
  - The set of evaluation points is  $A = \{a_0, \dots, a_{n-1}\}$  where for any two distinct  $i$  and  $j$  the sets  $\{a_i, a_i\gamma, \dots, a_i\gamma^{s-1}\}$  and  $\{a_j, a_j\gamma, \dots, a_j\gamma^{s-1}\}$  are disjoint.
- **Multiplicity Codes:** Then,  $MULT[k, n]$  codes of order  $s$  are linearly-extendible linear operator codes  $LELO_{\mathcal{L}, A}$  where:
  - $\mathcal{L} = (L_0, \dots, L_{s-1})$  with  $L_1(f(X)) = \frac{\partial f(X)}{\partial X}$  for  $f(X) \in \mathbb{F}_q[X]$  and  $L_i = L_1^i$  for  $i \in \{0, 1, \dots, s-1\}$ .
  - For the above family of operators  $M(X)$  is given by  $M(X)_{ij} = X \cdot \mathbb{I}[i = j] + i \cdot \mathbb{I}[i-1 = j]$  for  $i, j \in [s]$ .
  - The set of evaluation points is  $A = \{a_0, \dots, a_{n-1}\}$  where  $a_i$ s are all distinct.
- **Additive Folded Reed-Solomon Codes:** Let  $\beta \in \mathbb{F}_q$  be a non-zero element and the characteristic of  $\mathbb{F}_q$  be at least  $s$ . Then, Additive- $FRS[k, n]$  codes with folding parameter  $s$  are linearly-extendible linear operator codes  $LELO_{\mathcal{L}, A}$  where:

## 56:14 Ideal-Theoretic Explanation of Capacity-Achieving Decoding

- $\mathcal{L} = (L_0, \dots, L_{s-1})$  with  $L_1(f(X)) = f(X + \beta)$  for  $f(X) \in \mathbb{F}_q[X]$  and  $L_i = L_1^i$  for  $i \in \{0, 1, \dots, s-1\}$ .
- For the above family of operators  $M(X)$  is given by  $M(X)_{ij} = (X + i\beta) \cdot \mathbb{I}[i = j]$  for  $i, j \in [s]$ .
- The set of evaluation points is  $A = \{a_0, \dots, a_{n-1}\}$  where  $a_i - a_j \notin \{0, \beta, 2\beta, \dots, (s-1)\beta\}$  for distinct  $i$  and  $j$ .
- **Affine Folded Reed-Solomon Codes:** Let  $\alpha \in \mathbb{F}_q^*$  and  $\beta \in \mathbb{F}_q$ . Further, let  $\ell(X) = \alpha X + \beta$  with  $\text{ord}(\ell) = u$ . Then Affine-FRS $[k, n]$  codes with folding parameter  $s$  are linearly-extendible codes  $LELO_{\mathcal{L}, A}$  described below. (See Observation A.7 for more details.)  
 Define  $D_1 : \mathbb{F}[X] \rightarrow \mathbb{F}[X]$  as  $D_1(f(X)) = \frac{\partial f(X)}{\partial X}$  and  $S_1 : \mathbb{F}[X] \rightarrow \mathbb{F}[X]$  as  $S_1(f(X)) = f(\ell(X))$ . Further, for  $i \geq 0$  let  $D_i = D_1^i$  and  $S_i = S_1^i$ . Recall, that the order of  $\alpha$  is  $u$ . For any integer  $r \in [s]$  let  $r = r_1 u + r_0$ , with  $r_0 < u$ , be the unique representation of  $r$ .
  - Define  $L_r : \mathbb{F}[X] \rightarrow \mathbb{F}[X]$  as  $L_r(f(X)) = S_{r_0}(D_{r_1} f(X))$ . Set  $\mathcal{L} = (L_0, \dots, L_{s-1})$ . Clearly,  $\mathcal{L}$  is a family of linear operators.
  - $L_r(Xf) = S_{r_0}(D_{r_1} f) = S_{r_0}(r_1 \cdot D_{r_1-1} f + X \cdot D_{r_1} f) = r_1 \cdot L_{r-u} f + S_{r_0}(X) \cdot L_r f$ : hence,  $\mathcal{L}$  is a set of linearly-extendible linear operators.
  - The set of evaluation points  $A = \{a_0, \dots, a_{n-1}\}$  is such that for distinct  $i, j$  the sets  $\{\ell^{(0)}(a_i), \dots, \ell^{(s-1)}(a_i)\}$  and  $\{\ell^{(0)}(a_j), \dots, \ell^{(s-1)}(a_j)\}$  are disjoint.

### 5 List-decoding of polynomial ideal codes

In this section, we discuss the list-decoding of polynomial ideal codes.

#### 5.1 List-decoding up to to the Johnson radius

We first observe that polynomial ideal codes are list decodable in polynomial time, up to the Johnson radius.

► **Theorem 5.1.** *Let  $k, s, n \in \mathbb{N}$  be such that  $k < sn$  and  $s < k - 1$ . Let  $E_0(X), E_1(X), \dots, E_{n-1}(X) \in \mathbb{F}[X]$  be relatively prime monic polynomials of degree equal to  $s$  each. Let  $\text{Enc} : \mathbb{F}_{<k}[X] \rightarrow (\mathbb{F}_{<s}[X])^n$  be the encoding function defined as*

$$p(X) \mapsto (p(X) \pmod{E_i(X)})_{i=0}^{n-1}.$$

*Then, there is an algorithm, which takes as input a received word  $\mathbf{c} = (\mathbf{c}_0, \mathbf{c}_1, \dots, \mathbf{c}_n) \in \mathbb{F}_{<s}[X]^n$  and for every  $\varepsilon > 0$  outputs all polynomials  $f \in \mathbb{F}_{<k}[X]$  such that  $\text{Enc}(f)$  and  $\mathbf{c}$  agree on at least  $(k/sn)^{1/2} + \varepsilon$  fraction of coordinates in time  $\text{poly}(n, 1/\varepsilon)$ .*

Observe that the rate of this code is  $k/sn$  and distance is  $1 - (k-1)/sn$ , and thus Theorem 5.1 gives us an algorithmic analog of Theorem 2.2 for these codes.

The list decoding algorithm for polynomial ideal codes is an (almost immediate) extension of an algorithm of Guruswami, Sahai and Sudan [4] for list decoding codes based on Chinese Remainder Theorem to this setting. This algorithm, in turn, relies on ideas in an earlier algorithm of Guruswami and Sudan [5] for list decoding Reed-Solomon codes up to the Johnson radius.

As noted in the introduction, most of the ideas for the proof of Theorem 5.1 were already there in the work of Guruswami, Sahai and Sudan [4] and all we do in this section is to flush out some of the details. Due to space constraints, we refer the interested reader to full version [1] for details.

## 5.2 List-decoding beyond the Johnson radius

In this section, we use the linear operator viewpoint of polynomial ideal codes to study their list-decodability beyond the Johnson radius. We show that if the family of linear operators  $\mathcal{L}$  and the evaluation points satisfy some further properties, then the linear operator code is list-decodable all the way up to the distance of the code.

Let  $\mathcal{G} = (G_0, \dots, G_{m-1})$  and  $\mathcal{T} = (T_0, T_1, \dots, T_{r-1})$  be two families of linear operators such that  $G_i : \mathbb{F}[X] \rightarrow \mathbb{F}[X]$  and  $\mathcal{T}$  is a linearly-extendible family of linear operators. We say that the pair  $(\mathcal{T}, \mathcal{G})$  *list-composes* in terms of  $\mathcal{L}$  at the set of evaluation points  $A$  if we have the following. For every linear operator  $G \in \mathcal{G}$  and field element  $a \in A$ , there exists a linear function  $h_{G,a} : \mathbb{F}^s \rightarrow \mathbb{F}^r$  such that for every polynomial  $f \in \mathbb{F}[X]$  we have

$$\mathcal{T}(G(f))(a) = h_{G,a}(\mathcal{L}(f)(a)).$$

► **Theorem 5.2.** *If  $LO_k^A(\mathcal{L})$  is a linear operator code and there exists two families of linear operators  $\mathcal{G} = (G_0, \dots, G_{m-1})$  and  $\mathcal{T} = (T_0, \dots, T_{r-1})$  such that*

1.  $(\mathcal{T}, A)$  forms a linearly-extendible linear operator code  $LELO_{k+nr/m}^A(\mathcal{T})$
  2. The pair  $(\mathcal{T}, \mathcal{G})$  list-composes in terms of  $\mathcal{L}$  at the set of evaluation points
  3.  $\mathcal{G}$  is degree-preserving
  4.  $\text{Diag}(\mathcal{G}) \in \mathbb{F}^{|\mathcal{G}| \times k}$  is the generator matrix of a code with distance  $k - \ell$ .
- Then,  $LO_k^A(\mathcal{L})$  is list-decodable up to the distance  $1 - \frac{k}{rn} - \frac{1}{m}$  with list size  $q^\ell$ .

This theorem clearly implies Theorem 1.1. We refer the interested reader to the full version of the paper [1] for the proof.

---

### References

- 1 Siddharth Bhandari, Prahladh Harsha, Mrinal Kumar, and Madhu Sudan. Ideal-theoretic explanation of capacity-achieving decoding. (manuscript). [arXiv:2103.07930](https://arxiv.org/abs/2103.07930), [eccc:2021/TR21-036](https://eccc.watson.ibm.org/eccc/2021/TR21-036).
- 2 Venkatesan Guruswami. Linear-algebraic list decoding of folded Reed-Solomon codes. In *Proc. 26th IEEE Conf. on Comput. Complexity*, pages 77–85, 2011. doi:10.1109/CCC.2011.22.
- 3 Venkatesan Guruswami and Atri Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Trans. Inform. Theory*, 54(1):135–150, 2008. (Preliminary version in *38th STOC*, 2006). [eccc:2005/TR05-133](https://eccc.watson.ibm.org/eccc/2005/TR05-133), doi:10.1109/TIT.2007.911222.
- 4 Venkatesan Guruswami, Amit Sahai, and Madhu Sudan. "Soft-decision" decoding of Chinese Remainder Codes. In *Proc. 41st IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 159–168, 2000. doi:10.1109/SFCS.2000.892076.
- 5 Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Trans. Inform. Theory*, 45(6):1757–1767, 1999. (Preliminary version in *39th FOCS*, 1998). [eccc:1998/TR98-043](https://eccc.watson.ibm.org/eccc/1998/TR98-043), doi:10.1109/18.782097.
- 6 Venkatesan Guruswami and Carol Wang. Linear-algebraic list decoding for variants of Reed-Solomon codes. *IEEE Trans. Inform. Theory*, 59(6):3257–3268, 2013. (Preliminary version in *26th IEEE Conference on Computational Complexity*, 2011 and *15th RANDOM*, 2011). [eccc:2012/TR12-073](https://eccc.watson.ibm.org/eccc/2012/TR12-073), doi:10.1109/TIT.2013.2246813.
- 7 Durga Datt Joshi. A note on upper bounds for minimum distance codes. *Information and Control*, 1(3):289–295, 1958. doi:10.1016/S0019-9958(58)80006-6.
- 8 Yasuo Komamiya. Application of logical mathematics to information theory. *Proc. 3rd Japan. Nat. Cong. Appl. Math*, 437, 1953.
- 9 Swastik Kopparty. List-decoding multiplicity codes. *Theory of Computing*, 11:149–182, 2015. [eccc:2012/TR12-044](https://eccc.watson.ibm.org/eccc/2012/TR12-044), doi:10.4086/toc.2015.v011a005.

- 10 Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin. High-rate codes with sublinear-time decoding. *J. ACM*, 61(5):28:1–28:20, 2014. (Preliminary version in *43rd STOC*, 2011). [eccc:2010/TR10-148](#), [doi:10.1145/2629416](#).
- 11 Richard Collom Singleton. Maximum distance  $q$ -nary codes. *IEEE Trans. Inform. Theory*, 10(2):116–118, 1964. [doi:10.1109/TIT.1964.1053661](#).
- 12 Salil P. Vadhan. Pseudorandomness. *Found. Trends Theor. Comput. Sci.*, 7(1-3):1–336, 2012. [doi:10.1561/0400000010](#).

## A Example of Codes Achieving List-Decoding Capacity

In this section we will use Theorem 5.2 to (re)prove the list-decoding capacity of the Folded Reed-Solomon codes, multiplicity codes and additive Folded Reed-Solomon codes. We then introduce a common generalization of all these codes, which we refer to as affine Folded Reed-Solomon codes and prove the list-decoding up to capacity of these codes.

### A.1 Folded Reed-Solomon (*FRS*) Codes

Fix integers  $k, n, q$  with  $n \leq q$ . Fix  $\gamma \in \mathbb{F}_q^*$  of multiplicative order at least  $s$ . The message space of the  $FRS_s^\gamma[k, n]$  code with folding parameter  $s$  is polynomials of degree at most  $k-1$  over  $\mathbb{F}[X]$ , i.e.,  $\mathbb{F}_{<k}[X]$  where  $\mathbb{F} = \mathbb{F}_q$ . Then, *FRS* codes are linearly-extendible linear operator codes  $LELO_{\mathcal{L}, A}$  where:

- $\mathcal{L} = (L_0, \dots, L_{s-1})$  with  $L_1(f(X)) = f(\gamma X)$  for  $f(X) \in \mathbb{F}_q[X]$  and  $L_i = L_1^i$  for  $i \in \{0, 1, \dots, s-1\}$ .
- For the above family of operators  $M(X)$  is given by  $M(X)_{ij} = \gamma^i X \cdot \mathbb{I}[i = j]$  for  $i, j \in [s]$ .
- The set of evaluation points is  $A = \{a_0, \dots, a_{n-1}\}$  where for any two distinct  $i$  and  $j$  the sets  $\{a_i, a_i\gamma, \dots, a_i\gamma^{s-1}\}$  and  $\{a_j, a_j\gamma, \dots, a_j\gamma^{s-1}\}$  are disjoint.

► Remark A.1.

1. Recall that the bivariate polynomial  $E(X, Y)$  corresponding to the polynomial ideal code representation is  $E(X, Y) = \prod_{i=0}^{s-1} (X - \gamma^i Y)$ .
2. For the choice of  $A$  as above, the rate of the code is  $\frac{k}{sn}$  and its distance is  $1 - \frac{k-1}{sn}$  as the polynomials  $E_i = E(X, a_i)$  are pairwise co-prime.

► **Theorem A.2** ([6]). *Let  $\gamma \in \mathbb{F}_q^*$  be an element of order at least  $k$ . Further, let  $A = \{a_0, \dots, a_{n-1}\}$  be a set of evaluation points where for any two distinct  $i$  and  $j$  the sets  $\{a_i, a_i\gamma, \dots, a_i\gamma^{s-1}\}$  and  $\{a_j, a_j\gamma, \dots, a_j\gamma^{s-1}\}$  are disjoint. For every  $\varepsilon > 0$  there exists  $s$  large enough ( $s \geq \Omega(1/\varepsilon^2)$ ) such that  $FRS_s^\gamma[k, n]$  at the set of evaluation points  $A$  can be efficiently list-decoded up to distance  $1 - \frac{k}{sn} - \varepsilon$ .*

**Proof.** We will prove this by applying Theorem 5.2. Set  $\mathcal{G} = (L_0, \dots, L_{m-1})$  for some integer  $m < s$  to be set later and  $\mathcal{T} = (T_0, \dots, T_{r-1})$  with  $r = s - m + 1$  and  $T_i = L_i$ .

Theorem 5.2-Item 1: Clearly,  $(\mathcal{T}, A)$  forms a linearly-extendible linear operator code  $LELO_{k+nr/m}^A(\mathcal{T})$  which is  $FRS_r^\gamma[k + nr/m, n]$  at the set of evaluation points  $A$ .

Theorem 5.2-Item 2: For all  $G_i \in \mathcal{G}$ ,  $T_j \in \mathcal{T}$  and  $a \in A$ , we have that for every polynomial  $f \in \mathbb{F}[X]$ :  $T_j(G_i(f))(a) = L_{i+j}(f)(a)$ . Notice that  $L_{i+j} \in \mathcal{L}$  as  $i + j \leq s - 1$ .

Theorem 5.2-Item 3:  $G_i(x^j) = \gamma^{ij} y^j$ , and hence  $\mathcal{G}$  is degree preserving.

Theorem 5.2-Item 4: The matrix  $\text{Diag}(\mathcal{G})$  is given by  $\text{Diag}(\mathcal{G})_{ij} = \gamma^{ij}$  for  $i \in [m]$  and  $j \in [k]$ . Hence, as long as  $\gamma$  has order at least  $k$  this is the generator matrix of  $RS[m-1, k]$  and hence its distance is  $k - m + 1$ .

Thus  $FRS_s^\gamma[k, n]$  can be efficiently list-decoded up to distance  $1 - \frac{k-1}{rn} - \frac{1}{m}$  with list size  $q^{m-1}$ . By choosing a large enough  $m$  and  $s$  we can ensure that  $1 - \frac{k-1}{rn} - \frac{1}{m} > 1 - \frac{k}{sn} - \varepsilon$ . ◀



## A.2 Multiplicity (*MULT*) Codes

Fix integers  $k, n, q$  with  $n \leq q$ . The message space of the  $MULT_s[k, n]$  code of order  $s$  is polynomials of degree at most  $k-1$  over  $\mathbb{F}[X]$ , i.e.,  $\mathbb{F}_{<k}[X]$  where  $\mathbb{F} = \mathbb{F}_q$ . Then,  $MULT_s[k, n]$  codes are linearly-extendible linear operator codes  $LELO_{\mathcal{L}, A}$  where:

- $\mathcal{L} = (L_0, \dots, L_{s-1})$  with  $L_1(f(X)) = \frac{\partial f(X)}{\partial X}$  for  $f(X) \in \mathbb{F}_q[X]$  and  $L_i = L_1^i$  for  $i \in \{0, 1, \dots, s-1\}$ .
- For the above family of operators  $M(X)$  is given by  $M(X)_{ij} = X \cdot \mathbb{I}[i = j] + i \cdot \mathbb{I}[i-1 = j]$  for  $i, j \in [s]$ .
- The set of evaluation points is  $A = \{a_0, \dots, a_{n-1}\}$  where  $a_i$ s are all distinct.

► Remark A.3.

1. Recall that the bivariate polynomial  $E(X, Y)$  corresponding to the polynomial ideal code representation is  $E(X, Y) = (X - Y)^s$ .
2. For the choice of  $A$  as above,  $MULT_s[k, n]$  is a code with rate  $\frac{k}{sn}$  and distance  $1 - \frac{k-1}{sn}$  as the polynomials  $E_i = E(X, a_i)$  are pairwise co-prime.

► **Theorem A.4** ([6]). *Let the characteristic of  $\mathbb{F}_q$  be at least  $\max(s, k)$ . Further, let the set of evaluation points be  $A = \{a_0, \dots, a_{n-1}\}$  where  $a_i$ s are all distinct. Then, for every  $\varepsilon > 0$  there exists  $s$  large enough ( $s \geq \Omega(1/\varepsilon^2)$ ) such that  $MULT_s[k, n]$  can be efficiently list-decoded up to distance  $1 - \frac{k}{sn} - \varepsilon$ .*

**Proof.** We will again appeal to Theorem 5.2. Set  $\mathcal{G} = (G_0, \dots, G_{m-1})$  where  $G_i = \frac{X^i}{i!} \cdot L_i$  for  $i \in \{0, 1, \dots, m-1\}$  for some integer  $m < s$  to be set later and  $\mathcal{T} = (T_0, \dots, T_{r-1})$  with  $r = s - m + 1$  and  $T_i = L_i$ .

Theorem 5.2-Item 1: Clearly,  $(\mathcal{T}, A)$  forms a linearly-extendible linear operator code  $LELO_{k+nr/m}^A(\mathcal{T})$  which is  $MULT_r[k + nr/m, n]$  of order  $r$  at the set of evaluation points  $A$ .

Theorem 5.2-Item 2: For all  $G_i \in \mathcal{G}$ ,  $T_j \in \mathcal{T}$  and  $a \in A$ , we have that for every polynomial  $f \in \mathbb{F}[X]$ :

$$T_j(G_i(f))(a) = \left( \sum_{b=0}^j \binom{j}{b} \binom{i}{b} \cdot (b!/i!) \cdot X^{i-b} L_{i+b}(f) \right)(a).$$

Notice that the above expression only involves  $L_i$ s where  $i < s$ .

Theorem 5.2-Item 3:  $G_i(X^j) = \binom{j}{i} \cdot X^j$ , and hence  $\mathcal{G}$  is degree preserving.

Theorem 5.2-Item 4: The matrix  $\text{Diag}(\mathcal{G})$  is given by  $\text{Diag}(\mathcal{G})_{ij} = \binom{j}{i}$  for  $i \in [m]$  and  $j \in [k]$ . This matrix can be transformed via elementary row operations to a  $RS[m, k]$  generator matrix with points of evaluations as  $0, 1, \dots, k-1$ ; thus, as long as the characteristic of  $\mathbb{F}_q$  is at least  $k$  we have that the distance of  $\text{Diag}(\mathcal{G})$  is  $k - m + 1$ .

Thus  $MULT_s[k, n]$  can be efficiently list-decoded up to distance  $1 - \frac{k-1}{rn} - \frac{1}{m}$  with list size  $q^{m-1}$ . By choosing a large enough  $m$  and  $s$  we can ensure that  $1 - \frac{k-1}{rn} - \frac{1}{m} > 1 - \frac{k}{sn} - \varepsilon$ . ◀

## A.3 Additive Folded Reed-Solomon (Additive-FRS) Codes

Fix integers  $k, n, q$  with  $n \leq q$ . Let  $\beta \in \mathbb{F}_q$  be a non-zero element and characteristic of  $\mathbb{F}_q$  is at least  $s$ . The message space of the Additive-FRS $_s^\beta[k, n]$  code with folding parameter  $s$  is polynomials of degree at most  $k-1$  over  $\mathbb{F}[X]$ , i.e.,  $\mathbb{F}_{<k}[X]$  where  $\mathbb{F} = \mathbb{F}_q$ . Then, Additive-FRS $_s^\beta[k, n]$  codes are linearly-extendible linear operator codes  $LELO_{\mathcal{L}, A}$  where:

- $\mathcal{L} = (L_0, \dots, L_{s-1})$  with  $L_1(f(X)) = f(X + \beta)$  for  $f(X) \in \mathbb{F}_q[X]$  and  $L_i = L_1^i$  for  $i \in \{0, 1, \dots, s-1\}$ .

## 56:18 Ideal-Theoretic Explanation of Capacity-Achieving Decoding

- For the above family of operators  $M(X)$  is given by  $M(X)_{ij} = (X + i\beta) \cdot \mathbb{I}[i = j]$  for  $i, j \in [s]$ .
- The set of evaluation points is  $A = \{a_0, \dots, a_{n-1}\}$  where  $a_i - a_j \notin \{0, \beta, 2\beta, \dots, (s-1)\beta\}$  for distinct  $i$  and  $j$ .

► Remark A.5.

1. Recall that the bivariate polynomial  $E(X, Y)$  corresponding to the polynomial ideal code representation is  $E(X, Y) = \prod_{i=0}^{s-1} (X - Y - i\beta)$ .
2. For the choice of  $A$  as above, Additive-FRS $_s^\beta[k, n]$  is a code with rate  $\frac{k}{sn}$  and distance  $1 - \frac{k-1}{sn}$  as the polynomials  $E_i = E(X, a_i)$  are pairwise co-prime.

► **Theorem A.6.** *Let the characteristic of  $\mathbb{F}_q$  be at least  $\max(s, k)$  and  $\beta \in \mathbb{F}_q$  be a non-zero element. Further, let the set of evaluation points  $A = \{a_0, \dots, a_{n-1}\}$  be such that  $a_i - a_j \notin \{0, \beta, 2\beta, \dots, (s-1)\beta\}$  for distinct  $i$  and  $j$ . Then, for every  $\varepsilon > 0$  there exists  $s$  large enough ( $s \geq \Omega(1/\varepsilon^2)$ ) such that Additive-FRS $_s^\beta[k, n]$  over the set of evaluation points  $A$  can be efficiently list-decoded up to distance  $1 - \frac{k}{sn} - \varepsilon$ .*

**Proof.** We will again appeal to Theorem 5.2. To define  $\mathcal{G} = (G_0, \dots, G_{m-1})$  for some integer  $m < s$ , we need the following definitions. Let  $B \in \mathbb{F}_q^{m \times m}$  be a matrix where  $B_{ij} = (j)^i$  for  $i, j \in [m]$ , i.e, the transpose of the Vandermonde matrix at the points  $\{0, 1, \dots, m-1\}$ : these points are distinct since the characteristic of the field is at least  $k$ . Further, let  $\mathbf{b}_i \in \mathbb{F}_q^m$  be a vector such that  $B\mathbf{b}_i = e_i$  for  $i \in [m]$  where  $e_i$ s are the standard basis vectors:  $\mathbf{b}_i$ s exist because  $B$  is full rank. Now, define  $G_i = X^i \cdot \sum_{c=0}^{m-1} \mathbf{b}_i(c)L_c$  for  $i \in [m]$ . Set  $\mathcal{T} = (T_0, \dots, T_{r-1})$  with  $r = s - m + 1$  and  $T_i = L_i$ .

Theorem 5.2-Item 1: Clearly,  $(\mathcal{T}, A)$  forms a linearly-extendible linear operator code  $LELO_{k+nr/m}^A(\mathcal{T})$  which is Additive-FRS $_r^\beta[k + nr/m, n]$  with folding parameter  $r$  at the set of evaluation points  $A$ .

Theorem 5.2-Item 2: For all  $G_i \in \mathcal{G}$ ,  $T_j \in \mathcal{T}$  and  $a \in A$ , we have that for every polynomial  $f \in \mathbb{F}[X]$ :

$$\begin{aligned} T_j(G_i(f))(a) &= T_j \left( X^i \cdot \sum_{c=0}^{m-1} \mathbf{b}_i(c)L_c \right) (a) \\ &= \left( (X + j\beta)^i \cdot \sum_{c=0}^{m-1} \mathbf{b}_i(c)L_{c+j} \right) (a). \end{aligned}$$

Notice that the above expression only involves  $L_i$ s where  $i < s$ . Theorem 5.2-Item 3:

$$\begin{aligned} G_i(X^j) &= X^i \cdot \sum_{c=0}^{m-1} \mathbf{b}_i(c)L_c(X^j) \\ &= X^i \cdot \sum_{c=0}^{m-1} \mathbf{b}_i(c)(X + c\beta)^j \\ &= X^i \cdot \sum_{c=0}^{m-1} \mathbf{b}_i(c) \sum_{h \leq j} \binom{j}{h} X^h \cdot (c\beta)^{j-h} \\ &= X^i \cdot \left( \binom{j}{i} \beta^i X^{j-i} + \sum_{h \leq j-m} \alpha_h X^h \right) \end{aligned}$$

(this is because  $B\mathbf{b}_i = e_i$  which means that for  $h > j - m$  we have  $\sum_{c=0}^{m-1} \mathbf{b}_i(c) \cdot (c)^{j-h} = \mathbb{I}[j - h = i]$ ;  $\alpha_n$  are field constants)

$$= \binom{j}{i} \beta^{i-1} X^j + \dots,$$

and hence  $\mathcal{G}$  is degree preserving.

Theorem 5.2-Item 4: By the above, the matrix  $\text{Diag}(\mathcal{G})$  is given by  $\text{Diag}(\mathcal{G})_{ij} = \binom{j}{i} \beta^i$  for  $i \in [m]$  and  $j \in [k]$ . Up to scaling this is the same code as  $\text{Diag}(\mathcal{G})$  in Theorem A.4: and hence, if the characteristic of the field is at least  $k$  then its distance is  $k - m + 1$ .

Thus Additive-FRS $_s^\beta[k, n]$  can be efficiently list-decoded up to distance  $1 - \frac{k-1}{rn} - \frac{1}{m}$  with list size  $q^{m-1}$ . By choosing a large enough  $m$  and  $s$  we can ensure that  $1 - \frac{k-1}{rn} - \frac{1}{m} > 1 - \frac{k}{sn} - \varepsilon$ . ◀

### A.4 Affine Folded Reed-Solomon (Affine-FRS) Codes

We first recall the definition of Affine-FRS codes. Fix integers  $k, n, q$  with  $n \leq q$ . Let  $\alpha \in \mathbb{F}_q^*$  and  $\beta \in \mathbb{F}_q$  such that the multiplicative order of  $\alpha$  is  $u$ . Further, define  $\ell(X) = \alpha X + \beta$  and

$$\ell^{(i)}(X) = \underbrace{\ell(\ell \dots \ell(X))}_{i \text{ times}} = \alpha^i X + \beta \cdot \sum_{j=0}^{i-1} \alpha^j = \alpha_i X + \beta_i.$$

In fact, if  $\alpha \neq 1$ , i.e.,  $u > 1$  then,  $\ell^{(u)}(X) = \ell^{(0)}(X)$ . Let  $\text{ord}(\ell)$  denote the smallest positive integer  $t$  such that  $\ell^{(t)}(z) = z$ . The message space of the Affine-FRS $_s^{\alpha, \beta}[k, n]$  code with folding parameter  $s$  is polynomials of degree at most  $k - 1$  over  $\mathbb{F}[X]$ , i.e.,  $\mathbb{F}_{<k}[X]$  where  $\mathbb{F} = \mathbb{F}_q$ . Let the set of evaluation points be  $A = \{a_0, \dots, a_{n-1}\}$  such that for distinct  $i, j$  the sets  $\{\ell^{(0)}(a_i), \dots, \ell^{(s-1)}(a_i)\}$  and  $\{\ell^{(0)}(a_j), \dots, \ell^{(s-1)}(a_j)\}$  are disjoint. Then, Affine-FRS $_s^{\alpha, \beta}[k, n]$  codes are polynomial ideal codes where:

- The bivariate polynomial  $E(X, Y)$  corresponding to the polynomial ideal code representation is  $E(X, Y) = \prod_{i=0}^{s-1} (X - \alpha_i Y - \beta_i)$ .
- For the choice of  $A$  as above, Affine-FRS $_s^{\alpha, \beta}[k, n]$  is a code with rate  $\frac{k}{sn}$  and distance  $1 - \frac{k-1}{sn}$  as the polynomials  $E_i = E(X, a_i)$  are pairwise co-prime.

We will now recall the description of Affine-FRS codes in terms of linear operators which will be helpful while list-decoding. Define  $D_1 : \mathbb{F}[X] \rightarrow \mathbb{F}[X]$  as  $D_1(f(X)) = \frac{\partial f(X)}{\partial X}$  and  $S_1 : \mathbb{F}[X] \rightarrow \mathbb{F}[X]$  as  $S_1(f(X)) = f(\ell(X))$ . Further, for  $i \geq 0$  let  $D_i = D_1^i$  and  $S_i = S_1^i$ . Recall, that the order of  $\alpha$  is  $u$ . For any integer  $r \in [s]$  let  $r = r_1 u + r_0$ , with  $r_0 < u$ , be the unique representation of  $r$ . Then, define  $L_r : \mathbb{F}[X] \rightarrow \mathbb{F}[X]$  as  $L_r(f(X)) = S_{r_0}(D_{r_1} f(X))$ . Set  $\mathcal{L} = (L_0, \dots, L_{s-1})$ . Clearly,  $\mathcal{L}$  is a family of linear operators. Further,  $L_r(Xf) = S_{r_0}(D_{r_1} Xf) = S_{r_0}(r_1 \cdot D_{r_1-1} f + X \cdot D_{r_1} f) = r_1 \cdot L_{r-u} f + S_{r_0}(X) \cdot L_r f$ ; hence,  $\mathcal{L}$  is a set of linearly-extendible linear operators.

▶ **Observation A.7.** *If  $u > 1$  then at an evaluation point  $a \in \mathbb{F}_q$  the following pieces of information are the same:*

- $f(X) \bmod \prod_{i=0}^{s-1} (X - \alpha_i a - \beta_i)$
- $\mathcal{L}(f)(a)$ .

Hence, if  $u > 1$ , then, Affine-FRS $_s^{\alpha, \beta}[k, n]$  at the points of evaluation  $A$  is  $LELO_{\mathcal{L}, A}$ .

▶ **Theorem A.8.** *For every  $\varepsilon > 0$ , there exists a large enough  $s$  such that the follow holds. Let  $\mathbb{F}_q$  be a field,  $k$  a parameter and  $\ell(X) = \alpha \cdot X + \beta$  such that  $\alpha \in \mathbb{F}_q^*$  and  $\beta \in \mathbb{F}_q$ . Furthermore, let the evaluation points  $A = \{a_0, \dots, a_{n-1}\}$  be such that for distinct  $i, j$  the sets  $\{\ell^{(0)}(a_i), \dots, \ell^{(s-1)}(a_i)\}$  and  $\{\ell^{(0)}(a_j), \dots, \ell^{(s-1)}(a_j)\}$  are disjoint. Then, if either:*

## 56:20 Ideal-Theoretic Explanation of Capacity-Achieving Decoding

■  $\text{ord}(\ell) \geq k$  or

■  $\text{char}(\mathbb{F}_q) > k$  and  $\beta \neq 0$

holds, Affine-FRS $_{s}^{\alpha, \beta}[k, n]$  over the set of evaluation points  $A$  can be efficiently list-decoded up to distance  $1 - \frac{k}{sn} - \varepsilon$ .

**Proof.** We will again appeal to Theorem 5.2. Let  $u$  be the multiplicative order of  $\alpha$ . Let  $v = \lfloor s/u \rfloor$ .

**Case  $\text{ord}(\ell) \geq k$ .** This means that  $u \geq k$ . This is similar to decoding FRS codes. We skip the details.

Henceforth, we assume that  $\text{char}(\mathbb{F}_q) \geq k$  and  $\beta \neq 0$ .

**Case  $u = 1$ .** This is the same case as for Additive-FRS codes. Thus, by Theorem A.6 we are done.

**Case  $u > 1$  and  $v \geq \sqrt{s}$ .** (This case is similar to  $MULT_v[k, n]$ .)

Define  $\mathcal{G} = (G_0, \dots, G_{m-1})$  for some integer  $m < s$ , as  $G_i(f) = (X^i/i!) \cdot D_i f$ . Let  $r = (v - m)u$  and set  $\mathcal{T} = \{L_0, L_1, \dots, L_{r-1}\}$ .

Theorem 5.2-Item 1: Clearly,  $(\mathcal{T}, A)$  forms a linearly-extendible linear operator code  $LELO_{k+nr/m}^A(\mathcal{T})$  which is Affine-FRS $_r^{\alpha, \beta}[k + nr/m, n]$  at the set of evaluation points  $A$ .

Theorem 5.2-Item 2: For all  $G_i \in \mathcal{G}$ ,  $T_j \in \mathcal{T}$  and  $a \in A$  we have that for every polynomial  $f \in \mathbb{F}[X]$ :

$$\begin{aligned} T_j(G_i(f))(a) &= \left( S_{j_0} D_{j_1} \left( \frac{X^i}{i!} \cdot D_i(f) \right) \right) (a) \\ &= \left( S_{j_0} \sum_{b=0}^{j_1} \binom{j_1}{b} \binom{i}{b} \cdot (b!/i!) \cdot X^{i-b} D_{i+b}(f) \right) (a) \\ &= \left( \sum_{b=0}^{j_1} \binom{j_1}{b} \binom{i}{b} \cdot (b!/i!) \cdot (S_{j_0} X^{i-b}) \cdot L_{j_0+(i+b)u}(f) \right) (a). \end{aligned}$$

Notice that the above expression only involves  $L_i$ s where  $i < s$ .

Theorem 5.2-Items 3 and 4: are identical to the corresponding items in Theorem A.4.

Thus Affine-FRS $_s^{\beta}[k, n]$  can be efficiently list-decoded up to distance  $1 - \frac{k-1}{rn} - \frac{1}{m}$  with list size  $q^{m-1}$ . By choosing a large enough  $m$  and  $s$  we can ensure that  $1 - \frac{k-1}{rn} - \frac{1}{m} > 1 - \frac{k}{sn} - \varepsilon$ .

**Case  $u > \sqrt{s}$ .** (This case is similar to Additive-FRS $_u^{\beta}[k, n]$ .) As in Theorem A.6, to define  $\mathcal{G} = (G_0, \dots, G_{m-1})$  for some integer  $m < u$ , we need the following definitions. Let  $B \in \mathbb{F}_q^{m \times m}$  be a matrix where  $B_{ij} = (\beta(\alpha^j - 1)/(\alpha^j))^i$  for  $i, j \in [m]$ , i.e, the transpose of the Vandermonde matrix at the points  $\{\beta(\alpha^j - 1)/(\alpha^j) \mid j \in [m]\}$ : these points are distinct since the order of  $u$  is at least  $m$ . Further, let  $\mathbf{b}_i \in \mathbb{F}_q^m$  be a vector such that  $B\mathbf{b}_i = e_i$  for  $i \in [m]$  where  $e_i$ s are the standard basis vectors:  $\mathbf{b}_i$ s exist because  $B$  is full rank.

Define  $\mathcal{G} = (G_0, \dots, G_{m-1})$  for some integer  $m < s$ , as  $G_i = X^i \cdot \sum_{c=0}^{m-1} b_i(c) S_c$ . Let  $r = s - m + 1$  and set  $\mathcal{T} = \{L_0, \dots, L_{r-1}\}$ .

Theorem 5.2-Item 1: Clearly,  $(\mathcal{T}, A)$  forms a linearly-extendible linear operator code  $LELO_{k+nr/m}^A(\mathcal{T})$  which is Affine-FRS $_r^{\alpha, \beta}[k + nr/m, n]$  at the set of evaluation points  $A$ .

Theorem 5.2-Item 2: For all  $G_i \in \mathcal{G}$ ,  $T_j \in \mathcal{T}$  and  $a \in A$  we have that for every polynomial  $f \in \mathbb{F}[X]$ :

$$\begin{aligned}
T_j(G_i(f))(a) &= \left( S_{j_0} D_{j_1} \left( X^i \cdot \sum_{c=0}^{m-1} b_i(c) S_c f \right) \right) (a) \\
&= \left( S_{j_0} \sum_{b=0}^{j_1} \binom{j_1}{b} \binom{i}{b} \cdot (b!) \cdot X^{i-b} D_b \left( \sum_{c=0}^{m-1} b_i(c) S_c f \right) \right) (a) \\
&= \left( S_{j_0} \sum_{b=0}^{j_1} \binom{j_1}{b} \binom{i}{b} \cdot (b!) \cdot X^{i-b} \left( \sum_{c=0}^{m-1} (b_i(c) \alpha_c^b) S_c D_b f \right) \right) (a) \\
&= \left( S_{j_0} \sum_{b=0}^{j_1} \binom{j_1}{b} \binom{i}{b} \cdot (b!) \cdot X^{i-b} \left( \sum_{c=0}^{m-1} (b_i(c) \alpha_c^b) L_{bu+cf} \right) \right) (a).
\end{aligned}$$

Notice that the above expression only involves  $L_i$ s where  $i < s$ .

Theorem 5.2-Items 3 and 4: follow almost identically to the corresponding items in Theorem A.6.

Thus Affine-FRS $_s^\beta[k, n]$  can be efficiently list-decoded up to distance  $1 - \frac{k-1}{rn} - \frac{1}{m}$  with list size  $q^{m-1}$ . By choosing a large enough  $m$  and  $s$  we can ensure that  $1 - \frac{k-1}{rn} - \frac{1}{m} > 1 - \frac{k}{sn} - \varepsilon$ . ◀



# Visible Rank and Codes with Locality

Omar Alrabiah ✉

Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh, PA, USA

Venkatesan Guruswami ✉

Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

---

## Abstract

We propose a framework to study the effect of local recovery requirements of codeword symbols on the dimension of linear codes, based on a combinatorial proxy that we call *visible rank*. The locality constraints of a linear code are stipulated by a matrix  $H$  of  $\star$ 's and 0's (which we call a "stencil"), whose rows correspond to the local parity checks (with the  $\star$ 's indicating the support of the check). The visible rank of  $H$  is the largest  $r$  for which there is a  $r \times r$  submatrix in  $H$  with a unique generalized diagonal of  $\star$ 's. The visible rank yields a field-independent combinatorial lower bound on the rank of  $H$  and thus the co-dimension of the code.

We point out connections of the visible rank to other notions in the literature such as unique restricted graph matchings, matroids, spanoids, and min-rank. In particular, we prove a rank-nullity type theorem relating visible rank to the rank of an associated construct called *symmetric spanoid*, which was introduced by Dvir, Gopi, Gu, and Wigderson [5]. Using this connection and a construction of appropriate stencils, we answer a question posed in [5] and demonstrate that symmetric spanoid rank cannot improve the currently best known  $\tilde{O}(n^{(q-2)/(q-1)})$  upper bound on the dimension of  $q$ -query locally correctable codes (LCCs) of length  $n$ . This also pins down the efficacy of visible rank as a proxy for the dimension of LCCs.

We also study the  $t$ -Disjoint Repair Group Property ( $t$ -DRGP) of codes where each codeword symbol must belong to  $t$  disjoint check equations. It is known that linear codes with 2-DRGP must have co-dimension  $\Omega(\sqrt{n})$  (which is matched by a simple product code construction). We show that there are stencils corresponding to 2-DRGP with visible rank as small as  $O(\log n)$ . However, we show the second tensor of any 2-DRGP stencil has visible rank  $\Omega(n)$ , thus recovering the  $\Omega(\sqrt{n})$  lower bound for 2-DRGP. For  $q$ -LCC, however, the  $k$ 'th tensor power for  $k \leq n^{o(1)}$  is unable to improve the  $\tilde{O}(n^{(q-2)/(q-1)})$  upper bound on the dimension of  $q$ -LCCs by a polynomial factor. Inspired by this and as a notion of intrinsic interest, we define the notion of *visible capacity* of a stencil as the limiting visible rank of high tensor powers, analogous to Shannon capacity, and pose the question whether there can be large gaps between visible capacity and algebraic rank.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Error-correcting codes

**Keywords and phrases** Visible Rank, Stencils, Locality, DRGP Codes, Locally Correctable Codes

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.57

**Category** RANDOM

**Funding** Research supported in part by NSF grants CCF-1814603 and CCF-1908125.

## 1 Introduction

The notion of *locality* in error-correcting codes refers to the concept of recovering codeword symbols as a function of a small number of other codeword symbols. Local decoding requirements of various kinds have received a lot of attention in coding theory, due to both their theoretical and practical interest. For instance,  $q$ -query locally correctable codes (LCCs) aim to recover any codeword symbol as a function of  $q$  other codeword symbols in a manner



© Omar Alrabiah and Venkatesan Guruswami;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 57; pp. 57:1–57:18



Leibniz International Proceedings in Informatics

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

robust to a constant fraction of errors. On the other hand, locally recoverable codes (LRCs), in their simplest incarnation, require each codeword symbol to be a function of some  $\ell$  other codeword symbols, allowing local recovery from any single erasure.<sup>1</sup>

LCCs have been extensively studied in theoretical computer science, and have connections beyond coding theory to topics such as probabilistically checkable proofs and private information retrieval. We refer the reader to [31] and the introduction of [13] for excellent surveys on LCCs and their connections. LRCs were motivated by the need to balance global fault tolerance with extremely efficient repair of a small number of failed storage nodes in modern large-scale distributed storage systems [12]. They have led to intriguing new theoretical questions, and have also had significant practical impact with adoption in large scale systems such as Microsoft Azur [17] and Hadoop [25].

Let us define the above notions formally, in a convenient form that sets up this work. We will restrict attention to linear codes in this work, i.e., subspaces  $C$  of  $\mathbb{F}_q^n$  for some finite field  $\mathbb{F}_q$ . In this case, the  $i$ 'th symbol  $c_i$  of every codeword  $c = (c_1, \dots, c_n) \in C$  can be recovered as a function of the symbols  $c_j$ , for indices  $j$  in a (minimal) subset  $R_i \subset [n] \setminus \{i\}$ , iff  $c_i$  and  $\{c_j \mid j \in R_i\}$  satisfy a linear check equation, or in other words, there is a dual codeword whose support equals  $\{i\} \cup R_i$ . The set  $R_i$  is called a *repair group* for the  $i$ 'th codeword symbol (other terminology used in the literature includes regenerating sets and recovery sets).

The  $q$ -LCC property, for a fixed number of queries  $q$  and growing  $n$ , corresponds to having  $\Omega(n)$  disjoint groups of size  $\leq q$  for each position  $i \in [n]$ , or equivalently  $\Omega(n)$  dual codewords of Hamming weight at most  $(q + 1)$  whose support includes  $i$  and are otherwise disjoint. The  $\ell$ -LRC corresponds to having a dual codeword of Hamming weight at most  $(\ell + 1)$  whose support includes  $i$ , for each  $i \in [n]$ . A property that interpolates between these extremes of a single repair group and  $\Omega(n)$  disjoint repair groups is the Disjoint Repair Group Property ( $t$ -DRGP) where we require  $t$  disjoint repair groups for each position  $i \in [n]$  (equivalently  $t$  dual codewords whose support includes  $i$  but are otherwise disjoint).

There is an exponentially large gap between upper and lower bounds on the trade-off between code dimension and code length for  $q$ -LCCs. The best known code constructions have dimension only  $O((\log n)^{q-1})$  (achieved by generalized Reed-Muller codes or certain lifted codes [14]), whereas the best known upper bound on the dimension of  $q$ -LCCs is much larger and equals  $\widetilde{O}(n^{(q-2)/(q-1)})$  [19, 29, 18]<sup>2</sup>. Narrowing this huge gap has remained open for over two decades.

In contrast the best possible dimension of a  $\ell$ -LRC is easily determined to be  $\lfloor \frac{\ell n}{\ell+1} \rfloor$ .<sup>3</sup> However, for  $t$ -DRGP, there are again some intriguing mysteries. For 2-DRGP, we have tight bounds – the minimum possible redundancy (co-dimension) equals  $\sqrt{2n} \pm \Theta(1)$ . The lower bound is established via very elegant proofs based on the polynomial method [30] or rank arguments [24]. However, for fixed  $t > 2$ , we do not know better lower bounds, and the best known constructions have co-dimension  $\approx t\sqrt{n}$  [6]. There are better constructions known for some values of  $t = n^{\Theta(1)}$  [8, 20]. A lower bound on the co-dimension of  $c(t)\sqrt{n}$  for some function  $c(t)$  that grows with  $t$  seems likely, but has been elusive despite various attempts, and so far for any fixed  $t$ , the bound for  $t = 2$  is the best known.

<sup>1</sup> There is also a distance requirement on LRCs to provide more global error/erasure resilience.

<sup>2</sup> The  $\widetilde{O}(\cdot)$  and  $\widetilde{\Omega}(\cdot)$  are used to suppress factors poly-logarithmic in  $n$ .

<sup>3</sup> In this case, a more interesting trade-off is a Singleton-type bound that also factors in the distance of the code [12].



This work was motivated in part by these major gaps in our knowledge concerning  $q$ -LCCs and  $t$ -DRGPs. Our investigation follows a new perspective based on *visible rank* (to be defined soon), which is a combinatorial proxy for (linear-algebraic) rank that we believe is of broader interest. This is similar in spirit to a thought-provoking recent work [5] that introduced a combinatorial abstraction of spanning structures called *spanoids*<sup>4</sup> to shed light on the limitations of current techniques to prove better upper bounds on the dimension of  $q$ -LCCs. They noted that current techniques to bound LCC dimension apply more generally to the associated spanoids, which they showed could have rank as large as  $\tilde{\Omega}(n^{(q-2)/(q-1)})$ . Therefore to improve the LCC bound one needs techniques that are more specific than spanoids and better tailored to the LCC setting. One such possibility mentioned in [5] is to restrict attention to *symmetric spanoids*, which have a natural symmetry property that linear LCCs imply.

Our visible rank notion turns out to be intimately related to symmetric spanoids via a rank-nullity type theorem (Theorem 14). While technically simple in hindsight, it offers a powerful viewpoint on symmetric spanoids which in particular resolves a question posed in [5] – we show that symmetric spanoids are also too coarse a technique to beat the  $\tilde{O}(n^{(q-2)/(q-1)})$  upper bound on  $q$ -LCC dimension.

## 1.1 Stencils and visible rank

With the above backdrop, we now proceed to describe the setup we use to study these questions, based on the rank of certain matrix templates which we call “stencils.” We can represent the support structure of the check equations (i.e., dual codewords) governing a locality property by an  $n$ -column matrix of 0’s and  $\star$ ’s. For each check equation involving the  $i$ ’th symbol and a repair group  $R_i \subset [n] \setminus \{i\}$ , we place a row in the stencil with  $\star$ ’s precisely at  $R_i \cup \{i\}$  (i.e., with  $\star$ ’s at the support of the associated dual codeword). For the  $\ell$ -LRC property for instance, an associated stencil would be an  $n \times n$  matrix with  $\star$ ’s on the diagonal and  $\ell$  other  $\star$ ’s in each row. For  $q$ -LCC, we would have a  $\delta n^2 \times n$  matrix whose rows are split into  $n$  groups with the rows in the  $i$ ’th group having a  $\star$  in the  $i$ ’th column and  $q$  other  $\star$ ’s in disjoint columns.

The smallest co-dimension of linear codes over a field  $\mathbb{F}$  with certain locality property is, by design, the minimum rank  $\text{rk}_{\mathbb{F}}(H)$  of the associated stencil  $H$  when the  $\star$ ’s are replaced by arbitrary nonzero entries from  $\mathbb{F}$ . In this work, our goal is to understand this quantity via field oblivious methods based only on the combinatorial structure of the stencil of  $\star$ ’s.

The tool we put forth for this purpose is the *visible rank* of  $H$ , denoted  $\text{vrk}(H)$  and defined to be the largest  $r$  for which there is a  $r \times r$  submatrix of  $H$  that has exactly one general diagonal whose entries are all  $\star$ ’s. By the Leibniz formula, the determinant of such a submatrix is nonzero for any substitution of nonzero entries for the  $\star$ ’s. Thus  $\text{rk}_{\mathbb{F}}(H) \geq \text{vrk}(H)$  for every field  $\mathbb{F}$ .

Our goal in this work is to understand the interrelationship between visible rank and the co-dimension of linear codes under various locality requirements. This can shed further light on the bottleneck in known techniques to study trade-offs between locality and code dimension, and optimistically could also lead to better constructions.

---

<sup>4</sup> We defer a precise description of spanoids, along with their strong connection to visible rank, to Section 2.4.

## 1.2 Visible rank and Locality

For  $\ell$ -LRCs, a simple greedy argument shows that its associated parity-check stencil  $H$  satisfies  $\text{vrk}(H) \geq n/(\ell + 1)$ . Thus visible rank captures the optimal trade-off between code dimension and locality  $\ell$ .

For  $q$ -LCCs with  $q \geq 3$ , an argument similar to (in fact a bit simpler than and implied by) the one for spanoids in [5] shows that the stencil corresponding to  $q$ -LCCs has visible rank at least  $n - \tilde{O}(n^{(q-2)/(q-1)})$ , showing an upper bound of  $\tilde{O}(n^{(q-2)/(q-1)})$  on the dimension of  $q$ -LCCs. We show that visible rank suffers the same bottleneck as spanoids in terms of bounding the dimension of  $q$ -LCCs.

► **Theorem 1.** *For  $q \geq 3$ , there exist  $n$ -column stencils  $H$  with  $\star$ 's structure compatible with  $q$ -LCCs for which  $\text{vrk}(H) \leq n - \tilde{\Omega}(n^{(q-2)/(q-1)})$ .*

Through the precise connection we establish between visible rank and symmetric spanoids, this shows the same limitation for symmetric spanoids, thus answering a question posed in [5].

For the  $t$ -DRGP property, we focus on the  $t = 2$  case, with the goal of finding a combinatorial substitute for the currently known  $\Omega(\sqrt{n})$  lower bounds on co-dimension [30, 8] which are algebraic. Unfortunately, we show that visible rank, in its basic form, is too weak in this context.

► **Theorem 2.** *There exist  $2n \times n$  stencils  $H$  with  $\star$ 's structure compatible with 2-DRGP for which  $\text{vrk}(H) \leq O(\log n)$ .*

## 1.3 Visible rank and tensor powers

In view of Theorem 2, we investigate avenues to get better bounds out of the visible rank approach. Specifically, we study the visible rank of tensor powers of the matrix. It turns out that the visible rank is super-multiplicative:  $\text{vrk}(H \otimes H) \geq \text{vrk}(H)^2$ , while on the other hand algebraic rank is sub-multiplicative, so higher tensor powers could yield better lower bounds on the rank. Indeed, we are able to show precisely this for 2-DRGP:

► **Theorem 3.** *For every  $2n \times n$  stencil  $H$  with  $\star$ 's structure compatible with 2-DRGP, we have  $\text{vrk}(H \otimes H) \geq \Omega(n)$ , and thus  $\text{rk}_{\mathbb{F}}(H) \geq \Omega(\sqrt{n})$  for every field  $\mathbb{F}$ .*

On the other hand, for  $q$ -LCCs with  $q \geq 3$ , we show that higher tensor powers suffer the same bottleneck as Theorem 1.

► **Theorem 4.** *For  $q \geq 3$ , there exist  $n$ -column stencils  $H$  with  $\star$ 's structure compatible with  $q$ -LCCs for which  $\text{vrk}(H^{\otimes k})^{1/k} \leq n - \tilde{\Omega}(n^{(q-2)/(q-1)})/k$  for any integer  $k$ . In particular even for  $k = n^{o(1)}$ , we get no polynomial improvements to the current upper bounds on dimension of  $q$ -LCCs.*

## 1.4 Visible capacity

Given the super-multiplicativity of visible rank under tensor powers, and drawing inspiration from the Shannon capacity of graphs, we put forth the notion of *visual capacity* of a matrix  $H$  of 0's and  $\star$ 's, defined as  $\Upsilon(H) := \sup_k \text{vrk}(H^{\otimes k})^{1/k}$ . The visual capacity is also a field oblivious lower bound on algebraic rank  $\text{rk}_{\mathbb{F}}(H)$  for any field  $\mathbb{F}$ . It is not known whether there are stencils that exhibit a gap between visible capacity and its minimum possible  $\text{rk}_{\mathbb{F}}(H)$  over all fields  $\mathbb{F}$ .

The proofs of our results are technically simple, once the framework is set up. Our contributions are more on the conceptual side, via the introduction and initial systematic study of visible rank and its diverse connections. Our inquiry also raises interesting questions and directions for future work, some of which are outlined in Section 7, including the relationship between visible capacity and algebraic rank.

## 1.5 Connections and related work

Studying the interplay between the combinatorial structure of a matrix and its rank is a natural quest that arises in several contexts. See Chapter 3 of [26] for a survey of works on lower bounding the algebraic rank. For works specific to codes with locality, the work of [3] analyzed the combinatorial properties of design matrices over the reals to improve bounds on LCCs over the real numbers, although the methods used are particular to the field of reals and do not carry over to any field.

Visible rank in particular turns out to have a diverse array of connections, some of which we briefly discuss here. The connection to spanoids, that we already mentioned, is described in more detail in Section 2.4.

**Uniquely restricted matchings.** Given a stencil  $H \in \{0, \star\}^{m \times n}$ , there is a canonical bipartite graph  $G$  between the rows and columns of  $H$ , where a row connects to a column if and only if their shared entry has a star. Visual rank has a nice graph-theoretic formulation: it turns out (see Section 2.3) that a submatrix of  $H$  has a unique general diagonal of  $\star$ 's iff the corresponding induced subgraph has a unique perfect matching. Such induced bipartite graphs are known in the literature as *Uniquely Restricted Matchings (URMs)* and have been extensively studied [11, 9, 16, 22, 27, 7]. They were first introduced in [11], wherein they proved that computing the maximum URM of a bipartite graph is NP-complete. It was later shown in [22] that  $n^{1/3-o(1)}$  approximations of the maximum URM is also NP-hard unless  $\text{NP} = \text{ZPP}$  and additionally that the problem of finding the maximum URM is APX-complete.

**Matroids.** One can encode any matroid into a stencil. Recall that a circuit of a matroid is a minimal dependent set – that is, a dependent set whose proper subsets are all independent (the terminology reflects the fact that in a graphic matroid, the circuits are cycles of the graph). Given a matroid  $\mathcal{M}$  on universe  $[n]$  and a set  $\mathcal{C} = \{C_1, \dots, C_m\}$  of circuits of  $\mathcal{M}$ , we consider a  $m \times n$  stencil  $H$  where the entry at  $(i, j)$  is a  $\star$  if and only if  $j \in C_i$ . For this matrix, one can show that a collection of visibly independent columns (see Section 2.2 for the definition of visible independence) is an independent set in the dual matroid. Therefore, we have  $\text{rk}(\mathcal{M}) + \text{vrk}(H) \leq n$  – this also follows from our rank-nullity theorem for symmetric spanoids as one can associate a symmetric spanoid with any matroid (the collection of sets in Definition 13 will just be the circuits of the matroid).

**Min-rank.** The minimum possible rank of a square  $0\text{-}\star$  stencil over assignments to the  $\star$ 's from some field has been well studied in combinatorics.<sup>5</sup> For example, we have Haemers' classic bound on independent set of a graph and its applications to Shannon capacity [15].

<sup>5</sup> There is a slight difference in the minrank setup, in that the  $\star$ 's can take any value including 0, except the  $\star$ 's on the diagonal which must take nonzero values.

Note that in this case we are using a linear-algebraic tool to understand a combinatorial quantity, whereas visible rank goes the other way, serving as a combinatorial proxy for a linear-algebraic quantity. Recent interest in minrank has included their characterization of the most efficient linear index codes [2]. The minrank of stencils corresponding to  $n$ -vertex random Erdős-Rényi graphs was recently shown to be  $\Theta(n/\log n)$  over any field that is polynomially bounded [10].

**Matrix Rigidity.** Given a square matrix  $A \in \mathbb{F}^{n \times n}$  and a natural number  $r \leq n$ , the *rigidity* of  $A$  is the minimal number of entries that one can perturb in  $A$  so that its rank becomes at most  $r$ . Matrix rigidity was introduced in the seminal work [28] and since then had expansive research on constructing explicit rigid matrices. See [23] for a recent survey on matrix rigidity and related connections. The visible rank provides a combinatorial guarantee on the rank of a matrix, and that conjures up the possibility of constructing explicit rigid matrices by finding explicit stencils whose visible rank is robust to small amounts of corruptions of its entries.

**Incidence Theorems.** Given an  $m \times n$  matrix  $A$  over the field  $\mathbb{F}$  with rank  $r$ , one can decompose  $A = MN$  where  $M$  and  $N$  are  $m \times r$  and  $r \times n$  matrices. If we consider the rows of  $M$  as hyperplanes over the projective plane  $\mathbb{P}\mathbb{F}^{r-1}$  of dimension  $(r-1)$  and the columns of  $N$  as points in  $\mathbb{P}\mathbb{F}^{r-1}$ , then the stencil of  $A$  defines a point-hyperplane incidence over  $\mathbb{P}\mathbb{F}^{r-1}$ . In particular, when  $r = 3$ , the stencil of  $A$  defines a point-line incidence over the field  $\mathbb{F}$ . Thus studying the combinatorial properties of a stencil whose  $\mathbb{F}$ -rank (see Definition 6) is at most 3 is equivalent to studying the combinatorics of point-line incidences over the field  $\mathbb{F}$ . For more on incidence theorems, see [4] for an excellent survey in the area.

**Communication complexity.** The visible rank provides a connection between deterministic and nondeterministic communication complexity [21]. For a communication problem  $f : X \times Y \rightarrow \{0, 1\}$ , define the stencil  $H_f \in \{0, \star\}^{X \times Y}$  by  $H_f(x, y) = \star$  if  $f(x, y) = 0$  and  $M_f(x, y) = 0$  if  $f(x, y) = 1$ . Then it is known that  $D(f) \leq (\log_2 \text{vrk}(H_f)) \cdot (N(f) + 1)$  where  $D(f)$  and  $N(f)$  are respectively the deterministic and nondeterministic communication complexity of  $f$  [21, Thm 3.5].

## 1.6 Organization

We begin in Section 2 by formally introducing the notations and terminology for stencils, and establishing some simple but very useful combinatorial facts about visible rank. We use these to show that there are  $q$ -LCC stencils for  $q \geq 3$  with visible rank at most  $n - \tilde{\Omega}(n^{(q-2)/(q-1)})$  (Section 3), and the existence of a 2-DRGP stencil with visible rank of at most  $O(\log n)$  (Section 4). In Section 5, we introduce a tensor product operation on stencils and prove various properties about them. In Section 6, we utilize tensor powers to show that the rank of a 2-DRGP over any field  $\mathbb{F}$  is at least  $\sqrt{n}$ , which asymptotically matches the current best lower bounds on  $t$ -DRGP codes. We also show that for  $q$ -LCC stencils, the tensor powers at the  $k$ 'th level for  $k \leq \text{polylog}(n)$  do not yield better lower bounds on the rank than the ones obtained from the visible rank. Finally, in Section 7, we discuss further directions and questions inspired by this work.

## 2 Stencils and their visible rank

In this section, we will be formally setting up the model of stencils and all the associated definitions and notations. We denote  $[n]$  to be the set  $\{1, 2, \dots, n\}$ . For any matrix  $H \in \{0, \star\}^{m \times n}$ , we denote it as a *stencil*. For an  $m \times n$  stencil  $H$ , we denote its entry in the  $i$ 'th row and  $j$ 'th column by  $H[i, j]$ . Any restriction to the specific sub-collection of the rows and columns of  $H$  is said to be a *sub-stencil* of  $H$ . For given sets  $A$  and  $B$ , a stencil  $H$  is said to be an  $A \times B$  if it is an  $|A| \times |B|$  stencil along with an associated indexing of the rows by  $A$  and the columns by  $B$ . Given a square stencil  $M \in \{0, \star\}^{n \times n}$ , a *general diagonal* of  $M$ , is a collection of entries  $\{M[1, \pi(1)], \dots, M[n, \pi(n)]\}$  where  $\pi$  is a permutation on  $[n]$ . We say that a general diagonal is a *star diagonal* if all its  $n$  entries are  $\star$ 's.

### 2.1 Algebraic witnesses of stencils

Instantiating a code with the locality properties stipulated by a stencil amounts to filling its  $\star$ 's with field entries, or realizing an algebraic witness as defined below.

► **Definition 5** (Algebraic witness). *For field  $\mathbb{F}$  and stencil  $H \in \{0, \star\}^{m \times n}$ , a matrix  $W \in \mathbb{F}^{m \times n}$  is said to be an  $\mathbb{F}$ -witness of  $H$  if it satisfies the property that  $W[i, j] \neq 0$  if and only if  $H[i, j] = \star$ . More generally, any  $\mathbb{F}$ -witness of  $H$  is said to be an algebraic witness of  $H$ .*

We stress that every  $\star$  in the stencil  $H$  must be replaced by a *nonzero* entry from  $\mathbb{F}$  and cannot be zero. Of the possible algebraic witnesses for  $H$ , we will be primarily focused in this paper on the algebraic witnesses that attain the smallest feasible rank, which leads us to the following definition.

► **Definition 6** (Rank). *Given an  $m \times n$  stencil  $H$ , the  $\mathbb{F}$ -rank of  $H$  is the smallest natural number  $r$  such that there exists a field  $\mathbb{F}$  and an  $\mathbb{F}$ -witness  $W \in \mathbb{F}^{m \times n}$  whose rank is equal to  $r$ . We denote the value  $r$  by  $\text{rk}_{\mathbb{F}}(H)$ .*

### 2.2 Visible Rank

In this section, we introduce our notion of the *visible rank* of a stencil. The main motivation of introducing the visible is to be able to determine the most optimal lower bound on the rank of a matrix with only the knowledge of knowing the support of a matrix and nothing else about the values of that support.

Consider a square matrix  $A \in \mathbb{F}^{n \times n}$ , and suppose we are interested in determining if it is full rank. A natural approach would be to inspect its determinant. From the Leibniz formula, we know that  $\det(A) = \sum_{\pi \in S_n} \prod_{i=1}^n (-1)^{\text{sgn}(\pi)} A_{i, \pi(i)}$ , where  $S_n$  denotes the symmetric group of order  $n$  and  $\text{sgn}(\pi)$  denotes the sign of a permutation  $\pi$ . From Leibniz formula, notice that  $\det(A)$  is a linear combination of the nonzero general diagonals of  $A$ . If our hope is to obtain  $\det(A) \neq 0$  without any knowledge of the values of the support of  $A$ , one way to guarantee it is to say that  $A$  has exactly one nonzero general diagonal. In such a case, we can guarantee that  $\det(A) \neq 0$ . As when  $A$  has more than one general diagonal, there is no guarantee if  $\det(A) \neq 0$  without inspecting the values of the support of  $A$ .

From the previous discussion, it seems natural to define the notion of a rank on stencils as follows.

► **Definition 7** (Visibly Full Rank). *For a square stencil  $M \in \{0, \star\}^{n \times n}$ , we say that  $M$  is visibly full rank if  $M$  has exactly one star diagonal. That is, a general diagonal whose entries are all  $\star$ 's.*

Of course, in most cases, when we are given a matrix  $A \in \mathbb{F}^{m \times n}$ , we would be interested in determining its rank. One way to define the rank of the matrix  $A$  is to say that  $\text{rank}(A)$  is the size of the largest square submatrix in  $A$  that is full-rank. From this viewpoint, it seems clear to define the rank of a stencil in a similar fashion.

► **Definition 8** (Visible Rank). *For a stencil  $H \in \{0, \star\}^{m \times n}$ , the visible rank of  $H$ , denoted  $\text{vrk}(H)$ , is the largest square sub-stencil in  $H$  that is visibly full rank.*

We also say that a set of  $k$  columns in  $H$  is *visibly independent* if there exists a  $k \times k$  sub-stencil within these  $k$  columns that is visibly full rank. Of course, not all full-rank square matrices  $A \in \mathbb{F}^{m \times n}$  necessarily have exactly one nonzero general diagonal, but all squares that have exactly one nonzero general diagonal are necessarily full-rank. Thus if we are interested in determining  $\text{rank}(A)$  by finding the size of the largest square submatrix in  $A$  that is full-rank, we can instead search for the largest square submatrix in  $A$  that has exactly one nonzero general diagonal. Since that square submatrix has rank at most the rank of  $A$ , this leads us to the following proposition.

► **Proposition 9.** *Given a field  $\mathbb{F}$  and stencil  $H \in \{0, \star\}^{m \times n}$ , we have  $\text{rk}_{\mathbb{F}}(H) \geq \text{vrk}(H)$ .*

### 2.3 Combinatorial properties of visible rank

In this subsection, we will be proving some properties about visible rank. In particular, we will show that any visibly independent stencil  $M$  is permutationally equivalent to an upper triangular stencil, and from this observation, we will be able to upper bound the visible rank by the largest rectangle of zeros in the stencil, which will be our main tool in our constructions of  $q$ -LCC and  $t$ -DRGP stencils. We also show an upper bound on the rank of a stencil by the maximum number of zeros in each row.

Given two stencils  $H_1, H_2 \in \{0, \star\}^{m \times n}$ , we say that  $H_1$  is *permutationally equivalent* to  $H_2$  if there are permutations  $\pi : [m] \rightarrow [m]$  and  $\sigma : [n] \rightarrow [n]$  such that  $H_1[i, j] = H_2[\pi(i), \sigma(j)]$  for all  $i \in [m]$  and  $j \in [n]$ . For such  $H_1$  and  $H_2$ , we introduce the notation  $H_2 = (H_1)_{\pi, \sigma}$  to say that  $H_2$  is obtained from  $H_1$  by permuting the rows with the permutation  $\pi$  and the columns by the permutation  $\sigma$  (We remark that row permutations commute with column permutations).

► **Lemma 10.** *Let  $M \in \{0, \star\}^{n \times n}$  be visibly full rank. Then there exists permutations  $\pi$  and  $\sigma$  on  $[n]$  such that  $N := M_{\pi, \sigma}$  is an upper triangular stencil. That is,  $N[i, i] = \star$  and  $N[i, j] = 0$  for all  $i, j \in [n]$  with  $i > j$ .*

**Proof.** First, we claim that for any visibly full rank stencil  $M \in \{0, \star\}^{n \times n}$ , there exists a row in  $M$  with exactly one star. Indeed, assume (for the sake of a contradiction) that such a row doesn't exist. Since no row can be all zeros in  $M$ , then each row has at least two  $\star$ 's. Index the rows of  $M$  by  $R = \{r_1, \dots, r_n\}$  and the columns by  $C = \{c_1, \dots, c_n\}$ . Let  $G = (R, C, S)$  be a bipartite graph on the rows and columns of  $M$  with edges  $S$ , where  $S$  is the set of  $\star$ 's in  $M$ . Because  $M$  is visibly independent,  $G$  has a unique perfect matching. Moreover, by our initial assumption,  $d_G(v) \geq 2$  for all  $v \in R$ . Thus if we color the edges of the unique

matching of  $G$  red and all remaining edges of  $G$  blue, then by the fact that  $d_G(v) \geq 2$  for all  $v \in R$ , we can find an alternating cycle  $C$  with red edges  $R_C$  and blue edges  $B_C$ . Since  $R_C$  and  $B_C$  match the same vertex sets, then replacing the edges in  $R_C$  with those of  $B_C$  in the unique matching will produce another matching, but that's a contradiction as  $G$  has a unique perfect matching. This proves our claim.

Next, we proceed by induction on  $n$ . The base case  $n = 1$  is immediate to see. As for the induction step, we know by the previous claim that there exists some row in  $M$  with exactly one  $\star$ . Thus we can find a permutationally equivalent matrix  $M'$  of  $M$  with the  $n$ 'th row having exactly one  $\star$  at the  $n$ 'th column. Because  $M$  is visibly independent, then so is  $M'$ . Moreover, any general diagonal in  $M'$  must contain  $M[n, n]$ . This means that the  $(n - 1) \times (n - 1)$  minor  $M'_0$ , which is obtained by deleting row  $n$  and column  $n$  of  $M'$ , is visibly independent. By our induction hypothesis, we can permute the rows and columns of  $M'_0$  to make it upper triangular, and thus we conclude that  $M$ 's rows and columns can be permuted to make it upper triangular. ◀

Thus we can characterize all visibly full rank matrices, and that help us obtain the following upper bound on the visible rank.

► **Lemma 11.** *Given an  $m \times n$  stencil  $H$ , if there are natural numbers  $a, b$  such that  $H$  has no  $a \times b$  sub-stencil of zeros, then we have  $\text{vrk}(H) < a + b$ .*

**Proof.** Assume (for the sake of a contradiction) that  $H$  has a  $(a + b) \times (a + b)$  square sub-stencil  $H_0$  that is visibly independent. By Lemma 10, we know that  $H_0$  is permutationally equivalent to an  $(a + b) \times (a + b)$  upper triangle. Since such triangle has a  $a \times b$  sub-stencil of zeros, we arrive to a contradiction. ◀

We also provide an upper bound on the rank of a stencil by the maximum number of zeros in each rows.

► **Proposition 12.** *For any  $m \times n$  stencil  $H$ . If each row of  $H$  has at most  $d$  zeros, then we have that  $\text{rk}_{\mathbb{F}}(H) \leq d + 1$  for all fields  $\mathbb{F}$  such that  $|\mathbb{F}| \geq n$ .*

**Proof.** Pick a field  $|\mathbb{F}| \geq n$ . Label the columns of  $H$  by pairwise distinct entries  $a_1, \dots, a_n \in \mathbb{F}$ . For row  $i$ , let the columns that are zero along row  $i$  be  $Z_i \subseteq \{a_1, \dots, a_n\}$ . Consider the polynomial  $p_i(x) := \prod_{a \in Z_i} (x - a)$ . Notice that  $p_i$  evaluates to zero on  $Z_i$ . On everywhere else, it evaluates to a nonzero value. Thus the matrix  $E \in \mathbb{F}^{m \times n}$  defined by  $E_{ij} = p_i(a_j)$  is an  $\mathbb{F}$ -witness of  $H$ . Moreover, since  $|Z_i| \leq d$  for each  $i \in [m]$ , then we know that the monomials  $\{1, x, \dots, x^d\}$  span the polynomials  $\{p_1, \dots, p_m\}$ . This shows that  $\text{rank}(E) \leq d + 1$  and thus  $\text{rk}_{\mathbb{F}}(H) \leq d + 1$ . ◀

We remark that the bound  $|\mathbb{F}| \geq n$  is crucial for Proposition 12. Consider the stencil  $D \in \{0, \star\}^{n \times n}$  that has  $\star$ 's everywhere except on the diagonal. Such a stencil has a visible rank of 2, but one can show that its rank over  $\mathbb{F}_2$  is at least  $n - 1$ .

## 2.4 A Rank-Nullity Type Theorem Between Stencils and Symmetric Spanoids

In this subsection, we formally setup spanoids and prove a rank-nullity type theorem between symmetric spanoids and stencils.

A *spanoid*  $\mathcal{S}$  is a collection of inference rules in the form of pairs  $(S, i)$ , which are written in the form  $S \rightarrow i$ , where  $S \subseteq [n]$  and  $i \in [n]$ . The objective in spanoids is to determine the size of the smallest subset  $B \subseteq [n]$  such that one can use the inference rules of  $\mathcal{S}$  to obtain all of  $[n]$ . Spanoids were introduced in [5] as an abstraction of LCCs, wherein they proved that the spanoid analog of  $q$ -LCCs satisfy the upper bound  $\tilde{O}(n^{(q-2)/(q-1)})$  on the rank. Moreover, they also showed that there are  $q$ -LCC spanoids for which their rank is  $\tilde{\Omega}(n^{(q-2)/(q-1)})$ .

Let us setup the definitions needed for spanoids. A derivation in  $\mathcal{S}$  of  $i \in [n]$  from a set  $T \subseteq [n]$  is a sequence of sets  $T_0 = T, T_1, \dots, T_r$  satisfying  $T_j = T_{j-1} \cup \{i_j\}$  for some  $i_j \in [n]$ ,  $j \in [r]$ , and with  $i_r = i$ . Further, for every  $j \in [r]$ , there is a rule  $(S_{j-1}, i_j)$  in  $\mathcal{S}$  for some  $S_{j-1} \subseteq T_{j-1}$ . The *span* of a set  $T \subseteq [n]$ , denoted  $\text{span}_{\mathcal{S}}(T)$ , is the set of all  $i \in [n]$  for which there is a derivation of  $i$  from  $T$ . The rank of a spanoid, denoted  $\text{rank}(\mathcal{S})$ , is the size of the smallest set  $T \subseteq [n]$  such that  $\text{span}_{\mathcal{S}}(T) = [n]$ . Finally, we define symmetric spanoids below.

► **Definition 13** (Symmetric Spanoids). *A spanoid  $\mathcal{S}$  over  $[n]$  is a symmetric spanoid if there are a collection of sets  $\{S_1, \dots, S_m\}$  so that the inference rules of  $\mathcal{S}$  are of the form  $S_j \setminus \{i\} \rightarrow \{i\}$  for any  $i \in S_j$  and  $j \in [m]$ .*

Now we may proceed to prove our theorem that relates the rank of symmetric spanoids with the visible rank of an associated stencil.

► **Theorem 14.** *For any symmetric spanoid  $\mathcal{S}$  over  $[n]$  with  $m$  sets, there exists a canonical stencil  $H$  of size  $m \times n$  such that for any collection of columns  $C \subseteq [n]$  in  $H$ , they are visibly independent if and only if  $\text{span}_{\mathcal{S}}([n] \setminus C) = [n]$ . Moreover, we have  $\text{vrk}(H) + \text{rank}(\mathcal{S}) = n$ .*

**Proof.** Define  $H[i, j] = \star$  if  $j \in S_i$  and zero otherwise. We claim that such  $H$  satisfies the conditions. Indeed, suppose that the columns  $C = \{c_1, \dots, c_k\}$  are visibly independent. Then that means there are rows  $r_1, \dots, r_k$  so that the  $k \times k$  sub-stencil formed by these columns and rows is visibly full rank. Denote this matrix as  $H_C$ . By Lemma 10, we can find permutations  $\pi, \sigma$  over  $[k]$  such that the matrix  $H'_C := H_{\pi, \sigma}$  is upper triangular. In terms of spanoids, that means  $c_{\sigma(i)} \in S_{r_{\pi(i)}}$  and  $S_{r_{\pi(i)}} \subseteq [n] \setminus \{c_{\sigma(1)}, \dots, c_{\sigma(i-1)}\}$  for all  $i \in [k]$ . We can rewrite the last set containment as  $S_{r_{\pi(i)}} \subseteq ([n] \setminus C) \cup \{c_{\sigma(i)}, \dots, c_{\sigma(k)}\}$ . Thus we apply the inference rules  $S_{r_{\pi(k)}} \setminus c_{\sigma(k)} \rightarrow c_{\sigma(k)}, S_{r_{\pi(k-1)}} \setminus c_{\sigma(k-1)} \rightarrow c_{\sigma(k-1)}, \dots, S_{r_{\pi(1)}} \setminus c_{\sigma(1)} \rightarrow c_{\sigma(1)}$  in that order with the set  $[n] \setminus C$ , to deduce that the set  $[n] \setminus C$  spans the set  $[n]$  in  $\mathcal{S}$ . Thus  $\text{span}_{\mathcal{S}}([n] \setminus C) = [n]$ .

Now, suppose that  $\text{span}_{\mathcal{S}}([n] \setminus C) = [n]$ . Then that means that we can find sets  $S_{i_1}, \dots, S_{i_k}$  and a permutation  $\sigma$  over  $[k]$  such that  $c_{\sigma(j)} \in S_{i_j}$  and we can apply the inference rules  $S_{i_1} \setminus \{c_{\sigma(1)}\} \rightarrow c_{\sigma(1)}, \dots, S_{i_k} \setminus \{c_{\sigma(k)}\} \rightarrow c_{\sigma(k)}$  in that order. That implies then that  $S_{i_j} \subseteq ([n] \setminus C) \cup \{c_{\sigma(1)}, \dots, c_{\sigma(j)}\}$ . In terms of the stencil  $H$ , that means  $H[i_j, \sigma(j)] = \star$  and  $H[i_\ell, \sigma(j)] = 0$  for  $\ell < j$ . Thus the  $k \times k$  sub-stencil  $H'$  that is restricted to the columns  $c_{\sigma(1)}, \dots, c_{\sigma(k)}$  and rows  $i_1, \dots, i_k$  in that order forms a lower triangular stencil, which is permutationally equivalent to an upper triangular stencil. Thus we deduce that the set of columns  $C$  is visibly independent.

Now, for any set  $S \subseteq [n]$  such that  $\text{span}_{\mathcal{S}}(S) = [n]$ , we know that  $[n] \setminus S$  is visibly independent in  $H$ . Thus  $n - |S| \leq \text{vrk}(H)$ . Since this holds for any such set  $S$ , then we deduce that  $\text{rank}(\mathcal{S}) + \text{vrk}(H) \geq n$ . On the other hand, for any collection of columns  $C$  in  $H$  that is visibly independent, we know that  $\text{span}_{\mathcal{S}}([n] \setminus C) = [n]$ . This implies  $n - |C| \geq \text{rank}(\mathcal{S})$ . Since this holds for any visibly independent set of columns  $C$  in  $H$ , then we find that  $n \geq \text{rank}(\mathcal{S}) + \text{vrk}(H)$ . Hence  $\text{rank}(\mathcal{S}) + \text{vrk}(H) = n$ . ◀



### 3 Constructing $q$ -LCC Stencils

In this section, we define  $q$ -LCC stencils and construct  $q$ -LCC stencils whose visible rank achieves the known lower bounds up to polylog factors.

► **Definition 15** ( $q$ -LCC Stencils). For  $\delta > 0$ , a  $\delta n^2 \times n$  stencil  $H$  whose rows are labelled by  $[n] \times [\delta n]$  is said to be a  $q$ -LCC stencil if  $H[(i, j), i] = \star$  for all  $(i, j) \in [n] \times [\delta n]$ . Moreover, for every  $k \in [n] \setminus \{i\}$ , the collection of entries  $\{M[(i, 1), k], \dots, M[(i, t), k]\}$  has at most one star, and the number of  $\star$ 's in each row of  $H$  is at most  $q + 1$ .

Now we proceed to prove our main theorem for this section.

► **Theorem 16.** For  $q \geq 3$ , there exists a  $q$ -LCC stencil  $M$  for which  $\text{vrk}(M) \leq n - \tilde{\Omega}\left(n^{(q-2)/(q-1)}\right)$ .

**Proof.** For  $(i, j) \in [n] \times [\delta n]$ . Define  $r_j^i := \{k \in [n] : H[(i, j), k] = \star\}$  to be the support of row  $(i, j)$ , and let  $G_i := \{r_j^i : j \in [\delta n]\}$  be the  $\delta n$  groups for column  $i$ . We shall show that by picking the groups  $G_i$  uniformly at random, the visible rank will at most be  $n - n^{(q-2)/(q-1)}/\log n$  with high probability.

Consider natural numbers  $s > k$  where  $s = n^{\frac{q-2}{q-1}}$  and  $k = s/\log n = n^{\frac{q-2}{q-1}}/\log n$ . Let  $E_{s,k}$  be the event that there aren't any  $(s - k) \times (n - s)$  sub-stencils in  $H$  that are all zeros. We shall show that  $E_{s,k}$  occurs with high probability, which will yield an upper bound of  $n - k$  on the visible rank via Lemma 11. We will use an equivalent form of the event  $E_{s,k}$ , which is the event that there are no  $s - k$  rows in  $H$  whose union of supports is at most  $s$ . Now, consider a collection of columns  $C$  of size  $s$  and a collection of  $s - k$  rows  $R$ , where  $R$  and  $C$  denote the collection of their supports. Enumerate  $R \cap G_i = \{r_1^i, \dots, r_{a_i}^i\}$ . By the chain rule, the definition of  $G_i$ , and the independence of the  $G_i$ 's, we find that

$$\begin{aligned} \Pr \left[ \bigwedge_{r \in R} r \subseteq C \right] &= \prod_{i=1}^n \Pr \left[ \bigwedge_{r \in R \cap G_i} r \subseteq C \right] \\ &= \prod_{i=1}^n \prod_{j=1}^{a_i} \Pr \left[ r_j^i \subseteq C \mid r_k^i \subseteq C \text{ for } k \in [j - 1] \right] \\ &= \prod_{i=1}^n \prod_{j=1}^{a_i} \Pr \left[ r_j^i \setminus \{i\} \subseteq C \setminus \{r_1^i, \dots, r_{j-1}^i\} \mid r_k^i \subseteq C \text{ for } k \in [j - 1] \right] \\ &= \prod_{i=1}^n \prod_{j=1}^{a_i} \frac{\binom{s-(j-1)q-1}{q}}{\binom{n-(j-1)q-1}{q}} \leq \left( \frac{\binom{s-1}{q}}{\binom{n-1}{q}} \right)^{s-k} \leq \left( \frac{s}{n} \right)^{q(s-k)} \end{aligned}$$

Therefore, from the definition of  $E_{s,k}$ , by applying a Union Bound over all possible collections of columns  $C$  of size  $s$  and  $s - k$  collections of rows  $R$  and use the bound  $\binom{a}{b} \leq \left(\frac{ea}{b}\right)^b$ , we

deduce that

$$\begin{aligned}
\Pr[E_{s,k}] &\leq \binom{n}{s} \binom{\delta ns}{s-k} \left(\frac{s}{n}\right)^{q(s-k)} \leq \left(\frac{en}{s}\right)^s \left(\frac{e\delta ns}{s-k}\right)^{s-k} \left(\frac{s}{n}\right)^{q(s-k)} \\
&= \left(\frac{en}{s}\right)^k \left(\frac{e^2 \delta s^{q-1}}{\left(1 - \frac{k}{s}\right) n^{q-2}}\right)^{s-k} \\
&= \left(en^{1/(q-1)}\right)^k \left(\frac{e^2 \delta}{\left(1 - \frac{1}{\log n}\right)}\right)^{n^{\frac{q-2}{q-1}} \left(1 - \frac{1}{\log n}\right)}
\end{aligned}$$

Thus for small enough  $\delta < e^{-2}$ , the quantity above becomes  $\exp\left(-\Omega(n^{(q-2)/(q-1)})\right)$ . Thus we can find a  $q$ -LCC stencil  $M$  such that no  $s-k$  rows whose support is entirely contained within  $s$  columns. That is equivalent to saying that there is no  $(s-k) \times (n-s)$  sub-stencil that is all zeros. By Lemma 11, we therefore conclude that  $\text{vrk}(M) < n-k = n - \Omega\left(n^{\frac{q-2}{q-1}}/\log n\right)$ . ◀

#### 4 $t$ -DRGP Stencils

We now define the stencils that capture the requirement of each codeword symbol having  $t$  disjoint recovery groups.

► **Definition 17** ( $t$ -DRGP Stencils). *A  $tn \times n$  stencil  $H$  whose rows are labelled by  $[n] \times [t]$  is said to be a  $t$ -DRGP stencil if  $H[(i, j), i] = \star$  for all  $(i, j) \in [n] \times [t]$ . Moreover, for every  $k \in [n] \setminus \{i\}$ , the collection of entries  $\{M[(i, 1), k], \dots, M[(i, t), k]\}$  has at most one star.*

Now we proceed to prove our main theorem for this section.

► **Theorem 18.** *For any fixed natural number  $t \geq 2$ , there exists a  $t$ -DRGP stencil  $H$  satisfying  $\text{vrk}(H) \leq O(t^2 \log n)$ .*

**Proof.** Consider a random  $t$ -DRGP stencil  $H$  as follows: define the set of entries  $S_{i,j} := \{(i, s), j \mid s \in [t]\}$ . Set for each  $i \neq j \in [n]$ , set all of the entries  $S_{i,i}$  to be  $\star$ 's, and uniformly sample an entry from  $S_{i,j}$  to be a  $\star$  while everything else in  $S_{i,j}$  is set to be zero.

We will show that  $\text{vrk}(H) \leq c_1 \log n$  occurs with high probability. Indeed, fix  $k \in \mathbb{N}$ . Given any square sub-stencil  $H_0$  of  $H$  of size  $k$ , we have by Lemma 10 that if  $H_0$  is visibly independent, then we must have at least  $\binom{k}{2}$  zeros. Let this set of entries be  $Z \subseteq ([n] \times [t]) \times [n]$ . Since  $Z$  must all be zeros, then we deduce that  $Z \subseteq \cup_{i \neq j} S_{i,j}$ . Since each  $S_{i,j}$  has size  $t$ , then  $Z$  has at least  $\binom{k}{2}/t$  entries, each of which belongs to an  $S_{i,j}$  that is different than the other. That is, for each  $i \neq j \in [n]$  such that  $Z \cap S_{i,j} \neq \emptyset$ , arbitrarily pick an entry  $e \in Z \cap S_{i,j}$ , and let  $T$  be those set of entries. Then we know that  $T \geq \binom{k}{2}/t$ . Moreover, the events that the entries in  $T$  are zero are all independent, with each having a chance of at most  $1 - 1/t$  of being zero. Thus the chance that  $Z$  is all-zeros is at most  $(1 - 1/t)^{\binom{k}{2}/t}$ . By a Union Bound over all possible such  $Z$ 's, which is enumerated over all  $(k!)^2$  different permutations of the rows and columns, we deduce that

$$\Pr[H_0 \text{ is visibly independent}] \leq (k!)^2 \left(1 - \frac{1}{t}\right)^{\frac{\binom{k}{2}}{t}} \leq \left(k^2 \left(1 - \frac{1}{t}\right)^{\frac{k-1}{2t}}\right)^k.$$

And by applying another Union Bound over all such  $H_0$ , we find that

$$\Pr[\text{vrk}(H) \geq k] \leq \binom{tn}{k} \binom{n}{k} \left( k^2 \left( 1 - \frac{1}{t} \right)^{\frac{k-1}{2t}} \right)^k \leq \left( k^2 t n^2 \left( 1 - \frac{1}{t} \right)^{\frac{k-1}{2t}} \right)^k.$$

Picking  $k = 6t^2 \ln n + 1$  makes the right hand side less than 1 for large enough  $n$ . ◀

## 5 Tensor Products

In this section, we will be introducing a tensor product operation on stencils and explore its properties. Given the natural tensor product  $A \otimes B$  for matrices  $A$  and  $B$ , notice that the support of  $A \otimes B$  is determined completely by the support of the matrices  $A$  and  $B$ . As a consequence of this observation, we will be able to define the stencil of  $A \otimes B$  based solely on the stencils of  $A$  and  $B$ . This leads us to our definition of a tensor product over stencils.

► **Definition 19** (Tensor product). *Given an  $A_1 \times B_1$  stencil  $H_1$  and an  $A_2 \times B_2$  stencil  $H_2$ , let  $H_1 \otimes H_2$  be a  $(A_1 \times A_2) \times (B_1 \times B_2)$  stencil such that*

$$(H_1 \otimes H_2)[(a_1, a_2), (b_1, b_2)] = \begin{cases} \star & \text{if } H_1[a_1, b_1] \text{ and } H_2[a_2, b_2] \text{ both equal } \star, \\ 0 & \text{if at least one of } H_1[a_1, b_1] \text{ and } H_2[a_2, b_2] \text{ equals } 0. \end{cases}$$

We remark that our tensor product follows similar properties as the natural tensor product for matrices, such as associativity and non-commutativity.

### 5.1 Algebraic witnesses of tensor products

In this subsection, we will be proving that any algebraic witnesses of the stencils  $H_1$  and  $H_2$  is also an algebraic witness of  $H_1 \otimes H_2$ . This will therefore show us that the  $\mathbb{F}$ -rank is a *sub-multiplicative* function with respect to the tensor product.

► **Proposition 20.** *Let  $M$  and  $N$  be matrices over a field  $\mathbb{F}$  who are  $\mathbb{F}$ -witnesses to stencils  $H_1, H_2$ , respectively. Then  $M \otimes N$  is an  $\mathbb{F}$ -witness of  $H_1 \otimes H_2$ .*

**Proof.** For every entry in  $H_1 \otimes H_2$ , we know that  $(H_1 \otimes H_2)[(i_1, i_2), (j_1, j_2)]$  is a  $\star$  if and only if  $H_1[i_1, j_1]$  and  $H_2[i_2, j_2]$  are both  $\star$ 's. This holds if and only if  $M_{i_1 j_1}$  and  $N_{i_2 j_2}$  are both nonzero. Because  $(M \otimes N)_{(i_1, i_2), (j_1, j_2)} = M_{i_1 j_1} N_{i_2 j_2}$ , then the entry  $(M \otimes N)_{(i_1, i_2), (j_1, j_2)}$  is nonzero if and only if  $M_{i_1 j_1}$  and  $N_{i_2 j_2}$  are both nonzero. Thus  $M \otimes N$  is an  $\mathbb{F}$ -witness of  $H_1 \otimes H_2$ . ◀

By applying Proposition 20 on the  $\mathbb{F}$ -witnesses of  $H_1$  and  $H_2$  with the smallest ranks, we deduce the following corollary.

► **Corollary 21.** *For a field  $\mathbb{F}$ , we have the inequality  $rk_{\mathbb{F}}(H_1)rk_{\mathbb{F}}(H_2) \geq rk_{\mathbb{F}}(H_1 \otimes H_2)$*

### 5.2 Visible rank and tensor products

In this subsection, we will show that the tensor product of two visibly full rank stencils is also visibly full rank. This will therefore show us that the visible rank is *super-multiplicative* with respect to the tensor product. We will also show an upper bound on the visible rank of the tensor product with respect to the visible rank of one of the stencils.

► **Proposition 22.** *Given visibly independent matrices  $A$  and  $B$  of size  $n$ , their tensor  $A \otimes B$  is also visibly independent.*

**Proof.** By Lemma 10, we know that there are permutations  $\pi_A, \sigma_A, \pi_B, \sigma_B$  on  $[n]$  such that the stencils  $A_0 = (A)_{\pi_A, \sigma_A}$  and  $B_0 = (B)_{\pi_B, \sigma_B}$  are both upper triangular stencils. Moreover, we see that  $A_0 \otimes B_0 = (A \otimes B)_{(\pi_A, \pi_B), (\sigma_A, \sigma_B)}$ . Therefore, it suffices for us to show that  $A_0 \otimes B_0$  is an upper triangular stencil.

Consider the lexicographical ordering on  $[n] \times [n]$ . When  $(i_1, i_2) > (j_1, j_2)$ , then we know that one of the inequalities  $i_1 > j_1$  and  $i_2 > j_2$  must hold, which means one of  $A_0[i_1, j_1]$  or  $B_0[i_2, j_2]$  must be a zero. This proves that  $(A_0 \otimes B_0)[(i_1, i_2), (j_1, j_2)] = 0$  whenever  $(i_1, i_2) > (j_1, j_2)$ . As for when  $(i_1, i_2) = (j_1, j_2)$ , then we immediately know that  $(A_0 \otimes B_0)[(i_1, i_2), (i_1, i_2)] = \star$  as  $A_0[i_1, i_1] = B_0[i_2, i_2] = \star$ . Hence  $A_0 \otimes B_0$  is an upper triangular stencil with respect to the lexicographical ordering. ◀

Given stencils  $H_1$  and  $H_2$ , we know by Proposition 22 that the tensor product of any of their visibly full rank sub-stencils will also be visibly full rank in  $H_1 \otimes H_2$ . This yields us the following corollary.

► **Corollary 23.** *For stencils  $H_1$  and  $H_2$ , We have the inequality  $\text{vrk}(H_1 \otimes H_2) \geq \text{vrk}(H_1)\text{vrk}(H_2)$ .*

Lastly, we end this subsection with an upper bound on the visible rank of  $H_1 \otimes H_2$ .

► **Proposition 24.** *For stencils  $H_1$  and  $H_2$  of sizes  $m_1 \times n_1$  and  $m_2 \times n_2$ , respectively, We have the inequality  $\text{vrk}(H_1 \otimes H_2) \leq \text{vrk}(H_1)n_2$ .*

**Proof.** Consider a visibly full rank substencil  $M$  in  $H_1 \otimes H_2$  of size  $k \times k$ . By Lemma 10, we can find a  $k \times k$  permutationally equivalent matrix  $M'$  of  $M$ . Let the columns and rows of  $M'$  be indexed as  $(a_1, b_1), \dots, (a_k, b_k)$  and  $(c_1, d_1), \dots, (c_k, d_k)$ . Define  $b_{\max}$  to be the most frequent column of  $H_2$  in  $\{b_1, \dots, b_k\}$ . Let  $I := \{i_1, \dots, i_s\}$  be the indices such that  $b_{i_j} = b_{\max}$ . We know that  $s \geq k/n_2$  by definition of  $b_{\max}$ . Moreover, the substencil  $M_0$  of  $M'$  attained by taking the rows and columns with index in  $I$  is upper triangular. Since  $M'[(c_{i_j}, d_{i_j}), (a_{i_j}, b_{\max})] = \star$ , then  $H_2[d_{i_j}, b_{\max}] = \star$  for all  $j \in [s]$ . Because  $M_0$  is upper triangular, then if we consider the  $s \times s$  substencil  $N_1$  in  $H_1$  with columns and rows  $\{a_{i_1}, \dots, a_{i_s}\}$  and  $\{c_{i_1}, \dots, c_{i_s}\}$ , we deduce that  $N_1$  is upper triangular. Thus  $\text{vrk}(H_1) \geq \text{vrk}(N_1) = s \geq k/n_2$ . Hence  $n_2 \text{vrk}(H_1) \geq k$  for any visibly full rank  $k \times k$  substencil  $M$  in  $H_1 \otimes H_2$ . ◀

### 5.3 Visible rank of the tensor powers

Naturally, one would be interested in tensoring a stencil  $H$  with itself several times and examine such a stencil.

► **Definition 25 (Tensor power).** *Given an  $m \times n$  stencil  $H$ , the  $k$ 'th tensor of  $H$  is the  $m^k \times n^k$  stencil  $H^{\otimes k}$  defined as  $H^{\otimes k} := \underbrace{H \otimes H \otimes \dots \otimes H}_{k \text{ times}}$ .*

By combining all the results from the previous subsections, we obtain the following corollary.

► **Corollary 26.** *For a natural number  $k$  and an  $m \times n$  stencil  $H$ , we have the inequality*

$$\text{rk}_{\mathbb{F}}(H) \geq \text{rk}_{\mathbb{F}}(H^{\otimes k})^{1/k} \geq \text{vrk}(H^{\otimes k})^{1/k} \geq \text{vrk}(H)$$

*Moreover, we also have the inequality  $\text{vrk}(H^{\otimes k}) \leq n^{k-1} \text{vrk}(H)$ .*

From the previous corollary, we can see that by consider the visible rank of the higher tensor powers of a stencil  $H$ , one might hope to attain better lower bounds on the  $\mathbb{F}$ -rank. Naturally, one would define the highest possible bound achieved through this vein.

► **Definition 27 (Visible Capacity).** *The visible capacity of a stencil  $H$ , denoted as  $\Upsilon(H)$ , is defined as  $\Upsilon(H) := \sup_k \text{vrk}(H^{\otimes k})^{1/k}$ .*

By Corollary 26, we deduce that  $\text{rk}_{\mathbb{F}}(H) \geq \Upsilon(H)$  over any field  $\mathbb{F}$ . It is not known to us if there are stencils for which there is a gap between its visible capacity and all its  $\mathbb{F}$ -ranks. We leave the discussion of this point to Question 1 in Section 7.

## 6 Tensor Powers of Stencils for 2-DRGP Codes and $q$ -LCCs

In this section, we will be considering the tensor product in the previous section and use it to analyze the visible rank of the tensor powers of 2-DRGP and  $q$ -LCC stencils to see if they might yield better lower bounds on the rank via Corollary 26.

### 6.1 2-DRGP stencils

In this subsection, we prove that the second tensor power of an arbitrary 2-DRGP stencil has a large visible rank.

► **Theorem 28.** *For any 2-DRGP stencil  $H$ , we have  $\text{vrk}(H \otimes H) \geq n$ .*

**Proof.** We cite [24, 30] for the proof of this part. We will follow the notations given in [30] closely. While both proofs show that  $\text{rk}_{\mathbb{F}}(H) \geq \sqrt{2n} - O(1)$ , we will prove that  $\text{rk}_{\mathbb{F}}(H) \geq \sqrt{n}$  by showing that  $\text{vrk}(H \otimes H) \geq n$  and then applying Corollary 26. We rewrite their proofs in terms of tensor powers.

Consider the  $n \times n$  sub-stencil  $D$  in  $H \otimes H$  whose columns are  $(1, 1), \dots, (n, n)$  and whose rows are  $((1, 1), (1, 2)), \dots, ((n, 1), (n, 2))$ . We claim that  $D$  has  $\star$ 's along the diagonal and zero everywhere else, which implies that it is visibly full rank. Indeed, the entry  $D[((i, 1), (i, 2)), (j, j)]$  equals  $\star$  if and only if both  $H[(i, 1), j]$  and  $H[(i, 2), j]$  are  $\star$ 's. Since  $H$  is a 2-DRGP stencil, this happens precisely when  $i = j$ . Thus  $D$  is a diagonal stencil. ◀

Thus by Corollary 26, we obtain a lower bound  $\text{rk}_{\mathbb{F}}(H) \geq \sqrt{n}$  for any field  $\mathbb{F}$ . On the other hand, the best known lower bounds yield  $\text{rk}_{\mathbb{F}}(H) \geq \sqrt{2n} - O(1)$ , and so one might be interested in achieving this lower bound through the viewpoint of tensor products. In order to improve the lower bound that we have, we first have to translate our proof into linear-algebraic terms.

Given a field  $\mathbb{F}$ , suppose that we have an  $\mathbb{F}$ -witness  $A$  of the 2-DRGP stencil  $H$  whose rank is  $r$ . Decompose  $A = MN$  where  $M$  is an  $2n \times r$  matrix and  $N$  is an  $r \times n$  matrix. Denote the  $i$ 'th column of  $N$  by  $w_i$ . Then the proof of Theorem 28 is equivalent to saying that the tensors  $\{w_i \otimes w_i\}_{i=1}^n$  are linearly independent. Since they live in a space  $\mathbb{F}^r \otimes \mathbb{F}^r$ , then we obtain the inequality  $r^2 \geq n$ , which gives us the same lower bound as we obtained

in Theorem 28. Now, if one is more careful about the vector space, one can notice that the tensors  $\{w_i \otimes w_i\}_{i=1}^n$  belong to the space of *symmetric* tensors, which has a dimension of  $\binom{r+1}{2}$ . Thus we obtain the inequality  $\binom{r+1}{2} \geq n$ , which gives us  $r \geq \sqrt{2n} - O(1)$ .

## 6.2 $q$ -LCC stencils

In this subsection, we will show that the visible ranks of the  $k$ 'th tensor power of a  $q$ -LCC stencil would not improve the current bound of  $n - \tilde{\Omega}(n^{(q-2)/(q-1)})$  in Theorem 16 for  $k \leq \text{polylog}(n)$ . More generally, we will show that the visible rank of small tensor powers could be not significantly bigger than the visible rank for the regime of high-rate stencils.

► **Proposition 29.** *Let  $H$  be an  $m \times n$  stencil whose visible rank is at most  $n - s$ . For any fixed natural number  $k$ , we have  $\text{vrk}(H^{\otimes k})^{1/k} \leq n - \frac{s}{k}$ .*

**Proof.** By Corollary 26 and the inequality  $(1 - x)^{1/k} \leq 1 - \frac{x}{k}$  for  $x \geq 0$ , we have that

$$\text{vrk}(H^{\otimes k})^{1/k} \leq \left(n^{k-1} \text{vrk}(H)\right)^{1/k} \leq \left(n^{k-1}(n - s)\right)^{1/k} = n \left(1 - \frac{s}{n}\right)^{1/k} \leq n \left(1 - \frac{s}{kn}\right) = n - \frac{s}{k}. \blacktriangleleft$$

From Proposition 29, we notice that looking at the visible rank of the  $n^{o(1)}$  level tensor powers of a  $q$ -LCC stencil would not improve the current bounds on  $q$ -LCCS by a polynomial factor. We state it more formally in the following corollary.

► **Corollary 30.** *Let  $H$  be a  $q$ -LCC stencil whose visible rank is at most  $n - \tilde{\Omega}(n^{(q-2)/(q-1)})$ . For any natural number  $k$ , we have  $\text{vrk}(H^{\otimes k})^{1/k} \leq n - \tilde{\Omega}(n^{(q-2)/(q-1)})/k$ .*

## 7 Further Directions and Discussion

Stencils provide an initial framework toward combinatorial methods for effectively lower bounding the rank of a matrix. However, we have seen the limitations of the visible rank with 2-DRGP stencils as well as small tensor powers of  $q$ -LCC stencils. We leave the reader with questions that remain open about the current framework and possibilities of imposing further restrictions on the model to obtain sharper lower bounds on the rank.

1. While we may have shown that the  $k$ 'th tensor power of a  $q$ -LCC does not yield better lower bounds for  $k \leq n^{o(1)}$ , this does not rule out the possibility that the visible capacity might yield better lower bounds. In fact, we do not know if there are any stencils for which the visible capacity does not match the lowest possible rank for the stencil. In other words, does there exist a stencil  $H$  such that  $\text{rk}_{\mathbb{F}}(H) > \Upsilon(H)$  for every field  $\mathbb{F}$ ?
2. Random 2-DRGP patterns have shown an exponential gap between the visible ranks of the first and second tensor powers, but one might be curious to see an exponential separation between the visible ranks of the  $(t - 1)$ 'th and  $t$ 'th tensor powers. Formally speaking, for every natural number  $t$  greater than 1, does there exist a  $m \times n$  stencil  $H$  and a constant  $c > 0$  such that  $\text{vrk}(H^{\otimes i}) = O(\log^{ci} n)$  for  $i = 1, 2, \dots, t - 1$  while  $\text{vrk}(H^{\otimes t}) = \Omega(n)$ ? Such a phenomena holds with Shannon capacity [1].
3. In this paper, we shown a polynomial gap between  $\text{rk}_{\mathbb{F}}(H)$  and  $\text{vrk}(H)$  by proving that there are 2-DRGP stencils  $H$  with  $\text{vrk}(H) = O(\log n)$  and  $\text{rk}_{\mathbb{F}}(H) = \Omega(\sqrt{n})$ . On the other hand, can there also be a similar polynomial gap with the quantities  $n - \text{rk}_{\mathbb{F}}(H)$  and  $n - \text{vrk}(H)$ ? From Proposition 29, we have seen that the visible ranks of the  $k$ 'th tensor

power for  $k \leq n^{o(1)}$  would not suffice to show this polynomial gap. Nonetheless, it still leaves the possibility of using the visible capacity to showing this polynomial gap, but we do not know of any methods that can lower bound the visible capacity other than the visible ranks of finite tensor powers. Note that this question is the symmetric spanoid version of Question 2 posed in [5].

---

## References

- 1 Noga Alon and Eyal Lubetzky. The shannon capacity of a graph and the independence numbers of its powers. *IEEE Transactions on Information Theory*, 52(5):2172–2176, 2006.
- 2 Ziv Bar-Yossef, Yitzhak Birk, T. S. Jayram, and Tomer Kol. Index coding with side information. *IEEE Trans. Inf. Theory*, 57(3):1479–1494, 2011.
- 3 Boaz Barak, Zeev Dvir, Amir Yehudayoff, and Avi Wigderson. Rank bounds for design matrices with applications to combinatorial geometry and locally correctable codes. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 519–528, 2011.
- 4 Zeev Dvir. Incidence theorems and their applications. *arXiv preprint arXiv:1208.5073*, 2012.
- 5 Zeev Dvir, Sivakanth Gopi, Yuzhou Gu, and Avi Wigderson. Spanoids - an abstraction of spanning structures, and a barrier for LCCs. *SIAM J. Comput.*, 49(3):465–496, 2020.
- 6 Arman Fazeli, Alexander Vardy, and Eitan Yaakobi. Codes for distributed PIR with low storage overhead. In *IEEE International Symposium on Information Theory*, pages 2852–2856, 2015.
- 7 Mathew C Francis, Dalu Jacob, and Satyabrata Jana. Uniquely restricted matchings in interval graphs. *SIAM Journal on Discrete Mathematics*, 32(1):148–172, 2018.
- 8 S. Luna Frank-Fischer, Venkatesan Guruswami, and Mary Wootters. Locality via partially lifted codes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, volume 81 of *LIPICs*, pages 43:1–43:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.
- 9 Harold N Gabow, Haim Kaplan, and Robert E Tarjan. Unique maximum matching algorithms. *Journal of Algorithms*, 40(2):159–183, 2001.
- 10 Alexander Golovnev, Oded Regev, and Omri Weinstein. The minrank of random graphs. *IEEE Trans. Inf. Theory*, 64(11):6990–6995, 2018.
- 11 Martin Charles Golumbic, Tirza Hirst, and Moshe Lewenstein. Uniquely restricted matchings. *Algorithmica*, 31(2):139–154, 2001.
- 12 Parikshit Gopalan, Cheng Huang, Huseyin Simitci, and Sergey Yekhanin. On the locality of codeword symbols. *IEEE Transactions on Information theory*, 58(11):6925–6934, 2012.
- 13 Sivakanth Gopi. *Locality in coding theory*. PhD thesis, Princeton University, 2018.
- 14 Alan Guo, Swastik Kopparty, and Madhu Sudan. New affine-invariant codes from lifting. In *Proceedings of the Innovations in Theoretical Computer Science Conference*, pages 529–540, 2013.
- 15 Willem H. Haemers. On some problems of Lovász concerning the Shannon capacity of a graph. *IEEE Trans. Inf. Theory*, 25(2):231–232, 1979.
- 16 Thanh Minh Hoang, Meena Mahajan, and Thomas Thierauf. On the bipartite unique perfect matching problem. In *International Colloquium on Automata, Languages, and Programming*, pages 453–464. Springer, 2006.
- 17 Cheng Huang, Huseyin Simitci, Yikang Xu, Aaron Ogus, Brad Calder, Parikshit Gopalan, Jin Li, and Sergey Yekhanin. Erasure coding in windows azure storage. In *USENIX Annual Technical Conference*, 2012.
- 18 Eran Iceland and Alex Samorodnitsky. On coset leader graphs of structured linear codes. *Discrete and Computational Geometry*, 63:560–576, 2020.

- 19 Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 80–86, 2000.
- 20 Ray Li and Mary Wootters. Lifted multiplicity codes and the disjoint repair group property. *IEEE Trans. Inf. Theory*, 67(2):716–725, 2021.
- 21 László Lovász. Communication complexity: A survey. Technical report, Princeton University TR-204-89, February 1989. University of Zurich, Department of Informatics.
- 22 Sounaka Mishra. On the maximum uniquely restricted matching for bipartite graphs. *Electronic Notes in Discrete Mathematics*, 37:345–350, 2011.
- 23 C. Ramya. Recent progress on matrix rigidity—a survey. *arXiv preprint arXiv:2009.09460*, 2020.
- 24 Sankeerth Rao and Alexander Vardy. Lower bound on the redundancy of PIR codes. *arXiv preprint arXiv:1605.01869*, 2016.
- 25 Maheswaran Sathiamoorthy, Megasthenis Asteris, Dimitris Papailiopoulos, Alexandros Dimakis, Ramkumar Vadali, Scott Chen, and Dhruba Borthakur. XORing elephants: Novel erasure codes for big data. *Proc. VLDB Endow.*, 6:325–336, 2013.
- 26 Maguy Tréfois. *Topics in combinatorial matrix theory*. PhD thesis, PhD thesis, UCL, 2016.
- 27 Maguy Trefois and Jean-Charles Delvenne. Zero forcing sets, constrained matchings and minimum rank. *Linear and Multilinear Algebra*, 2013.
- 28 Leslie G Valiant. Graph-theoretic arguments in low-level complexity. In *International Symposium on Mathematical Foundations of Computer Science*, pages 162–176. Springer, 1977.
- 29 David P. Woodruff. A quadratic lower bound for three-query linear locally decodable codes over any field. *J. Comput. Sci. Technol.*, 27(4):678–686, 2012.
- 30 Mary Wootters. Linear codes with disjoint repair groups. *unpublished manuscript*, 2016.
- 31 Sergey Yekhanin. Locally decodable codes. *Foundations and Trends in Theoretical Computer Science*, 6(3):139–255, 2012. doi:10.1561/04000000030.



# Pseudorandom Generators for Read-Once Monotone Branching Programs

Dean Doron  



Department of Computer Science, Stanford University, CA, USA

Raghu Meka  

Department of Computer Science, University of California at Los Angeles, CA, USA

Omer Reingold  

Department of Computer Science, Stanford University, CA, USA

Avishay Tal  

Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, CA, USA

Salil Vadhan  

John A. Paulson School of Engineering & Applied Sciences, Harvard University, Cambridge, MA, USA

---

## Abstract

---

Motivated by the derandomization of space-bounded computation, there has been a long line of work on constructing pseudorandom generators (PRGs) against various forms of read-once branching programs (ROBPs), with a goal of improving the  $O(\log^2 n)$  seed length of Nisan’s classic construction [33] to the optimal  $O(\log n)$ .

In this work, we construct an explicit PRG with seed length  $\tilde{O}(\log n)$  for constant-width ROBPs that are *monotone*, meaning that the states at each time step can be ordered so that edges with the same labels never cross each other. Equivalently, for each fixed input, the transition functions are a monotone function of the state. This result is complementary to a line of work that gave PRGs with seed length  $O(\log n)$  for (ordered) *permutation* ROBPs of constant width [7, 26, 12, 37], since the monotonicity constraint can be seen as the “opposite” of the permutation constraint.

Our PRG also works for monotone ROBPs that can read the input bits in any order, which are strictly more powerful than read-once  $AC^0$ . Our PRG achieves better parameters (in terms of the dependence on the depth of the circuit) than the best previous pseudorandom generator for read-once  $AC^0$ , due to Doron, Hatami, and Hoza [13].

Our pseudorandom generator construction follows Ajtai and Wigderson’s approach of iterated pseudorandom restrictions [1, 18]. We give a randomness-efficient width-reduction process which proves that the branching program simplifies to an  $O(\log n)$ -junta after only  $O(\log \log n)$  independent applications of the Forbes–Kelley pseudorandom restrictions [16].

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

**Keywords and phrases** Branching programs, pseudorandom generators, constant depth circuits

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.58

**Category** RANDOM

**Funding** *Dean Doron:* Supported by NSF award CCF-1763311 and Simons Foundation investigators award 689988.

*Raghu Meka:* Supported by NSF Career award CCF-1553605 and NSF award CCF-2007682.

*Omer Reingold:* Supported by Supported by NSF award CCF-1763311 and Simons Foundation investigators award 689988.

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

**Acknowledgements** We are grateful to Kristoffer Hansen for pointing us to [3] and explaining how their results imply Theorem 3.



© Dean Doron, Raghu Meka, Omer Reingold, Avishay Tal, and Salil Vadhan; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 58; pp. 58:1–58:21



Leibniz International Proceedings in Informatics

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

## 1 Introduction

Branching programs are a fundamental model in computational complexity, capturing both space-bounded computation and circuit classes. In this paper, we study a restricted class of branching programs we call *monotone*, giving a new pseudorandom generator for their read-once version.

### 1.1 Monotone Branching Programs

First we recall the standard definition of a layered branching program:

► **Definition 1.** For  $w, n, s \in \mathbb{N}$ , a (layered) branching program (BP)  $B$  on  $n$  variables, with length  $s$  and width  $w$ , or an  $[n, s, w]$  BP, is specified by a start state  $v_0 \in [w]$ , a set of accept states  $V_{\text{acc}} \subseteq [w]$ , a sequence of variable indices  $i_1, \dots, i_s \in [n]$ , and sequence of transition functions  $E_j: \{0, 1\} \times [w] \rightarrow [w]$  for  $j = 1, \dots, s$ .

A branching program  $B$  as above naturally defines a function  $B: \{0, 1\}^n \rightarrow \{0, 1\}$ : Start at the starting state  $v_0$ , and then for  $j = 1, \dots, s$ , read the input bit  $x_{i_j}$  and then transition to state  $v_j = E_j(x_{i_j}, v_{j-1})$ . The branching program accepts ( $B(x) = 1$ ) if  $v_n \in V_{\text{acc}}$  and rejects ( $B(x) = 0$ ) otherwise.

$B$  is a read-once branching program, or an  $[n, w]$  ROBP, if  $s = n$  and  $i_1, \dots, i_s$  is a permutation of  $[n]$ . If this is the identity permutation (i.e. the variables are read in order), then we say  $B$  is an ordered branching program.

A layered branching program  $B$  has an associated directed graph. The vertex set has  $s+1$  layers of  $w$  vertices each. For each  $j = 1, \dots, s$ , layer  $j$  is labelled with an input variable, namely  $x_{i_j}$ , and there are two edges, labelled 0 and 1, going from each vertex  $v$  in layer  $j$  to vertices layer  $j+1$ , namely  $E_j(0, v)$  and  $E_j(1, v)$ .

We now introduce the model of *monotone* programs that we consider.

► **Definition 2 (monotone branching program (MBP)).** We say a BP  $B$  is monotone if for every  $j \in [s]$  and  $\sigma \in \{0, 1\}$ , the  $j$ -th transition function with input bit restricted to  $\sigma$ , denoted  $E_j^\sigma \triangleq E_j(\sigma, \cdot): [w] \rightarrow [w]$ , is a monotone function according to the standard ordering of  $[w]$ , i.e. if  $v \geq v'$ , then  $E_j^\sigma(v) \geq E_j^\sigma(v')$ .

That is, put differently, if we draw the layered graph as an  $w \times (s+1)$  grid, then whenever we consider the edges associated with a fixed input  $x$ , there are no edges crossing. We will refer to BPs that are both monotone and read once as read-once MBPs.

It is important to note that this definition only requires monotonicity with respect to the *state* of the branching program; MBPs can easily compute functions that are non-monotone as a function of their *input* (as we will see below). We remark that the definition of read-once MBPs as defined here is different from the notion of *locally monotone* studied in [10]. Importantly, the latter property is not preserved under restrictions and hence is less nice structurally. The read-once definition also coincides with the notion of *monotone ROBPs* as defined in [31], if we require all reject states to precede the accepting ones in the last layer.<sup>1</sup> However, the formulation above is more convenient for us.

<sup>1</sup> In fact, for the sake of constructing PRGs, we can remove this requirement by replacing  $\varepsilon$  with  $\varepsilon/w$ .

## 1.2 Monotone Branching Programs and $AC^0$

Recall Barrington’s celebrated theorem that constant-width branching programs are equivalent in power to  $NC^1$  circuits [2]. However, when we restrict to *monotone* branching programs, then they become equivalent in power to the much weaker  $AC^0$  circuits. Our model of monotone branching programs is closely related to the models of planar branching programs studied in [3, 4] and the following can be deduced from their results:<sup>2</sup>

► **Theorem 3** (corollary of [4]). *A sequence of functions  $f_n : \{0, 1\}^n \rightarrow \{0, 1\}$  is in  $AC^0$  if and only if it is computable by a constant-width MBP of polynomial length.*

In this paper, our focus is on read-once MBPs. We prove that these are *strictly stronger* than read-once  $AC^0$ :

► **Proposition 4.**

1. *If a sequence of functions  $f_n : \{0, 1\}^n \rightarrow \{0, 1\}$  is in read-once  $AC^0$ , then it can be computed by constant-width read-once MBP. Moreover, if  $f_n$  can be computed in depth  $w$  read-once  $AC^0$ , then it can be computed by width  $w + 1$  read-once MBPs.*
2. *For every  $n \geq 3$ , there exists a function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  computable by a width 3 read-once MBP, but not computable by any read-once De Morgan formula (regardless of depth).*

Item 1 is proven in the same way as the easier direction of Theorem 3, noting that if we start with a read-once  $AC^0$  circuit, we end up with a read-once MBP. Item 2 is proven by showing that simple functions, like checking whether the input contains at least two ones cannot be computed by a read-once De Morgan formula, but can be computed by width three MBPs. We give the proof for Item 2 in Section 4.

Thus, constant-width read-once MBPs form an intermediate class between read-once  $AC^0$  and  $AC^0$ .<sup>3</sup>

## 1.3 Pseudorandom Generators for Read-Once Branching Programs

A longstanding quest in complexity theory is to understand the power of randomness in relation to space complexity. A central challenge in this direction is to construct pseudorandom generators for read-once branching programs.

In this work we study the question of designing explicit PRGs for small-width ROBPs.

► **Definition 5.** *Given a class of functions  $\mathcal{F} : \{0, 1\}^n \rightarrow \{0, 1\}$ , a function  $G : \{0, 1\}^r \rightarrow \{0, 1\}^n$  is a PRG for  $\mathcal{F}$  with error  $\varepsilon$  if for any  $f \in \mathcal{F}$ , we have*

$$\left| \Pr_{y \in_u \{0, 1\}^r} [f(G(y)) = 1] - \Pr_{x \in_u \{0, 1\}^n} [f(x) = 1] \right| \leq \varepsilon.$$

*We call  $r$  the seed length of the generator and the generator is explicit if its output can be computed in polynomial time (in  $n$ ). We often say  $G$   $\varepsilon$ -fools  $\mathcal{F}$ .*

<sup>2</sup> In an earlier version of our paper [15], we claimed Theorem 3 as a new contribution. Kristoffer Hansen then explained to us how the result follows from [3]. More precisely, there are simple gadget reductions to show that a constant-width monotone branching program according to our definition can be simulated by a planar branching program according to the definition of [3]. Thus the result of [3] establishing the equivalence of the latter with  $AC^0$  implies the “if” direction of Theorem 3. The “only if” direction is much easier, and amounts to observing that the standard simulation of  $AC^0$  by constant-width branching programs yields a monotone program. For completeness, we provide a direct and self-contained proof of both directions of Theorem 3 in Appendix A.

<sup>3</sup> Both inclusions are strict, since there are functions in  $AC^0$  that cannot be computed by circuits of size smaller than  $n^4$ , and the proof of Theorem 3 shows that any constant-width read-once MBP can be simulated by an  $AC^0$  circuit of size  $O(n^3)$ .

Designing *pseudorandom generators* against ordered ROBPs has received a lot of attention and is intimately connected to the question of understanding the power of randomness vs. space. It has also found a number of applications beyond derandomizing space. ([29, 35, 21, 24, 25, 19] are just few examples.)

The best known PRGs for ordered ROBPs to date are those of Nisan [33] and Impagliazzo–Nisan–Wigderson [23] which give seed length  $O(\log^2 n)$  when  $w = n^{O(1)}$  and  $\varepsilon = 1/n^{O(1)}$ . However, even for width four and constant error their construction requiring seed length  $O(\log^2 n)$  is still the best. Improving on this seed length, even for constant width, has been a longstanding barrier. We do have better PRGs for various special classes of ROBPs that are independently interesting:

- Braverman, Rao, Raz, and Yehudayoff [7] construct PRGs with  $\tilde{O}(\log n)$  seed length for constant-width ordered *regular* branching programs and  $\varepsilon = 1/\text{poly}(\log n)$ . Regular branching programs are a special class of ROBPs where we require the structural condition that each vertex has the same in-degree in the underlying layered graph.
- Starting with the work of Koucký, Nimbhorkar, and Pudlák [26], several works [12, 37, 22] have achieved a seed length of  $O(\log n)$  (with no  $\log \log n$  factors) for the further restricted model of ordered *permutation branching programs*. In these, we require that at each layer  $j$ , and for each symbol  $\sigma$ , the transition function  $E_i^\sigma$  is a permutation of  $[w]$ .
- Meka, Reingold, and Tal [30] construct PRGs with  $\tilde{O}(\log n)$  seed length for width three ordered ROBPs and  $\varepsilon = 1/\text{poly}(\log n)$ , as well as for unordered ones with  $\varepsilon = 1/\text{poly}(\log \log n)$ .
- [16] gave a PRG that is significantly different from that of [33, 23] and achieves seed length  $O(\log^3 n)$  for polynomial width and  $\tilde{O}(\log^2 n)$  for constant-width ROBPs (again, even unordered).

## 1.4 Our Main Result

We give an explicit PRG with seed length  $\tilde{O}(\log(n/\varepsilon))$  for read-once MBPs.

► **Theorem 6** (see Section 3.2). *For any positive integers  $n$ ,  $w \leq n$ , and  $\varepsilon \in (0, 1/2)$ , there is an explicit PRG that  $\varepsilon$ -fools  $[n, w]$  read-once MBPs (even unordered ones) with seed length  $O(w^2 \log(n/\varepsilon) \cdot (\log \log(n/\varepsilon))^2)$ .*

We believe that fooling read-once MBPs is an important (and clearly necessary) step toward breaking the  $O(\log^2 n)$ -barrier for constant-width ROBPs. The class of (ordered) branching programs that we understand best from the perspective of pseudorandomness is that of *permutation branching programs*, thanks to the aforementioned works of [7, 26, 12, 37, 22], all of which obtain their results by showing that the Impagliazzo–Nisan–Wigderson [23] pseudorandom generator can be analyzed better for such programs. Monotone BPs can be seen as the extreme opposite of permutation BPs: the only monotone function  $E: [w] \rightarrow [w]$  that is also a permutation is the identity. Thus the only layers a monotone BP can share with a permutation BP are redundant (can be eliminated from the branching program without changing its functionality). Furthermore, in stark contrast to the case of ordered permutation branching programs, it is known that instantiations of the classical constructions of [33, 23] with  $\tilde{O}(\log n)$  seed length provably do not work against ordered MBPs of width 3 [8].

Technically, our arguments build on the paradigm of using random restrictions for fooling ROBPs as studied in the works of [18, 34, 38, 11, 20, 28, 10, 16, 30, 27, 13, 14]. This gives more evidence that this approach can perhaps lead to  $\tilde{O}(\log n)$  seed length for constant-width ROBPs. Our analysis introduces the idea of exploiting *width reduction* combined with *alphabet reduction* that could be useful for the general problem.

By Proposition 4, the PRG of Theorem 6 is also a PRG for read-once De Morgan formulas. For read-once  $\text{AC}^0$ , corresponding to width  $w = O(1)$ , it achieves a better dependence on the width  $w$  than the previous generator for read-once  $\text{AC}^0$ , which has a seed of length  $\log(n/\varepsilon) \cdot O(w \log \log(n/\varepsilon))^{2w+2}$  for depth- $w$  formulas [13] (our dependence on  $w$  is  $w^2$ ). For read-once formulas of arbitrary depth, the best PRG prior to our work was the PRG of Forbes and Kelley [16], which has seed length  $O(\log(n/\varepsilon) \log^2 n)$ . Thus, the PRG of Theorem 6 attains better seed length for read-once formulas that have depth up to  $w = o\left(\frac{\log n}{\log \log(n/\varepsilon)}\right)$ .

We note that constructing PRGs that are not sensitive to the ordering of the bits in which the input is read is a natural question. First, fooling read-once  $\text{AC}^0$ , for example, is inherently an unordered task. But also, PRGs for ROBPs that follows the “classical” and successful seed recycling approach due to Nisan (e.g., [33, 23, 26, 7, 6]) heavily depends on the ordering of the bits. In fact, Tzur [39] proved that Nisan’s PRG can in fact be distinguished from uniform by an unordered constant width branching program (see also [5]). Thus, the hope is that PRGs that are not sensitive to the ordering will help make progress on the problem of fooling ordered ROBPs with seed length  $o(\log^2 n)$ .

## 1.5 Techniques

We proceed by giving an overview of the construction of our PRG and of the techniques we use.

### 1.5.1 The Iterated Restrictions Approach

We construct our PRG using the *iterated pseudorandom restrictions* approach, pioneered by Ajtai and Wigderson [1] and further developed by Gopalan et al. [18]. That is, we pseudorandomly assign values to a pseudorandomly chosen subset of the variables, and then repeat the process until we assigned values to all variables. Intuitively, designing a pseudorandom restriction for some function  $f$  is easier than fooling  $f$  outright, because designing a pseudorandom restriction amounts to fooling a “smoothed out” version of  $f$  [18], or equivalently, designing a PRG that would fool  $f$  after some noise was added [20]. Previous works that used this approach include PRGs for unordered ROBPs [34, 10, 16], PRGs for width-3 ROBPs [18, 38, 30], PRGs for bounded-depth read-once formulas [18, 11, 13, 14], and PRGs for arbitrary-order product tests [20, 28, 27].

Following the iterated restrictions approach, we need our pseudorandom distribution  $X$  over restrictions to satisfy two key properties. The first property is that the restriction should approximately *preserve the expectation* of the function. i.e., in expectation over  $X$ , the restricted function  $f|_X$  should have approximately the same bias as  $f$  itself, i.e.  $\mathbb{E}_X[\mathbb{E}_U[f|_X(U)]] \approx \mathbb{E}_U[f(U)]$ , where  $U$  denotes the uniform distribution on the appropriate number of bits. This feature ensures that after sampling the restriction  $X$ , our remaining task is simply to fool  $f|_X$ . The second property is the *simplification* property. That is, we want that the restricted function, for a typical restriction, should be in a sense simpler than  $f$  itself. Clearly, simplifying  $f$  would make it easier to fool.

To achieve the first property of preserving the expectation, we follow Forbes and Kelley [16], who constructed a simple pseudorandom distribution over restrictions that approximately preserves the expectation of any constant-width ROBP. In the Forbes–Kelley distribution, we determine which coordinates stay alive in an almost  $k$ -wise independent manner, and sample the fixed coordinates using a small-bias space. This distribution, per a single restriction, can be sampled using  $\tilde{O}(\log(n/\varepsilon))$  uniform bits. Next, we proceed to discuss how to achieve the simplification property.

### 1.5.2 Iterative Width Reduction

In our setting, we design our restrictions in a way that fits nicely with the [16] distribution. Thus, the remaining challenge is indeed to ensure that such restrictions *simplify* constant width monotone ROBPs. In [16], the measure of complexity was simply the number of remaining unset variables. That is, Forbes and Kelley argued that after applying  $O(\log n)$  independent pseudorandom restrictions, with high probability, all variables are set, and hence there is nothing left to fool. Such an analysis gives seed length of  $\tilde{O}(\log(n/\varepsilon) \cdot \log n)$ , and recent works used more sophisticated measures of complexity to show that for more restricted classes of bounded width ROBPs, one can reach a function which is simple enough after only  $O(\log \log n)$  independent pseudorandom restrictions [18, 30, 13, 14]. In this work, we continue this line of research, and show that after  $O(\log \log n)$  iterations, roughly speaking, the width of the ROBP decreases by 1.

Since the construction and analysis of PRG will not depend on the ordering of the input bits, for simplicity we will describe it here assuming that the monotone ROBP  $B$  is ordered (to avoid the indexing  $i_j$  of input variables). Before getting to the construction, we highlight the key concept of *colliding layers* in a branching program, which was also paramount in [8, 36, 38, 10, 30]. We say that a BP layer  $i$  is a collision one, if there exist two edges with the same label  $\sigma$  that are mapped to the same vertex, i.e.  $E_i^\sigma$  is not a permutation. We say a collision is *realized* if a restriction fixes  $x_i$  to  $\sigma$ , and thus effectively introduces a layer with smaller width. The property of monotone BPs we use is that every non-identity layer is a collision one, and crucially, that this property is preserved under restrictions.

Another technical, yet powerful, component of our analysis is treating a branching program with edges labeled 0 and 1, i.e., over the alphabet  $\{0, 1\}$ , as a branching program over a much larger alphabet. Expressing the branching program over a larger alphabet preserves monotonicity and allows us to reduce the width of the BP in some cases. In fact, we will treat both the width and the alphabet size as progress measures.

Towards describing our iterative simplifying process, express our ROBP  $B$ , over  $\{0, 1\}$ , as a branching program over  $\Sigma = \{0, 1\}^\ell$  in the straightforward way, where  $\ell$  will start out as  $O(\log(n/\varepsilon))$  and eventually will be reduced to  $O(\log \log(n/\varepsilon))$ . Each “layer” is now a function from  $\Sigma \times [w]$  to  $[w]$ . Initially, moving to a larger alphabet only makes our task more difficult, but the generality will be useful as we induct on the width below (i.e. even if we start out with a width  $w$  program with alphabet  $\{0, 1\}$ , the argument below will force us to handle width  $w - 1$  programs having alphabet  $\{0, 1\}^{O(\log(n/\varepsilon))}$ ).

We iteratively apply the following two observations.

1. **Realizing a collision.** After a suitable pseudorandom restriction  $X^1$ , in every sequence of  $\exp(O(\ell)) \cdot \log(n/\varepsilon) = \exp(O(\ell))$  collision layers, we will have a collision in one of these layers. As each layer in a read-once MBP is either an identity layer or a collision layer, and this remains true also after transitioning to a larger alphabet, we can deduce that after  $X^1$  every  $C^\ell$  consecutive nontrivial layers contains a layer of width at most  $w - 1$ , for a sufficiently large constant  $C$ .
2. **Alphabet reduction.** After a suitable pseudorandom restriction  $X^2$ , up to a few “unruly” layers, we can shrink the alphabet size of  $B$  so that all layers are effectively over  $\{0, 1\}^{\ell/2}$ . Specifically, we can assume that in every sequence of  $C^\ell$  consecutive layers of alphabet size  $B$ , all but  $O(\log(n/\varepsilon))$  of them will have their alphabet size reduced to  $\{0, 1\}^{\ell/2}$ .

Both  $X^1$  and  $X^2$  will consist of almost  $k$ -wise independent distributions on  $\{0, 1, \star\}$  where  $\star$  represents the bits not assigned by the restriction and we take  $X^1$  to have  $\star$ -probability  $1/2$  and  $X^2$  to have a smaller, yet still constant,  $\star$ -probability.

Equipped with the above two observations, aiming at reducing the width of  $B$ , we apply, independently, the above  $X^1$  and  $X^2$  for  $t = O(\log \log(n/\varepsilon))$  iterations. After the first application of  $X^1$ , we can write  $B$  as  $B = B_1 \circ \dots \circ B_r$ , each  $B_i$  is of length at most  $C^\ell$  over an  $\ell$ -bit alphabet, starting and ending in a layer of width  $w - 1$ . Then the first application of  $X^2$  will reduce the alphabet of each of the  $B_i$ -s to consist of  $\ell/2$  bits, except for  $O(\log(n/\varepsilon))$  unruly layers within each  $B_i$ . The second application of  $X^1$  will now create collisions every  $C^{\ell/2}$  non-unruly layers, refining the program further into  $B = B'_1 \circ \dots \circ B'_{r'}$ , where each  $B'_i$  is of length at most  $C^{\ell/2}$  over an  $\ell/2$ -bit alphabet (except for  $O(\log(n/\varepsilon))$  unruly layers), starting and ending with a layer of width  $w - 1$ . The second of application of  $X^2$  will then reduce the alphabet size of each  $B'_i$  to at most  $\ell/4$  except for  $O(\log(n/\varepsilon))$  additional unruly layers within each  $B'_i$ . In general, each iteration *reduces the distance* between consecutive layers of width  $w - 1$  and *reduces the alphabet size*, except for increasing the number of unruly layers by  $O(\log(n/\varepsilon))$  within each interval. Finally after  $t = O(\log \log(n/\varepsilon))$  iterations, we will have an alphabet where each symbol consists of  $\ell^* = O(\log \log(n/\varepsilon))$  bits, so the distance between width  $w - 1$  layers is at most  $C^{\ell^*} = \text{poly}(\log(n/\varepsilon))$ . Even including the unruly layers, we can now view our as a width  $w - 1$  read-once MBP over  $\Sigma' = \{0, 1\}^{\text{poly}(\log(n/\varepsilon))}$ .

Before we can repeat the above process and reduce the width from  $w - 1$  to  $w - 2$ , etc., we need to reduce  $\Sigma'$  back to  $\Sigma = \{0, 1\}^{O(\log(n/\varepsilon))}$ . We can achieve this by an additional alphabet reduction using an almost  $k$ -wise independent distribution with  $\star$ -probability  $1/\text{poly} \log(n/\varepsilon)$  suffices.

Recall that due to [16], we can set the above restrictions to preserve the expectation of our original  $B$ , up to a small error. Hence, with seed of length  $\tilde{O}(\log(n/\varepsilon))$  we can both preserve the expectation and reduce the width by 1. Applying this  $w - 1$  times, with high probability our program will be very simple – a function depending on only  $O(\log(n/\varepsilon))$  bits (i.e. a junta), which is fooled by an almost  $O(\log(n/\varepsilon))$ -wise independent distribution.

All in all, our construction consists of commonly used primitives for PRGs: pseudorandom restrictions in which both the choice of live variables and the the choice of fixed coordinates are sampled from an almost  $k$ -wise independent distributions, with varying parameters. The analysis of iterative width reduction via resorting to larger alphabets is new, and we believe can be of use for designing PRGs for other models of computation. Naturally, there are some additional subtleties in the analysis and the choice of parameters, which we leave to the complete analysis in Section 3.

## 2 Preliminaries

We denote by  $U_n$  the uniform distribution over  $\{0, 1\}^n$ . Suppose  $\mathcal{C}$  is a class of functions in  $\{0, 1\}^n \rightarrow \mathbb{R}$  and  $G$  is a distribution over  $\{0, 1\}^n$ . We say that  $G$   $\varepsilon$ -fools  $\mathcal{C}$  if for every  $f \in \mathcal{C}$  it holds that  $|\mathbb{E}[f(G)] - \mathbb{E}[f(U_n)]| \leq \varepsilon$ . Recall that a PRG  $\varepsilon$ -fooling  $\mathcal{C}$  is a function  $G: \{0, 1\}^s \rightarrow \{0, 1\}^n$  such that  $G(U_s)$   $\varepsilon$ -fools  $\mathcal{C}$ . As a shorthand, we often write  $\mathbb{E}[f]$  to denote  $\mathbb{E}[f(U_n)]$ , and omit the subscript  $n$  when the number of input bits is clear from context.

### 2.1 Branching Programs

We extend the definition of branching programs from Definition 1 to large alphabets. We do so by grouping together at most  $\ell$  consecutive bits in a single edge-layer of the program. The main advantage in such a transformation is that we can potentially express a width  $w$  program over  $\{0, 1\}$  as a width  $w' < w$  program over  $\{0, 1\}^\ell$ . This will be crucial in our analysis.

We say that a *read-once branching program* (ROBP)  $B$  is a  $[n, w']_\ell$  ROBP if  $B$  can be written as a directed layered graph with  $m+1$  layers (for some  $m \leq n$ ) denoted  $V_1, \dots, V_{m+1}$ . Each  $V_i$  consists of at most  $w'$  many vertices. Furthermore, there exists a partition of  $[n]$  to disjoint subsets  $S_1, \dots, S_m \subseteq [n]$  of size at most  $\ell$  each, and between every consecutive layers of vertices  $V_i$  and  $V_{i+1}$  there exists a set of directed edges such that any vertex in  $V_i$  has  $2^{|S_i|}$  edges going towards  $V_{i+1}$ . We can treat the  $i$ -th layer of edges as a transition function  $E_i: \{0, 1\}^{S_i} \times [w'] \rightarrow [w']$  between  $V_i$  and  $V_{i+1}$ . Namely, for each  $\sigma \in \{0, 1\}^{S_i}$  we have the function  $E_i^\sigma \triangleq E_i(\sigma, \cdot): [w'] \rightarrow [w']$  that is defined in the natural way by following the edges labeled  $\sigma$  from  $V_i$  to  $V_{i+1}$ . Such a program naturally describes a read-once computation on  $x \in \{0, 1\}^n$ , where in the  $i$ -th step we follow the edge marked with  $x_{S_i} \in \{0, 1\}^{S_i}$  from a vertex in  $V_i$  to a vertex in  $V_{i+1}$ . We often denote  $\ell$  as the *alphabet length* of  $B$  and  $2^\ell$  as the *alphabet size* of  $B$ .

We say that an  $[n, w']_\ell$  ROBP is monotone if for every  $i \in [m]$ , its  $i$ -th layer  $E_i$  satisfies the following. For any  $\sigma \in \{0, 1\}^{S_i}$  and distinct  $x_1, x_2 \in [w']$ ,  $x_1 \geq x_2$  implies  $E_i^\sigma(x_1) \geq E_i^\sigma(x_2)$ . We say  $E_i$  is an *identity layer* if for any  $\sigma \in \{0, 1\}^{S_i}$  it holds that  $E_i^\sigma$  is the identity function. We say that  $E_i$  is a *collision layer* if there exists  $\sigma \in \{0, 1\}^{S_i}$  such that  $E_i^\sigma$  contains a collision, i.e., there exist distinct  $x_1, x_2 \in [w']$  such that  $E_i^\sigma(x_1) = E_i^\sigma(x_2)$ . We will make use of the following key observation.

▷ **Claim 7.** In a read-once MBP, every layer is either an identity layer or a collision layer.

As noted above, our techniques will also hold for the unordered setting, so we may assume that the bits of  $x$  are permuted by some permutation  $\pi \in S_n$ , i.e., the  $i$ -th layer of the program follow the edge marked by  $x_{\pi(i)}$ . Since we are in the unordered setting we can assume without loss of generality that there are no identity layers in the program, by skipping these layers.

Observe that if an (unordered)  $[n, w]$  ROBP  $B$  over  $\{0, 1\}$  has  $m+1$  of its  $n+1$  vertex-layers with width at most  $w'$  and of distance at most  $\ell$  apart, then we can write  $B$  as a  $[n, w']_\ell$  ROBP  $B'$ . Furthermore, if  $B$  is monotone so is  $B'$ .

## 2.2 $k$ -Wise and $\delta$ -Biased Distributions

We say that a random variable  $Y \sim \{0, 1\}^n$  is  $\delta$ -biased if it  $\delta$ -fools all parity functions. Namely, if for any nonempty  $I \subseteq [n]$  it holds that

$$\left| \Pr \left[ \bigoplus_{i \in I} Y_i = 1 \right] - \frac{1}{2} \right| \leq \delta.$$

There are explicit constructions of  $\delta$ -biased distributions over  $\{0, 1\}^n$  that can be sampled efficiently with  $O(\log n + \log \frac{1}{\delta})$  truly random bits [32].

► **Lemma 8** (Vazirani's XOR Lemma, See e.g., [17, Section 1]). *Let  $Y \sim \{0, 1\}^n$  be a  $\delta$ -biased distribution, and let  $S \subseteq [n]$ . Then,  $|Y_S - U_{|S|}| \leq 2^{|S|/2} \cdot \delta$ .*

For  $p \in [0, 1]$ , we denote by  $\text{Bernoulli}(p)^{\otimes n}$  the distribution over  $\{0, 1\}^n$  where the bits are i.i.d. and each bit has expectation  $p$ . We say that  $Z \sim \{0, 1\}^n$  is  $\gamma$ -almost  $k$ -wise independent with marginals  $p$  if for every set  $I \subseteq [n]$  satisfying  $|I| \leq k$  it holds that  $|Z_I - \text{Bernoulli}(p)^{\otimes |I|}| \leq \gamma$ . We can sample such distributions efficiently.

▷ **Claim 9** (see, e.g., in [14]). For any positive integers  $n, k, C$ , and any  $\gamma > 0$ , there is an explicit  $\gamma$ -almost  $k$ -wise independent distribution with marginals  $p = 2^{-C}$  that can be sampled efficiently with  $O(Ck + \log \frac{1}{\gamma} + \log \log n)$  truly random bits.



Moreover, we have good tail bounds for almost  $k$ -wise distribution.

► **Lemma 10** (following [9, 38]). *Let  $X_1, \dots, X_n$  be  $\gamma$ -almost  $k$ -wise independent random variables over  $\{0, 1\}$  with marginals  $q$ , and let  $\alpha > 0$ . Then, for an even  $k \leq qn$ ,*

$$\Pr \left[ \left| \sum_{i \in [n]} X_i - qn \right| \geq \alpha qn \right] \leq \left( \frac{16k}{\alpha^2 qn} \right)^{k/2} + 2k\gamma \left( \frac{1}{\alpha q} \right)^k.$$

► **Corollary 11.** *Let  $X'_1, \dots, X'_n$  be  $\gamma$ -almost  $k$ -wise independent random variables over  $\{0, 1\}$  with marginals  $\geq q$ . Then, for an even  $k \leq qn$ ,*

$$\Pr \left[ \sum_{i \in [n]} X'_i = 0 \right] \leq \left( \frac{16k}{qn} \right)^{k/2} + 2k\gamma \left( \frac{1}{q} \right)^k.$$

**Proof.** Take  $X_i = X'_i \wedge Y_i$  where  $Y_i$  is a coin toss with  $\Pr[Y_i = 1] = q/\mathbb{E}[X'_i]$ . We have that  $\mathbb{E}[X_i] = q$ , and that  $X_1, \dots, X_n$  are  $\gamma$ -almost  $k$ -wise independent with marginals  $q$ . Applying Lemma 10 with  $\alpha = 1$  implies that

$$\Pr \left[ \sum_{i \in [n]} X_i = 0 \right] \leq \left( \frac{16k}{qn} \right)^{k/2} + 2k\gamma \left( \frac{1}{q} \right)^k.$$

The proof is complete since  $\Pr \left[ \sum_{i \in [n]} X'_i = 0 \right] \leq \Pr \left[ \sum_{i \in [n]} X_i = 0 \right]$ . ◀

### 2.3 Restrictions and Pseudorandom Restrictions

A *restriction* is a string  $x \in \{0, 1, \star\}^n$ . Intuitively,  $x_i = \star$  means the  $i$ -th coordinate has not been set by the restriction. A restriction  $x$  can be specified by two strings  $y, z \in \{0, 1\}^n$  where  $z$  determines the  $\star$  locations and  $y$  determines the assigned values in the non- $\star$  locations. Namely, we define  $\text{Res}: \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1, \star\}^n$  by

$$\text{Res}(y, z)_i = \begin{cases} \star & z_i = 1, \\ y_i & z_i = 0. \end{cases}$$

We define a *composition* operation on restrictions, by

$$(x \circ x')_i = \begin{cases} x_i & x_i \neq \star, \\ x'_i & \text{otherwise.} \end{cases}$$

For a function  $f$  on  $\{0, 1\}^n$ , the restricted function  $f|_x$  on  $\{0, 1\}^n$  is defined by  $f|_x(x') = f(x \circ x')$ .

We will repeatedly use the following fact.

▷ **Claim 12.** Let  $B$  be a read-once MBP of length  $n$ , and let  $x \in \{0, 1, \star\}^n$  be any restriction. Then,  $B|_x$  is a read-once MBP.

Given a function  $f: \{0, 1\}^n \rightarrow \mathbb{R}$  and a distribution  $X \sim \{0, 1, \star\}^n$ , we say that  $X$  *preserves the expectation* of  $f$  with error  $\varepsilon$  if  $|\mathbb{E}[f|_X(U)] - \mathbb{E}[f]| \leq \varepsilon$ .

Forbes and Kelley showed that pseudorandom restrictions preserve the expectation of constant-width ROBPs. We give a “with high probability” version of their result, proved in [14].

► **Lemma 13** ([16], restated). *There exists a constant  $c \geq 1$  such that the following holds for any positive integers  $n, w$ , and  $\eta > 0$ . Let  $Z$  be a  $\gamma$ -almost  $k$ -wise independent distribution over  $\{0, 1\}^n$ , where  $k \geq c \log \frac{nw}{\eta}$  and  $\gamma \leq 2^{-k}$ . Let  $Y$  be a  $\delta$ -biased distribution over  $\{0, 1\}^n$ , where  $\log \frac{1}{\delta} \geq cwk \log \log n$ . Then, for any  $[n, w, \{0, 1\}]$  BP  $B$  it holds that with probability at least  $1 - \eta$  over  $z \sim Z$ ,*

$$\left| \mathbb{E}_{Y,U} [B|_{\text{Res}(Y,z)}(U)] - \mathbb{E}[B] \right| \leq \eta.$$

For  $X \sim \{0, 1, \star\}^n$  and a positive integer  $t$ , we denote by  $X^{\text{ot}}$  the distribution over  $\{0, 1, \star\}^n$  obtained by drawing independent samples  $x^{(1)}, \dots, x^{(t)} \sim X$  and composing them, namely  $x = x^{(1)} \circ \dots \circ x^{(t)}$ . We record two easy claims.

▷ **Claim 14.** Let  $\mathcal{F} \subseteq \{0, 1\}^n \rightarrow \mathbb{R}$  be some function class which is closed under restrictions. Then, if  $X$  preserves the expectation of every  $f \in \mathcal{F}$  with error  $\varepsilon$ , then  $X^{\text{ot}}$  preserves the expectation of every  $f \in \mathcal{F}$  with error  $t \cdot \varepsilon$ .

▷ **Claim 15.** Let  $X = \text{Res}(Y, Z)$  where  $Y \sim \{0, 1\}^n$  and  $Z$  is  $\gamma$ -almost  $k$ -wise independent with marginals  $p$ . Then, for any positive integer  $t$ , the distribution of the  $\star$  positions in  $X^{\text{ot}}$  is  $(t\gamma)$ -almost  $k$ -wise independent with marginals  $p^t$ .

Finally, we turn to define the notion of realizing a collision, in which a restriction “hits” a symbol in a collision layer that indeed causes a collision.

► **Definition 16** (realizing a collision). *Let  $B$  be an  $[n, w]_\ell$  ROBP and let  $E_i : \{0, 1\}^{S_i} \times [w] \rightarrow [w]$  be a collision layer in  $B$  for some  $i \in [n]$ . We say a string  $(y, z) \in \{0, 1\}^n \times \{0, 1\}^n$  realizes a collision in  $E_i$  if for any symbol  $\sigma \in \{0, 1\}^{S_i}$  consistent with the restriction  $\text{Res}(y, z)$  (i.e.,  $\sigma_j = y_j$  for all  $j \in S_i$  with  $z_j = 0$ ) we have that  $E_i^\sigma$  contains a collision (i.e.  $E_i^\sigma(v) = E_i^\sigma(v')$  for two distinct states  $v, v'$ ). We say  $(y, z)$  realizes a collision in  $B$  if it realizes a collision in some layer  $E_i$ .*

We will always use the special case where a collision is realized by  $z_{S_i} = 0^{|S_i|}$  and  $E_i^{y_{S_i}}$  having a collision.

### 3 PRGs for Constant-Width Read-Once MBPs

We set forth two auxiliary lemmas that will serve as the building blocks for our iterative argument.

The first claim states that in a read-once MBP with enough colliding layers from  $[w]$  to  $[w]$ , each depending on at most  $\ell$  bits, it is likely that one of the layers will realize a collision under a pseudorandom restriction. The second claim will help us implement *alphabet reduction* as outlined in the introduction.

► **Lemma 17** (realizing a collision). *Let  $\ell \in \mathbb{N}$  and  $m \geq 16^\ell$ . For  $i = 1, \dots, m$ , let  $E_i : \{0, 1\}^{S_i} \times [w] \rightarrow [w]$  where  $S_1, \dots, S_m \subseteq [n]$  are disjoint sets of size at most  $\ell$ . Suppose that each  $E_i$  is a collision layer. Let  $Y, Z \sim \{0, 1\}^n$  be  $\gamma$ -almost  $k$ -wise independent distributions, for  $\ell \leq k \leq 2^\ell/16$ . Then,*

$$\Pr_{Z,Y} [\exists i : (Y, Z) \text{ realizes a collision in } E_i] \geq 1 - 2^{-k/2} - \gamma \cdot 8^k.$$

**Proof.** For  $j \in [m]$  let  $\mathcal{E}_j$  be the event that  $z_{S_j} = 0^{|S_j|}$  and  $y_{S_j} = \sigma_j$ , where  $\sigma_j$  is an arbitrary choice of a string for which  $E_j^{\sigma_j}$  collides. Observe that when  $\mathcal{E}_j$  occurs,  $(Y, Z)$  realizes a collision in  $E_j$ . Thus, it suffices to lower bound the probability that some of the  $\mathcal{E}_j$  occurs.

The key observation is that the events  $\mathcal{E}_1, \dots, \mathcal{E}_m$  are  $2\gamma$ -almost  $k/\ell$ -wise independent with marginals  $\geq 4^{-\ell}$ . Indeed, for any test that depends on  $k/\ell$  of the events  $\mathcal{E}_1, \dots, \mathcal{E}_m$ , the test can be written as a function of  $k$  bits from  $Y$  and  $k$  bits from  $Z$ , and since any  $k$  bits from  $Y$  are  $\gamma$ -almost uniform and any  $k$  bits of  $Z$  are  $\gamma$ -almost uniform, we get that the test is fooled by the distribution with error at most  $2\gamma$ . Since on the uniform distribution  $z_{S_i} = 0^{|S_i|}$  and  $y_{S_i} = \sigma_i$  has probability  $4^{-|S_i|} \geq 4^{-\ell}$ , we get that  $\mathcal{E}_1, \dots, \mathcal{E}_m$  are  $2\gamma$ -almost  $k/\ell$ -wise independent with marginals  $\geq 4^{-\ell}$ .

By Corollary 11,

$$\begin{aligned} \Pr \left[ \sum_{j=1}^m 1_{\mathcal{E}_j} = 0 \right] &\leq \left( \frac{16k/\ell}{4^{-\ell} 16^\ell} \right)^{k/2\ell} + 4(k/\ell)\gamma \left( \frac{1}{4^{-\ell}} \right)^{k/\ell} \\ &\leq (2^\ell/4^\ell)^{k/2\ell} + \gamma \cdot 2^\ell \cdot (4^\ell)^{k/\ell} \leq 2^{-k/2} + \gamma \cdot 8^k. \end{aligned}$$

Thus, we get

$$\Pr_{Z,Y} [\exists i : (Y, Z) \text{ realizes a collision in } E_i] \geq \Pr \left[ \sum_{j=1}^m 1_{\mathcal{E}_j} > 0 \right] \geq 1 - 2^{-k/2} + \gamma \cdot 8^k. \quad \blacktriangleleft$$

► **Lemma 18** (alphabet reduction). *For every constant  $C > 1$  there exists a constant  $p \in (0, 1)$  such that the following holds. Let  $\ell \in \mathbb{N}$  and  $m \leq C^\ell$ . For  $i = 1, \dots, m$ , let  $E_i: \{0, 1\}^{S_i} \times [w] \rightarrow [w]$  where  $S_1, \dots, S_m \subseteq [n]$  are disjoint sets of size at most  $\ell$ . Let  $Z \sim \{0, 1\}^n$  be a  $\gamma$ -almost  $k$ -wise independent distribution with marginals  $p$ , for  $k \geq \ell$ . For  $j = 1, \dots, m$  let  $B_j$  be the indicator that  $Z_{S_j}$  has more than  $\ell/2$  ones. Then,*

$$\Pr \left[ \sum_{j=1}^m B_j \geq \frac{k}{\ell} \right] \leq C^k \cdot \gamma + 2^{-k}$$

**Proof.** Fix a set  $T \subseteq [m]$  of size  $t = \frac{k}{\ell}$ . For  $j \in T$ , let  $B_j(z)$  be the indicator random variable that is 1 if and only if  $z_{S_j}$  has more than  $\frac{\ell}{2}$  ones. We bound  $\Pr_Z[\forall j \in T : B_j(Z) = 1]$ . Note that this event depends only on  $k$  bits of  $Z$  and thus

$$\Pr_Z[\forall j \in T, B_j(Z) = 1] \leq \Pr_U[\forall j \in T : B_j(U) = 1] + \gamma.$$

To bound the probability of  $\forall j \in T, B_j(U)$  we note that each  $B_j$  happens with probability at most  $\binom{\ell}{\ell/2} p^{\ell/2} \leq 2^\ell p^{\ell/2}$  and that  $k/\ell$  of these events happen simultaneously with probability at most  $(2^\ell p^{\ell/2})^{k/\ell} = 2^k p^{k/2}$ .

Taking the union-bound over all subsets, we get the probability there exists  $T \subseteq [m]$  of size  $t$  for which  $B_j = 1$  for every  $j \in T$  is at most

$$\binom{C^\ell}{t} \left( \gamma + 2^k p^{k/2} \right) \leq C^{\ell t} \cdot \left( \gamma + 2^k p^{k/2} \right) = C^k \cdot \gamma + 2^{-k},$$

for  $p = \frac{1}{16C^2}$ . ◀

### 3.1 Width Reduction

► **Lemma 19.** *Let  $B$  be an  $[n, w]_\ell$  read-once MBP, and let  $\varepsilon > 0$ . Let  $k = \max(\ell, 4 \log(2n/\varepsilon))$  and  $\gamma = 32^{-k}$ . Set  $t = \log(\ell/\log(16k))$ . Also, for every  $j \in [t]$ ,*

■ *Let  $Y_1^j \sim \{0, 1\}^n$  and  $Z_1^j \sim \{0, 1\}^{\ell n}$  be  $\gamma$ -almost  $k$ -wise independent distribution;*

## 58:12 PRGs for Read-Once Monotone Branching Programs

- Let  $Y_2^j \sim \{0, 1\}^n$  be any distribution; and,
- Let  $Z_2^j \sim \{0, 1\}^n$  be a  $\gamma$ -almost  $k$ -wise independent distribution with marginal probability  $p$  as obtained from Lemma 18 for the constant  $C = 16$ .

For every  $j \in [t]$  we denote the  $j$ -th restriction as

$$X^j = X^{j,1} \circ X^{j,2} = \text{Res}(Y_1^j, Z_1^j) \circ \text{Res}(Y_2^j, Z_2^j),$$

and we set the pseudorandom restriction  $\bar{X} = X^1 \circ \dots \circ X^t$ .

Then, with probability at least  $1 - \varepsilon$  over  $\bar{x} \sim \bar{X}$ ,  $B|_{\bar{x}}$  can be written as an  $[n, w - 1]_{\ell'}$  read-once MBP for  $\ell' = O(k^9)$ .

**Proof.** Consider  $\ell_0 = \ell, \ell_1 = \ell/2, \dots, \ell_t = \ell/2^t$ . Note that for all  $i$  we have  $\ell_i \leq k \leq 2^{\ell_i}/16$ . Denote  $\Sigma^{(0)} = \Sigma$  and  $\ell_0 = \ell$ . Consider the pseudorandom restriction  $X^{1,1}$ , denoting  $A^1 = B|_{X^{1,1}}$ . By Lemma 17, followed by a union bound, we get that with probability at least

$$1 - n \cdot \left(2^{-k/2} + \gamma \cdot 8^k\right) \geq 1 - \varepsilon/2n,$$

every  $16^{\ell_0}$  consecutive layers of  $A^1$  contains a layer of vertices of width  $w - 1$ .<sup>4</sup> In the following, we condition on the event mentioned in the previous sentence. After the restriction we identify all layers of width  $w - 1$  and decompose the program to a concatenation of subprograms starting and ending with width at most  $w - 1$ . That is, we can write  $A^1$  as  $A_1^1 \circ \dots \circ A_r^1$ , where  $A_i^1$  has initial and final width at most  $w - 1$ , and length at most  $16^{\ell_0}$  over alphabet  $\Sigma^{(0)} = \{0, 1\}^{\ell_0}$ .

Next, consider the application of  $X^{1,2}$  on  $A^1 = A_1^1 \circ \dots \circ A_r^1$ . By Lemma 18 and a union bound, with probability at least

$$1 - n(16^k \gamma + 2^{-k}) \geq 1 - \varepsilon/2n,$$

we can reduce the alphabet in each  $A_i^1|_{X^{1,2}}$  to  $\Sigma^{(1)} = \{0, 1\}^{\ell_0/2}$ , except for  $k/\ell_0 \leq k$  “unruly” wide layers whose alphabet is a subset of  $\{0, 1\}^{\ell_0}$ . To sum up, after the first restriction, with probability at least  $1 - \varepsilon/n$ ,  $B^1 = B|_{X^1}$  can be written as a read-once MBP  $\tilde{B}_1^1 \circ \dots \tilde{B}_r^1$ , such that for every subprogram  $\tilde{B}_i^1$ : (i) starts and ends with width  $w - 1$  (ii) has at most  $16^{\ell_0}$  good layers with alphabet length  $\leq \ell_1$ , and (iii) has up to  $k$  unruly layers with alphabet length  $\leq \ell_0$ .

We show by induction on  $j$  that, with probability at least  $1 - \varepsilon j/n$ , after the  $j$ -th restriction  $B^j = B|_{X^1 \circ \dots \circ X^j}$  can be written as  $\tilde{B}_1^j \circ \dots \tilde{B}_{r_j}^j$ , such that for every subprogram  $\tilde{B}_i^j$ :

- Starts and ends with width  $w - 1$ ,
- Has at most  $16^{\ell_{j-1}}$  good layers with alphabet length  $\leq \ell_j$ , and,
- Has up to  $jk$  unruly layers with alphabet length  $\leq \ell_0$ .

Assume this to be the case for some  $j < t$ , we show how to prove it to be the case for  $j + 1$ . We denote by  $A^{j+1} = B^j|_{X^{j+1,1}}$ . By Lemma 17, with probability at least  $1 - n \cdot (2^{-k} + \gamma \cdot 8^k) \geq 1 - \varepsilon/2n$ , every  $16^{\ell_j}$  consecutive good layers of  $B^{j+1}$  realizes a collision in  $A^{j+1}$ . We write  $A^{j+1}$  as

$$A_1^{j+1} \circ \dots \circ A_{r_{j+1}}^{j+1},$$

<sup>4</sup> Observe that if  $16^{\ell_0} > n$  we may not apply Lemma 17. However, then the statement that “every  $16^{\ell_0}$  consecutive layers of  $A^1$  contains a layer of vertices of width  $w - 1$ ” is always true.

where each subprogram  $A_i^{j+1}$ : (i) starts and ends with width  $w - 1$ , (ii) has at most  $16^{\ell_j}$  good layers with alphabet length  $\leq \ell_j$ , and (iii) has up to  $jk$  unruly layers with alphabet length  $\leq \ell_0$ . To see Item (iii) note that the partition to subprograms is a refinement of the previous partition and thus cannot increase the maximal number of “unruly” layers in a subprogram.

Applying  $X^{j+1,2}$ , by Lemma 18, with probability at least  $1 - n(16^k\gamma + 2^{-k}) \geq 1 - \varepsilon/2n$ , in each subprogram  $A_i^{j+1}$  we can reduce the alphabet to  $\Sigma_{j+1} = \{0, 1\}^{\ell_{j+1}}$  except for at most the previous  $jk$  unruly layers and potentially  $k$  new unruly layers.

Overall, with probability at least  $1 - t\varepsilon/n \geq 1 - \varepsilon$ , the branching program  $B^t$  can be written as  $B^t = B_1^t \circ \dots \circ B_{r_t}^t$ , where  $B_i^t$  starts and ends with width  $w - 1$ , has at most  $16^{\ell_{t-1}}$  good layers and at most  $kt$  unruly layers. Thus, each  $B_i^t$  is a function of at most

$$16^{\ell_{t-1}} \cdot \ell_t + kt \cdot \ell_0 \leq k \cdot 16^{2(4+\log(k))} + k^3 = O(k^9)$$

bits. We can merge all bits participating in  $B_i^t$  to a single symbol in  $\Sigma' = \{0, 1\}^{\ell'}$  where  $\ell' = O(k^9)$ . We can thus write  $B^t$  as an  $[n, w - 1]_{\ell'}$  read-once MBP. ◀

As a second step, we reduce the alphabet size from  $\text{poly}(\log(n/\varepsilon))$  down to  $O(\log(n/\varepsilon))$ .

► **Lemma 20.** *Let  $\varepsilon > 0$ ,  $k = 4 \log(n/\varepsilon)$ ,  $\gamma = 1/(16\ell)^k$ . Let  $B$  be an  $[n, w]_{\ell}$  read-once MBP. Let  $Z$  be a  $\gamma$ -almost  $k$ -wise independent distribution over  $\{0, 1\}^n$  with marginals  $p_2 = 1/2\ell$ ; Let  $Y$  be any distribution over  $\{0, 1\}^n$ . Let  $X = \text{Res}(Y, Z)$ .*

*Then, with probability at least  $1 - \varepsilon$  over  $\bar{x} \sim \bar{X}$ ,  $B|_{\bar{x}}$  can be written as an  $[n, w]_k$  read-once MBP.*

**Proof.** Let  $z \sim Z$ . As in Lemma 18, for each layer  $j$  let  $B_j$  be the indicator random variable that is 1 if and only if  $z_j$  has more than  $k$  ones. By the union bound,

$$\Pr_Z [B_j = 1] \leq \binom{\ell}{k} \cdot (p_2^k + \gamma) \leq \ell^k p_2^k + \ell^k \cdot \gamma \leq 2 \cdot 2^{-k}.$$

Union bounding over all layers, the probability that we failed to reduce the alphabet size to  $2^k$  in any of the layers is at most  $1 - 2n2^{-k} \geq 1 - \varepsilon$ . ◀

### 3.2 Putting It Together

Our process will apply a sequence of  $w - 1$  restrictions sampled using Lemma 13, reducing the program width one at a time, with high probability, while preserving the acceptance probability.

Let  $c$  be a large enough constant. Set  $k = c \log(nw/\varepsilon)$  and  $t = \log(k)$ . Set  $\gamma = 1/(ck^9)^k$  and  $\delta = \min\{\gamma/2^k, 2^{-cwk \log \log n}\}$ . Set  $C = 16$ ,  $p_1 = \frac{1}{16C^2}$  and  $p_2 = \frac{1}{ck^9}$ .

For  $i \in [w - 2]$  and for  $j \in [t]$ :

- Let  $X^{i,j,1} = \text{Res}(Y^{i,j,1}, Z^{i,j,1})$  be a restriction from Lemma 13 with parameters  $k$ ,  $\gamma$  and  $\delta$  as above. We have that  $Y^{i,j,1}$  is a  $\delta$ -biased distribution, which is also a  $\gamma$ -almost  $k$ -wise independent distribution (due to Lemma 8). We have that  $Z^{i,j,1}$  is  $\gamma$ -almost  $k$ -wise independent (with marginals  $1/2$ ).
- Let  $X^{i,j,2} = \text{Res}(Y^{i,j,2}, Z^{i,j,2})$  be a composition of  $\log(1/p_1) = O(1)$  restrictions from Lemma 13 with parameters  $k$ ,  $\gamma$  and  $\delta$  as above. We have that  $Y^{i,j,2}$  is a  $\delta$ -biased distribution, which is also a  $\gamma$ -almost  $k$ -wise independent distribution. By Claim 15 we have that  $Z^{i,j,2}$  is  $\log(1/p_1)\gamma$ -almost  $k$ -wise independent with marginals  $p_1$ .

## 58:14 PRGs for Read-Once Monotone Branching Programs

- Let  $\tilde{X}^i = \text{Res}(\tilde{Y}^i, \tilde{Z}^i)$  be a composition of  $\log(1/p_2) = O(\log \log(nw/\varepsilon))$  restrictions from Lemma 13 with parameters  $k, \gamma$  and  $\delta$  as above. By Claim 15 we have that  $\tilde{Z}^i$  is  $\log(1/p_2)\gamma$ -almost  $k$ -wise independent with marginals  $p_2$ .

We define  $X^{i,j} = (X^{i,j,1} \circ X^{i,j,2})$  and  $X^i = (X^{i,1} \circ X^{i,2} \circ \dots \circ X^{i,t}) \circ \tilde{X}^i$ . And finally,  $X = X^1 \circ X^2 \circ \dots \circ X^{w-1}$ . Let  $S \sim \{0, 1\}^n$  be a  $\varepsilon$ -almost  $k$ -wise independent distribution. Our PRG  $G$  is given by

$$G = X \circ S.$$

Let  $s = s(n, w, \varepsilon)$  be the seed length required to sample from  $G$ . Following the seed lengths of the above primitives in Section 2, we can give the following bound.

▷ Claim 21. It holds that  $s = O(w^2 \log(n/\varepsilon) \cdot (\log \log(n/\varepsilon))^2)$ .

▷ Claim 22.  $G$  fools width- $w$  read-once MBPs of length  $n$  with error at most  $4\varepsilon n$ .

Proof. Let  $B$  be an  $[n, w]_1$  read-once MBP, which can also be written as an  $[n, w]_k$  read-once MBP by grouping every  $k$ -consecutive layers. Note that this transformation preserves monotonicity. Since our restriction is picked as a  $m = O(t \cdot w + w \log k) \leq n$  compositions of restrictions that each maintain the acceptance probability of the ROBP up to error  $\varepsilon$  (Lemma 13), we see that

$$\left| \mathbb{E}_{X,U}[B|_X(U)] - \mathbb{E}_U[B(U)] \right| \leq \varepsilon \cdot n.$$

It remains to show that  $\mathbb{E}_{X,U}[B|_X(U)] \approx \mathbb{E}_{X,S}[B|_X(S)]$ . For that we show that with high probability  $B|_X$  can be expressed as a  $[n, 1]_k$  read-once MBP. Let  $E = E(\bar{X})$  be the union of the following bad events:

- There exists an  $i \in [w-1]$  such that  $(X^{i,1} \circ X^{i,2} \circ \dots \circ X^{i,t})$  fails to reduce the width, in the sense of Lemma 19.
- There exists an  $i \in [w-1]$  such that  $\tilde{X}^i$  fails to reduce the alphabet size from  $O(k^9)$  to  $k$ , in the sense of Lemma 20.

By Lemmas 19 and 20,  $\Pr[E] \leq 2w\varepsilon$ . Note that in the case that  $E$  does not occur, we have that  $B|_X$  is a  $[n, 1]_k$  ROBP or in other words that it is a junta that depends on at most  $k$  bits. In such a case,  $B|_X$  will be  $\varepsilon$ -fooled by  $S$ . Overall we have

$$\begin{aligned} \left| \mathbb{E}_{X,U}[B|_X(U)] - \mathbb{E}_{X,S}[B|_X(S)] \right| &\leq \Pr[E] + \Pr[\bar{E}] \cdot \left| \mathbb{E}_X \left[ \mathbb{E}_U[B|_X(U)] - \mathbb{E}_S[B|_X(S)] \mid \bar{E} \right] \right| \\ &\leq 2w\varepsilon + \varepsilon. \end{aligned}$$

Combining both estimates we see that

$$\left| \mathbb{E}_G[B(G)] - \mathbb{E}_U[B(U)] \right| \leq \varepsilon \cdot (n + 2w + 1) \leq 4\varepsilon n. \quad \triangleleft$$

► **Theorem 23.** *Let  $n \in \mathbb{N}$ ,  $\varepsilon' > 0$  and  $w \leq n$ . There exists a generator  $G$  that fools width- $w$  read-once MBPs of length  $n$ , with error at most  $\varepsilon'$  and seed-length  $O(w^2 \cdot \log(n/\varepsilon') \cdot (\log \log(n/\varepsilon'))^2)$ .*

**Proof.** Apply Claim 22 and Claim 21 with  $\varepsilon = \varepsilon'/4n$ . ◀

## 4 Relation to Read-Once $AC^0$

In this section we study the relation between constant-width read-once MBPs and read-once  $AC^0$ :

► **Proposition 4.**

1. If a sequence of functions  $f_n: \{0,1\}^n \rightarrow \{0,1\}$  is in read-once  $AC^0$ , then it can be computed by constant-width read-once MBP. Moreover, if  $f_n$  can be computed in depth  $w$  read-once  $AC^0$ , then it can be computed by width  $w + 1$  read-once MBPs.
2. For every  $n \geq 3$ , there exists a function  $f: \{0,1\}^n \rightarrow \{0,1\}$  computable by a width 3 read-once MBP, but not computable by any read-once De Morgan formula (regardless of depth).

First, we establish Item 1 of Proposition 4 by observing that the known implication, that read-once  $AC^0$  formulas can be computed by constant-width ROBPs, yields a monotone ROBP.

► **Lemma 24.** *Let  $f: \{0,1\}^n \rightarrow \{0,1\}$  be a function computable by a read-once, depth- $w$   $AC^0$  formula. Then,  $f$  can also be computed by an  $[n, w + 1]$  read-once MBP.*

**Proof.** We prove the claim by induction on the depth  $w$ , and further prove that the ‘accept’ states in our read-once MBP are above the ‘reject’ states (that is, if  $s$  is an accept state and  $s'$  is a reject state then  $s \geq s'$ ). For  $w = 1$ ,  $f$  computes either the disjunction or the conjunction of at most  $s$  literals. This can clearly be done by an  $[n, s, 2]$  read-once MBP, if we set state 2 to be an accept state (and so state 1 is a reject one).

Next, fix some  $f$  computable by a formula  $F: \{0,1\}^n \rightarrow \{0,1\}$  of depth  $w > 1$  and size  $s$ , and assume that its top gate is an AND gate (the other case is similar). We denote the subformulas feeding into the top gate as  $F_1, \dots, F_m$ , and these are on disjoint variables because the formula is read-once. By the induction’s hypothesis, each subformula  $F_i$  is computable by a width  $w$  read-once MBP over its variables with the accept states being on top.

To construct  $B$  that computes  $F$ , we can concatenate the  $B_i$ -s and add another “sudden reject” level at level  $s = 1$ .<sup>5</sup> The starting vertex of  $B$  is the starting vertex of  $B_1$ . Whenever a computation of some  $B_i$ , for some  $i < m$ , reaches its final layer, we rewire the edges in that layer to either the sudden reject level, if  $B_i$  did not reach an accepting vertex, or to the starting vertex of  $B_{i+1}$ . The accept vertices of  $B$  are the accept vertices of  $B_m$ . Note that this transformation preserves the ordering between accepts and reject states, since  $B_m$  does.

The fact that  $B$  computes  $f$  readily follows, and  $B$  is read-once because the  $B_i$ -s are on disjoint variables. To argue that monotonicity is preserved, simply observe that the rewiring preserves the order: In the AND case, accept vertices are rewired to the next starting vertex, which is indeed above the sudden reject level, to which all reject vertices are rewired. The OR case is similar. ◀

We now prove Item 2 of Proposition 4, giving a family of functions computable by read-once MBPs but not by read-once formulas. Our proof also gives a new characterization of read-once formulas.

<sup>5</sup> In a sudden reject level, each vertex transitions to the same level with both its edges, and the last vertex in that level is a reject vertex. When the top gate is an OR gate, we would replace the sudden reject level with a sudden accept level at  $s = w + 1$ , and make the last vertex of the sudden accept level an accepting vertex.

► **Lemma 25.** *For every  $n \geq 3$ , there exists a function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  computable by a width 3 read-once MBP, but not computable by any read-once De Morgan formula (regardless of depth).*

**Proof.** We first give a property of functions computable by read-once formulas. Given  $g : \{0, 1\}^m \rightarrow \{0, 1\}$  and  $b \in \{0, 1\}$ , let  $W_b(g) \in \{0, \dots, m\}$  denote the size of the smallest set of coordinates  $I \subseteq [m]$  for which there exists a  $z \in \{0, 1\}^{|I|}$  such that for every  $x \in \{0, 1\}^m$  it holds that  $x_I = z$  implies  $g(x) = b$ .

► **Lemma 26.** *Let  $g : \{0, 1\}^n \rightarrow \{0, 1\}$  be a function computable by a read-once De Morgan formula. Then,  $W_0(g) \cdot W_1(g) \leq n$ .*

Roughly speaking, this lemma says that for a function computable by a read-once formula, we can either find a short witness for it being 0, or a short witness for it being 1. In particular, it cannot be highly resilient.

**Proof.** We prove the lemma by induction on the formula's depth  $d$ . For  $d = 1$ ,  $g$  is either an AND of literals or an OR of literals. For the AND function,  $W_0(\text{AND}) = 1$  and  $W_1(\text{AND}) = n$ . For the OR function,  $W_0(\text{OR}) = n$  and  $W_1(\text{OR}) = 1$ . Thus, indeed,  $W_0(g) \cdot W_1(g) \leq n$ .

Assume our lemma holds for formulas of depth  $d \geq 1$ , and let  $g$  be some formula of depth  $d + 1$ , say with an AND top gate, so  $g = \text{AND}(f_1, \dots, f_k)$ , each  $f_i : \{0, 1\}^{n_i} \rightarrow \{0, 1\}$  is a depth- $d$  formula. In this case,  $W_0(g) = \min_{j \in [k]} W_0(f_j)$  and  $W_1(g) = \sum_{i \in [k]} W_1(f_i)$ . By our induction's hypothesis, we get that

$$W_0(g) \cdot W_1(g) = \left( \min_{j \in [k]} W_0(f_j) \right) \cdot \sum_{i \in [k]} W_1(f_i) \leq \sum_{i \in [k]} W_0(f_i) \cdot W_1(f_i) \leq \sum_{i \in [k]} n_i = n.$$

The case of an OR top gate is analogous. ◀

Now, our function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  will simply be the  $\text{Thr}_2^n$  function, that returns 1 if and only if the Hamming weight of the input string  $x \in \{0, 1\}^n$  is at least 2. There,  $W_1(f) = 2$  and  $W_0(f) = n - 1$ , so it is not computable by read-once formulas, however  $f$  is computable by a simple width-3 read-once MBP. ◀

We note that we can also construct balanced functions  $f$  separating read-once MBPs from read-once De Morgan formulas. In particular,  $f = \text{AND}_m \circ \text{Thr}_2^w$  for  $m = O(2^w/w)$  (which resembles the Tribes function) has this property. More generally, one can consider, say,  $\text{Thr}_2$ , as a “gadget” to construct richer families of read-once MBPs not computable by read-once formulas.

---

## References

- 1 Miklos Ajtai and Avi Wigderson. Deterministic simulation of probabilistic constant depth circuits. *Advances in Computing Research*, 5(199-222):1, 1989.
- 2 David A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in  $\text{NC}^1$ . *Journal of Computer and System Sciences*, 38(1):150–164, 1989.
- 3 David A. Barrington, Chi-Jen Lu, Peter Bro Miltersen, and Sven Skyum. Searching constant width mazes captures the  $\text{AC}^0$  hierarchy. In *Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science (STACS 1998)*, pages 73–83. Springer, 1998.
- 4 David A. Barrington, Chi-Jen Lu, Peter Bro Miltersen, and Sven Skyum. On monotone planar circuits. In *Proceedings of the 14th Annual IEEE Conference on Computational Complexity (CCC 1999)*, pages 24–31. IEEE, 1999.



- 5 Andrej Bogdanov, Periklis A Papakonstantinou, and Andrew Wan. Pseudorandomness for read-once formulas. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2011)*, pages 240–246. IEEE, 2011.
- 6 Mark Braverman, Gil Cohen, and Sumegha Garg. Pseudorandom pseudo-distributions with near-optimal error for read-once branching programs. *SIAM Journal on Computing*, 49(5):STOC18–242–STOC18–299, 2020. doi:10.1137/18M1197734.
- 7 Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom generators for regular branching programs. *SIAM Journal on Computing*, 43(3):973–986, 2014.
- 8 J. Brody and E. Verbin. The coin problem and pseudorandomness for branching programs. In *Proceedings of the 51st IEEE Annual Symposium on Foundations of Computer Science (FOCS 2012)*, pages 30–39, October 2010. doi:10.1109/FOCS.2010.10.
- 9 L. Elisa Celis, Omer Reingold, Gil Segev, and Udi Wieder. Balls and bins: Smaller hash families and faster evaluation. *SIAM Journal on Computing*, 42(3):1030–1050, 2013.
- 10 Eshan Chattopadhyay, Pooya Hatami, Omer Reingold, and Avishay Tal. Improved pseudorandomness for unordered branching programs through local monotonicity. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC 2018)*, pages 363–375. ACM, 2018.
- 11 Sitan Chen, Thomas Steinke, and Salil Vadhan. Pseudorandomness for read-once, constant-depth circuits. *arXiv preprint arXiv:1504.04675*, 2015.
- 12 Anindya De. Pseudorandomness for permutation and regular branching programs. In *Proceedings of the 26th Annual IEEE 26th Annual Conference on Computational Complexity (CCC 2011)*, pages 221–231. IEEE, 2011.
- 13 Dean Doron, Pooya Hatami, and William M. Hoza. Near-optimal pseudorandom generators for constant-depth read-once formulas. In *Proceedings of the 34th Computational Complexity Conference (CCC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2019.
- 14 Dean Doron, Pooya Hatami, and William M. Hoza. Log-seed pseudorandom generators via iterated restrictions. In *Proceedings of the 35th Computational Complexity Conference (CCC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
- 15 Dean Doron, Raghu Meka, Omer Reingold, Avishay Tal, and Salil Vadhan. Monotone branching programs: Pseudorandomness and circuit complexity. In *Electronic Colloquium on Computational Complexity (ECCC)*, number TR21-018, February 2021. Version 1.
- 16 Michael A. Forbes and Zander Kelley. Pseudorandom generators for read-once branching programs, in any order. In *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2018)*. IEEE, 2018.
- 17 Oded Goldreich. Three XOR-lemmas – an exposition. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 248–272. Springer, 2011.
- 18 Parikshit Gopalan, Raghu Meka, Omer Reingold, Luca Trevisan, and Salil Vadhan. Better pseudorandom generators from milder pseudorandom restrictions. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2012)*, pages 120–129. IEEE, 2012.
- 19 Iftach Haitner, Danny Harnik, and Omer Reingold. On the power of the randomized iterate. *SIAM Journal on Computing*, 40(6):1486–1528, 2011.
- 20 Elad Haramaty, Chin Ho Lee, and Emanuele Viola. Bounded independence plus noise fools products. *SIAM Journal on Computing*, 47(2):493–523, 2018. doi:10.1137/17M1129088.
- 21 Alexander Healy, Salil Vadhan, and Emanuele Viola. Using nondeterminism to amplify hardness. *SIAM Journal on Computing*, 35(4):903–931, 2006.
- 22 William M. Hoza, Edward Pyne, and Salil P. Vadhan. Pseudorandom generators for unbounded-width permutation branching programs. 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 7:1–7:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ITCS.2021.7.

- 23 Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC 1994)*, pages 356–364. ACM, 1994.
- 24 Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM*, 53(3):307–323, 2006.
- 25 Adam R. Klivans, Homin Lee, and Andrew Wan. Mansour’s conjecture is true for random DNF formulas. In *Proceedings of the 23rd Annual Conference on Learning Theory (COLT 2010)*, 2010.
- 26 Michal Koucký, Prajakta Nimbhorkar, and Pavel Pudlák. Pseudorandom generators for group products. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC 2011)*, pages 263–272. ACM, New York, 2011. doi:10.1145/1993636.1993672.
- 27 Chin Ho Lee. Fourier bounds and pseudorandom generators for product tests. In *Proceedings of the 34th Computational Complexity Conference (CCC 2019)*, pages 7:1–7:25. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019.
- 28 Chin Ho Lee and Emanuele Viola. More on bounded independence plus noise: Pseudorandom generators for read-once polynomials. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 24, page 167, 2017.
- 29 Michael Luby, Boban Velickovic, and Avi Wigderson. Deterministic approximate counting of depth-2 circuits. In *Proceedings of the 2nd Annual Israel Symposium on Theory and Computing Systems (ISTCS 1993)*, pages 18–24. IEEE, 1993.
- 30 Raghu Meka, Omer Reingold, and Avishay Tal. Pseudorandom generators for width-3 branching programs. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC 2019)*, pages 626–637. ACM, New York, 2019.
- 31 Raghu Meka and David Zuckerman. Pseudorandom generators for polynomial threshold functions. *SIAM Journal on Computing*, 42(3):1275–1301, 2013.
- 32 Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, 1993.
- 33 Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.
- 34 Omer Reingold, Thomas Steinke, and Salil Vadhan. Pseudorandomness for regular branching programs via Fourier analysis. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 655–670. Springer, 2013.
- 35 D. Sivakumar. Algorithmic derandomization via complexity theory. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC 2002)*, pages 619–626. ACM, 2002.
- 36 John Steinberger. The distinguishability of product distributions by read-once branching programs. In *Proceedings of the 28th IEEE Conference on Computational Complexity (CCC 2013)*, pages 248–254. IEEE, 2013.
- 37 Thomas Steinke. Pseudorandomness for permutation branching programs without the group theory. Technical Report TR12-083, Electronic Colloquium on Computational Complexity (ECCC), July 2012. URL: <http://eccc.hpi-web.de/report/2012/083/>.
- 38 Thomas Steinke, Salil Vadhan, and Andrew Wan. Pseudorandomness and Fourier-growth bounds for width-3 branching programs. *Theory of Computing*, 13(1):1–50, 2017.
- 39 Yoav Tzur. Notions of weak pseudorandomness and  $\text{GF}(2^n)$ -polynomials. *Master’s thesis, Weizmann Institute of Science*, 2009.

## **A** Monotone Branching Programs and $\text{AC}^0$ Circuits

In this section we give a self-contained proof of the equivalence between constant width MBPs and  $\text{AC}^0$  circuits, proving Theorem 3:

► **Theorem 3** (corollary of [4]). *A sequence of functions  $f_n : \{0, 1\}^n \rightarrow \{0, 1\}$  is in  $\text{AC}^0$  if and only if it is computable by a constant-width MBP of polynomial length.*

First we note that the “if” direction follows from Lemma 24.

► **Lemma 27.** *Let  $f: \{0,1\}^n \rightarrow \{0,1\}$  be a function computable by a (read-many)  $AC^0$  formula of depth  $w$  and size  $s$ . Then,  $f$  can also be computed by an  $[n, s, w + 1]$  MBP.*

(Here take the *size* of an  $AC^0$  formula to be the number of leaves.)

**Proof.** Let  $F(x_1, \dots, x_n)$  be depth- $w$   $AC^0$  formula of size  $s$ . Then by putting distinct variables on the leaves of  $F$  we obtain a read-once  $AC^0$  formula  $G(y_1, \dots, y_s)$  on  $s$  variables such that  $F(x_1, \dots, x_n) = G(\sigma_1 x_{i_1}, \dots, \sigma_s x_{i_s})$  where  $i_1, \dots, i_s \in [n]$  and  $\sigma_1, \dots, \sigma_s \in \{\pm 1\}$  (referring to whether or not the variable is negated at each leaf). By Lemma 24, there is a read-once MBP  $B(y_1, \dots, y_s)$  of width  $w + 1$  computing  $G$ . Then,  $B(\sigma_1 x_{i_1}, \dots, \sigma_s x_{i_s})$  is a (read-many) MBP of width  $w + 1$  and length  $s$  computing  $F$ . ◀

The result naturally extends to  $AC^0$  circuits, due to the standard transformation expressing a size  $s$  depth  $w$   $AC^0$  circuit as a size  $s^w$  depth  $w$   $AC^0$  formula.

► **Corollary 28.** *Let  $f: \{0,1\}^n \rightarrow \{0,1\}$  be a function computable by a depth- $w$   $AC^0$  circuit of size  $s$ . Then,  $f$  can also be computed by an  $[n, s^w, w + 1]$  MBP.*

Next, we give the other direction of Theorem 3. Similarly to the other direction, we start by showing that *read-once* MBPs can be simulated by (read-many)  $AC^0$ :

► **Lemma 29.** *Let  $f: \{0,1\}^n \rightarrow \{0,1\}$  be a function computable by a read-once MBP of width  $w$ . Then,  $f$  can also be computed by a circuit of depth  $O(w)$  and size  $O(w^4 n^3)$ .*

**Proof.** We prove this lemma by induction on the width. For  $w = 1$  the claim is trivial. Fix some  $w > 1$  and let  $B$  be an  $[n, w]$  read-once MBP. We define two BPs,  $B^u$  and  $B^\ell$ , each of width  $w - 1$ , as follows.

- For  $B^u$ , we remove the first level of vertices (that is, removing state number 1 in each layer) and reroute edges that go into state 1 to state 2. Formally, each transition  $U_i^b: \{2, \dots, w\} \rightarrow \{2, \dots, w\}$  of  $B^u$  is defined by  $U_i^b(x) = \max\{E_i^b(x), 2\}$ , for  $E_i^b: [w] \rightarrow [w]$  being the corresponding transition of  $B$ .
- Similarly, for  $B^\ell$ , we remove the last level of vertices: Each transition  $L_i^b: [w - 1] \rightarrow [w - 1]$  of  $B^\ell$  is defined by  $L_i^b(x) = \min\{E_i^b(x), w - 1\}$ .

Notice that these transformations preserve monotonicity. Roughly speaking, our goal is to first argue that at each transition,  $B$  acts the same as either  $B^u$  or  $B^\ell$ , depending on whether  $B$  last reached the state 1 or the state  $w$ . Then, we show that we can efficiently detect, given any layer  $j$  and an input  $x$ , if indeed  $B(x)$  passed through the state 1 or through the state  $w$  before reaching the layer  $j$ .

Let  $s_0$  be the starting vertex of  $B$ , and denote  $u_0 = \max\{s_0, 2\}$  and  $\ell_0 = \min\{s_0, w - 1\}$ . Given some input  $x \in \{0,1\}^n$ , we consider the computation path of all three BPs on  $x$ . Towards this end, denote by  $s_1, \dots, s_n \in [w]$  the states that  $x$  traverses in  $B$ ,  $u_1, \dots, u_n \in \{2, \dots, w\}$  the states that  $x$  traverses in  $B^u$  and  $\ell_1, \dots, \ell_n \in [w - 1]$  the states that  $x$  traverses in  $B^\ell$ . First, observe that:

▷ **Claim 30.** For every  $i \in [n]$ ,  $u_i \geq s_i \geq \ell_i$ .

The above claim readily follows by induction on  $i$ , using the monotonicity property. Next, we argue:

▷ **Claim 31.** For every  $i \in [n]$ , let  $j \leq i$  be the largest integer such that  $s_j \in \{1, w\}$ , if it exists. Thus, if  $s_j = w$  then  $u_i = s_i$  and if  $s_j = 1$  then  $\ell_i = s_i$ .

Proof. Fix some  $i \in [n]$  and assume that  $j \leq i$  is the largest integer such that  $s_j \in \{1, w\}$ , say  $s_j = 1$ . By Claim 30, we must also have  $\ell_j = 1$ . Then by induction, we also have  $\ell_{j'} = s_{j'}$  for all  $j' = j, j+1, \dots, i$ , because the only way in which the transition in  $B^\ell$  and  $B$  can differ is if  $s_{j'} = w$ , which by assumption does not occur in this interval.  $\triangleleft$

Hence, for each layer  $i$ , we know that either  $s_i = u_i$  or  $s_i = \ell_i$ , and we know which is the case by looking at the last place the original path reached either 1 or  $w$ .

By our induction's hypothesis, for every  $i \in [n]$  and  $s \in \{2, \dots, w\}$  there exists a circuit  $C_{i,s}^u: \{0, 1\}^n \rightarrow \{0, 1\}$  such that  $C_{i,s}^u(x) = 1$  if and only if  $B^u$  reached the state  $s$  after reading  $x_1, \dots, x_i$ . Similarly, there exists a circuit  $C_{i,s}^\ell$  that detects whether  $B^\ell$  reached  $s \in [w-1]$  in the  $i$ -th layer upon traversing with  $x$ . Using these circuits, for each  $s \in [w]$ , we will construct a circuit  $C_s(x)$  that determined whether  $s_n = s$ .

The construction goes as follows. The circuit will determine the last  $j$  where there was a “switch” between the two cases of Claim 31, i.e., the smallest  $j \in [n]$  such that  $s_j \in \{1, w\}$  and for every  $k \geq j$  it holds that  $s_k \in \{2, \dots, w-1\} \cup \{s_j\}$ . Observe that if  $s_j = 1$  then  $s_{j-1} = u_{j-1}$ , so  $E_j^{x_j}(u_{j-1}) = 1$ . Afterward, we keep following  $B^\ell$ , i.e.,  $s_k = \ell_k$  and so  $E_{k+1}^{x_{k+1}}(\ell_k) \neq w$  for all  $k \geq j$ . The converse also holds. Namely,  $E_j^{x_j}(u_{j-1}) = 1$  implies that  $s_j = \ell_j = 1$  (since  $\ell_{j-1} \leq u_{j-1}$  and the program is monotone) and  $E_{k+1}^{x_{k+1}}(\ell_k) \neq w$  for all  $k \geq j$  implies that indeed  $s_{k+1} = \ell_{k+1}$ . Thus, the predicate

$$P_L(x) = \left( \bigvee_{j \in [n]} \left( (E_j^{x_j}(u_{j-1}) = 1) \wedge \bigwedge_{k \geq j} (E_{k+1}^{x_{k+1}}(\ell_k) \neq w) \right) \right) \vee \left( u_1 \neq w \wedge \bigwedge_{k \geq 1} E_k^{x_k}(\ell_k) \neq w \right)$$

evaluates to 1 if and only if the largest integer  $j \leq n$  such that  $s_j \in \{1, w\}$  has  $s_j = 1$ , or  $s_j$  never equals  $w$  (and hence  $s_n = \ell_n$ ). Following the same reasoning,

$$P_U(x) = \left( \bigvee_{j \in [n]} \left( (E_j^{x_j}(\ell_{j-1}) = w) \wedge \bigwedge_{k \geq j} (E_{k+1}^{x_{k+1}}(u_k) \neq 1) \right) \right) \vee \left( \ell_1 \neq 1 \wedge \bigwedge_{k \geq 1} E_k^{x_k}(u_k) \neq 1 \right)$$

evaluates to 1 if and only if the largest integer  $j \leq n$  such that  $s_j \in \{1, w\}$  has  $s_j = w$ , or  $s_j$  never equals 1 (and hence  $s_n = u_n$ ).

We now wish to compute  $P_L: \{0, 1\}^n \rightarrow \{0, 1\}$  by a shallow circuit. Determining  $u_{j-1}$  can be done by querying  $C_{j-1,s}^u(x)$  for each  $s \in \{2, \dots, w\}$ . Similarly, determining  $\ell_k$  can be done by querying  $C_{k,s}^\ell(x)$  for each  $s \in [w-1]$ . The functions  $E_j^b$  and  $E_{k+1}^b$ , for each  $b \in \{0, 1\}$ , are determined solely by  $B$  and can be hardwired. Letting  $\text{size}(w-1)$  and  $\text{depth}(w-1)$  be the size and depth upper bound for the circuits guaranteed to us by the hypothesis, we can bound  $\text{size}(P_L)$  by  $2nw \cdot \text{size}(w-1) + O(wn^2)$  and  $\text{depth}(P_L)$  by  $\text{depth}(w-1) + O(1)$ . The same bounds for  $P_U: \{0, 1\}^n \rightarrow \{0, 1\}$  also hold.

Equipped with circuits  $C_L$  and  $C_U$  computing  $P_L$  and  $P_U$  respectively, we are ready to compute  $B$ . Indeed, all that is left is to determine whether  $s_n = \ell_n$  or  $s_n = u_n$  and invoke the relevant circuit from the previous level. This incurs additional constant depth and  $O(wn)$  size. Overall, the size and depth of  $C$  satisfies the recurrence relations  $\text{size}(w) = O(nw) \cdot \text{size}(w-1) + O(wn^2)$  and  $\text{depth}(w) = \text{depth}(w-1) + O(1)$ . As  $\text{size}(1) = O(n)$  and  $\text{depth}(1) = O(1)$ , this gives us depth  $O(w)$  and size  $w^{O(w)} \cdot n^w$ .

We can improve the size of the circuit by a dynamic programming approach. For  $1 \leq a \leq b \leq w$ , let  $B^{[a,b]}$  be the ROBP in which we keep only the levels  $a, \dots, b$  and rewire edges accordingly. Namely, we replace each  $E_i^\sigma(x)$  with  $\max\{a, \min\{b, E_i^\sigma(x)\}\}$ . Observe that for when  $a < b$ ,  $(B^{[a,b]})^\ell = B^{[a,b-1]}$  and  $(B^{[a,b]})^u = B^{[a+1,b]}$ .

For every  $1 \leq a \leq b \leq w$  and  $s \in \{a, \dots, b\}$ , let  $X_i^{a,b,s}$  be the indicator which is 1 if and only if upon reading the first  $i$  bits of  $x$ , the program  $B^{[a,b]}$  reached the state  $s$ . Note that there are at most  $w^3 \cdot n$  such indicators overall.

Fix some integer  $\Delta \in \{0, \dots, w-1\}$ . We can compute the values

$$\mathcal{I}_\Delta = \left\{ X_i^{a,a+\Delta,s} : a \in [w-\Delta], s \in [a, a+\Delta], i \in [n] \right\}$$

in the following manner. For  $\Delta = 0$ , all indicators are true. For  $\Delta \geq 1$ , assume we already computed the values

$$\mathcal{I}_{\Delta-1} = \left\{ X_i^{a,a+\Delta-1,s} : a \in [w-(\Delta-1)], s \in [a, a+\Delta-1], i \in [n] \right\}.$$

Thus, to compute a single indicator from  $\mathcal{I}_\Delta$  given  $\mathcal{I}_{\Delta-1}$ , we can use the above recurrence relations, as each  $\ell_i$  and  $u_i$  correspond to some indicator from  $\mathcal{I}_{\Delta-1}$ . This takes  $O(wn^2)$  size and  $O(1)$  depth. Computing the entire  $\mathcal{I}_\Delta$  thus takes  $O(w^3n^3)$  size and  $O(1)$  depth. Overall, computing  $\blacktriangleleft$

Similarly to the proof of Lemma 27, we can handle the read-many case by noting that a read-many MBP of length  $s$  can be obtained from a read-once MBP on  $s$  variables by a variable substitution. This gives us the “only if” direction of Theorem 3:

► **Corollary 32.** *Let  $f: \{0, 1\}^n \rightarrow \{0, 1\}$  be a function computable by an  $[n, s, w]$  MBP for  $s \geq n$ . Then,  $f$  can also be computed by a circuit of depth  $O(w)$  and size  $O(w^4s^3)$ .*



# On the Power of Choice for $k$ -Colorability of Random Graphs

Varsha Dani ✉🏠

Ronin Institute, Montclair, NJ, USA

Dept. of Computer Science, Rochester Institute of Technology, NY, USA

Diksha Gupta ✉

School of Computing, National University of Singapore, Singapore

Thomas P. Hayes ✉🏠

Dept. of Computer Science, University of New Mexico, Albuquerque, NM, USA

---

## Abstract

In an  $r$ -choice Achlioptas process, random edges are generated  $r$  at a time, and an online strategy is used to select one of them for inclusion in a graph. We investigate the problem of whether such a selection strategy can shift the  $k$ -colorability transition; that is, the number of edges at which the graph goes from being  $k$ -colorable to non- $k$ -colorable.

We show that, for  $k \geq 9$ , two choices suffice to delay the  $k$ -colorability threshold, and that for every  $k \geq 2$ , six choices suffice.

**2012 ACM Subject Classification** Mathematics of computing → Stochastic processes; Theory of computation → Generating random combinatorial structures

**Keywords and phrases** Random graphs, Achlioptas Processes, Phase Transition, Graph Colorability

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.59

**Category** RANDOM

**Acknowledgements** The authors would like to thank Will Perkins for suggesting the problem of shifting the threshold for  $k$ -coloring, and Will Perkins and Cris Moore for helpful conversations.

## 1 Introduction

In studying the evolution of a random graph, a field launched by the seminal paper of Erdős and Rényi [7], one starts from an empty graph, and adds edges one by one, generating each one independently and uniformly at random. In this context, a common object of study is the size of the graph at which some property of interest changes. For instance, if we are interested in  $k$ -colorability, there will eventually be some edge whose addition changes the graph from being  $k$ -colorable to non- $k$ -colorable.

The  $k$ -colorability transition threshold conjecture states that there is a particular threshold  $d(k)$  such that, almost surely, the  $k$ -colorability transition occurs when  $G$  has average degree approximately  $d(k)$ ; more precisely, when the average degree lies between  $(1 - \varepsilon)d(k)$  and  $(1 + \varepsilon)d(k)$ , for any fixed  $\varepsilon > 0$ . Substantial progress has been made on pinning down this transition threshold, especially by Achlioptas and Naor [2] and by Coja-Oghlan and Vilenchik [5], culminating in a rather precise formula for the asymptotics of  $d(k)$  for large  $k$ . However, for fixed  $k \geq 3$ , the conjecture remains open.

An interesting twist on the evolution of the random graph was proposed by Achlioptas in 2001: Suppose that *two* random edges are sampled at each step in the construction of  $G$ , and an online algorithm selects one of them, which is then added to  $G$ . A more general version of this process proposes  $r$  random edges in each step, from which the algorithm selects one. After  $m$  edges have been chosen in this way, how different can the resulting graph be from the usual Erdős-Rényi random graph  $G(n, m)$ ?



© Varsha Dani, Diksha Gupta, and Thomas P. Hayes;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 59; pp. 59:1–59:17



Leibniz International Proceedings in Informatics

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

Earlier work on the “power of choice” to affect the outcome of random processes has investigated questions like load-balancing in balls and bins models, scheduling, routing and more; for more, see the excellent survey by Richa, Mitzenmacher and Sitaraman [10]. More specifically, Achlioptas processes have been studied in the context of formation of the giant component in a random graph [3, 4, 13, 11], and the satisfiability threshold for random boolean formulas [12, 6, 9]. In each of these cases, the upshot has been that fairly simple heuristics are capable of shifting the thresholds to a significant extent. However, the heuristics and their analyses remain fairly problem-specific.

The main contribution of the present work is a proof that, for every  $k \geq 2$ , there exist fairly simple choice strategies that significantly delay the  $k$ -coloring threshold, given a constant number of choices for each edge. Our proof leverages existing upper and lower bounds on the  $k$ -colorability threshold, and works even if the  $k$ -Colorability Threshold Conjecture turns out to be false. More precisely, we establish the following result.

► **Theorem 1.** *For every  $k \geq 2$ , there exist  $2 \leq r \leq 6$ , an explicit edge selection strategy for the  $r$ -choice Achlioptas process, and a real number  $d$  such that, if  $G$  is the graph produced by running our strategy for  $dn/2$  steps, and  $H$  is an Erdős-Rényi random graph with the same number of edges, then  $G$  is almost surely  $k$ -colorable and  $H$  is almost surely not  $k$ -colorable. In particular,  $r = 2$  choices suffice for  $k = 2$  and  $k \geq 9$ ,  $r = 3$  suffices for all  $k \neq 3$ , and  $r = 6$  suffices for all  $k$ .*

If, rather than delaying, one wants to *hasten* the  $k$ -colorability threshold, this can be done very easily by “densifying” the graph, an idea used in [6] to hasten the  $k$ -SAT threshold for random boolean formulas. Unlike our main result, this technique easily extends to any monotone graph property that has a sharp threshold in the Erdős-Rényi model. More precisely,

► **Observation 2.** *Let  $P$  be any graph property that is monotone in the sense that  $P(G)$  implies  $P(G')$  for every subgraph  $G'$  of  $G$ . Then, if the threshold conjecture is true for  $P$ , we can lower the threshold using  $r$  choices, whenever  $r \geq 2$ . Moreover, even without the threshold conjecture for  $P$ , if there exist real numbers  $0 < \alpha_1 < \alpha_2$  such that  $P$  almost surely holds for  $G(n, \alpha_1 n)$ , and almost surely fails to hold for  $G(n, \alpha_2 n)$ , then there exists  $r = r(\alpha_1, \alpha_2)$  and  $d = d(\alpha_1, \alpha_2)$ , and an explicit edge selection strategy for the  $r$ -choice Achlioptas process, such that, if  $G$  is the graph produced by running our strategy for  $dn/2$  steps, and  $H$  is an Erdős-Rényi random graph with the same number of edges, then  $H$  almost surely has property  $P$ , and  $G$  almost surely does not. In the case when  $P$  is  $k$ -colorability,  $r = 2$  choices suffices to lower the  $k$ -coloring threshold when  $k = 2$  or  $k \geq 12$ ,  $r = 3$  suffices when  $k \geq 6$ ,  $r = 4$  suffices when  $k \neq 4$ , and  $r = 5$  suffices for all  $k$ .*

The interested reader may refer to Appendix A for additional details.

## 1.1 Strategy for Delaying the $k$ -Colorability Transition

Our basic strategy for delaying the  $k$ -colorability transition is to try to create a large bipartite subgraph. This can be achieved very simply by, *ab initio*, partitioning the vertex set into two equal parts, and then by choosing, whenever possible, a *crossing edge*, that is, one whose endpoints lie in both sides of the partition. As we shall see, this extremely simple heuristic suffices to establish Theorem 1 when  $k \geq 6$ , and with slight modifications, for  $k \leq 5$  as well.

For intuition about why this approach works, think about what happens in the limit as  $r$  becomes very large. Since the probability of being offered all non-crossing edges in a particular step is less than  $2^{-r}$ , by choosing crossing edges whenever possible, our graph



becomes “more and more bipartite” as  $r$  increases. Indeed, when  $r = 3 \log n$ ,  $G$  will almost surely become a complete bipartite graph before it is forced to include any non-crossing edges! Obviously, this is a huge delay to any of the  $k$ -colorability thresholds, which all take place after linearly many edges.

For more intuition, consider the case when  $k$  is very large, but  $r \geq 2$  is constant. We expect about a  $2^{-r}$  fraction of the edges to be non-crossing, and hence the average degree of the graph induced by one side of  $G$  will be about  $2^{-r}$  times the average degree of  $G$ . Since, asymptotically for large  $k$ , we know that the  $k$ -colorability occurs somewhere around  $d \approx 2k \ln(k)$ , (See Theorem 3 for a more precise statement.) which is a nearly-linear function, this tells us that each side of  $G$  should need almost  $2^r$  times fewer colors than  $G(n, m)$ . Hence, if we color the two sides with disjoint sets of colors, so that the crossing edges cannot cause any monochromatic edges, we would expect to need almost  $2^{r-1}$  times fewer colors to color our graph than a random graph with the same average degree.

The above approach works as stated for  $k \geq 6$ . For smaller values of  $k$ , it is necessary to improve the above strategy by adding an additional “filtering” step that checks to see whether the edge proposed by the basic strategy would create an obstacle to  $k$ -coloring; in this case, we make a different edge choice. This is the most technical part of the paper, particularly the case  $k = 3$ , for which the filtering algorithm is fairly complicated.

For  $k = 4$  and  $k = 5$ , since we are splitting the colors among the two sides of  $G$ , at least one side gets only two colors. This is a bit of a special case because, unlike with more colors, two-coloring does not have a sharp phase transition at a particular average degree. Instead, the transition for  $G(n, d/n)$  is spread out over the range  $0 < d < 1$ . However, as we shall see, for Achlioptas processes, it is possible to delay this threshold until the emergence of a giant component at  $d = 1$  (and even beyond!).

For  $k = 3$ , we need a further modification to our plan as outlined above. With only three colors, one side of the graph would only get one color, and would need to remain empty of edges! Since this is clearly impossible, we modify our plan of prescribing disjoint sets of colors for the two sides of the graph. Instead, we allow one of the three colors to be used on both sides. As will be seen, this complicates both the edge selection process and its analysis, and increases the number of choices we need, to  $r = 6$ .

We point out an interesting qualitative difference between the problem of delaying the  $k$ -coloring threshold and that of delaying the  $k$ -SAT threshold. Earlier work on delaying the  $k$ -SAT threshold, in particular by Perkins [9] and by Dani et al. [6], took advantage of the fact that, with enough choices, the 2-SAT threshold can be shifted past the  $k$ -SAT threshold. The analogous statement for  $k$ -coloring would require us to keep our graph bipartite past the formation of a giant component. Although Bohman and Frieze [3] showed that it is possible to delay the formation of a giant component, it obviously cannot be delayed past  $d = 2$ , and indeed, as shown by Bohman, Frieze and Wormald [4, Theorem 1(d)], not past  $d = 1.93$ . After a linear-size giant component has formed, each step of our Achlioptas process has a constant probability that all  $r$  offered edges will fall within the giant component, and moreover all violate bipartiteness. Thus, there is no hope of keeping the graph 2-colorable past the 3-colorability threshold, for any constant (or indeed sub-logarithmic) number of choices. This “fragility” of the property of 2-colorability may provide some intuition for the increased difficulty of our attempts to shift the  $k$ -colorability threshold for small values of  $k$ .

## 1.2 Organization of the Paper

The remainder of the paper is divided into numbered sections. For the most part, each section introduces one or two new ideas that are needed for a particular range of the number of colors,  $k$ . Many of the sections depend on concepts introduced in earlier sections, so it is easiest to read them in order.

In Section 2 (Preliminaries), we introduce various notation and terminology, as well as stating the key results from past work that we will need for our work. In Section 3 we formally state the PreferCrossing strategy, and show how it can be used directly to raise the  $k$ -colorability threshold for  $k \geq 6$ . In Section 4, we handle the case  $k = 2$  by showing that odd cycles (indeed all cycles) can be delayed until a giant component forms, and that this idea can be combined with previous work on delaying the birth of the giant component. In Section 5, we handle the cases  $k = 4$  and  $k = 5$ . These are treated separately from the large  $k$  cases because now one of the two sides will be colored using only two colors, which requires the cycle-avoidance technique developed in Section 4. In Section 6, we handle the hardest case:  $k = 3$ , which involves a significant extension to the technique for avoiding cycles introduced in Section 4. In Section 7, we show how an improved bound on the 3-coloring transition threshold, due to Achlioptas and Moore [1], can be used to reduce the number of choices we need for  $k = 9$  from 3 to 2.

Finally, Appendix A presents a proof of Observation 2, about hastening the transition for (almost) any monotone graph property.

## 2 Preliminaries

Let  $V$  be a fixed vertex set, of size  $n$ . In the rest of the paper, unless otherwise specified, whenever we use asymptotic notations such as big-O and little-O, these refer to limits as  $n \rightarrow \infty$ , while all the other key parameters, namely, average degree  $\bar{d}$ , number of choices,  $r$ , and number of colors,  $k$ , are held constant. When we state that something happens “almost surely,” we mean that the corresponding event has probability  $1 - o(1)$ .

When we talk about the Erdős-Rényi random graph,  $G(n, m)$ , we assume that  $m$  independent random edges are sampled from  $\binom{V}{2}$ , with replacement. Edges are undirected and self-loops are not allowed.

In an  $r$ -choice Achlioptas process, at each step,  $r$  independent random edges are sampled from  $\binom{V}{2}$ , with replacement. An online algorithm, which we call a “strategy” is used to select one of these edges for inclusion in the edge set of the graph, which is initially empty. We allow duplicate edges both in the set of proposed edges, as well as the graph itself. However, observe that, when the total number of edges is linear in  $n$ , and  $r = O(1)$ , the expected number of duplicate edges seen during the entire process is  $O(1)$ . Consequently, in this range of parameters, it should be easy to see that very similar results hold even when duplicate edges are not allowed.

### Key Results from Prior Work

The following result is due to Achlioptas and Naor [2, See Lemma 3 and Proposition 4]

► **Theorem 3** (Achlioptas and Naor). *Suppose  $k$  is a positive integer, and  $d < 2(k-1) \ln(k-1)$ . Then, almost surely,  $G(n, dn/2)$  is  $k$ -colorable. If, instead,  $d > (2k-1) \ln(k)$ , then, almost surely,  $G(n, dn/2)$  is not  $k$ -colorable.*

For notational convenience, we introduce a shorthand for the upper and lower bounds on the transition threshold from Theorem 3.

► **Definition 4.** *For  $k$  a positive integer, denote*

$$L_k = 2(k-1) \ln(k-1) \quad \text{and} \quad U_k = (2k-1) \ln(k).$$

Subsequent work by Coja-Oghlan and Vilenchik [5] established an asymptotically sharper bound, pinning down the chromatic number for a set of degrees having asymptotic density one. However, their bounds are only stated asymptotically in  $k$ , and do not lead to improved bounds for fixed values of  $k$ .

For the case  $k = 3$ , Achlioptas and Moore [1] proved a tighter lower bound on the 3-colorability threshold by analysing the success probability of a naive 3-coloring algorithm using the differential equations method.

► **Theorem 5** (Achlioptas and Moore). *Almost all graphs with average degree 4.03 are 3-colorable.*

Although Theorem 3 is sharp enough to derive most of our bounds, we will need Theorem 5 in order to shift the transition threshold for  $k = 7$  using  $r = 3$  choices, and  $k = 9$  using only  $r = 2$  choices. We note that future improvements to the bounds on the  $k$ -coloring transition thresholds for  $G(n, m)$  might produce further improvements to our bounds.

For the cases whose analysis involve 2-coloring, we will make use of past work on accelerating or delaying the formation of the giant component. We start with a classical result of Erdős and Rényi:

► **Theorem 6.** *When  $d < 1$ , almost surely, all connected components of  $G(n, dn/2)$  have size  $O(\log n)$ , but when  $d > 1$ , almost surely,  $G(n, dn/2)$  has a “giant” component of size  $\Theta(n)$ .*

Bohman and Frieze [3] showed that, in an Achlioptas process, it is possible to delay this threshold, inspiring many related papers. The following result is due to Spencer and Wormald [13].

► **Theorem 7.** *There exists an edge selection strategy for the 2-choice Achlioptas process, in which, almost surely, the largest component size is still  $O(\log n)$  after the inclusion of  $dn/2$  edges, where  $d = 1.6587$ .*

The details of Spencer and Wormald’s elegant algorithm will not be important in the present work. In Section 4 we will show how to modify their strategy to additionally delay  $G$ ’s first cycle until the giant component forms, but these modifications treat the original strategy as a black box. We note that, in the same paper, Spencer and Wormald presented another strategy for hastening the arrival of giant component, causing it to appear at average degree  $d = 0.6671$ .

### 3 Main Idea, Many Colors

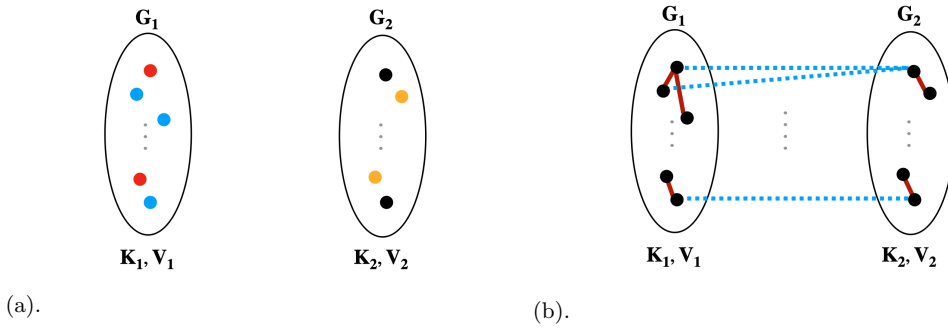
Our general approach to delaying the  $k$ -colorability threshold is to partition both the vertex and color sets into two parts, and then to assign a disjoint set of colors to each side of the graph. The intuition for this was already discussed in Section 1.1. We now formalize some of these ideas.

Let  $V$  be the set of vertices and  $K$  the set of colors. Then  $|V| = n$  and  $|K| = k$ . We will partition  $V$  into disjoint subsets  $V_1$  and  $V_2$ , called “sides,” each of size  $n/2$ . (Since we are interested in the asymptotic behaviour in  $n$  we do not need to worry about its parity.)

We also partition  $K$  into disjoint sets  $K_1$  and  $K_2$ . When we color the graph, we will use colors in  $K_i$  to color side  $V_i$ . Most of the time we will partition the set of colors so that  $|K_1| = \lfloor k/2 \rfloor$  and  $|K_2| = \lceil k/2 \rceil$ , although we will have some occasions to deviate from this.

We will use an Achlioptas process to build a graph  $G$  with  $m$  edges on  $V$ .  $G_1$  and  $G_2$  will denote the subgraphs of  $G$  induced by  $V_1$  and  $V_2$ . By abuse of notation, we will also refer to the graphs obtained partway through the Achlioptas process as  $G$ ,  $G_1$  and  $G_2$ .

Based on the partition  $V = V_1 \sqcup V_2$ , we classify the possible edges into two types:



■ **Figure 1** Illustration for: (a) Disjoint color sets  $K_1$  and  $K_2$ , and vertex sets  $V_1$  and  $V_2$  assigned to  $G_1$  and  $G_2$ , respectively; and (b) Types of edges using solid lines for non-crossing edges and dashed lines for crossing edges.

- **a crossing edge:** An (undirected) edge  $\{u, v\}$  with  $u \in V_1$  and  $v \in V_2$ .
- **a non-crossing edge on side  $i$ , or an edge in  $G_i$  :** An (undirected) edge  $\{u, v\}$  where both  $u, v \in V_i$ .

Note that since we are using disjoint sets of colors for  $V_1$  and  $V_2$ , a crossing edge is never violated by a coloring.

Each edge offered to us in the Achlioptas process is sampled uniformly at random from all  $\binom{n}{2}$  pairs of vertices. The probability of a single offered edge being a crossing edge is

$$\frac{\binom{n/2}{2}}{\binom{n}{2}} = \frac{1}{2} + \frac{1}{2(n-1)} = 1/2 + o(1) \approx 1/2$$

while for  $i = 1, 2$ , the probability of a single offered edge being a non-crossing edge on side  $i$  is

$$\frac{\binom{n/2}{2}}{\binom{n}{2}} = \frac{1}{4} - \frac{1}{4(n-1)} = 1/4 - o(1) \approx 1/4$$

Let  $r$  denote the number of edges offered to the algorithm at each step of the Achlioptas process. As a reminder, each edge is sampled independently and uniformly from  $\binom{V}{2}$ . We use the following strategy to select an edge at every step, unless stated otherwise:

► **Strategy 1. PreferCrossing**  
*Select the first crossing edge, if any. Otherwise, select the first edge.*

Note that in the event that no crossing is available, the selected edge is equally likely to be on either side, and is a uniformly random edge conditioned on being on the side it is.

Let  $m$  be the total number of edges inserted into  $G$ , so the average degree of  $G$  is  $\bar{d} = 2m/n$ . For  $i \in \{1, 2\}$ , let  $\bar{d}_i$  denote the average degree of the graph  $G_i$ .

We use the PreferCrossing strategy to choose the edge to be inserted into  $G$  at each step. A non-crossing edge is inserted only if all  $r$  candidate edges are non-crossing, so the probability of inserting a non-crossing edge is at most  $1/2^r$ . Also, in this case we insert the first edge, which is equally likely to be on either side. So the probability of inserting an edge into  $G_i$  is  $1/2^{r+1}$ .

It follows that in expectation, there are  $m/2^{r+1}$  edges in each  $G_i$  and the rest are crossing edges. Using this we can calculate the expected average degrees on the two sides as follows:

$$\mathbb{E}[\bar{d}_i] < \frac{2m/2^{r+1}}{n/2} = \left(\frac{2m}{n}\right) \frac{1}{2^r} = \frac{\bar{d}}{2^r}$$

By the Law of Large Numbers, it follows that, almost surely,

$$\bar{d}_i < (1 + o(1)) \frac{\bar{d}}{2^r}. \tag{1}$$

Now, since whichever of the three classes of edge (crossing, non-crossing on side 1, non-crossing on side 2) the PreferCrossing strategy selects, the edge is uniformly random within that class, it follows that, conditioned on  $\bar{d}_1$  and  $\bar{d}_2$ ,  $G_1$  and  $G_2$  are uniformly random graphs with that number of edges. Therefore, assuming each  $\bar{d}_i$  is below a known lower bound on the  $k_i$ -colorability transition, it will follow that each  $G_i$  is almost surely  $k_i$ -colorable, and hence  $G$  is  $(k_1 + k_2)$ -colorable. If, additionally,  $\bar{d}$  is greater than a known upper bound on the  $k$ -colorability threshold, and  $k = k_1 + k_2$ , we will have shifted the  $k$ -colorability transition threshold.

Theorem 3 tells us that for  $\kappa \geq 3$ , the  $\kappa$ -colorability transition threshold (if it exists) lies between  $L_\kappa$  and  $U_\kappa$  (see Definition 4.) Additionally, we will sometimes also use the improved lower bound  $L'_3 = 4.03$  from Theorem 5

Since the expression for  $L_\kappa$  is monotone, the graphs  $G_i$  are  $k_i$ -colorable (and hence  $G$  is  $k$ -colorable) until  $\bar{d}_1 = \bar{d}_2 = \min\{L_{k_1}, L_{k_2}\}$ . It therefore makes sense to split the colors as evenly as possible. We will set  $k_1 = \lfloor k/2 \rfloor$ ,  $k_2 = \lceil k/2 \rceil$ . Then  $G_1$  and  $G_2$  are  $k_1$ - and  $k_2$ -colorable respectively until  $\bar{d}_1 = \bar{d}_2 = L_{\lfloor k/2 \rfloor}$

Now, we know from Eq. (1) that

$$\bar{d} \geq 2^r \bar{d}_1 \geq 2^r L_{\lfloor k/2 \rfloor}$$

and we will have delayed the  $k$ -colorability transition if this exceeds  $U_k$

Since  $L_k$  and  $U_k$  are both asymptotically equal to  $2k \ln k$ , this shows that for sufficiently large  $k$  two choices suffice to raise the  $k$ -colorability threshold. Indeed, using Mathematica to solve the inequalities

$$2^r L_{\lfloor k/2 \rfloor} \geq U_k$$

for  $r = 2, 3$  and 4, we see that

- two choices suffice for even  $k \geq 10$  and odd  $k \geq 13$
- three choice suffice for even  $k \geq 6$  and odd  $k \geq 9$  and
- four choices suffice for  $k = 7$ .

Moreover, if we use the improved lower bound  $L'_3 = 4.03$ , instead of  $L_3$  for the case of  $k = 7$ , then we see that

$$8L'_3 = 8 \times 4.03 = 32.24 > 25.3 = U_7$$

so that three choices suffice  $k = 7$ . This establishes Theorem 1 for  $k \geq 6$ , except for the cases  $k = 9$  and  $k = 11$ .

For  $k = 9, 11$  we have established that three choices suffice, but we want to show that in fact we only need two. We will tackle the case  $k = 11$  here and leave  $k = 9$  for Section 7.

When  $k = 11$ , we allocate five colors to side 1, and six colors to side 2. The five-colorability of  $G_1$  is only guaranteed until  $\bar{d}_1 = 8 \ln 4 \approx 11.09$ . With  $r = 2$  choices, at this point  $\bar{d}$  is about 44.36, smaller than  $20 \ln 10 = 46.05$ , so that although  $G$  is 11-colorable, so is

## 59:8 Power of Choice for $k$ -Colorability

$G(n, m = 44.36n/2)$ , so we have not shifted the threshold. In order to increase  $\bar{d}$  past  $U_{11} = 21 \ln 11 = 50.356$ , we note that  $\bar{d}_2$  is *also* about 11.09, since we are equally likely to add a non-crossing edge to side 2 as to side 1. But  $\bar{d}_2$  is allowed to go to  $10 \ln 5 \approx 16.09$  before we can no longer guarantee the 6-colorability of  $G_2$ . This means we have a fair bit of slack to favor  $G_2$  when adding non-crossing edges. Suppose we put a  $\varphi < 1/2$  fraction of the non-crossing edges into  $G_1$  and a  $(1 - \varphi)$  fraction of them into  $G_2$ . What should  $\varphi$  be to ensure the best outcome? Note that we need  $\varphi \geq 2^{-r}$ , since if all the non-crossing choices are on side 1, then we cannot add an edge in side 2. However, subject to this constraint, we are adding  $m\varphi/2^r$  edges to  $G_1$  and  $m(1 - \varphi)/2^r$  edges to  $G_2$  in expectation. But this means that  $\mathbb{E}[\bar{d}_1] = \bar{d}\varphi/2^{r-1}$  and  $\mathbb{E}[\bar{d}_2] = \bar{d}(1 - \varphi)/2^{r-1}$ . Since these random variables stay close to their expectations, it follows that  $\bar{d}_1$  and  $\bar{d}_2$  are in the ratio  $\varphi/(1 - \varphi)$ . Now, it is best if we can arrange it so that both  $G_1$  and  $G_2$  lose their guarantee of colorability at the same time (so that there is no slack). But this means

$$\frac{L_5}{L_6} = \frac{\varphi}{1 - \varphi}$$

But this means we should set

$$\varphi = \frac{L_5}{L_5 + L_6} = \frac{11.09}{11.09 + 16.09} \approx \frac{2}{5}$$

Since  $2/5 > 1/4$ , it is possible to achieve a  $2/5 - 3/5$  split of the non-crossing edges, when there are two choices.

Finally, what does this make the average degree of the graph  $G$  at the time when 11-colorability can no longer be guaranteed?? Since

$$\bar{d} \approx \frac{2^{r-1}\bar{d}_1}{\varphi} \approx \frac{2^{r-1}\bar{d}_2}{1 - \varphi}$$

when  $r = 2$  and  $\varphi = 2/5$  we get

$$\bar{d} \approx \frac{2L_5}{2/5} = 55.45 > 50.356 = U_{11}.$$

Thus two choices suffice to raise the 11-colorability threshold.

To write down an *explicit* edge selection strategy, note that if when we are not forced to take an edge on a particular side, we toss a biased coin that selects side 1 with probability  $\gamma$ , then the overall probability of adding an edge to side 1 conditioned on adding a non-crossing edge is  $1/4 + \gamma/2$ . Since we want this to be  $2/5$  we should set  $\gamma = 3/10$ . Here is the strategy we use.

► **Strategy 2. BiasedPreferCrossing for  $k = 11$**

*Given two edges, select the first crossing edge, if any.*

*Otherwise if both non crossing edges are on the same side, select the first one*

*Otherwise there is one edge offered on each side. Select the one on side 1 with probability 0.3, and the one on side 2 with probability 0.7.*

## 4 Emergence of Giant component and Emergence of Cycles

The case  $k = 2$  differs from larger  $k$  in one very important way: namely, the  $k$ -Colorability Threshold Conjecture is false when  $k = 2$ ; for  $G(n, p)$  where  $p = d/n$ , rather than a sharp transition from colorable to non-colorable at a critical value of  $d$ , instead this transition is spread across the whole range  $0 < d < 1$ .

To see this, we observe that the expected number of triangles is  $\binom{n}{3}p^3 \approx d^3/6$ , which is a positive constant for all  $0 < d < 1$ . It is not much harder to prove that the probability that at least one triangle exists is also  $\Theta(1)$  whenever  $d = \Theta(1)$ , and hence the probability that  $G(n, p)$  is not 2-colorable is bounded away from zero.

On the other hand, it is also not hard to prove that, as long as  $p < (1 - \epsilon)/n$ ,  $G(n, p)$  is a forest with probability bounded below by a constant, and hence the probability that  $G(n, p)$  is 2-colorable is also bounded away from zero. In other words, the transition from  $G(n, d/n)$  being almost surely 2-colorable to being almost surely not 2-colorable is not sharp, but is rather spread over the entire interval  $0 < d < 1$ .

Even though there isn't a sharp threshold for 2-colorability in  $G(n, p)$ , we will prove in this section that, given  $r = 2$  choices, we can both create a sharp threshold, and shift it.

Two-colorability is of course, equivalent to the absence of odd cycles, and it turns out that the presence of odd cycles—indeed, of any cycles—is intimately linked with the emergence of the giant component.

Consider a 2-choice Achlioptas process, using the following, very simple, edge selection rule:

► **Strategy 3. SimpleAvoidCycles**

*Select the first edge, unless it would create a cycle, in which case, select the second edge.*

SimpleAvoidCycles manages to avoid the emergence of cycles until the average degree is 1, the threshold for the emergence of the giant component. On the other hand, once a giant component forms, it very quickly grows to size  $\omega(\sqrt{n})$ , at which point it is almost certain that a pair of edges will be offered within  $o(n)$  steps, both of which lie within the giant component. Therefore it is not possible to avoid cycles for more than a few steps after the formation of a giant component. Thus, with two choices, this very simple heuristic results in a sharp threshold for the emergence of cycles (and similarly for odd cycles, a.k.a. non-2-colorability).

#### 4.1 Analysis of SimpleAvoidCycles

As before, let  $m = dn/2$ , where  $d < 1$ . Consider the graph  $G' = G(n, m')$ , where  $m' = m + \log n$ .

For our purposes,  $G(n, m')$  means the graph obtained from sampling  $m'$  independent edges uniformly from  $\binom{n}{2}$  (with replacement).

► **Lemma 8.** *The number of edges of  $G'$  contained in one or more cycles is  $o(\log n)$ , almost surely.*

**Proof.** This is a standard result, so we present an abbreviated proof. The expected number of cycles of length  $k$  in the  $G(n, p)$  model is

$$\binom{n}{k} \frac{k!}{2k} p^k < \frac{(np)^k}{2k}.$$

Since each  $k$ -cycle contains  $k$  edges, it follows that the expected number of edges in  $k$ -cycles is less than  $(np)^k/2$ . If we set  $np = 1 - \epsilon$  and sum over all  $k \geq 3$ , we get

$$\sum_{k=3}^n \frac{(1 - \epsilon)^k}{2} < \sum_{k=3}^{\infty} \frac{(1 - \epsilon)^k}{2} = \frac{(1 - \epsilon)^3}{2\epsilon} = O(1).$$

We omit the details of the comparison between the  $G(n, m)$  model and the  $G(n, p)$  model, which are standard. Since the expected number of edges in cycles is  $O(1)$ , whereas  $\log(n)$  tends to infinity, by Markov's inequality it is almost certain that the actual number of edges in cycles is  $o(\log n)$ . ◀

► **Lemma 9.** *The probability that any of the edges  $e_{m+1}, \dots, e_{m'}$  are contained in a cycle of  $G'$  is  $O(\log(n)/n)$ .*

**Proof.** Since, by Lemma 8, the expected number of edges in cycles is  $O(1)$ , and since the  $m'$  edges of  $G'$  are identically distributed, it follows that each edge  $e_j$  has probability  $O(1/m')$  to be part of a cycle. Hence, by linearity of expectation and Markov's inequality, the probability that any of the edges  $e_{m+1}, \dots, e_{m'}$  is part of a cycle is  $O((m' - m)/m') = O(\log(n)/n)$ . ◀

► **Theorem 10.** *For  $d < 1$ , SimpleAvoidCycles outputs a cycle-free graph, almost surely.*

**Proof.** We couple the  $m$  choices made by SimpleAvoidCycles with the edges chosen in  $G(n, m')$ . For each  $1 \leq i \leq m$ , let  $e_i$  be the first edge offered to SimpleAvoidCycles. For each  $j$ 'th edge rejected by SimpleAvoidCycles, we let  $e_{m+j}$  be the second edge offered to SimpleAvoidCycles. When  $j$  is greater than the number of edges rejected by SimpleAvoidCycles, we let  $e_{m+j}$  be a uniformly random edge, chosen independently from all others.

Our first observation is that the sequence of edges  $e_1, \dots, e_{m'}$  is uniformly random in  $\binom{[n]}{m'}$ . This is because each  $e_j$  is uniformly random, conditioned on  $e_1, \dots, e_{j-1}$ .

Now, suppose the output of SimpleAvoidCycles contains a cycle. This means that at least one of the “second edges” chosen by SimpleAvoidCycles is contained in a cycle in the output of SimpleAvoidCycles. This implies that either SimpleAvoidCycles rejected more than  $m' - m$  first edges, in which case  $e_1, \dots, e_m$  contains more than  $m' - m$  cycles, and hence so does  $G'$ . This is unlikely by Lemma 8. Or SimpleAvoidCycles rejected fewer than  $m' - m$  edges, but one of the second edges formed a cycle in its output, which is a subgraph of  $G'$ . But Lemma 9 bounds the probability of this event. Applying the union bound to these two events, we get the desired upper bound on the probability that the output of SimpleAvoidCycles contains a cycle. ◀

## 4.2 Avoiding Cycles Longer

Next we will show how to keep  $G$  a forest as long as the average degree is less than 1.6587, the threshold from Theorem 7. More generically, we will show how, if any strategy for a 2-choice Achlioptas process can delay the giant component until average degree  $d$ , we can tweak it to additionally keep  $G$  a forest up to the same average degree threshold. We will refer to this strategy as DelayGiant. To be more precise, we will assume that, for every  $d' < d$ , DelayGiant run for  $d'n/2$  steps almost surely outputs a graph whose components all have  $O(n^{1/4})$  vertices.

First, we argue that, without loss of generality, DelayGiant can be assumed to have the following two properties:

1. If exactly one of the two offered edges make a cycle, DelayGiant selects it.
2. In this case, the subsequent behavior of DelayGiant is independent of the second, unselected edge.

The first property is obvious, since if an edge forms a cycle, adding it to  $G$  does not increase any of the component sizes; therefore it dominates any edge that doesn't form a cycle. The second property is less obvious, but the idea is that any strategy can be made “forgetful” by making it resample any state information it might be maintaining, from its conditional



distribution, conditioned on the edges it has accepted so far. It follows from the Law of Total Probability that this does not change the distribution of the output. An algorithm that is forgetful in this sense, and satisfies property 1, necessarily satisfies property 2 as well. The motivation for property 2 is that it will allow us to apply the Principle of Deferred Decisions to the edges chosen by our strategy in steps when it deviates from DelayGiant's choices.

Now our strategy for delaying the appearance of the first cycle in  $G$  can be described in one sentence:

► **Strategy 4. AvoidCycles**

*Select the edge chosen by the DelayGiant algorithm, unless it would form a cycle, in which case, select the other edge.*

► **Theorem 11.** *For  $d < 1.6587$ , with high probability, the 2-choice Achlioptas process run for  $m = dn/2$  steps using strategy AvoidCycles outputs a cycle-free graph.*

Consider a run of the DelayGiant algorithm. Let  $\{(e_1, e'_1), (e_2, e'_2) \dots (e_m, e'_m)\}$  be the edges that are offered to the algorithm during this run. Let  $G_i$  be the graph produced by DelayGiant after the first  $i$  steps, *i.e.*  $G_i$  has  $i$  edges, one out of each pair  $(e_j, e'_j)$ ,  $1 \leq j \leq i$ . Let

$$S := \{i \mid \text{neither of the edges } e_i, e'_i \text{ forms a cycle when added to } G_{i-1}\}$$

Let DelayGiant' be an algorithm that emulates DelayGiant on the steps in  $S$ , but adds no edge on the  $m - |S|$  steps when DelayGiant would add a cycle-forming edge. Let  $G'_i$  be the intermediate graph produced by DelayGiant' after  $i$  steps. Note that for all  $i$ ,  $G'_i$  is a spanning forest of  $G_i$ .

By assumption, almost surely, all the components of  $G_m$  have size  $o(n^{1/4})$ . Hence also, for all  $1 \leq i \leq m$ , the components of  $G_i$ , and therefore also  $G'_i$  have size  $O(n^{1/4})$ . Now, consider an arbitrary forest all of whose components are of size at most  $t$ . We make two observations:

► **Observation 12.** *Let  $G$  be a graph, all of whose components are of size at most  $t$ . Then the probability that adding one random edge to  $G$  creates a cycle is at most  $\frac{t-1}{n-1}$ .*

► **Observation 13.** *Let  $G$  be a graph, all of whose components are of size at most  $t$ . Add any  $\ell$  edges to  $G$ . Then, the largest component of the resulting graph has size at most  $\ell t$*

Applying Observation 12 inductively to each  $G'_i$ , with  $t = O(n^{1/4})$ , we see that the expected number of steps on which DelayGiant' adds no edge,  $\mathbb{E}[m - |S|]$ , is at most  $m \left( \frac{t-1}{n-1} \right)$ , which is  $O(n^{1/4})$ .

When DelayGiant' has run for  $m$  steps, the resulting graph  $G'_m$  is a forest with  $|S|$  edges, whose components are size  $O(n^{1/4})$ . Let DelayGiant'' be the algorithm that runs DelayGiant' and then expands  $G'_m$  to a graph with  $m$  edges by adding  $m - |S| = O(n^{1/4})$  uniformly random edges. Applying Observation 13, the components of this graph have size at most  $O(n^{1/2})$ . Since each of the  $O(n^{1/4})$  random edges to be added has at most  $O(n^{-1/2})$  chance of forming a cycle, by Markov's inequality, the probability that this graph contains a cycle is at most  $O(n^{-1/4})$ . Thus, the graph produced by DelayGiant'' is almost surely a forest.

The proof of Theorem 11 will be complete once we establish the following Lemma, relating AvoidCycles to DelayGiant''.

► **Lemma 14.** *AvoidCycles is better at avoiding cycles than DelayGiant'', i.e., for every  $m$ ,*

$$\mathbb{P}(\text{AvoidCycles is cycle-free after } m \text{ edges}) \geq \mathbb{P}(\text{DelayGiant'' is cycle-free after } m \text{ edges}).$$

**Proof.** It will suffice to couple the choices made by the two algorithms in such a way that each edge chosen by DelayGiant'' is either the same as the one chosen by AvoidCycles, or forms a cycle. Consider the edge chosen by AvoidCycles at a particular timestep  $i \in [m] \setminus S$ . Also, let  $A_i$  denote the set of all possible edges that would form a cycle if added to  $G_{i-1}$ , and let  $B_i = \binom{n}{2} \setminus A_i$ . We apply the principle of deferred decisions to the edges  $(e_i, e'_i)$ . Conditioned on  $G_{i-1}$  and the event that  $i \notin S$ , the distribution of  $(e_i, e'_i)$  is uniform in  $(A_i \cup B_i)^2 \setminus B_i^2$ . This means that the edge selected by AvoidCycles in step  $i$  has a conditional distribution which is uniform in  $A_i$  with probability  $\frac{|A_i|}{|A_i|+2|B_i|}$  and uniform in  $B_i$  with probability  $\frac{2|B_i|}{|A_i|+2|B_i|}$ .

Let us compare this distribution with that of a uniformly random edge. A uniformly random edge is uniform in  $A_i$  with probability  $\frac{|A_i|}{|A_i|+|B_i|}$ , and uniform in  $B_i$  with probability  $\frac{|B_i|}{|A_i|+|B_i|}$ . Now, observing that

$$\frac{a}{a+2b} < \frac{a}{a+b}$$

whenever  $a, b > 0$ , we see that the edge selected by AvoidCycles can be coupled with the uniformly random edge so that either the two edges are either equal, or the edge selected by AvoidCycles is in  $B_i$  and the uniformly random edge is in  $A_i$ . Since an edge in  $A_i$  would have formed a cycle even at step  $i$ , it definitely forms a cycle when added to the final result of DelayGiant.

Moreover, conditioned on the edges  $e_1, \dots, e_m$ , the deferred edges  $e_{m+j}$  are fully independent, since Property 2 tells us that DelayGiant does not take the identities of previously rejected edges into account when making its decisions. Thus, the sequence of edges  $e_{m+1}, \dots, e_{m'}$  is less likely to make a cycle than a sequence of  $m' - m$  uniformly random edges. It follows that there is a coupling between the output of AvoidCycles and DelayGiant'' such that the graphs produced are always identical except when DelayGiant'' contains at least one cycle. ◀

## 5 Four or Five Colors

When we get down to fewer than six colors, the basic PreferCrossing strategy runs into some difficulties, since at least one of the sides has fewer than three colors. This is problematic because even at low edge densities,  $G(n, m)$  has a constant chance of having an odd cycle and therefore cannot be two-colored. This means that the subgraph  $G_i$  of  $G$  on the side with only two colors will stop being two-colorable even before it has a linear number of edges. Fortunately, as we saw in the previous section, given a choice of two edges to choose from, we can avoid the appearance of cycles and keep the graph two-colorable until it reaches an average degree of about 1.6587.

When  $k = 4$ , we partition  $V$  into two sides as usual, and assign two of the four colors to each side. We prefer crossing edges as usual, and select a crossing edge whenever we are offered one. If there are at least three edges to choose from, and we are not offered any crossing edges, then at least two of the offered non-crossing edges are on the same side, and we have some room to be selective about the edge we are adding, and avoid cycles in the graph. Note that either side is equally likely to have two or more edges, and conditioned on the side, the edge choices are uniformly random from that side.

Here is an explicit description of the edge-selection strategy used:

► **Strategy 5. PreferCrossing with Two-sided Cycle Avoidance (PCTCA)**

Choose  $r = 3$  edges independently and uniformly at random

if there are any crossing edges then

| Select the first crossing edge.

else

| Let  $G_i$  be the side with more candidate edges

| Select the edge chosen by AvoidCycles on  $G_i$ .

end

Using the above edge selection strategy, we can show that

► **Theorem 15.** *Three choices suffice to increase the 4-colorability threshold.*

**Proof.** Let  $m$  be the total number of edges inserted into  $G$ , so the average degree of  $G$  is  $\bar{d} = 2m/n$ . For  $i \in \{1, 2\}$ , let  $\bar{d}_i$  denote the average degree of the graph  $G_i$ .

The probability of inserting a crossing edge into  $G$  is  $7/8$ . When there are no crossing edges, the chance that a particular side has two edge choices is  $1/16$ . We choose one of the two or more offered edges using the AvoidCycles strategy so that for  $i \in \{1, 2\}$  the expected number of edges inserted into  $G_i$  is  $m/16$ . Thus  $\mathbb{E}[\bar{d}_i] = \frac{2m/16}{n/2} = \frac{\bar{d}}{8}$ , and as usual,

$$\bar{d}_i \leq (1 + o(1))\bar{d}/8$$

Since we are using the AvoidCycles strategy to insert edges into  $G_1$  and  $G_2$ , by Theorem 11 we can push  $\bar{d}_1$  to 1.6587 before  $G_i$  stops being two-colorable. At that point,

$$\bar{d} = 8 \times 1.6587 = 13.2696 > 9.704 = 7 \ln 4 = U_4$$

so that  $G$  is 4-colorable at a density where  $G(n, m)$  isn't, and we have shifted the threshold. ◀

When  $k = 5$  we assign two colors to  $G_1$  and three colors to  $G_2$ . Again, we choose crossing edges whenever we can; if there are  $r = 3$  choices we can do this about  $7/8$  of the time.

What happens when we can't choose a crossing edge? Half the time, there will be two edges offered on side 1 and we can use AvoidCycles to choose one of them. If we choose an edge on side 2 the other half the time, then we will have  $\bar{d}_1 = \bar{d}_2 = \bar{d}/8$  and as we know from the four-colorability analysis above, we can push this up to  $\bar{d}_1 = 1.6587$  and  $\bar{d} = 13.2696$  before the 2-coloring on  $G_1$  breaks down. But  $13.2696 < 14.485 = 9 \ln 5 = U_5$  so we haven't shifted the 5-colorability threshold. Of course, at this point,  $\bar{d}_2$  is also only 1.6587, and has a lot of slack before it reaches  $L'_3 = 4.03$ , or even  $L_3 = 2.77$ .

So we want to use a biased strategy that favors choosing edges from side 2 when no crossing edges are available. We could figure out the optimal bias that makes both sides reach their limits at the same time, as we did in the  $k = 11$  case. Instead we opt for the following simple explicit strategy.

► **Strategy 6. PreferCrossing with One-sided Cycle Avoidance (PCOCA)**

Choose  $r = 3$  edges independently and uniformly at random

if there are any crossing edges then

| Select the first crossing edge.

else

| if the first two edges are both in  $V_1^2$  then

| | Select one of them according to *AvoidCycles*, run on  $G_1$

| else

| | (In this case at least one edge is in  $V_2^2$ )

| | Select the first edge in  $V_2^2$ .

| end

end

► **Theorem 16.** *Three choices suffice to increase the 5-colorability threshold.*

**Proof.** Let  $m$  be the total number of edges inserted into  $G$  so the average degree of  $G$  is  $\bar{d} = 2m/n$ . Similarly, for  $i \in \{1, 2\}$ , let  $\bar{d}_i$  denote the average degree of the graph  $G_i$ .

The probability of choosing a crossing edge is  $7/8$ . The probability of choosing an edge on side 1 is  $(1/4)(1/4)(1/2) = 1/32$ , and the probability of choosing an edge on side 2 is  $3/32$ . Then  $\mathbb{E}[\bar{d}_1] = \bar{d}/16$  and  $\mathbb{E}[\bar{d}_2] = 3\bar{d}/16$

If we set  $m = 8n$  then  $\bar{d} = 16 > U_5$  is a density at which  $G(n, m)$  is not 5-colorable. On the other hand if  $\bar{d} = 16$  in  $G$  constructed using PCOCA, then  $\bar{d}_1 = 1 < 1.6586$  and  $\bar{d}_2 = 3 < 4.03 = L'_3$ , so that  $G_1$  is two-colorable,  $G_2$  is 3-colorable and hence  $G$  is 5-colorable. ◀

## 6 Three Colors

For  $k = 3$ , we face a new challenge to our approach, namely: there is no longer any hope of using disjoint color sets to color the two sides of our graph. Instead, we try to make the color sets as disjoint as possible. Specifically, we try to color  $G$  using red and yellow for the first side, and blue and yellow for the second side. Although the crossing edges may cause problems now, at least the only bad color assignment for a crossing edge is (yellow, yellow). We call this kind of 3-coloring a  $(Y, *)$ -coloring, since the non-yellow colors are determined by their side.

Note that this specific type of coloring can be found in linear time, since it is a special case of Constrained Graph 3-Coloring, which is reducible to 2-SAT (see [8, Problem 5.6]). Here is our strategy:

► **Strategy 7. PreferCrossingButCheck (PCBC)**

Choose  $r = 6$  edges independently and uniformly at random.

Let  $e$  be the edge chosen by the *PreferCrossing* heuristic.

Check whether  $G \cup \{e\}$  remains  $(Y, *)$ -colorable. If it is, select  $e$ . Otherwise, select the first edge other than  $e$ .

We note that with an appropriate data structure, all  $m$  of the colorability checks can be performed in combined expected time  $O(n)$ . However, since our goal is just to show that the colorability transition can be shifted, we leave the details as an exercise.

We claim that, when  $r = 6$ , the output of PCBC is almost surely  $(Y, *)$ -colorable. To see this, observe that, in order for a greedy approach to coloring to fail, the graph must have a cycle of length  $2k + 1$  with edges (in order)  $(e_1, e_2, \dots, e_{2k+1})$ , where the  $k$  even edges  $e_{2i}$  are all non-crossing. This is analogous to the fact that a graph fails to be 2-colorable if and only if it has an odd cycle. However, note that in the case of 2-coloring, the criterion is “if and only if,” whereas here there is only an implication; the cycle is only guaranteed to cause a problem if we start by coloring the wrong vertex yellow. We call a cycle of this type a “bad odd cycle.”

► **Proposition 17.** *Let  $d > 0$ . Let  $G$  be the output of an Achlioptas process with  $r$  choices, running the PreferCrossing heuristic, for  $dn/2$  steps. Also suppose that  $d^2 < 2^r$ . Then the expected number of edges contained in bad odd cycles of  $G$  is  $O(1)$ .*

**Proof.** Note that, for every vertex  $v$ , the expected degree is  $d$ , but the expected number of non-crossing edges incident with  $v$  is  $d2^{-r}$ . With a little work we can see that the expected number of walks of length  $2k$  starting at a particular node, in which all the even steps are along non-crossing edges is at most  $d^k(d2^{-r})^k$ . In order to complete such a walk to a cycle of length  $2k + 1$ , we need a particular edge to be present, which is an event of probability at most  $d/(n/2)$ . Since there are  $n$  possible starting points for our walk, this gives the following bound on the number of edges contained in a bad odd cycle:

$$n \sum_{k \geq 1} \frac{d}{n/2} (d^2 2^{-r})^k (2k + 1) = 2d \sum_{k \geq 1} \sum_{k \geq 1} (2k + 1) (d^2 2^{-r})^k,$$

which since  $d^2 < 2^r$ , is a convergent sum. ◀

Thus, in expectation, PCBC deviates from the choices made by PreferCrossing on only  $O(1)$  steps. Denote this number of steps by  $m' - m$ . On the steps when it deviates, it takes the first alternative edge. Since PreferCrossing makes its edge choice based only on which edges are crossing or not, this alternative edge must be uniformly random, conditioned on whether it is a crossing edge or not. It follows that PCBC succeeds at least as often as a variant PCBC' that, instead of taking each rejected edge from PreferCrossing, instead adds one uniformly random crossing edge and one uniformly random non-crossing edge.

PCBC', in turn, will almost surely perform at least as well as another variant, PCBC'', which, instead of adding one uniformly random crossing edge, and one uniformly random non-crossing edge, instead adds  $C2^r$  edges chosen by an Achlioptas process running the PreferCrossings strategy, where  $C \rightarrow \infty$ . But now, observe that PCBC'' is just PreferCrossings run for  $m'' = m + o(n)$  steps, with all of its bad odd cycles from the first  $m$  steps broken up. Since PreferCrossings run for  $m''$  steps still has, in expectation,  $O(1)$  edges involved in bad odd cycles, and these edges are uniformly randomly distributed among the  $m'$  steps, the probability that any of them occur in the last  $m'' - m$  steps is  $O((m'' - m)/m'') = o(1)$ . Hence the output of PCBC'' almost surely has no bad odd cycles, and is therefore  $(Y, *)$ -colorable. Since by our earlier remarks, PCBC almost surely performs at least as well as PCBC'', this establishes the result.

## 7 Two choices for 9 colors

In Section 3 as part of a unified analysis for  $k \geq 6$  we showed that three choices were enough to raise the 9-colorability threshold. In this section we will show that in fact just two choices suffice. Surprisingly, this result involves a more uneven split of the colors, with three colors

reserved for  $V_1$  and six for  $V_2$ . This helps partly because, for  $k = 3$ , the improved lower bound  $L'_3 = 4.03$  of Theorem 5 is significantly better than the bound of Theorem 3.

The main idea is to use a biased PreferCrossing strategy which favors the six color side when a non-crossing edge is forced. We have two choices, so we will be putting in a non-crossing edge only a fourth of the time. Conditioned in that, we want to make the colorability on the two sides break at roughly the same time. As we saw before, this means that we should add edges to side 1 (with three colors) with probability  $\varphi$ , where

$$\frac{\varphi}{1 - \varphi} = \frac{L'_3}{L_6} = \frac{4.03}{16.094} \approx \frac{1}{4}$$

from which we get that  $\varphi$  should be approximately  $1/5$ ... and that is a problem. If the probability of selecting an edge from side 1 conditioned on a non-crossing edge is  $1/5$ , then the overall probability is  $1/20$ , but this is not achievable with two choices, since there is a  $1/16$  chance that both edges are on side 1!

So where does that leave us? It turns out that we can still tweak this to make it work. From the beginning, we have made the *a priori* division of the vertex set into two equal sized disjoint subsets because that maximizes our ability to put in crossing edges. But having found ourselves in a situation where we want to put in fewer edges into side one than is possible, the obvious solution seems to be to make side one smaller. So let's start over, and partition  $V$  into disjoint sets  $V_1$  and  $V_2$ , where  $|V_1| = \alpha n$  and  $|V_2| = (1 - \alpha)n$ . It turns out that  $\alpha = 0.47$  works well. With this parameter setting, we choose crossing edges whenever possible, and failing that, edges in  $G_2$ , with edges in  $G_1$  as a last resort. This leads to average degrees  $\bar{d}_1 = 0.1038\bar{d}$  and  $\bar{d}_2 = 0.3830\bar{d}$ . Since  $0.1038U_9 \leq L_3$  and  $0.3830U_9 \leq L_6$ , this shows that we have shifted the 9-coloring threshold with  $r = 2$  choices.

---

## References

- 1 Dimitris Achlioptas and Cristopher Moore. Almost all graphs with average degree 4 are 3-colorable. *Journal of Computer and System Sciences*, 67(2):441–471, 2003.
- 2 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.
- 3 Tom Bohman and Alan Frieze. Avoiding a giant component. *Random Structures & Algorithms*, 19(1):75–85, 2001.
- 4 Tom Bohman, Alan Frieze, and Nicholas C Wormald. Avoidance of a giant component in half the edge set of a random graph. *Random Structures & Algorithms*, 25(4):432–449, 2004.
- 5 Amin Coja-Oghlan and Dan Vilenchik. Chasing the  $k$ -colorability threshold. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 380–389. IEEE, 2013.
- 6 Varsha Dani, Josep Diaz, Thomas Hayes, and Cristopher Moore. The power of choice for random satisfiability. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 484–496. Springer, 2013.
- 7 Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.*, 5(1):17–60, 1960.
- 8 Cristopher Moore and Stephan Mertens. *The nature of computation*. OUP Oxford, 2011.
- 9 Will Perkins. Random  $k$ -SAT and the power of two choices. *Random Structures & Algorithms*, 47(1):163–173, 2015.
- 10 Andrea W Richa, M Mitzenmacher, and R Sitaraman. The power of two random choices: A survey of techniques and results. *Combinatorial Optimization*, 9:255–304, 2001.
- 11 Oliver Riordan and Lutz Warnke. Achlioptas process phase transitions are continuous. *The Annals of Applied Probability*, 22(4):1450–1464, 2012.
- 12 Alistair Sinclair and Dan Vilenchik. Delaying satisfiability for random 2-SAT. *Random Structures & Algorithms*, 43(2):251–263, 2013.

- 13 Joel Spencer and Nicholas Wormald. Birth control for giants. *Combinatorica*, 27(5):587–628, 2007.

## Appendix A: Hastening the threshold

Here we present a sketch of the proof of Observation 2. Since the choice strategy and the proof technique are exactly the same as in [6], we omit most of the details.

**Proof Sketch for Observation 2.** The choice strategy is to favor some vertex set  $S$ , where  $|S| = \gamma n$ . For instance, let  $S = \{1, 2, \dots, \gamma n\}$ . By always choosing a random edge in  $\binom{S}{2}$  when one is available, we find that the induced graph on  $S$  is uniformly random, but denser than  $G$  as a whole, having average degree asymptotically equal to  $(1 - (1 - \gamma^2)^r)/\gamma$  times the average degree of  $G$ . Choosing  $\gamma$  to maximize this expression, we obtain the desired choice strategy. For instance, setting  $\gamma = 1/\sqrt{r}$ , we can see that  $(1 - (1 - \gamma^2)^r)/\gamma = \Theta(\sqrt{r})$ , which tends to infinity. This shows that the favored subgraph can be made arbitrarily more dense than  $G$ , thus bridging the gap between any upper and lower bounds on the threshold. ◀





# Memory-Sample Lower Bounds for Learning Parity with Noise

Sumegha Garg ✉

Department of Computer Science, Harvard University, Cambridge, MA, USA

Pravesh K. Kothari ✉

Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

Pengda Liu ✉

Department of Computer Science, Stanford University, CA, USA

Ran Raz ✉

Department of Computer Science, Princeton University, NJ, USA

---

## Abstract

In this work, we show, for the well-studied problem of learning parity under noise, where a learner tries to learn  $x = (x_1, \dots, x_n) \in \{0, 1\}^n$  from a stream of random linear equations over  $\mathbb{F}_2$  that are correct with probability  $\frac{1}{2} + \varepsilon$  and flipped with probability  $\frac{1}{2} - \varepsilon$  ( $0 < \varepsilon < \frac{1}{2}$ ), that any learning algorithm requires either a memory of size  $\Omega(n^2/\varepsilon)$  or an exponential number of samples.

In fact, we study memory-sample lower bounds for a large class of learning problems, as characterized by [8], when the samples are noisy. A matrix  $M : A \times X \rightarrow \{-1, 1\}$  corresponds to the following learning problem with error parameter  $\varepsilon$ : an unknown element  $x \in X$  is chosen uniformly at random. A learner tries to learn  $x$  from a stream of samples,  $(a_1, b_1), (a_2, b_2) \dots$ , where for every  $i$ ,  $a_i \in A$  is chosen uniformly at random and  $b_i = M(a_i, x)$  with probability  $1/2 + \varepsilon$  and  $b_i = -M(a_i, x)$  with probability  $1/2 - \varepsilon$  ( $0 < \varepsilon < \frac{1}{2}$ ). Assume that  $k, \ell, r$  are such that any submatrix of  $M$  of at least  $2^{-k} \cdot |A|$  rows and at least  $2^{-\ell} \cdot |X|$  columns, has a bias of at most  $2^{-r}$ . We show that any learning algorithm for the learning problem corresponding to  $M$ , with error parameter  $\varepsilon$ , requires either a memory of size at least  $\Omega\left(\frac{k \cdot \ell}{\varepsilon}\right)$ , or at least  $2^{\Omega(r)}$  samples. The result holds even if the learner has an exponentially small success probability (of  $2^{-\Omega(r)}$ ). In particular, this shows that for a large class of learning problems, same as those in [8], any learning algorithm requires either a memory of size at least  $\Omega\left(\frac{(\log |X|) \cdot (\log |A|)}{\varepsilon}\right)$  or an exponential number of noisy samples.

Our proof is based on adapting the arguments in [21, 8] to the noisy case.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Machine learning theory

**Keywords and phrases** memory-sample tradeoffs, learning parity under noise, space lower bound, branching program

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.60

**Category** RANDOM

**Funding** *Sumegha Garg*: Research supported by Michael O. Rabin Postdoctoral Fellowship.

*Pravesh K. Kothari*: Research supported by NSF CAREER Award No. 2047933.

*Ran Raz*: Research supported by the Simons Collaboration on Algorithms and Geometry, by a Simons Investigator Award and by the National Science Foundation grants No. CCF-1714779, CCF-2007462.

**Acknowledgements** We would like to thank Avishay Tal and Greg Valiant for the helpful discussions.

## 1 Introduction

In this work, we study the number of samples needed for learning under noise and memory constraints. The study of the resources needed for learning, under memory constraints was initiated by Shamir [22] and Steinhardt, Valiant and Wager [24], and has been studied



© Sumegha Garg, Pravesh K. Kothari, Pengda Liu, and Ran Raz;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 60; pp. 60:1–60:19



Leibniz International Proceedings in Informatics

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

in the streaming setting. In addition to being a natural question in learning theory and complexity theory, lower bounds in this model also have direct applications to bounded storage cryptography [20, 26, 16, 25, 11, 13, 6, 12]. [24] conjectured that any algorithm for learning parities of size  $n$  (that is, learning  $x \in \{0, 1\}^n$  from a stream of random linear equations in  $\mathbb{F}_2$ ) requires either a memory of size  $\Omega(n^2)$  or an exponential number of samples. This conjecture was proven in [20] and in follow up works, this was generalized to learning sparse parities in [16] and more general learning problems in [21, 17, 19, 8, 2, 5, 18, 23, 9, 4, 10].

In this work, we extend this line of work to *noisy* Boolean function learning problems. In particular, we consider the well-studied problem of learning parity under noise (LPN). In this problem, a learner wants to learn  $x \in \{0, 1\}^n$  from independent and uniformly random linear equations in  $\mathbb{F}_2$  where the right hand sides are obtained by independently flipping the evaluation of an unknown parity function with probability  $\frac{1}{2} - \varepsilon$ . Learning Parity with Noise (LPN) is a central problem in Learning and Coding Theory (often referred to as decoding random linear codes) and has been extensively studied. Even without memory constraints, coming up with algorithms for the problem has proven to be challenging and the current state-of-the-art for solving the problem is still the celebrated work of Blum, Kalai and Wasserman [3] that runs in time  $2^{O(n/\log_2(n))}$ . Over time, the hardness of LPN (and its generalization to non-binary finite fields) has been used as a starting point in several hardness results [14, 7] and constructing cryptographic primitives [1]. On the other hand, lower-bounds for the problem are known only in restricted models such as Statistical Query Learning<sup>1</sup> [15].

Learning under noise is at least as hard as learning without noise and thus, memory-sample lower bounds for parity learning [20] holds for learning parity under noise too. It is natural to ask – can we get better space lower bounds for learning parities under noise? In this work, we are able to strengthen the memory lower bound to  $\Omega(n^2/\varepsilon)$  for parity learning with noise.

Our results actually extend to a broad class of learning problems – under noise. As in [21] and follow up works, we represent a learning problem using a matrix. Let  $X, A$  be two finite sets (where  $X$  represents the concept-class that we are trying to learn and  $A$  represents the set of possible samples). Let  $M : A \times X \rightarrow \{-1, 1\}$  be a matrix. The matrix  $M$  represents the following learning problem with error parameter  $\varepsilon$  ( $0 < \varepsilon < \frac{1}{2}$ ): An unknown element  $x \in X$  was chosen uniformly at random. A learner tries to learn  $x$  from a stream of samples,  $(a_1, b_1), (a_2, b_2) \dots$ , where for every  $i$ ,  $a_i \in A$  is chosen uniformly at random and  $b_i = M(a_i, x)$  with probability  $\frac{1}{2} + \varepsilon$ .

## 1.1 Our Results

We use extractor-based characterization of the matrix  $M$  to prove our lower bounds, as done in [8]. Our main result can be stated as follows (Corollary 19): Assume that  $k, \ell, r$  are such that any submatrix of  $M$  of at least  $2^{-k} \cdot |A|$  rows and at least  $2^{-\ell} \cdot |X|$  columns, has a bias of at most  $2^{-r}$ . Then, any learning algorithm for the learning problem corresponding to  $M$  with error parameter  $\varepsilon$  requires either a memory of size at least  $\Omega(k \cdot \ell/\varepsilon)$ , or at least  $2^{\Omega(r)}$  samples. Thus, we get an extra factor of  $\frac{1}{\varepsilon}$  in the space lower bound for all the bounds on learning problems that [8] imply, some of which are as follows (see [8] for details on why the corresponding matrices satisfy the extractor-based property):

---

<sup>1</sup> The SQ model does not seem to distinguish between noisy and noiseless variants of parity learning and yields the same lower bound in both cases.

1. **Parities with noise:** A learner tries to learn  $x = (x_1, \dots, x_n) \in \{0, 1\}^n$ , from (a stream of) random linear equations over  $\mathbb{F}_2$  which are correct with probability  $\frac{1}{2} + \varepsilon$  and flipped with probability  $\frac{1}{2} - \varepsilon$ . Any learning algorithm requires either a memory of size  $\Omega(n^2/\varepsilon)$  or an exponential number of samples.
2. **Sparse parities with noise:** A learner tries to learn  $x = (x_1, \dots, x_n) \in \{0, 1\}^n$  of sparsity  $\ell$ , from (a stream of) random linear equations over  $\mathbb{F}_2$  which are correct with probability  $\frac{1}{2} + \varepsilon$  and flipped with probability  $\frac{1}{2} - \varepsilon$ . Any learning algorithm requires:
  - a. Assuming  $\ell \leq n/2$ : either a memory of size  $\Omega(n \cdot \ell/\varepsilon)$  or  $2^{\Omega(\ell)}$  samples.
  - b. Assuming  $\ell \leq n^{0.9}$ : either a memory of size  $\Omega(n \cdot \ell^{0.99}/\varepsilon)$  or  $\ell^{\Omega(\ell)}$  samples.
3. **Learning from noisy sparse linear equations:** A learner tries to learn  $x = (x_1, \dots, x_n) \in \{0, 1\}^n$ , from (a stream of) random sparse linear equations, of sparsity  $\ell$ , over  $\mathbb{F}_2$ , which are correct with probability  $\frac{1}{2} + \varepsilon$  and flipped with probability  $\frac{1}{2} - \varepsilon$ . Any learning algorithm requires:
  - a. Assuming  $\ell \leq n/2$ : either a memory of size  $\Omega(n \cdot \ell/\varepsilon)$  or  $2^{\Omega(\ell)}$  samples.
  - b. Assuming  $\ell \leq n^{0.9}$ : either a memory of size  $\Omega(n \cdot \ell^{0.99}/\varepsilon)$  or  $\ell^{\Omega(\ell)}$  samples.
4. **Learning from noisy low-degree equations:** A learner tries to learn  $x = (x_1, \dots, x_n) \in \{0, 1\}^n$ , from (a stream of) random multilinear polynomial equations of degree at most  $d$ , over  $\mathbb{F}_2$ , which are correct with probability  $\frac{1}{2} + \varepsilon$  and flipped with probability  $\frac{1}{2} - \varepsilon$ . We prove that if  $d \leq 0.99 \cdot n$ , any learning algorithm requires either a memory of size  $\Omega\left(\binom{n}{\leq d} \frac{n}{d \cdot \varepsilon}\right)$  or  $2^{\Omega(n/d)}$  samples (where  $\binom{n}{\leq d} = \binom{n}{0} + \binom{n}{1} + \dots + \binom{n}{d}$ ).
5. **Low-degree polynomials with noise:** A learner tries to learn an  $n'$ -variate multilinear polynomial  $p$  of degree at most  $d$  over  $\mathbb{F}_2$ , from (a stream of) random evaluations of  $p$  over  $\mathbb{F}_2^{n'}$ , which are correct with probability  $\frac{1}{2} + \varepsilon$  and flipped with probability  $\frac{1}{2} - \varepsilon$ . We prove that if  $d \leq 0.99 \cdot n'$ , any learning algorithm requires either a memory of size  $\Omega\left(\binom{n'}{\leq d} \cdot \frac{n'}{d \cdot \varepsilon}\right)$  or  $2^{\Omega(n'/d)}$  samples.

## 1.2 Techniques

Our proof follows the proof of [21, 8] very closely and builds on that proof. We extend the extractor-based result of [8] to the noisy case and a straightforward adaptation to its proof gives the stronger lower bound for the noisy case (which reflects on the strength of the current techniques). The main contribution of this paper is not a technical one but establishing stronger space lower bounds for a well-studied problem of learning parity with noise, using the current techniques.

## 1.3 Discussion and Open Problem

Let's look at a space upper bound for the problem of learning parity with noise, that is, a learner tries to learn  $x \in \{0, 1\}^n$  from a stream of samples of the form  $(a, b)$ , where  $a \in \{0, 1\}^n$  is chosen uniformly at random and  $b = a \cdot x$  with probability  $\frac{1}{2} + \varepsilon$  and  $b = 1 - a \cdot x$  with probability  $\frac{1}{2} - \varepsilon$  (here,  $a \cdot x$  represents the inner product of  $a$  and  $x$  in  $\mathbb{F}_2$ , that is,  $a \cdot x = \sum_i a_i x_i \pmod{2}$ ).

### Upper Bound

Consider the following algorithm  $A$ : Store the first  $m = O(n/\varepsilon^2)$  samples. Check for every  $x' \in \{0, 1\}^n$ , if for at least  $(\frac{1}{2} + \frac{\varepsilon}{2})$  fraction of the samples  $(a_1, b_1), \dots, (a_m, b_m)$ ,  $a_i \cdot x'$  agrees with  $b_i$ . Output the first  $x'$  that satisfies the check. In expectation,  $a_i \cdot x$  would agree with  $b_i$  for  $(\frac{1}{2} + \varepsilon)$  fraction of the samples, and otherwise for  $x' \neq x$ , in expectation,  $a_i \cdot x'$  would

agree with  $b_i$  for half the samples. Therefore, for large enough  $m$ , using Chernoff bound and a union bound, with high probability  $(1 - o(1))$  over the  $m$  samples,  $x'$  satisfies the check if and only if  $x' = x$ , and  $A$  outputs the correct answer under such an event.  $A$  uses  $O(n/\varepsilon^2)$  samples and  $O(n^2/\varepsilon^2)$  bits of space.

In this paper, we prove that any algorithm that learns parity with noise from a stream of samples (as defined above) requires  $\Omega(n^2/\varepsilon)$  bits of space or exponential number of samples. Improving the lower bound to match the upper bound (or vice versa) is a fascinating open problem and we conjecture that the upper bound is tight. As each sample gives at most  $O(\varepsilon^2)$  bits of information about  $x$ , we can at least show that a learning algorithm requires  $O(n/\varepsilon^2)$  samples to learn  $x$  (which corresponds to using  $O(n^2/\varepsilon^2)$  bits of space if each sample is stored).

► **Conjecture 1.** *Any learner that tries to learn  $x \in \{0, 1\}^n$  from a stream of samples of the form  $(a, b)$ , where  $a \in \{0, 1\}^n$  is chosen uniformly at random and  $b = a \cdot x$  with probability  $\frac{1}{2} + \varepsilon$  and  $b = 1 - a \cdot x$  with probability  $\frac{1}{2} - \varepsilon$ , requires either  $\Omega(n^2/\varepsilon^2)$  bits of memory or  $2^{\Omega(n)}$  samples.*

The proof of the conjecture, if true, would lead to new technical insights (beyond extractor-based techniques) into proving time-space (or memory-sample) lower bounds for learning problems.

## 1.4 Outline of the Paper

In Section 2, we establish certain notations and definitions, which are borrowed from [21, 8]. We give a proof overview in Section 3 and prove the main theorem in Section 4.

## 2 Preliminaries

Denote by  $\mathcal{U}_X : X \rightarrow \mathbb{R}^+$  the uniform distribution over  $X$ . Denote by  $\log$  the logarithm to base 2. For a random variable  $Z$  and an event  $E$ , we denote by  $\mathbb{P}_Z$  the distribution of the random variables  $Z$ , and we denote by  $\mathbb{P}_{Z|E}$  the distribution of the random variable  $Z$  conditioned on the event  $E$ .

### Viewing a Learning Problem, with error $\frac{1}{2} - \varepsilon$ , as a Matrix

Let  $X, A$  be two finite sets of size larger than 1. Let  $n = \log_2 |X|$  and  $n' = \log_2 |A|$ .

Let  $M : A \times X \rightarrow \{-1, 1\}$  be a matrix. The matrix  $M$  corresponds to the following learning problem with error parameter  $\varepsilon$  ( $0 < \varepsilon < \frac{1}{2}$ ). There is an unknown element  $x \in X$  that was chosen uniformly at random. A learner tries to learn  $x$  from samples  $(a, b)$ , where  $a \in A$  is chosen uniformly at random, and  $b = M(a, x)$  with probability  $\frac{1}{2} + \varepsilon$  and  $b = -M(a, x)$  with probability  $\frac{1}{2} - \varepsilon$ . That is, the learning algorithm is given a stream of samples,  $(a_1, b_1), (a_2, b_2) \dots$ , where each  $a_t$  is uniformly distributed, and  $b_t = M(a_t, x)$  with probability  $\frac{1}{2} + \varepsilon$  and  $b_t = -M(a_t, x)$  with probability  $\frac{1}{2} - \varepsilon$ .

### Norms and Inner Products

Let  $p \geq 1$ . For a function  $f : X \rightarrow \mathbb{R}$ , denote by  $\|f\|_p$  the  $\ell_p$  norm of  $f$ , with respect to the uniform distribution over  $X$ , that is:

$$\|f\|_p = \left( \mathbf{E}_{x \in_R X} [|f(x)|^p] \right)^{1/p}.$$

For two functions  $f, g : X \rightarrow \mathbb{R}$ , define their inner product with respect to the uniform distribution over  $X$  as

$$\langle f, g \rangle = \mathbf{E}_{x \in_R X} [f(x) \cdot g(x)].$$

For a matrix  $M : A \times X \rightarrow \mathbb{R}$  and a row  $a \in A$ , we denote by  $M_a : X \rightarrow \mathbb{R}$  the function corresponding to the  $a$ -th row of  $M$ . Note that for a function  $f : X \rightarrow \mathbb{R}$ , we have  $\langle M_a, f \rangle = \frac{(M \cdot f)_a}{|X|}$ . Here,  $M \cdot f$  represents the matrix multiplication of  $M$  with  $f$ .

## $L_2$ -Extractors and $L_\infty$ -Extractors

► **Definition 2** ( $L_2$ -Extractor). *Let  $X, A$  be two finite sets. A matrix  $M : A \times X \rightarrow \{-1, 1\}$  is a  $(k, \ell)$ - $L_2$ -Extractor with error  $2^{-r}$ , if for every non-negative  $f : X \rightarrow \mathbb{R}$  with  $\frac{\|f\|_2}{\|f\|_1} \leq 2^\ell$  there are at most  $2^{-k} \cdot |A|$  rows  $a$  in  $A$  with*

$$\frac{|\langle M_a, f \rangle|}{\|f\|_1} \geq 2^{-r}.$$

Let  $\Omega$  be a finite set. We denote a distribution over  $\Omega$  as a function  $f : \Omega \rightarrow \mathbb{R}^+$  such that  $\sum_{x \in \Omega} f(x) = 1$ . We say that a distribution  $f : \Omega \rightarrow \mathbb{R}^+$  has min-entropy  $k$  if for all  $x \in \Omega$ , we have  $f(x) \leq 2^{-k}$ .

► **Definition 3** ( $L_\infty$ -Extractor). *Let  $X, A$  be two finite sets. A matrix  $M : A \times X \rightarrow \{-1, 1\}$  is a  $(k, \ell \sim r)$ - $L_\infty$ -Extractor if for every distribution  $p_x : X \rightarrow \mathbb{R}^+$  with min-entropy at least  $(\log(|X|) - \ell)$  and every distribution  $p_a : A \rightarrow \mathbb{R}^+$  with min-entropy at least  $(\log(|A|) - k)$ ,*

$$\left| \sum_{a' \in A} \sum_{x' \in X} p_a(a') \cdot p_x(x') \cdot M(a', x') \right| \leq 2^{-r}.$$

## Branching Program for a Learning Problem

In the following definition, we model the learner for the learning problem that corresponds to the matrix  $M$ , by a *branching program*, as done by previous papers starting with [20].

► **Definition 4. Branching Program for a Learning Problem:** *A branching program of length  $m$  and width  $d$ , for learning, is a directed (multi) graph with vertices arranged in  $m + 1$  layers containing at most  $d$  vertices each. In the first layer, that we think of as layer 0, there is only one vertex, called the start vertex. A vertex of outdegree 0 is called a leaf. All vertices in the last layer are leaves (but there may be additional leaves). Every non-leaf vertex in the program has  $2|A|$  outgoing edges, labeled by elements  $(a, b) \in A \times \{-1, 1\}$ , with exactly one edge labeled by each such  $(a, b)$ , and all these edges going into vertices in the next layer. Each leaf  $v$  in the program is labeled by an element  $\tilde{x}(v) \in X$ , that we think of as the output of the program on that leaf.*

**Computation-Path:** *The samples  $(a_1, b_1), \dots, (a_m, b_m) \in A \times \{-1, 1\}$  that are given as input, define a computation-path in the branching program, by starting from the start vertex and following at step  $t$  the edge labeled by  $(a_t, b_t)$ , until reaching a leaf. The program outputs the label  $\tilde{x}(v)$  of the leaf  $v$  reached by the computation-path.*

**Success Probability:** *The success probability of the program is the probability that  $\tilde{x} = x$ , where  $\tilde{x}$  is the element that the program outputs, and the probability is over  $x, a_1, \dots, a_m, b_1, \dots, b_m$  (where  $x$  is uniformly distributed over  $X$  and  $a_1, \dots, a_m$  are uniformly distributed over  $A$ , and for every  $t$ ,  $b_t = M(a_t, x)$  with probability  $\frac{1}{2} + \varepsilon$  and  $-M(a_t, x)$  with probability  $\frac{1}{2} - \varepsilon$ ).*

A learning algorithm, using  $m$  samples and a memory of  $s$  bits, can be modeled as a branching program<sup>2</sup> of length  $m$  and width  $2^{O(s)}$ . Thus, we will focus on proving width-length tradeoffs for any branching program that learns an extractor-based learning problem with noise, and such tradeoffs would translate into memory-sample tradeoffs for the learning algorithms.

### 3 Overview of the Proof

The proof adapts the extractor-based time-space lower bound of [8] to the *noisy* case, which in turn built on [21] that gave a general technique for proving memory-samples lower bounds. We recall the arguments in [21, 8] for convenience.

Assume that  $M$  is a  $(k', \ell')$ - $L_2$ -extractor with error  $2^{-r'}$ , and let  $r = \min\{k', \ell', r'\}$ . Let  $B$  be a branching program for the noisy learning problem that corresponds to the matrix  $M$ . We want to prove that  $B$  has at least  $2^{\Omega(r)}$  length or requires at least  $2^{\Omega(\frac{k'\ell'}{\varepsilon})}$  width (that is, any learning algorithm solving the learning problem corresponding to the matrix  $M$  with error parameter  $\varepsilon$ , requires either  $\Omega(\frac{k'\ell'}{\varepsilon})$  memory or exponential number of samples). Assume for a contradiction that  $B$  is of length  $m = 2^{cr}$  and width  $d = 2^{c\frac{k'\ell'}{\varepsilon}}$ , where  $c > 0$  is a small constant.

We define the *truncated-path*,  $\mathcal{T}$ , to be the same as the computation-path of  $B$ , except that it sometimes stops before reaching a leaf. Roughly speaking,  $\mathcal{T}$  stops before reaching a leaf if certain “bad” events occur. Nevertheless, we show that the probability that  $\mathcal{T}$  stops before reaching a leaf is negligible, so we can think of  $\mathcal{T}$  as almost identical to the computation-path.

For a vertex  $v$  of  $B$ , we denote by  $E_v$  the event that  $\mathcal{T}$  reaches the vertex  $v$ . We denote by  $\Pr(v) = \Pr(E_v)$  the probability for  $E_v$  (where the probability is over  $x, a_1, \dots, a_m, b_1, \dots, b_m$ ), and we denote by  $\mathbb{P}_{x|v} = \mathbb{P}_{x|E_v}$  the distribution of the random variable  $x$  conditioned on the event  $E_v$ . Similarly, for an edge  $e$  of the branching program  $B$ , let  $E_e$  be the event that  $\mathcal{T}$  traverses the edge  $e$ . Denote,  $\Pr(e) = \Pr(E_e)$ , and  $\mathbb{P}_{x|e} = \mathbb{P}_{x|E_e}$ .

A vertex  $v$  of  $B$  is called *significant* if

$$\|\mathbb{P}_{x|v}\|_2 > 2^{\ell'} \cdot 2^{-n}.$$

Roughly speaking, this means that conditioning on the event that  $\mathcal{T}$  reaches the vertex  $v$ , a non-negligible amount of information is known about  $x$ . In order to guess  $x$  with a non-negligible success probability,  $\mathcal{T}$  must reach a significant vertex. Lemma 6 shows that the probability that  $\mathcal{T}$  reaches any significant vertex is negligible, and thus the main result follows.

To prove Lemma 6, we show that for every fixed significant vertex  $s$ , the probability that  $\mathcal{T}$  reaches  $s$  is at most  $2^{-\Omega(k'\ell'/\varepsilon)}$  (which is smaller than one over the number of vertices in  $B$ ). Hence, we can use a union bound to prove the lemma.

The proof that the probability that  $\mathcal{T}$  reaches  $s$  is extremely small is the main part of the proof. To that end, we use the following functions to measure the progress made by the branching program towards reaching  $s$ .

<sup>2</sup> The lower bound holds for randomized learning algorithms because a branching program is a non-uniform model of computation, and we can fix a *good* randomization for the computation without affecting the width.

Let  $L_i$  be the set of vertices  $v$  in layer- $i$  of  $B$ , such that  $\Pr(v) > 0$ . Let  $\Gamma_i$  be the set of edges  $e$  from layer- $(i-1)$  of  $B$  to layer- $i$  of  $B$ , such that  $\Pr(e) > 0$ . Let

$$\mathcal{Z}_i = \sum_{v \in L_i} \Pr(v) \cdot \langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^{k'/2\varepsilon},$$

$$\mathcal{Z}'_i = \sum_{e \in \Gamma_i} \Pr(e) \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^{k'/2\varepsilon}.$$

We think of  $\mathcal{Z}_i, \mathcal{Z}'_i$  as measuring the progress made by the branching program, towards reaching a state with distribution similar to  $\mathbb{P}_{x|s}$ .

We show that each  $\mathcal{Z}_i$  may only be negligibly larger than  $\mathcal{Z}_{i-1}$ . Hence, since it's easy to calculate that  $\mathcal{Z}_0 = 2^{-\frac{2nk'}{2\varepsilon}}$ , it follows that  $\mathcal{Z}_i$  is close to  $2^{-\frac{2nk'}{2\varepsilon}}$ , for every  $i$ . On the other hand, if  $s$  is in layer- $i$  then  $\mathcal{Z}_i$  is at least  $\Pr(s) \cdot \langle \mathbb{P}_{x|s}, \mathbb{P}_{x|s} \rangle^{\frac{k'}{2\varepsilon}}$ . Thus,  $\Pr(s) \cdot \langle \mathbb{P}_{x|s}, \mathbb{P}_{x|s} \rangle^{\frac{k'}{2\varepsilon}}$  cannot be much larger than  $2^{-2n\frac{k'}{2\varepsilon}}$ . Since  $s$  is significant,  $\langle \mathbb{P}_{x|s}, \mathbb{P}_{x|s} \rangle^{\frac{k'}{2\varepsilon}} > 2^{(2\ell' - 2n)\frac{k'}{2\varepsilon}}$  and hence  $\Pr(s)$  is at most  $2^{-\Omega(\frac{k'\ell'}{\varepsilon})}$ .

The proof that  $\mathcal{Z}_i$  may only be negligibly larger than  $\mathcal{Z}_{i-1}$  is done in two steps: Claim 17 shows by a simple convexity argument that  $\mathcal{Z}_i \leq \mathcal{Z}'_i$ . The hard part, that is done in Claim 15 and Claim 16, is to prove that  $\mathcal{Z}'_i$  may only be negligibly larger than  $\mathcal{Z}_{i-1}$ .

For this proof, we define for every vertex  $v$ , the set of edges  $\Gamma_{out}(v)$  that are going out of  $v$ , such that  $\Pr(e) > 0$ . Claim 15 shows that for every vertex  $v$ ,

$$\sum_{e \in \Gamma_{out}(v)} \Pr(e) \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^{k'/2\varepsilon}$$

may only be negligibly higher than

$$\Pr(v) \cdot \langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^{k'/2\varepsilon}.$$

For the proof of Claim 15, which is the hardest proof in the paper, we follow [21, 8] and consider the function  $\mathbb{P}_{x|v} \cdot \mathbb{P}_{x|s}$ . We first show how to bound  $\|\mathbb{P}_{x|v} \cdot \mathbb{P}_{x|s}\|_2$ . We then consider two cases: If  $\|\mathbb{P}_{x|v} \cdot \mathbb{P}_{x|s}\|_1$  is negligible, then  $\langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^{k'/2\varepsilon}$  is negligible and doesn't contribute much, and we show that for every  $e \in \Gamma_{out}(v)$ ,  $\langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^{k'/2\varepsilon}$  is also negligible and doesn't contribute much. If  $\|\mathbb{P}_{x|v} \cdot \mathbb{P}_{x|s}\|_1$  is non-negligible, we use the bound on  $\|\mathbb{P}_{x|v} \cdot \mathbb{P}_{x|s}\|_2$  and the assumption that  $M$  is a  $(k', \ell')$ - $L_2$ -extractor to show that for almost all edges  $e \in \Gamma_{out}(v)$ , we have that  $\langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^{k'/2\varepsilon}$  is very close to  $\langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^{k'/2\varepsilon}$ . Only an exponentially small  $(2^{-k'})$  fraction of edges are “bad” and give a significantly larger  $\langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^{k'/2\varepsilon}$ . In the noiseless case, any “bad” edge can increase  $\langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle$  by a factor of 2 in the worst case, and hence [8] raised  $\langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle$  and  $\langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle$  to the power of  $k'$ , as it is the largest power for which the contribution of the “bad” edges is still small (as their fraction is  $2^{-k'}$ ). But in the noisy case, any “bad” edge can increase  $\langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle$  by a factor of at most  $(1 + 2\varepsilon)$  in the worst case, and thus, we can afford to raise  $\langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle$  and  $\langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle$  to the power of  $k'/2\varepsilon$ . *This is where our proof differs from that of [8].*

This outline oversimplifies many details. To make the argument work, we force  $\mathcal{T}$  to stop at significant vertices and whenever  $\mathbb{P}_{x|v}(x)$  is large, that is, at significant values, as done in previous papers. And we force  $\mathcal{T}$  to stop before traversing some edges, that are so “bad” that their contribution to  $\mathcal{Z}'_i$  is huge and they cannot be ignored. We show that the total probability that  $\mathcal{T}$  stops before reaching a leaf is negligible.

## 4 Main Result

► **Theorem 5.** Let  $\frac{1}{100} < c < \frac{\ln 2}{3}$ . Fix  $\gamma$  to be such that  $\frac{3c}{\ln 2} < \gamma^2 < 1$ . Let  $X, A$  be two finite sets. Let  $n = \log_2 |X|$ . Let  $M : A \times X \rightarrow \{-1, 1\}$  be a matrix which is a  $(k', \ell')$ - $L_2$ -extractor with error  $2^{-r'}$ , for sufficiently large<sup>3</sup>  $k', \ell'$  and  $r'$ , where  $\ell' \leq n$ . Let

$$r := \min \left\{ \frac{r'}{2}, \frac{(1-\gamma)k'}{2}, \frac{(1-\gamma)\ell'}{2} - 1 \right\}. \quad (1)$$

Let  $B$  be a branching program, of length at most  $2^r$  and width at most  $2^{c \cdot k' \cdot \ell' / \varepsilon}$ , for the learning problem that corresponds to the matrix  $M$  with error parameter  $\varepsilon$ . Then, the success probability of  $B$  is at most  $O(2^{-r})$ .

**Proof.** We recall the proof in [8, 21] and adapt it to the noisy case. Let

$$k := \frac{\gamma \ln 2}{2\varepsilon} k' \quad \text{and} \quad \ell := \gamma \ell' / 3. \quad (2)$$

Our proof differs from [8] starting with Claim 10, which allows us to set  $k$  to a larger value of  $\frac{\gamma \ln 2}{2\varepsilon} k'$  instead of  $\gamma(\ln 2)k'$  as set in [8]. Note that by the assumption that  $k', \ell'$  and  $r'$  are sufficiently large, we get that  $k, \ell$  and  $r$  are also sufficiently large. Since  $\ell' \leq n$ , we have  $\ell + r \leq \frac{\gamma \ell'}{3} + \frac{(1-\gamma)\ell'}{2} < \frac{\ell'}{2} \leq \frac{n}{2}$ . Thus,

$$r < n/2 - \ell. \quad (3)$$

Let  $B$  be a branching program of length  $m = 2^r$  and width<sup>4</sup>  $d = 2^{c \cdot k' \cdot \ell' / \varepsilon}$  for the learning problem that corresponds to the matrix  $M$  with error parameter  $\varepsilon$ . We will show that the success probability of  $B$  is at most  $O(2^{-r})$ .

### 4.1 The Truncated-Path and Additional Definitions and Notation

We will define the **truncated-path**,  $\mathcal{T}$ , to be the same as the computation-path of  $B$ , except that it sometimes stops before reaching a leaf. Formally, we define  $\mathcal{T}$ , together with several other definitions and notations, by induction on the layers of the branching program  $B$ .

Assume that we already defined the truncated-path  $\mathcal{T}$ , until it reaches layer- $i$  of  $B$ . For a vertex  $v$  in layer- $i$  of  $B$ , let  $E_v$  be the event that  $\mathcal{T}$  reaches the vertex  $v$ . For simplicity, we denote by  $\Pr(v) = \Pr(E_v)$  the probability for  $E_v$  (where the probability is over  $x, a_1, \dots, a_m, b_1, \dots, b_m$ ), and we denote by  $\mathbb{P}_{x|v} = \mathbb{P}_{x|E_v}$  the distribution of the random variable  $x$  conditioned on the event  $E_v$ .

There will be three cases in which the truncated-path  $\mathcal{T}$  stops on a non-leaf  $v$ :

1. If  $v$  is a, so called, significant vertex, where the  $\ell_2$  norm of  $\mathbb{P}_{x|v}$  is non-negligible. (Intuitively, this means that conditioned on the event that  $\mathcal{T}$  reaches  $v$ , a non-negligible amount of information is known about  $x$ ).
2. If  $\mathbb{P}_{x|v}(x)$  is non-negligible. (Intuitively, this means that conditioned on the event that  $\mathcal{T}$  reaches  $v$ , the correct element  $x$  could have been guessed with a non-negligible probability).
3. If  $(M \cdot \mathbb{P}_{x|v})(a_{i+1})$  is non-negligible. (Intuitively, this means that  $\mathcal{T}$  is about to traverse a “bad” edge, which is traversed with a non-negligibly higher or lower probability than probability of traversal under uniform distribution on  $x$ ).

Next, we describe these three cases more formally.

<sup>3</sup> By “sufficiently large” we mean that  $k', \ell', r'$  are larger than some constant that depends on  $\gamma$ .

<sup>4</sup> width lower bound is vacuous for  $\varepsilon < 2^{-r/2}$  as regardless of the width,  $\Omega(n/\varepsilon^2) > 2^r$  samples are needed to learn.



## Significant Vertices

We say that a vertex  $v$  in layer- $i$  of  $B$  is **significant** if

$$\|\mathbb{P}_{x|v}\|_2 > 2^\ell \cdot 2^{-n}.$$

## Significant Values

Even if  $v$  is not significant,  $\mathbb{P}_{x|v}$  may have relatively large values. For a vertex  $v$  in layer- $i$  of  $B$ , denote by  $\text{Sig}(v)$  the set of all  $x' \in X$ , such that,

$$\mathbb{P}_{x|v}(x') > 2^{2\ell+2r} \cdot 2^{-n}.$$

## Bad Edges

For a vertex  $v$  in layer- $i$  of  $B$ , denote by  $\text{Bad}(v)$  the set of all  $\alpha \in A$ , such that,

$$|(M \cdot \mathbb{P}_{x|v})(\alpha)| \geq 2^{-r'}.$$

## The Truncated-Path $\mathcal{T}$

We define  $\mathcal{T}$  by induction on the layers of the branching program  $B$ . Assume that we already defined  $\mathcal{T}$  until it reaches a vertex  $v$  in layer- $i$  of  $B$ . The path  $\mathcal{T}$  stops on  $v$  if (at least) one of the following occurs:

1.  $v$  is significant.
2.  $x \in \text{Sig}(v)$ .
3.  $a_{i+1} \in \text{Bad}(v)$ .
4.  $v$  is a leaf.

Otherwise,  $\mathcal{T}$  proceeds by following the edge labeled by  $(a_{i+1}, b_{i+1})$  (same as the computational-path).

## 4.2 Proof of Theorem 5

Since  $\mathcal{T}$  follows the computation-path of  $B$ , except that it sometimes stops before reaching a leaf, the success probability of  $B$  is bounded (from above) by the probability that  $\mathcal{T}$  stops before reaching a leaf, plus the probability that  $\mathcal{T}$  reaches a leaf  $v$  and  $\tilde{x}(v) = x$ .

The main lemma needed for the proof of Theorem 5 is Lemma 6 that shows that the probability that  $\mathcal{T}$  reaches a significant vertex is at most  $O(2^{-r})$ .

► **Lemma 6.** *The probability that  $\mathcal{T}$  reaches a significant vertex is at most  $O(2^{-r})$ .*

Lemma 6 is proved in Section 4.3. We will now show how the proof of Theorem 5 follows from that lemma.

Lemma 6 shows that the probability that  $\mathcal{T}$  stops on a non-leaf vertex, because of the first reason (i.e., that the vertex is significant), is small. The next two claims imply that the probabilities that  $\mathcal{T}$  stops on a non-leaf vertex, because of the second and third reasons, are also small. We defer the proofs to Appendix A (proved as in [8]).

▷ **Claim 7.** If  $v$  is a non-significant vertex of  $B$  then

$$\Pr_x[x \in \text{Sig}(v) \mid E_v] \leq 2^{-2r}.$$

## 60:10 Memory-Sample Lower Bounds for Learning Parity with Noise

▷ **Claim 8.** If  $v$  is a non-significant vertex of  $B$  then

$$\Pr_{a_{i+1}} [a_{i+1} \in \text{Bad}(v)] \leq 2^{-2r}.$$

We can now use Lemma 6, Claim 7 and Claim 8 to prove that the probability that  $\mathcal{T}$  stops before reaching a leaf is at most  $O(2^{-r})$ . Lemma 6 shows that the probability that  $\mathcal{T}$  reaches a significant vertex and hence stops because of the first reason, is at most  $O(2^{-r})$ . Assuming that  $\mathcal{T}$  doesn't reach any significant vertex (in which case it would have stopped because of the first reason), Claim 7 shows that in each step, the probability that  $\mathcal{T}$  stops because of the second reason, is at most  $2^{-2r}$ . Taking a union bound over the  $m = 2^r$  steps, the total probability that  $\mathcal{T}$  stops because of the second reason, is at most  $2^{-r}$ . In the same way, assuming that  $\mathcal{T}$  doesn't reach any significant vertex (in which case it would have stopped because of the first reason), Claim 8 shows that in each step, the probability that  $\mathcal{T}$  stops because of the third reason, is at most  $2^{-2r}$ . Again, taking a union bound over the  $2^r$  steps, the total probability that  $\mathcal{T}$  stops because of the third reason, is at most  $2^{-r}$ . Thus, the total probability that  $\mathcal{T}$  stops (for any reason) before reaching a leaf is at most  $O(2^{-r})$ .

Recall that if  $\mathcal{T}$  doesn't stop before reaching a leaf, it just follows the computation-path of  $B$ . Recall also that by Lemma 6, the probability that  $\mathcal{T}$  reaches a significant leaf is at most  $O(2^{-r})$ . Thus, to bound (from above) the success probability of  $B$  by  $O(2^{-r})$ , it remains to bound the probability that  $\mathcal{T}$  reaches a non-significant leaf  $v$  and  $\tilde{x}(v) = x$ . Claim 9 shows that for any non-significant leaf  $v$ , conditioned on the event that  $\mathcal{T}$  reaches  $v$ , the probability for  $\tilde{x}(v) = x$  is at most  $2^{-r}$ , which completes the proof of Theorem 5.

▷ **Claim 9.** If  $v$  is a non-significant leaf of  $B$  then

$$\Pr[\tilde{x}(v) = x \mid E_v] \leq 2^{-r}.$$

Refer to Appendix A for the proof (proved as in [8]). This completes the proof of Theorem 5. ◀

### 4.3 Proof of Lemma 6

**Proof.** We need to prove that the probability that  $\mathcal{T}$  reaches any significant vertex is at most  $O(2^{-r})$ . Let  $s$  be a significant vertex of  $B$ . We will bound from above the probability that  $\mathcal{T}$  reaches  $s$ , and then use a union bound over all significant vertices of  $B$ . Interestingly, the upper bound on the width of  $B$  is used only in the union bound.

#### The Distributions $\mathbb{P}_{x|v}$ and $\mathbb{P}_{x|e}$

Recall that for a vertex  $v$  of  $B$ , we denote by  $E_v$  the event that  $\mathcal{T}$  reaches the vertex  $v$ . For simplicity, we denote by  $\Pr(v) = \Pr(E_v)$  the probability for  $E_v$  (where the probability is over  $x, a_1, \dots, a_m, b_1, \dots, b_m$ ), and we denote by  $\mathbb{P}_{x|v} = \mathbb{P}_{x|E_v}$  the distribution of the random variable  $x$  conditioned on the event  $E_v$ .

Similarly, for an edge  $e$  of the branching program  $B$ , let  $E_e$  be the event that  $\mathcal{T}$  traverses the edge  $e$ . Denote,  $\Pr(e) = \Pr(E_e)$  (where the probability is over  $x, a_1, \dots, a_m, b_1, \dots, b_m$ ), and  $\mathbb{P}_{x|e} = \mathbb{P}_{x|E_e}$ .

▷ **Claim 10.** For any edge  $e = (v, u)$  of  $B$ , labeled by  $(a, b)$ , such that  $\Pr(e) > 0$ , for any  $x' \in X$ ,

$$\mathbb{P}_{x|e}(x') = \begin{cases} 0 & \text{if } x' \in \text{Sig}(v) \\ \mathbb{P}_{x|v}(x')(1 + 2\varepsilon) \cdot c_e^{-1} & \text{if } x' \notin \text{Sig}(v) \text{ and } M(a, x') = b \\ \mathbb{P}_{x|v}(x')(1 - 2\varepsilon) \cdot c_e^{-1} & \text{if } x' \notin \text{Sig}(v) \text{ and } M(a, x') \neq b \end{cases}$$

where  $c_e$  is a normalization factor that satisfies,

$$c_e \geq 1 - 4 \cdot 2^{-2r}.$$

Proof. Let  $e = (v, u)$  be an edge of  $B$ , labeled by  $(a, b)$ , and such that  $\Pr(e) > 0$ . Since  $\Pr(e) > 0$ , the vertex  $v$  is not significant (as otherwise  $\mathcal{T}$  always stops on  $v$  and hence  $\Pr(e) = 0$ ). Also, since  $\Pr(e) > 0$ , we know that  $a \notin \text{Bad}(v)$  (as otherwise  $\mathcal{T}$  never traverses  $e$  and hence  $\Pr(e) = 0$ ).

If  $\mathcal{T}$  reaches  $v$ , it traverses the edge  $e$  if and only if:  $x \notin \text{Sig}(v)$  (as otherwise  $\mathcal{T}$  stops on  $v$ ) and  $a_{i+1} = a$ ,  $b_{i+1} = b$ . Therefore, by Bayes' rule, for any  $x' \in X$ ,

$$\mathbb{P}_{x|e}(x') = \begin{cases} 0 & \text{if } x' \in \text{Sig}(v) \\ \mathbb{P}_{x|v}(x')(1 + 2\varepsilon) \cdot c_e^{-1} & \text{if } x' \notin \text{Sig}(v) \text{ and } M(a, x') = b \\ \mathbb{P}_{x|v}(x')(1 - 2\varepsilon) \cdot c_e^{-1} & \text{if } x' \notin \text{Sig}(v) \text{ and } M(a, x') \neq b \end{cases}$$

where  $c_e$  is a normalization factor, given by

$$\begin{aligned} c_e &= \sum_{\{x' : x' \notin \text{Sig}(v) \wedge M(a, x') = b\}} \mathbb{P}_{x|v}(x')(1 + 2\varepsilon) \\ &\quad + \sum_{\{x' : x' \notin \text{Sig}(v) \wedge M(a, x') \neq b\}} \mathbb{P}_{x|v}(x')(1 - 2\varepsilon) \\ &= (1 + 2\varepsilon) \cdot \Pr_x[(x \notin \text{Sig}(v)) \wedge (M(a, x) = b) \mid E_v] \\ &\quad + (1 - 2\varepsilon) \cdot \Pr_x[(x \notin \text{Sig}(v)) \wedge (M(a, x) \neq b) \mid E_v]. \end{aligned}$$

Since  $v$  is not significant, by Claim 7,

$$\Pr_x[x \in \text{Sig}(v) \mid E_v] \leq 2^{-2r}.$$

Since  $a \notin \text{Bad}(v)$ ,

$$\left| \Pr_x[M(a, x) = 1 \mid E_v] - \Pr_x[M(a, x) = -1 \mid E_v] \right| = |(M \cdot \mathbb{P}_{x|v})(a)| \leq 2^{-r'},$$

and hence for every  $b' \in \{-1, 1\}$ ,

$$\Pr_x[M(a, x) = b' \mid E_v] \geq \frac{1}{2} - 2^{-r'}.$$

Hence, by the union bound,

$$c_e \geq (1 + 2\varepsilon) \cdot (\frac{1}{2} - 2^{-r'} - 2^{-2r}) + (1 - 2\varepsilon) \cdot (\frac{1}{2} - 2^{-r'} - 2^{-2r}) \geq 1 - 4 \cdot 2^{-2r}$$

(where the last inequality follows since  $r \leq r'/2$ , by Equation (1)).  $\triangleleft$

### Bounding the Norm of $\mathbb{P}_{x|s}$

We will show that  $\|\mathbb{P}_{x|s}\|_2$  cannot be too large. Towards this, we will first prove that for every edge  $e$  of  $B$  that is traversed by  $\mathcal{T}$  with probability larger than zero,  $\|\mathbb{P}_{x|e}\|_2$  cannot be too large. We defer the proofs of the following claims to Appendix A (proved as in [8]).

▷ **Claim 11.** For any edge  $e$  of  $B$ , such that  $\Pr(e) > 0$ ,

$$\|\mathbb{P}_{x|e}\|_2 \leq 4 \cdot 2^\ell \cdot 2^{-n}.$$

▷ **Claim 12.**

$$\|\mathbb{P}_{x|s}\|_2 \leq 4 \cdot 2^\ell \cdot 2^{-n}.$$

## 60:12 Memory-Sample Lower Bounds for Learning Parity with Noise

### Similarity to a Target Distribution

Recall that for two functions  $f, g : X \rightarrow \mathbb{R}^+$ , we defined

$$\langle f, g \rangle = \mathbf{E}_{z \in_R X} [f(z) \cdot g(z)].$$

We think of  $\langle f, g \rangle$  as a measure for the similarity between a function  $f$  and a target function  $g$ . Typically  $f, g$  will be distributions.

▷ Claim 13.

$$\langle \mathbb{P}_{x|s}, \mathbb{P}_{x|s} \rangle > 2^{2\ell} \cdot 2^{-2n}.$$

Proof. Since  $s$  is significant,

$$\langle \mathbb{P}_{x|s}, \mathbb{P}_{x|s} \rangle = \|\mathbb{P}_{x|s}\|_2^2 > 2^{2\ell} \cdot 2^{-2n}. \quad \triangleleft$$

▷ Claim 14.

$$\langle \mathcal{U}_X, \mathbb{P}_{x|s} \rangle = 2^{-2n},$$

where  $\mathcal{U}_X$  is the uniform distribution over  $X$ .

Proof. Since  $\mathbb{P}_{x|s}$  is a distribution,

$$\langle \mathcal{U}_X, \mathbb{P}_{x|s} \rangle = 2^{-2n} \cdot \sum_{z \in X} \mathbb{P}_{x|s}(z) = 2^{-2n}. \quad \triangleleft$$

### Measuring the Progress

For  $i \in \{0, \dots, m\}$ , let  $L_i$  be the set of vertices  $v$  in layer- $i$  of  $B$ , such that  $\Pr(v) > 0$ . For  $i \in \{1, \dots, m\}$ , let  $\Gamma_i$  be the set of edges  $e$  from layer- $(i-1)$  of  $B$  to layer- $i$  of  $B$ , such that  $\Pr(e) > 0$ . Recall that  $k = \frac{\gamma \ln 2}{2\varepsilon} k'$  (Equation (2)).

For  $i \in \{0, \dots, m\}$ , let

$$\mathcal{Z}_i = \sum_{v \in L_i} \Pr(v) \cdot \langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^k.$$

For  $i \in \{1, \dots, m\}$ , let

$$\mathcal{Z}'_i = \sum_{e \in \Gamma_i} \Pr(e) \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k.$$

We think of  $\mathcal{Z}_i, \mathcal{Z}'_i$  as measuring the progress made by the branching program, towards reaching a state with distribution similar to  $\mathbb{P}_{x|s}$ .

For a vertex  $v$  of  $B$ , let  $\Gamma_{out}(v)$  be the set of all edges  $e$  of  $B$ , that are going out of  $v$ , such that  $\Pr(e) > 0$ . Note that

$$\sum_{e \in \Gamma_{out}(v)} \Pr(e) \leq \Pr(v).$$

(We don't always have an equality here, since sometimes  $\mathcal{T}$  stops on  $v$ ).

The next four claims show that the progress made by the branching program is slow.

▷ **Claim 15.** For every vertex  $v$  of  $B$ , such that  $\Pr(v) > 0$ ,

$$\sum_{e \in \Gamma_{out}(v)} \frac{\Pr(e)}{\Pr(v)} \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k \leq \langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^k \cdot (1 + 2^{-r})^k + (2^{-2n+2})^k.$$

*Proof.* If  $v$  is significant or  $v$  is a leaf, then  $\mathcal{T}$  always stops on  $v$  and hence  $\Gamma_{out}(v)$  is empty and thus the left hand side is equal to zero and the right hand side is positive, so the claim follows trivially. Thus, we can assume that  $v$  is not significant and is not a leaf.

Define  $P : X \rightarrow \mathbb{R}^+$  as follows. For any  $x' \in X$ ,

$$P(x') = \begin{cases} 0 & \text{if } x' \in \text{Sig}(v) \\ \mathbb{P}_{x|v}(x') & \text{if } x' \notin \text{Sig}(v) \end{cases}$$

Note that by the definition of  $\text{Sig}(v)$ , for any  $x' \in X$ ,

$$P(x') \leq 2^{2\ell+2r} \cdot 2^{-n}. \quad (4)$$

Define  $f : X \rightarrow \mathbb{R}^+$  as follows. For any  $x' \in X$ ,

$$f(x') = P(x') \cdot \mathbb{P}_{x|s}(x').$$

By Claim 12 and Equation (4),

$$\|f\|_2 \leq 2^{2\ell+2r} \cdot 2^{-n} \cdot \|\mathbb{P}_{x|s}\|_2 \leq 2^{2\ell+2r} \cdot 2^{-n} \cdot 4 \cdot 2^\ell \cdot 2^{-n} = 2^{3\ell+2r+2} \cdot 2^{-2n}. \quad (5)$$

By Claim 10, for any edge  $e \in \Gamma_{out}(v)$ , labeled by  $(a, b)$ , for any  $x' \in X$ ,

$$\mathbb{P}_{x|e}(x') = \begin{cases} 0 & \text{if } x' \in \text{Sig}(v) \\ \mathbb{P}_{x|v}(x')(1 + 2\varepsilon) \cdot c_e^{-1} & \text{if } x' \notin \text{Sig}(v) \text{ and } M(a, x') = b \\ \mathbb{P}_{x|v}(x')(1 - 2\varepsilon) \cdot c_e^{-1} & \text{if } x' \notin \text{Sig}(v) \text{ and } M(a, x') \neq b \end{cases}$$

where  $c_e$  is a normalization factor that satisfies,

$$c_e \geq 1 - 4 \cdot 2^{-2r}.$$

Therefore, for any edge  $e \in \Gamma_{out}(v)$ , labeled by  $(a, b)$ , for any  $x' \in X$ ,

$$\mathbb{P}_{x|e}(x') \cdot \mathbb{P}_{x|s}(x') = f(x') \cdot (1 + 2\varepsilon \cdot b \cdot M(a, x')) \cdot c_e^{-1}$$

and hence, we have

$$\begin{aligned} \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle &= \mathbf{E}_{x' \in \mathcal{R}X} [\mathbb{P}_{x|e}(x') \cdot \mathbb{P}_{x|s}(x')] = \mathbf{E}_{x' \in \mathcal{R}X} [f(x') \cdot (1 + 2\varepsilon \cdot b \cdot M(a, x')) \cdot c_e^{-1}] \\ &= (\|f\|_1 + 2\varepsilon \cdot b \cdot \langle M_a, f \rangle) \cdot (c_e)^{-1} \\ &< (\|f\|_1 + 2\varepsilon |\langle M_a, f \rangle|) \cdot (1 + 2^{-2r+3}) \end{aligned} \quad (6)$$

(where the last inequality holds by the bound that we have on  $c_e$ , because we assume that  $k', \ell', r'$  and thus  $r$  are sufficiently large).

We will now consider two cases:

**Case I:**  $\|f\|_1 < 2^{-2n}$ . In this case, we bound  $|\langle M_a, f \rangle| \leq \|f\|_1$  (since  $f$  is non-negative and the entries of  $M$  are in  $\{-1, 1\}$ ) and  $(1 + 2^{-2r+3}) < 2$  (since we assume that  $k', \ell', r'$  and thus  $r$  are sufficiently large) and obtain for any edge  $e \in \Gamma_{out}(v)$ ,

$$\langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle < 4 \cdot 2^{-2n}.$$

Since  $\sum_{e \in \Gamma_{out}(v)} \frac{\Pr(e)}{\Pr(v)} \leq 1$ , Claim 15 follows, as the left hand side of the claim is smaller than the second term on the right hand side.

## 60:14 Memory-Sample Lower Bounds for Learning Parity with Noise

**Case II:**  $\|f\|_1 \geq 2^{-2n}$ . For every  $a \in A$ , define

$$t(a) = \frac{|\langle M_a, f \rangle|}{\|f\|_1}.$$

By Equation (6),

$$\langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k < \|f\|_1^k \cdot (1 + 2\varepsilon \cdot t(a))^k \cdot (1 + 2^{-2r+3})^k. \quad (7)$$

Note that by the definitions of  $P$  and  $f$ ,

$$\|f\|_1 = \mathbf{E}_{x' \in_{RA} X} [f(x')] = \langle P, \mathbb{P}_{x|s} \rangle \leq \langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle.$$

Note also that for every  $a \in A$ , there is at most one edge  $e_{(a,1)} \in \Gamma_{out}(v)$ , labeled by  $(a, 1)$ , and at most one edge  $e_{(a,-1)} \in \Gamma_{out}(v)$ , labeled by  $(a, -1)$ , and we have

$$\frac{\Pr(e_{(a,1)})}{\Pr(v)} + \frac{\Pr(e_{(a,-1)})}{\Pr(v)} \leq \frac{1}{|A|},$$

since  $\frac{1}{|A|}$  is the probability that the next sample read by the program is  $a$ . Thus, summing over all  $e \in \Gamma_{out}(v)$ , by Equation (7),

$$\sum_{e \in \Gamma_{out}(v)} \frac{\Pr(e)}{\Pr(v)} \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k < \langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^k \cdot \mathbf{E}_{a \in_{RA} A} \left[ (1 + 2\varepsilon \cdot t(a))^k \right] \cdot (1 + 2^{-2r+3})^k. \quad (8)$$

It remains to bound

$$\mathbf{E}_{a \in_{RA} A} \left[ (1 + 2\varepsilon \cdot t(a))^k \right], \quad (9)$$

using the properties of the matrix  $M$  and the bounds on the  $\ell_2$  versus  $\ell_1$  norms of  $f$ .

By Equation (5), the assumption that  $\|f\|_1 \geq 2^{-2n}$ , Equation (1) and Equation (2), we get

$$\frac{\|f\|_2}{\|f\|_1} \leq 2^{3\ell+2r+2} \leq 2^{\ell'}.$$

Since  $M$  is a  $(k', \ell')$ - $L_2$ -extractor with error  $2^{-r'}$ , there are at most  $2^{-k'} \cdot |A|$  rows  $a \in A$  with  $t(a) = \frac{|\langle M_a, f \rangle|}{\|f\|_1} \geq 2^{-r'}$ . We bound the expectation in Equation (9), by splitting the expectation into two sums

$$\mathbf{E}_{a \in_{RA} A} \left[ (1 + 2\varepsilon \cdot t(a))^k \right] = \frac{1}{|A|} \cdot \sum_{a : t(a) \leq 2^{-r'}} (1 + 2\varepsilon \cdot t(a))^k + \frac{1}{|A|} \cdot \sum_{a : t(a) > 2^{-r'}} (1 + 2\varepsilon \cdot t(a))^k. \quad (10)$$

We bound the first sum in Equation (10) by  $(1 + 2\varepsilon \cdot 2^{-r'})^k$ . As for the second sum in Equation (10), we know that it is a sum of at most  $2^{-k'} \cdot |A|$  elements, and since for every  $a \in A$ , we have  $t(a) \leq 1$ , we have

$$\frac{1}{|A|} \cdot \sum_{a : t(a) > 2^{-r'}} (1 + 2\varepsilon \cdot t(a))^k \leq 2^{-k'} \cdot (1 + 2\varepsilon)^k \leq 2^{-k'} e^{2\varepsilon k} \leq 2^{-2r}$$

(where in the last inequality we used Equations (1) and (2)). Overall, using Equation (1) again, we get

$$\mathbf{E}_{a \in_{RA} A} \left[ (1 + 2\varepsilon \cdot t(a))^k \right] \leq (1 + 2\varepsilon \cdot 2^{-r'})^k + 2^{-2r} \leq (1 + 2^{-2r})^{k+1}. \quad (11)$$

Substituting Equation (11) into Equation (8), we obtain

$$\begin{aligned} \sum_{e \in \Gamma_{out}(v)} \frac{\Pr(e)}{\Pr(v)} \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k &< \langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^k \cdot (1 + 2^{-2r})^{k+1} \cdot (1 + 2^{-2r+3})^k \\ &< \langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^k \cdot (1 + 2^{-r})^k \end{aligned}$$

(where the last inequality uses the assumption that  $r$  is sufficiently large). This completes the proof of Claim 15.  $\triangleleft$

The following three claims use Claim 15 to quantify the progress over the layers and we defer the proofs to Appendix A (proved as in [8]).

▷ **Claim 16.** For every  $i \in \{1, \dots, m\}$ ,

$$\mathcal{Z}'_i \leq \mathcal{Z}_{i-1} \cdot (1 + 2^{-r})^k + (2^{-2n+2})^k.$$

▷ **Claim 17.** For every  $i \in \{1, \dots, m\}$ ,

$$\mathcal{Z}_i \leq \mathcal{Z}'_i.$$

▷ **Claim 18.** For every  $i \in \{1, \dots, m\}$ ,

$$\mathcal{Z}_i \leq 2^{4k+2r} \cdot 2^{-2k \cdot n}.$$

## Proof of Lemma 6

We can now complete the proof of Lemma 6. Assume that  $s$  is in layer- $i$  of  $B$ . By Claim 13,

$$\mathcal{Z}_i \geq \Pr(s) \cdot \langle \mathbb{P}_{x|s}, \mathbb{P}_{x|s} \rangle^k > \Pr(s) \cdot (2^{2\ell} \cdot 2^{-2n})^k = \Pr(s) \cdot 2^{2\ell \cdot k} \cdot 2^{-2k \cdot n}.$$

On the other hand, by Claim 18,

$$\mathcal{Z}_i \leq 2^{4k+2r} \cdot 2^{-2k \cdot n}.$$

Thus, using Equation (1) and Equation (2), we get

$$\Pr(s) \leq 2^{4k+2r} \cdot 2^{-2\ell \cdot k} \leq 2^{\frac{2k'}{\varepsilon}} \cdot 2^{-\frac{\gamma^2 \ln 2}{3\varepsilon} (k' \ell')}.$$

Recall that we assumed that the width of  $B$  is at most  $2^{ck' \ell' / \varepsilon}$  for some constant  $c < \ln 2/3$ , and that the length of  $B$  is at most  $2^r$ . Recall that we fixed  $\gamma$  such that  $\gamma^2(\ln 2)/3 > c$ . Taking a union bound over at most  $2^r \cdot 2^{ck' \ell' / \varepsilon} \leq 2^{k'} \cdot 2^{ck' \ell' / \varepsilon}$  significant vertices of  $B$ , we conclude that the probability that  $\mathcal{T}$  reaches any significant vertex is at most  $2^{-\Omega(k' \ell' / \varepsilon)}$ . Since we assume that  $k'$  and  $\ell'$  are sufficiently large,  $2^{-\Omega(k' \ell' / \varepsilon)}$  is certainly at most  $2^{-k'}$ , which is at most  $2^{-r}$ .  $\triangleleft$

► **Corollary 19.** *Let  $X, A$  be two finite sets. Let  $M : A \times X \rightarrow \{-1, 1\}$  be a matrix. Assume that  $k, \ell, r \in \mathbb{N}$  are large enough and such that any submatrix of  $M$  of at least  $2^{-k} \cdot |A|$  rows and at least  $2^{-\ell} \cdot |X|$  columns, has a bias of at most  $2^{-r}$ .*

*Then, any learning algorithm for the learning problem corresponding to  $M$  with error parameter  $\varepsilon$ , requires either a memory of size at least  $\Omega\left(\frac{k \cdot \ell}{\varepsilon}\right)$ , or at least  $2^{\Omega(r)}$  samples. The result holds even if the learner has an exponentially small success probability (of  $2^{-\Omega(r)}$ ).*

Corollary follows from the equivalence between  $L_2$ -Extractors and  $L_\infty$ -Extractors (up to constant factors) observed in [8].

---

**References**

---

- 1 Michael Alekhnovich. More on average case vs approximation complexity. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 298–307. IEEE Computer Society, 2003. doi:10.1109/SFCS.2003.1238204.
- 2 Paul Beame, Shayan Oveis Gharan, and Xin Yang. Time-space tradeoffs for learning finite functions from random evaluations, with applications to polynomials. In *Conference On Learning Theory*, pages 843–856, 2018.
- 3 Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003. doi:10.1145/792538.792543.
- 4 Yuval Dagan, Gil Kur, and Ohad Shamir. Space lower bounds for linear prediction in the streaming model. In *Conference on Learning Theory*, pages 929–954. PMLR, 2019.
- 5 Yuval Dagan and Ohad Shamir. Detecting correlations with little memory and communication. In *Conference On Learning Theory*, pages 1145–1198, 2018.
- 6 Wei Dai, Stefano Tessaro, and Xihu Zhang. Super-linear time-memory trade-offs for symmetric encryption. Cryptology ePrint Archive, Report 2020/663, 2020. URL: <https://eprint.iacr.org/2020/663>.
- 7 Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. On agnostic learning of parities, monomials, and halfspaces. *SIAM J. Comput.*, 39(2):606–645, 2009. doi:10.1137/070684914.
- 8 Sumegha Garg, Ran Raz, and Avishay Tal. Extractor-based time-space lower bounds for learning. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 990–1002. ACM, 2018.
- 9 Sumegha Garg, Ran Raz, and Avishay Tal. Time-space lower bounds for two-pass learning. In *34th Computational Complexity Conference (CCC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.
- 10 Uma Girish, Ran Raz, and Wei Zhan. Quantum logspace algorithm for powering matrices with bounded norm. *arXiv preprint arXiv:2006.04880*, 2020.
- 11 Jiabin Guan and Mark Zhandary. Simple schemes in the bounded storage model. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 500–524. Springer, 2019.
- 12 Jiabin Guan and Mark Zhandary. Disappearing cryptography in the bounded storage model. *IACR Cryptol. ePrint Arch.*, 2021:406, 2021.
- 13 Joseph Jaeger and Stefano Tessaro. Tight time-memory trade-offs for symmetric encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 467–497. Springer, 2019.
- 14 Adam Tauman Kalai, Adam R. Klivans, Yishay Mansour, and Rocco A. Servedio. Agnostically learning halfspaces. *SIAM J. Comput.*, 37(6):1777–1805, 2008. doi:10.1137/060649057.
- 15 Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, 1998. doi:10.1145/293347.293351.
- 16 Gillat Kol, Ran Raz, and Avishay Tal. Time-space hardness of learning sparse parities. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1067–1080. ACM, 2017.
- 17 Dana Moshkovitz and Michal Moshkovitz. Mixing implies lower bounds for space bounded learning. In *Conference on Learning Theory*, pages 1516–1566. PMLR, 2017.
- 18 Dana Moshkovitz and Michal Moshkovitz. Entropy samplers and strong generic lower bounds for space bounded learning. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.
- 19 Michal Moshkovitz and Naftali Tishby. Mixing complexity and its applications to neural networks. *arXiv preprint arXiv:1703.00729*, 2017.
- 20 Ran Raz. Fast learning requires good memory: A time-space lower bound for parity learning. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 266–275. IEEE, 2016.



- 21 Ran Raz. A time-space lower bound for a large class of learning problems. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 732–742, 2017.
- 22 Ohad Shamir. Fundamental limits of online and distributed algorithms for statistical learning and estimation. *Advances in Neural Information Processing Systems*, 27:163–171, 2014.
- 23 Vatsal Sharan, Aaron Sidford, and Gregory Valiant. Memory-sample tradeoffs for linear regression with small error. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 890–901, 2019.
- 24 Jacob Steinhardt, Gregory Valiant, and Stefan Wager. Memory, communication, and statistical queries. In *Conference on Learning Theory*, pages 1490–1516. PMLR, 2016.
- 25 Stefano Tessaro and Aishwarya Thiruvengadam. Provable time-memory trade-offs: symmetric cryptography against memory-bounded adversaries. In *Theory of Cryptography Conference*, pages 3–32. Springer, 2018.
- 26 Gregory Valiant and Paul Valiant. Information theoretically secure databases. *arXiv preprint arXiv:1605.02646*, 2016.

## A Omitted Proofs from Section 4

**Proof of Claim 7.** Since  $v$  is not significant,

$$\mathbf{E}_{x' \sim \mathbb{P}_{x|v}} [\mathbb{P}_{x|v}(x')] = \sum_{x' \in X} [\mathbb{P}_{x|v}(x')]^2 = 2^n \cdot \mathbf{E}_{x' \in \mathbb{R}X} [\mathbb{P}_{x|v}(x')] \leq 2^{2\ell} \cdot 2^{-n}.$$

Hence, by Markov's inequality,

$$\Pr_{x' \sim \mathbb{P}_{x|v}} [\mathbb{P}_{x|v}(x') > 2^{2r} \cdot 2^{2\ell} \cdot 2^{-n}] \leq 2^{-2r}.$$

Since conditioned on  $E_v$ , the distribution of  $x$  is  $\mathbb{P}_{x|v}$ , we obtain

$$\Pr_x [x \in \text{Sig}(v) \mid E_v] = \Pr_x [(\mathbb{P}_{x|v}(x) > 2^{2r} \cdot 2^{2\ell} \cdot 2^{-n}) \mid E_v] \leq 2^{-2r}. \quad \blacktriangleleft$$

**Proof of Claim 8.** Since  $v$  is not significant,  $\|\mathbb{P}_{x|v}\|_2 \leq 2^\ell \cdot 2^{-n}$ . Since  $\mathbb{P}_{x|v}$  is a distribution,  $\|\mathbb{P}_{x|v}\|_1 = 2^{-n}$ . Thus,

$$\frac{\|\mathbb{P}_{x|v}\|_2}{\|\mathbb{P}_{x|v}\|_1} \leq 2^\ell \leq 2^{\ell'}.$$

Since  $M$  is a  $(k', \ell')$ - $L_2$ -extractor with error  $2^{-r'}$ , there are at most  $2^{-k'} \cdot |A|$  elements  $\alpha \in A$  with

$$|\langle M_\alpha, \mathbb{P}_{x|v} \rangle| \geq 2^{-r'} \cdot \|\mathbb{P}_{x|v}\|_1 = 2^{-r'} \cdot 2^{-n}$$

The claim follows since  $a_{i+1}$  is uniformly distributed over  $A$  and since  $k' \geq 2r$  (Equation (1)).  $\blacktriangleleft$

**Proof of Claim 9.** Since  $v$  is not significant,

$$\mathbf{E}_{x' \in \mathbb{R}X} [\mathbb{P}_{x|v}(x')] \leq 2^{2\ell} \cdot 2^{-2n}.$$

Hence, for every  $x' \in X$ ,

$$\Pr[x = x' \mid E_v] = \mathbb{P}_{x|v}(x') \leq 2^\ell \cdot 2^{-n/2} \leq 2^{-r}$$

since  $r \leq n/2 - \ell$  (Equation (3)). In particular,  $\Pr[\tilde{x}(v) = x \mid E_v] \leq 2^{-r}$ .  $\blacktriangleleft$

## 60:18 Memory-Sample Lower Bounds for Learning Parity with Noise

**Proof of Claim 11.** Let  $e = (v, u)$  be an edge of  $B$ , labeled by  $(a, b)$ , and such that  $\Pr(e) > 0$ . Since  $\Pr(e) > 0$ , the vertex  $v$  is not significant (as otherwise  $\mathcal{T}$  always stops on  $v$  and hence  $\Pr(e) = 0$ ). Thus,

$$\|\mathbb{P}_{x|v}\|_2 \leq 2^\ell \cdot 2^{-n}.$$

By Claim 10, for any  $x' \in X$ ,

$$\mathbb{P}_{x|e}(x') = \begin{cases} 0 & \text{if } x' \in \text{Sig}(v) \\ \mathbb{P}_{x|v}(x')(1 + 2\varepsilon) \cdot c_e^{-1} & \text{if } x' \notin \text{Sig}(v) \text{ and } M(a, x') = b \\ \mathbb{P}_{x|v}(x')(1 - 2\varepsilon) \cdot c_e^{-1} & \text{if } x' \notin \text{Sig}(v) \text{ and } M(a, x') \neq b \end{cases}$$

where  $c_e$  is a normalization factor that satisfies,

$$c_e \geq 1 - 4 \cdot 2^{-2r} > \frac{1}{2}.$$

(where the last inequality holds because we assume that  $k', \ell', r'$  and thus  $r$  are sufficiently large.) Thus,  $\|\mathbb{P}_{x|e}\|_2 \leq c_e^{-1} \cdot (1 + 2\varepsilon) \|\mathbb{P}_{x|v}\|_2 \leq 4 \cdot 2^\ell \cdot 2^{-n}$ . ◀

**Proof of Claim 12.** Let  $\Gamma_{in}(s)$  be the set of all edges  $e$  of  $B$ , that are going into  $s$ , such that  $\Pr(e) > 0$ . Note that

$$\sum_{e \in \Gamma_{in}(s)} \Pr(e) = \Pr(s).$$

By the law of total probability, for every  $x' \in X$ ,

$$\mathbb{P}_{x|s}(x') = \sum_{e \in \Gamma_{in}(s)} \frac{\Pr(e)}{\Pr(s)} \cdot \mathbb{P}_{x|e}(x'),$$

and hence by Jensen's inequality,

$$\mathbb{P}_{x|s}(x')^2 \leq \sum_{e \in \Gamma_{in}(s)} \frac{\Pr(e)}{\Pr(s)} \cdot \mathbb{P}_{x|e}(x')^2.$$

Summing over  $x' \in X$ , we obtain,

$$\|\mathbb{P}_{x|s}\|_2^2 \leq \sum_{e \in \Gamma_{in}(s)} \frac{\Pr(e)}{\Pr(s)} \cdot \|\mathbb{P}_{x|e}\|_2^2.$$

By Claim 11, for any  $e \in \Gamma_{in}(s)$ ,

$$\|\mathbb{P}_{x|e}\|_2^2 \leq (4 \cdot 2^\ell \cdot 2^{-n})^2.$$

Hence,  $\|\mathbb{P}_{x|s}\|_2^2 \leq (4 \cdot 2^\ell \cdot 2^{-n})^2$ . ◀

**Proof of Claim 16.** By Claim 15,

$$\begin{aligned} \mathcal{Z}'_i &= \sum_{e \in \Gamma_i} \Pr(e) \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k = \sum_{v \in L_{i-1}} \Pr(v) \cdot \sum_{e \in \Gamma_{out}(v)} \frac{\Pr(e)}{\Pr(v)} \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k \\ &\leq \sum_{v \in L_{i-1}} \Pr(v) \cdot \left( \langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^k \cdot (1 + 2^{-r})^k + (2^{-2n+2})^k \right) \\ &= \mathcal{Z}_{i-1} \cdot (1 + 2^{-r})^k + \sum_{v \in L_{i-1}} \Pr(v) \cdot (2^{-2n+2})^k \\ &\leq \mathcal{Z}_{i-1} \cdot (1 + 2^{-r})^k + (2^{-2n+2})^k \end{aligned}$$

◀

**Proof of Claim 17.** For any  $v \in L_i$ , let  $\Gamma_{in}(v)$  be the set of all edges  $e \in \Gamma_i$ , that are going into  $v$ . Note that

$$\sum_{e \in \Gamma_{in}(v)} \Pr(e) = \Pr(v).$$

By the law of total probability, for every  $v \in L_i$  and every  $x' \in X$ ,

$$\mathbb{P}_{x|v}(x') = \sum_{e \in \Gamma_{in}(v)} \frac{\Pr(e)}{\Pr(v)} \cdot \mathbb{P}_{x|e}(x'),$$

and hence

$$\langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle = \sum_{e \in \Gamma_{in}(v)} \frac{\Pr(e)}{\Pr(v)} \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle.$$

Thus, by Jensen's inequality,

$$\langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^k \leq \sum_{e \in \Gamma_{in}(v)} \frac{\Pr(e)}{\Pr(v)} \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k.$$

Summing over all  $v \in L_i$ , we get

$$\begin{aligned} \mathcal{Z}_i &= \sum_{v \in L_i} \Pr(v) \cdot \langle \mathbb{P}_{x|v}, \mathbb{P}_{x|s} \rangle^k \\ &\leq \sum_{v \in L_i} \Pr(v) \cdot \sum_{e \in \Gamma_{in}(v)} \frac{\Pr(e)}{\Pr(v)} \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k \\ &= \sum_{e \in \Gamma_i} \Pr(e) \cdot \langle \mathbb{P}_{x|e}, \mathbb{P}_{x|s} \rangle^k \\ &= \mathcal{Z}'_i. \end{aligned} \quad \blacktriangleleft$$

**Proof of Claim 18.** By Claim 14,  $\mathcal{Z}_0 = (2^{-2n})^k$ . By Claim 16 and Claim 17, for every  $i \in \{1, \dots, m\}$ ,

$$\mathcal{Z}_i \leq \mathcal{Z}_{i-1} \cdot (1 + 2^{-r})^k + (2^{-2n+2})^k.$$

Hence, for every  $i \in \{1, \dots, m\}$ ,

$$\mathcal{Z}_i \leq (2^{-2n+2})^k \cdot (m+1) \cdot (1 + 2^{-r})^{km}.$$

Since  $m = 2^r$ ,

$$\mathcal{Z}_i \leq 2^{-2k \cdot n} \cdot 2^{2k} \cdot (2^r + 1) \cdot e^k \leq 2^{-2k \cdot n} \cdot 2^{4k+2r}. \quad \blacktriangleleft$$



# Testing Hamiltonicity (And Other Problems) in Minor-Free Graphs

Reut Levi  

Efi Arazi School of Computer Science, The Interdisciplinary Center Herzliya, Israel

Nadav Shoshan 

Efi Arazi School of Computer Science, The Interdisciplinary Center Herzliya, Israel

---

## Abstract

In this paper we provide sub-linear algorithms for several fundamental problems in the setting in which the input graph excludes a fixed minor, i.e., is a minor-free graph. In particular, we provide the following algorithms for minor-free unbounded degree graphs.

1. A tester for Hamiltonicity with two-sided error with  $\text{poly}(1/\epsilon)$ -query complexity, where  $\epsilon$  is the proximity parameter.
2. A local algorithm, as defined by Rubinfeld et al. (ICS 2011), for constructing a spanning subgraph with almost minimum weight, specifically, at most a factor  $(1 + \epsilon)$  of the optimum, with  $\text{poly}(1/\epsilon)$ -query complexity.

Both our algorithms use partition oracles, a tool introduced by Hassidim et al. (FOCS 2009), which are oracles that provide access to a partition of the graph such that the number of cut-edges is small and each part of the partition is small. The polynomial dependence in  $1/\epsilon$  of our algorithms is achieved by combining the recent  $\text{poly}(d/\epsilon)$ -query partition oracle of Kumar-Seshadhri-Stolman (ECCC 2021) for minor-free graphs with degree bounded by  $d$ .

For bounded degree minor-free graphs we introduce the notion of *covering partition oracles* which is a relaxed version of partition oracles and design a  $\text{poly}(d/\epsilon)$ -time covering partition oracle for this family of graphs. Using our covering partition oracle we provide the same results as above (except that the tester for Hamiltonicity has one-sided error) for minor-free bounded degree graphs, as well as showing that any property which is monotone and additive (e.g. bipartiteness) can be tested in minor-free graphs by making  $\text{poly}(d/\epsilon)$ -queries.

The benefit of using the covering partition oracle rather than the partition oracle in our algorithms is its simplicity and an improved polynomial dependence in  $1/\epsilon$  in the obtained query complexity.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Streaming, sublinear and near linear time algorithms

**Keywords and phrases** Property Testing, Hamiltonian path, minor free graphs, sparse spanning sub-graphs

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.61

**Category** RANDOM

**Related Version** *Full Version:* <https://arxiv.org/abs/2102.11728>

**Funding** *Reut Levi:* This research was supported by the Israel Science Foundation grant No. 1867/20.

**Acknowledgements** We would like to thank Dana Ron and Oded Goldreich for helpful comments.

## 1 Introduction

The family of minor-free graphs has been at the focus of attention ever since the theory of graph minors began many decades ago and has been drawing much attention in the field of computer science as well. Aside from being an important family that includes natural families of graphs such as planar graphs, it also has the appeal that some hard graph problems become easy when restricted to this family of graphs (e.g. Graph Isomorphism [15]).



© Reut Levi and Nadav Shoshan;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 61; pp. 61:1–61:23



Leibniz International Proceedings in Informatics

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

Minor-free graphs have been extensively studied also in the realm of sublinear algorithms and in particular property testing (see e.g. [4, 5, 29, 21, 17, 3, 10, 18, 7, 6, 19, 20]). In particular, in the general-graph model [30] where there is much less body of work, compared to the bounded-degree graph model [11] and the dense graph model [13], these graphs draw attention as they allow for better characterization compared to general unbounded degree graphs. A notable example is the result by Czumaj and Sohler [7] who recently gave a full characterization of properties that can be tested with one-sided error with query complexity that is independent of the size of the graph (i.e. *testable*). They showed that the latter is possible if and only if testing the property can be reduced to testing for a finite family of finite forbidden subgraphs. This raises the question regarding the testability of properties that can not be reduced to testing for a finite family of finite forbidden subgraphs when we allow the tester to have two-sided error. A well known example of such property is the property of being Hamiltonian.

Another question, which is also relevant for bounded degree graphs, is whether we can obtain algorithms with query complexity which is only polynomial in  $1/\epsilon$  where  $\epsilon$  is the proximity parameter. Newman and Sohler [29] showed that any property of bounded degree hyperfinite graphs and in particular minor-free graphs is testable. Their algorithm learns the graph up to modifications of  $\epsilon dn$  edges with query complexity which is super-polynomial in  $d$  and  $\epsilon$ . While this approach works for all properties of graphs, more efficient testers can be obtained for specific properties of graphs. In particular, properties of graphs which are monotone and additive can be tested by using  $O(d/\epsilon)$  queries to a partition oracle, a tool introduced by Hassidim et al. [14]. Thus, an implication of the recent  $\text{poly}(d/\epsilon)$ -query partition oracle of Kumar-Seshadhri-Stolman [20] is that monotone and additive properties are testable with  $\text{poly}(d/\epsilon)$ -queries. Thus the question of designing testers with  $\text{poly}(1/\epsilon)$ -queries remains open for properties which are not monotone or not additive, as the property of being Hamiltonian.

An additional motivation for studying Hamiltonicity in minor-free graphs is, as shown by Yoshida-Ito [33] and Goldreich [12], that it can not be tested with sublinear query complexity in general bounded degree graphs.

## 1.1 Our Results

All our algorithms work under the promise that the input graph is minor-free.

### 1.1.1 Testing Hamiltonicity

In the general graph model, we provide an algorithm for approximating the distance from Hamiltonicity up to an additive error of  $\epsilon n$  where  $n$  denotes the number of vertices in the input graph with query complexity which is  $\text{poly}(1/\epsilon)$  and time complexity which is exponential in  $\text{poly}(1/\epsilon)$ . This also implies a tolerant tester with two-sided error for testing Hamiltonicity with the same complexities.

In the bounded-degree graph model, we provide an algorithm for testing Hamiltonicity with one-sided error with query complexity which is  $\text{poly}(d/\epsilon)$ , where  $d$  denotes the bound on the degree, and time complexity which is exponential in  $\text{poly}(d/\epsilon)$ .

### 1.1.2 Local algorithm for constructing spanning subgraphs of almost optimum weight

In the general graph model, we provide a local algorithm for constructing a sparse spanning subgraph of weight at most  $(1 + \epsilon)\text{OPT}$  where  $\text{OPT}$  denotes the weight of the MST of the input graph. The algorithm receives as parameters  $\epsilon$  and an upper bound,  $W$ , on the maximum weight of an edge in the graph. Moreover, the number of edges of the output graph that do not belong to the MST of the input graph<sup>1</sup> is  $O(\epsilon n/W)$ . The query complexity and time complexity of the algorithm is  $\text{poly}(W/\epsilon)$ . We note that in addition to incidence queries our algorithm also use random neighbour queries.

In the bounded-degree graph model, we provide a simpler algorithm with the same guarantees whose query complexity and time complexity is  $\text{poly}(dW/\epsilon)$ , where the polynomial of the complexity is somewhat improved compared to the algorithm for graphs of unbounded degree.

### 1.1.3 Testing monotone and additive properties of graphs

We prove that any property which is monotone (closed under removal of edges and vertices) and additive (closed under the disjoint union of graphs) can be tested in the bounded degree model with  $\text{poly}(d/\epsilon)$ -query complexity under the promise that the input graph is minor-free. The same result was recently shown independently by Kumar-Seshadhri-Stolman [20]. While in [20] they use partition oracles in the proof, we use a relaxed notion of partition oracles (which we introduce in this paper) and consequently obtain a somewhat improved polynomial dependence in the query complexity and a simpler algorithm.

## 1.2 Our algorithms for minor-free unbounded degree graphs

### 1.2.1 Testing Hamiltonicity

We begin by proving that the distance from Hamiltonicity of any graph  $G = (V, E)$  equals the size of the minimum path cover of  $G$  minus 1, where a path cover of a graph is a set of disjoint paths such that each  $v \in V$  belongs to exactly one path (see Claim 11).

We then prove that if we remove  $O(\epsilon|V|)$  edges from  $G$  as well as edges that are incident to  $O(\epsilon|V|)$  vertices in  $G$  then the distance from Hamiltonicity may be increased by (at most)  $O(\epsilon|V|)$  (see Claims 12 and 13).

Thus, in order to obtain an approximation, with an additive error of  $O(\epsilon|V|)$ , to the size of the minimum path cover of  $G$  (and hence to its distance from Hamiltonicity) it suffices to obtain such approximation to the size of the minimum path cover of  $\hat{G}$  where  $\hat{G}$  is defined as follows. We obtain  $\hat{G}$  from  $G$  by first removing the edges incident to vertices of high degree, which we refer to as *heavy* vertices, then running the partition oracle on the resulting graph and then removing the cut-edges of the partition. An approximation to the size of the minimum path cover of  $\hat{G}$  can be obtained by sampling vertices u.a.r. from  $V$  and computing the size of the minimum path cover of their connected component in  $\hat{G}$ .

Since we obtain an approximation algorithm for the distance from being Hamiltonian we also obtain a tolerant tester with two-sided error for this property.

<sup>1</sup> Without loss of generality we assume that the weights of the edges are distinct and hence that there is a unique MST.

### 1.2.2 Constructing spanning subgraphs with almost optimum weight

We present our algorithm as a global algorithm and prove its correctness. Thereafter, we describe the local implementation of this global algorithm.

Our global algorithm proceeds as follows. In the first step, the algorithm adds all the edges between *heavy* vertices to the edges of the constructed spanning subgraph,  $E'$ , where a heavy vertex is defined to be a vertex of degree greater than some threshold.

It then runs the partition oracle on the graph induced on the vertices that are not heavy, i.e., *light* vertices and adds all the cut-edges of the partition to  $E'$ .

In the second step, each part of the partition is partitioned into subparts by running a controlled variant of Borůvka's algorithm for finding an MST on each part independently. The edges spanning the sub-parts are then added to  $E'$ . Then, for each each sub-part, the algorithm adds a single edge to a single heavy vertex which is adjacent to the sub-part (assuming there is one).

We prove that all the edges added in the second step belong to the minimum spanning forest (MSF) of a graph which is  $O(\epsilon/W_G)$ -close to  $G$ , where  $W_G$  denotes the maximum weight of an edge in  $G$ . Additionally we prove that if we remove  $O(\epsilon|V|/W_G)$  edges from  $G$ , then the weight of the minimum spanning forest (MSF) may increase by (at most)  $O(\epsilon|V|)$ .

The second step partitions the vertices of the graph into *clusters* and *isolated parts*, where isolated parts are parts that are not adjacent to any heavy vertex, and the clusters are defined as follows. Each cluster contains a single heavy vertex, which we refer to as the *center* of the cluster and sub-parts that are adjacent in the constructed graph to the center (each sub-part is adjacent to at most a single center).

In the third step the algorithm adds edges to  $E'$  between pairs of cluster that are adjacent to each other. For every edge  $\{u, v\}$  which is adjacent to two different clusters,  $A$  and  $B$ , the algorithm runs another algorithm that samples edges incident to  $A$  and  $B$  and returns the lightest one. The edge  $\{u, v\}$  is added to  $E'$  if it is lighter than the returned edge.

We note that the algorithm that samples edges incident to a pair of specific clusters,  $A$  and  $B$  may not sample sufficient number of edges or may not return any edge (this is likely when the degree of both centers is large compared to the number of edges which are incident to both  $A$  and  $B$ ). In the analysis which is adapted from [28], we show that nonetheless, w.h.p. the number of edges added in the third step is sufficiently small. The main idea is to consider the graph in which each cluster is contracted into a single vertex and then to analyse the sampling algorithm with respect to this graph which is also minor-free and hence has bounded arboricity<sup>2</sup>. The bounded arboricity of the contracted graph ensures that w.h.p. the sampling algorithm samples enough edges which are incident to  $A$  and  $B$  as long as the cut between these clusters is sufficiently large. On the other hand, if this is not the case then we show that we can afford to add to  $E'$  all the edges in the cut.

The local implementation of the above-mention global algorithm is quite straightforward and is presented in Section B.7.

## 1.3 Our algorithms for minor-free bounded degree graphs

### 1.3.1 Covering partition oracles

We introduce a relaxed version of partition oracles which we call *covering partition oracles* and design such an oracle for minor-free graphs with query complexity  $\text{poly}(d/\epsilon)$ . Given query access to a graph  $G = (V, E)$  and parameter  $\epsilon$  a partition oracle provides access to

---

<sup>2</sup> The arboricity of a graph is the minimum number of forests into which its edges can be partitioned.



a partition of  $V$ ,  $\mathcal{P}$ , such that the size of each part of  $\mathcal{P}$  is small (usually  $\text{poly}(1/\epsilon)$ ), the number of cut-edges of  $\mathcal{P}$  is at most  $\epsilon|V|$  (w.h.p.) and  $\mathcal{P}$  is determined only by  $G$  and the randomness of the oracle. On query  $v \in V$  the oracle returns the part of  $v$  in  $\mathcal{P}$ .

A covering partition oracle has the same guarantees only that the requirement to return the part of  $v$  on query  $v \in V$  is relaxed as follows. On query  $v \in V$  the oracle is required to return a (small) subset  $S$  such that  $S$  contains the part of  $v$  in  $\mathcal{P}$ .

Our covering partition oracle builds on a central theorem from the recent work of [19]. The theorem states that for any minor-free bounded degree graph there exists a partition of the graph into small parts with small number of cut-edges such that for each part of the partition,  $P$ , there exists a vertex,  $s \in V$ , such that if we perform sufficiently many (polynomial in  $1/\epsilon$ ) lazy random walks from  $s$  then w.h.p. we encounter all the vertices in  $P$ . Building on this theorem we prove that the simple algorithm that on query  $v \in V$  performs a set of lazy random walks from  $v$  (of different lengths) and then performs a set of lazy random walks from each endpoint of these walks is a covering partition oracle.

The algorithms described in Subsections 1.3.2-1.3.4, use our covering partition oracle.

We note that since a partition oracle is a special case of a covering partition oracle (with stronger guarantees) all our algorithms for bounded degree graphs also work when one replaces calls to the covering partition oracle by calls to a partition oracle.

As mentioned above, the use of covering partition oracles has two benefits. The first benefit is that the implementation of the covering partition oracles is much simpler and the second benefit is that the query complexity per oracle query is better (though both our covering partition oracle and the partition oracle of [20] have query complexity which is  $\text{poly}(d/\epsilon)$ ), which consequently affects the query complexity of the algorithms.

Another conceptual benefit in introducing covering partition oracles is that for some families of graphs the gap in the query complexity can be more dramatic. To give a concrete example consider  $(\epsilon, \rho(\epsilon))$ -hyperfinite graphs<sup>3</sup>. It is straightforward to obtain a covering partition oracle with query complexity  $O(d^{\rho(\epsilon)})$  for this family of graphs while the best known partition oracle for this family has query complexity which is  $O(2^{d^{\rho(c\epsilon^3)}})$  [14], where  $c$  is some constant. We note that all our algorithms for bounded degree graphs work for any family of graphs for which there is a covering partition oracle (including  $(\epsilon, \rho(\epsilon))$ -hyperfinite graphs).

### 1.3.2 Testing Hamiltonicity

In addition to relating the distance from Hamiltonicity of any graph  $G = (V, E)$  to the size of its minimum path cover, as mentioned above, we also prove that given a subset  $S \subset V$ , if the size of the minimum path cover of  $G[S]$  is greater than the number of edges in the cut of  $S$  and  $V \setminus S$  then  $G$  is not Hamiltonian. Using this claim it becomes straightforward to design a one-sided error tester for Hamiltonicity that uses  $O(1/\epsilon)$  queries to a partition oracle. We prove that it suffices to use the same number of queries to the covering partition oracle. We note that in this case there is a trade-off between the query complexity and time complexity. In particular while using the covering partition oracle rather than the partition oracle results in a better polynomial dependence of the query complexity it also results in a worse polynomial dependence in the exponent of the running time (in both cases the running time is exponential in  $\epsilon^{-1}$  since we find the size of the minimum path cover by brute force<sup>4</sup>).

<sup>3</sup> Let  $\rho$  be a function from  $\mathbb{R}_+$  to  $\mathbb{R}_+$ . A graph  $G = (V, E)$  is  $(\epsilon, \rho(\epsilon))$ -hyperfinite if for every  $\epsilon > 0$  it is possible to remove  $\epsilon|V|$  edges of the graph such that the remaining graph has connected components of size at most  $\rho(\epsilon)$ .

<sup>4</sup> Finding the minimum path cover is APX-hard since (as noted by Chandra Chekuri at stackexchange.com [16]) we can reduce the TSP-path problem in metrics with distances 1 and 2 to it. The latter problem is APX-hard [9]).

### 1.3.3 Constructing spanning subgraphs with almost optimum weight

As mentioned-above, given a weighted graph  $G = (V, E, w)$  if we remove  $O(\epsilon|V|/W_G)$  edges from  $G$  then the weight of the MSF of the resulting graph may increase by at most  $O(\epsilon|V|)$  compared to the weight of  $G$ . Thus given access to a partition of  $V$  such that the subgraph induced on each part is connected and the number of cut-edges is  $O(\epsilon|V|/W_G)$  we can proceed as follows. For each part of the partition we add to  $E'$  the edges of the MST of the subgraph induced on this part. In addition, we add to  $E'$  the cut-edges of the partition. Consequently, the total weight of the edges in  $E'$  is greater than the weight of the MST of  $G$  by at most  $O(\epsilon|V|/W_G)$ . Hence, if on query  $\{u, v\}$  we query the partition oracle on  $u$  and  $v$  then it is possible to determine whether  $\{u, v\} \in E'$  where  $E'$  is constructed as described above with respect to the partition of the oracle. We prove that the same approach works when we preform the same queries to the covering partition oracle.

### 1.3.4 Testing monotone and additive properties

One of the main applications of the partition oracle is a general reduction for testing monotone and additive properties of bounded degree minor-free graphs. The idea of the reduction (from testing to the partition oracle) is to sample  $O(d/\epsilon)$  vertices and for each vertex  $v$  in the sample to test whether the subgraph induced on the part of  $v$  has the properties. The tester accepts iff all sampled parts pass the test. We prove that the same reduction works when we replace the queries to the partition oracle by queries to the covering partition oracle.

## 1.4 Organization

Due to space limitations, an extended section on related work as well as omitted proofs and details of Section 3 appear in the appendix.

## 2 Preliminaries

In this section we introduce several definitions and some known results that will be used in the following sections. Unless stated explicitly otherwise, we consider simple graphs, that is, with no self-loops and no parallel edges.

Let  $G = (V, E)$  be a graph over  $n$  vertices. Each vertex  $v \in V$  has an id,  $id(v)$ , where there is a full order over the ids.

The total order over the vertices induces a total order (ranking)  $\rho$  over the edges of the graph in the following straightforward manner:  $\rho((u, v)) < \rho((u', v))$  if and only if  $\min\{u, v\} < \min\{u', v'\}$  or  $\min\{u, v\} = \min\{u', v'\}$  and  $\max\{u, v\} < \max\{u', v'\}$  (recall that  $V = [n]$ ). Thus, given a weighted graph, we may assume without loss of the generality that the weights of the edges are unique by breaking ties according to the order over the edges.

For a subset of vertices  $X$ , we let  $G[X]$  denote the subgraph of  $G$  induced by  $X$ .

When we consider bounded degree graphs, we consider the bounded-degree graph model [11]. The graphs we consider have a known degree bound  $d$ , and we assume we have query access to their incidence-lists representation. Namely, for any vertex  $v$  and index  $1 \leq i \leq d$  it is possible to obtain the  $i^{\text{th}}$  neighbor of  $v$  (where if  $v$  has less than  $i$  neighbors, then a special symbol is returned). If the graph is edge-weighted, then the weight of the edge is returned as well. When we consider graphs with unbounded degree we consider the general graph model [30] equipped with an additional type of query: random neighbor query. Namely, when we query any given vertex  $v$  a random neighbor of  $v$  is returned <sup>5</sup>.

<sup>5</sup> We note that we do not use the random neighbor query in our tester for Hamiltonicity.

For a graph  $G = (V, E)$  and two sets of vertices  $V_1, V_2 \subseteq V$ , we let  $E^G(V_1, V_2)$  denote the set of edges in  $G$  with one endpoint in  $V_1$  and one endpoint in  $V_2$ . That is  $E(V_1, V_2) \stackrel{\text{def}}{=} \{(v_1, v_2) \in E : v_1 \in V_1, v_2 \in V_2\}$ . If  $G$  is clear from the context we may omit it from the notation.

A tester with two-sided error for a property  $\mathcal{P}$  receives a parameter  $\epsilon$  and query access to a graph  $G = (V, E)$  and has the following guarantees. If  $G \in \mathcal{P}$  then the tester accepts with probability at least  $2/3$ . If  $G$  is  $\epsilon$ -far<sup>6</sup> from  $\mathcal{P}$  then the tester rejects with probability at least  $2/3$ . A tester with one-sided error accepts  $G \in \mathcal{P}$  with probability 1.

## 2.1 Partition oracles and covering partition oracles

► **Definition 1.** For  $\epsilon \in (0, 1]$ ,  $k \geq 1$  and a graph  $G = (V, E)$ , we say that a partition  $\mathcal{P} = (V_1, \dots, V_t)$  of  $V$  is an  $(\epsilon, k)$ -partition (w.r.t.  $G$ ), if the following conditions hold:

1. For every  $1 \leq i \leq t$  it holds that  $|V_i| \leq k$ ;
2. For every  $1 \leq i \leq t$  the subgraph induced by  $V_i$  in  $G$  is connected;
3. The total number of edges whose endpoints are in different parts of the partition is at most  $\epsilon|V|$  (that is,  $|\{(v_i, v_j) \in E : v_i \in V_j, v_j \in V_j, i \neq j\}| \leq \epsilon|V|$ ).

Let  $G = (V, E)$  be a graph and let  $\mathcal{P}$  be a partition of  $V$ . We denote by  $g_{\mathcal{P}}$  the function from  $v \in V$  to  $2^V$  (the set of all subsets of  $V$ ), that on input  $v \in V$ , returns the subset  $V_{\ell} \in \mathcal{P}$  such that  $v \in V_{\ell}$ . We denote the set of cut-edges of  $\mathcal{P}$  by  $E_{\mathcal{P}}^G = \{(u, v) \in E : g_{\mathcal{P}}(v) \neq g_{\mathcal{P}}(u)\}$  (we may omit  $G$  from the notation when it is clear from the context).

► **Definition 2** ([14]). An oracle  $\mathcal{O}$  is a partition oracle if, given query access to the incidence-lists representation of a graph  $G = (V, E)$ , the oracle  $\mathcal{O}$  provides query access to a partition  $\mathcal{P} = (V_1, \dots, V_t)$  of  $V$ , where  $\mathcal{P}$  is determined by  $G$  and the internal randomness of the oracle. Namely, on input  $v \in V$ , the oracle returns  $g_{\mathcal{P}}(v)$  and for any sequence of queries,  $\mathcal{O}$  answers consistently with the same  $\mathcal{P}$ . An oracle  $\mathcal{O}$  is an  $(\epsilon, k)$ -partition oracle with respect to a class of graphs  $\mathcal{C}$  if the partition  $\mathcal{P}$  it answers according to has the following properties.

1. For every  $V_{\ell} \in \mathcal{P}$ ,  $|V_{\ell}| \leq k$  and the subgraph induced by  $V_{\ell}$  in  $G$  is connected.
2. If  $G$  belongs to  $\mathcal{C}$ , then  $|E_{\mathcal{P}}| \leq \epsilon|V|$  with high constant probability, where the probability is taken over the internal coin flips of  $\mathcal{O}$ .

We consider the following relaxation of Definition 2.

► **Definition 3.** An oracle  $\mathcal{O}$  is a covering partition oracle if, given query access to the incidence-lists representation of a graph  $G = (V, E)$ , the oracle  $\mathcal{O}$ , on input  $v \in V$ , returns a subset  $S \subseteq V$  such that  $g_{\mathcal{P}}(v) \subseteq S$  where  $\mathcal{P}$  is a partition of  $V$  determined by  $G$  and the internal randomness of the oracle. For any sequence of queries,  $\mathcal{O}$  answers consistently according to the same  $\mathcal{P}$ . An oracle  $\mathcal{O}$  is an  $(\epsilon, k)$ -covering partition oracle with respect to a class of graphs  $\mathcal{C}$  if the following conditions hold.

1. On every query, the subgraph induced by the subset returned by  $\mathcal{O}$ ,  $S$ , is connected and  $|S| \leq k$ .
2. If  $G$  belongs to  $\mathcal{C}$ , then w.h.p.,  $|E_{\mathcal{P}}| \leq \epsilon|V|$ , where the probability is taken over the internal coin flips of  $\mathcal{O}$ .

<sup>6</sup> In the bounded degree model, a graph  $G = (V, E)$  is said to be  $\epsilon$ -far from a property  $\mathcal{P}$  if for every graph  $G' = (V, E') \in \mathcal{P}$  of maximum degree  $d$  it holds that the symmetric difference between  $E$  and  $E'$  has cardinality which is greater than  $\epsilon \cdot d|V|/2$ . In the general graph model, a graph  $G = (V, E)$  is said to be  $\epsilon$ -far from a property  $\mathcal{P}$  if for every graph  $G' = (V, E') \in \mathcal{P}$  it holds that the symmetric difference between  $E$  and  $E'$  has cardinality which is greater than  $\epsilon \cdot \max\{|E|, |E'|\}$ .

## 2.2 Graph minors

Recall that a graph  $R$  is called a *minor* of a graph  $G$  if  $R$  is isomorphic to a graph that can be obtained by zero or more edge contractions on a subgraph of  $G$ . A graph  $G$  is  *$R$ -minor-free* if  $R$  is not a minor of  $G$ . We next quote two results that will play a central role in this work.

► **Fact 4.** *Let  $R$  be a fixed graph with  $r$  edges. For every  $R$ -minor-free graph  $G = (V, E)$  it holds that:*

1.  $|E| \leq r \cdot |V|$ ;
2.  $E$  can be partitioned into at most  $r$  forests.

Unless stated otherwise, in all our algorithms, we assume that the input graph is  $R$ -minor-free graph where  $R$  is a fixed graph with  $r$  edges (we could receive  $r$  as a parameter but we make this assumption for the sake of brevity).

## 2.3 Hamiltonian path and minimum path cover

► **Definition 5** (Hamiltonian path). *A Hamiltonian path in  $G = (V, E)$  is a path between two vertices of  $G$  that visits each vertex of  $G$  exactly once.*

► **Definition 6** (minimum path cover). *Given an undirected graph  $G = (V, E)$ , a path cover is a set of disjoint paths such that every vertex  $v \in V$  belongs to exactly one path. The minimum path cover of  $G$  is a path cover of  $G$  having the least number of paths.*

## 2.4 Local algorithms for constructing sparse spanning subgraphs

► **Definition 7** ([28]). *An algorithm  $\mathcal{A}$  is a local sparse spanning graph (LSSG) algorithm if, given  $n \geq 1$ ,  $\epsilon > 0$ , and query access to the incidence-lists representation of a connected graph  $G = (V, E)$  over  $n$  vertices, it provides oracle access to a subgraph  $G' = (V, E')$  of  $G$  such that:*

1.  $G'$  is connected.
2.  $|E'| \leq (1 + \epsilon) \cdot n$  with high constant probability<sup>7</sup>, where  $E'$  is determined by  $G$  and the internal randomness of  $\mathcal{A}$ .

More specifically, on query  $u, v \in E$ ,  $\mathcal{A}$  returns whether  $(u, v) \in E'$ , and for any sequence of edges,  $\mathcal{A}$  answers consistently with the same  $G'$ .

An algorithm  $\mathcal{A}$  is an LSSG algorithm for a family of graphs  $\mathcal{C}$  if the above conditions hold, provided that the input graph  $G$  belongs to  $\mathcal{C}$ .

► **Definition 8** ([26]). *A local algorithm for  $(1 + \epsilon)$ -approximating the minimum weight spanning graph of a graph  $G = (V, E, w)$  with positive weights and  $\min_{e \in E} w(e) \geq 1$ , is a local algorithm for  $(1 + \epsilon)$ -sparse spanning graph of  $G = (V, E, w)$  for which the following holds:  $\sum_{e \in E'} w(e) \leq (1 + \epsilon)\alpha$ , where  $\alpha$  is the weight of a minimum weight spanning tree of  $G$ .*

For a graph  $G = (V, E, w)$  we define  $W_G = \max_{e \in E} w(e)$  (when it is clear from the context, we sometimes omit the subscript  $G$ ). We denote by  $\text{MSF}(G)$  the set of edges the minimum-spanning-forest of  $G$  (as mentioned above we assume without loss of generality that all weights are distinct and thus the minimum-spanning-forest is unique). For a connect weighted graph we denote by  $\text{MST}(G)$  the set of edges the minimum-spanning-forest of  $G$ . For a subset of edges  $S \subseteq E$ , we define  $w(S) \stackrel{\text{def}}{=} \sum_{e \in S} w(e)$ .

<sup>7</sup> In some papers the required success probability is high, i.e. at least  $1 - 1/\Omega(n)$ .

Our algorithms build on the following rules.

1. The *cut rule* states that for any cut of the graph (a cut is a partition of the vertices into two sets), the lightest edge that crosses the cut must be in the MST.
2. The *cycle rule* states that if we have a cycle, the heaviest edge on that cycle cannot be in the MST.

### 3 Algorithms for minor-free graphs with unbounded degrees

#### 3.1 Testing Hamiltonicity

In this section we prove the following Theorem.

► **Theorem 9.** *Given query access to an input graph  $G = (V, E)$  where  $G$  is a minor-free unbounded degree graph and parameters  $\epsilon$  and  $|V|$ , there exists an algorithm that accepts  $G$  with probability at least  $2/3$  if  $G$  is  $\epsilon/2$ -close to being Hamiltonian and rejects  $G$  with probability at least  $2/3$  if  $G$  is  $\epsilon$ -far from being Hamiltonian. The query complexity of the algorithm is  $\text{poly}(1/\epsilon)$  and the running time is exponential in  $\text{poly}(1/\epsilon)$ .*

Theorem 9 is a direct consequence of following claim (which is proved in the sequel).

▷ **Claim 10.** Given an input graph  $G = (V, E)$  which is a minor-free graph, and parameter  $\epsilon$ , Algorithm 1 outputs a value  $x$  such that with high constant probability  $\delta_{\text{HAM}}(G) - \epsilon|V| \leq x \leq \delta_{\text{HAM}}(G) + \epsilon|V|$ .

We next state a couple of claims which we use in the proof of Claim 10.

▷ **Claim 11.** Let  $G = (V, E)$  be a graph and let  $k$  be the size of a minimum path cover of  $G$ . Then the distance of  $G$  for being Hamiltonian,  $\delta_{\text{HAM}}(G)$ , is  $k - 1$ .

▷ **Claim 12.** Let  $G = (V, E)$  be a graph and let  $F \subseteq E$  be a subset of edges. Then

$$\delta_{\text{HAM}}(G) \leq \delta_{\text{HAM}}(G') \leq \delta_{\text{HAM}}(G) + |F|,$$

where  $G' = (V, E')$  and  $E' = E \setminus F$ .

▷ **Claim 13.** Let  $G = (V, E)$  be a graph and let  $S \subseteq V$  be a subset of vertices. Then

$$\delta_{\text{HAM}}(G) \leq \delta_{\text{HAM}}(G') \leq \delta_{\text{HAM}}(G) + 2|S|,$$

where  $G' = (V, E')$  and  $E'$  is the set of edges in  $E$  that are not incident to vertices in  $S$ .

Proof of Claim 10. Let  $\mathcal{P}$  denote the partition for which the partition oracle executed in Step 3(a)i answers according to. With high constant probability  $|E_{\mathcal{P}}| \leq \frac{\epsilon|V|}{4}$ . Let  $E_1$  denote this event.

Let  $G' = (V, E')$  be the graph such that  $E'$  is the set of edges that are not incident to vertices in  $H$  and are not in  $E_{\mathcal{P}}$ . By Claims 12 and 13,

$$\delta_{\text{HAM}}(G) \leq \delta_{\text{HAM}}(G') \leq \delta_{\text{HAM}}(G) + |E_{\mathcal{P}}| + 2|H|.$$

By Markov's inequality and Fact 4,  $|H| \leq \frac{\epsilon|V|}{4}$ . We prove that, conditioned that  $E_1$  occurs, Algorithm 1 outputs with high constant probability a  $(1 + \epsilon)$ -approximation to  $\delta_{\text{HAM}}(G')$ .

For each  $v \in V$  define the random variable  $x_v$  as defined in Step 3 of Algorithm 1. Let  $T \in \mathcal{P}$  be a part in  $\mathcal{P}$ , then  $\sum_{v \in T} x_v = k$  where  $k$  is the minimum path cover of  $G[T]$ . Thus  $\sum_{v \in V} x_v$  is the minimum path cover of  $G'$ . Since for every  $v \in V$ ,  $x_v \in (0, 1]$ , it follows by

## 61:10 Testing Hamiltonicity (And Other Problems) in Minor-Free Graphs

■ **Algorithm 1** Approximating the distance to Hamiltonicity in minor-free, unbounded degree, graphs.

**Input:** Oracle access to a minor-free, unbounded-degree, graph  $G = (V, E)$

**Output:**  $(1 + \epsilon)$ -approximation to  $\delta_{\text{HAM}}(G)$

1. Define  $\Delta \stackrel{\text{def}}{=} 8c(h)/\epsilon$ ,  $H$  to be the set of vertices of degree greater than  $\Delta$ , and  $L \stackrel{\text{def}}{=} V \setminus H$ .
2. Sample a set  $S$  of  $y = \Theta(1/\epsilon^2)$  vertices u.a.r.
3. For each vertex  $v \in S$ :
  - a. If  $v \in L$  then:
    - i. Query the partition oracle on  $v$  with parameter  $\epsilon/4$  w.r.t. the graph  $G[L]$ .  
Let  $S_v$  denote the returned set.
    - ii. Set  $x_v = k/|S_v|$  where  $k$  is the size of the minimum path cover of  $G[S_v]$ .
  - b. Otherwise, set  $x_v = 1$ .
4. Output  $\frac{\sum_{v \in S} x_v}{|S|} \cdot |V|$ .

the additive Chernoff's bound that with high constant probability  $\left| \frac{\sum_{v \in S} x_v}{|S|} - \frac{\sum_{v \in V} x_v}{|V|} \right| \leq \frac{\epsilon}{4}$ .  
Thus, with high constant probability,

$$\delta_{\text{HAM}}(G) - \frac{\epsilon|V|}{4} \leq \frac{\sum_{v \in S} x_v}{|S|} \cdot |V| \leq \delta_{\text{HAM}}(G) + \epsilon|V|,$$

as desired. ◁

### 3.2 A Local algorithm for constructing a spanning subgraph with almost optimum weight

In this section we prove the following theorem.

► **Theorem 14.** *There exists a local algorithm for  $(1 + \epsilon)$ -approximating the minimum weight spanning graph for the family of unbounded degree minor-free graphs, with positive weights and minimum weight which is at least 1. The query complexity and time complexity of the algorithm is  $\text{poly}(W/\epsilon)$  where  $W$  is an upper bound on the maximum weight. The algorithm receives  $\epsilon$  and  $W$  as parameters.*

▷ **Claim 15.** Let  $G = (V, E, w)$  be a weighted graph and let  $G' = (V, E', w)$  be a graph such that  $E' = E \setminus S$  where  $S \subseteq E$ . Then  $w(\text{MSF}(G')) \leq w(\text{MSF}(G)) + |S|W_G$ . Moreover,  $|\text{MSF}(G') \setminus \text{MSF}(G)| \leq |S|$ .

We next describe our algorithm from a global point of view.

#### 3.2.1 The global algorithm

Our global algorithm, which is listed in Algorithm 2, proceeds as follows. In Step 2, the algorithm adds all the edges between *heavy* vertices to  $E'$  where a heavy vertex is defined to be a vertex of degree greater than  $\Delta \stackrel{\text{def}}{=} 6r^2W/\epsilon$ .

It then runs the partition oracle on the graph induced on the *light* vertices, namely vertices that are not heavy, and adds all the cut-edges of the partition,  $\mathcal{P}$ , to  $E'$  (Step 4).

■ **Algorithm 2** Global algorithm for approximated-MST in unbounded-degree minor-free graphs.

**Input:** parameters  $\epsilon$  and  $W$  and access to a minor-free graph  $G = (V, E)$ .

**Output:**  $G = (V, E')$  which is an approximated-MST of  $G$ .

1. Let  $H$  denote the set of all vertices of degree greater than  $\Delta$  and let  $L = V \setminus H$ .
2. Add all the edges of  $G[H]$  to  $E'$ .
3. Run the partition oracle with parameter  $\epsilon/(6W)$  on all vertices in  $G[L]$ . Let  $\mathcal{P}$  denote the resulting partition.
4. Add all the edges in  $E_{\mathcal{P}}$  to  $E'$ .
5. For each  $S \in \mathcal{P}$ :
  - a. Run Algorithm 4 and let  $F = (S, A)$  denote the returned graph.
  - b. Add the edges in  $A$  to  $E'$ .
  - c. For each connected component of  $F$ ,  $C$ :
    - i. Add to  $E'$  the lightest edges which is adjacent to  $C$  and  $H$  (if such edge exists).
6. For each  $v \in H$ , define the *cluster* of  $v$ , denoted by  $\mathcal{C}(v)$ , to be subset of vertices that contains  $v$  and all the vertices in sub-parts  $B$  such that there exists an edge in  $E'$  that is incident to  $v$  and a vertex in  $B$ .  $v$  is referred to as the *center* of the cluster.
7. For each edges  $\{u, v\} \in E$  such that  $u$  and  $v$  belong to different clusters:
  - a. Run Algorithm 3 and add  $\{u, v\}$  to  $E'$  if it is lighter than the edge returned by the algorithm or if it returned null.
8. Return  $G' = (V, E')$ .

Step 5 consists of two parts. In Sub-step 5a the algorithm runs Algorithm 4 on each part in  $\mathcal{P}$ . Algorithm 4 partitions the parts into connected sub-parts by performing a controlled variant of Borůvka's algorithm. The edges added in Sub-step 5a are the edges that span the sub-parts. In Sub-step 5c, for each sub-part, the algorithm adds a single edge to a single vertex in  $H$  which is adjacent to the sub-part (assuming there is one).

This partitions the vertices of the graph into *clusters* and *isolated parts*, namely, parts in  $\mathcal{P}$  that are not adjacent to any vertex in  $H$ . Each cluster contains a single heavy vertex, which we refer to as the *center* of the cluster and sub-parts that are connected by a single edge to the center.

In Step 7 the algorithm adds edges to  $E'$  between pairs of clusters that are adjacent to each other in  $G' = (V, E \setminus E_{\mathcal{P}})$ . For every edge  $\{u, v\}$  which is adjacent to two different clusters,  $A$  and  $B$ , the algorithm runs Algorithm 3 which samples edges incident to  $A$  and  $B$  and returns the lightest one. The edge  $\{u, v\}$  is added to  $E'$  if it is lighter than the edge returned by Algorithm 3 (or if the algorithm returned null).

We note that we do not have direct access to uniform samples of edges incident to a pair of specific clusters,  $A$  and  $B$ . In fact, it could be the case that Algorithm 3 does not return any edge (this is likely when the degree of both centers is large compared to the number of edges which are incident to both  $A$  and  $B$ ). By adapting the analysis in [28], we show that nonetheless, w.h.p. the number of edges added in Step 7 is sufficiently small.

This concludes the description of the global algorithm. The proof of correctness as well as the local implementation (Algorithm 8) are deferred to the appendix.

## 61:12 Testing Hamiltonicity (And Other Problems) in Minor-Free Graphs

■ **Algorithm 3** Return sampled lightest edge.

**Input:**  $a \in H$  and  $b \in H$ .

**Output:** The sampled lightest edge between  $\mathcal{C}(a)$  and  $\mathcal{C}(b)$

1. Initially  $A = \emptyset$  and let  $x$  be an upper bound on the size of the parts returned by the partition oracle when executed with parameter  $\epsilon/(6W)$ .
2. Sample a set,  $S$ , of  $\tilde{\Theta}(W^2 r^3 x \Delta / \epsilon^2)$  edges incident to  $a$ .
3. For each  $\{a, u\} \in S$  do:
  - a. If  $u \in H$  then add it to  $A$  if and only if  $u = b$ .
  - b. Otherwise, find the part of  $u$  and the sub-parts of this part.
  - c. For each sub-part find its center.
  - d. Add to  $A$  all the edges between the sub-part of  $u$  and sub-parts that belong to  $\mathcal{C}(b)$ .
4. Repeat Steps 2-3 where  $a$  and  $b$  switch roles.
5. Return the lightest edge in  $A$  (if  $A = \emptyset$  then return null).

■ **Algorithm 4** Partition into sub-parts.

**Input:** Access to the input graph  $G = (V, E)$  and a subset  $S \subseteq L$  such that  $G[S]$  is connected.

**Output:** A graph  $F = (S, A)$  such that each connected component of  $F$  is a sub-part of  $S$

1. Initially every vertex in  $S$  is in its own sub-part (a singleton) and  $A = \emptyset$ .
2. We say a sub-part  $B$  is *active* if the lightest in  $E^G(B, S \setminus B)$  is lighter than the lightest edge in  $E^G(B, H)$  or if  $E^G(B, H) = \emptyset$ .
3. While there are still active sub-parts do:
  - a. Each active sub-part  $B$  selects the lightest edge in  $E^G(B, S \setminus B)$ , denoted by  $e_B$ .
  - b. For each active sub-part,  $B$ , add  $e_B$  to  $A$
  - c. Update the new sub-parts to be the connected components of the graph  $F = (S, A)$  (each connected component is a sub-part).
4. Return  $F$ .

## 4 Algorithms for minor-free graphs with bounded degrees

### 4.1 Covering partition oracle

In this section we prove the following theorem.

► **Theorem 16.** *Algorithm 5 is an  $(\epsilon, \text{poly}(\epsilon^{-1}))$ -covering-partition oracle for minor-free bounded degree graphs with query complexity  $\text{poly}(\epsilon^{-1})$ . Specifically, the size of the sets returned by the oracle is  $O(\epsilon^{-640} \log^2(1/\epsilon))$ .*

We begin with a couple of definitions and lemmas from [20] that we build on.

► **Definition 17** ([20]). *Given  $\mathbf{x} \in (\mathbb{R}^+)^{|V|}$  and parameter  $\xi \in [0, 1]$ , the  $\xi$ -clipped vector  $\text{cl}(\mathbf{x}, \xi)$  is the lexicographically least vector  $\mathbf{y}$  optimizing the program:  $\min \|\mathbf{y}\|_2$ , subject to  $\|\mathbf{x} - \mathbf{y}\|_1 \leq \xi$  and  $\forall v \in V, \mathbf{y}(v) \leq \mathbf{x}(v)$ .*

► **Lemma 18** ([20]). *There is an absolute constant  $\alpha$  such that the following holds. Let  $H$  be a graph on  $r$  vertices. Suppose  $G$  is a  $H$ -minor-free graph. Then for any  $h \geq \alpha r^3$ , there exists at least  $(1 - 1/h)n$  vertices such that  $\|\text{cl}(\mathbf{p}_{v,\ell}, 3/8)\|_2^2 \geq h^{-7}$ .*



Given two parameters  $\epsilon \in [0, 1/2]$ , and a graph  $R$  on  $r \geq 3$  vertices. The length of the random walk is  $\ell = \alpha r^3 + \lceil \epsilon^{-20} \rceil$  where  $\alpha$  is some absolute constant.

► **Theorem 19** ([20]). *Suppose there are at least  $(1 - 1/\ell^{1/5})n$  vertices  $s$  s.t.  $\|\text{cl}(\mathbf{p}_{s,\ell}, 1/4)\|_2^2 > \ell^{-c}$ . Then, there is a partition  $\{P_1, P_2, \dots, P_b\}$  of the vertices s.t.:*

1. For each  $P_i$ , there exists  $s \in V$  such that:  $\forall v \in P_i, \sum_{t < 10\ell^{c+1}} p_{s,t}(v) \geq 1/8\ell^{c+1}$ .
2. The total number of edges crossing the partition is at most  $8dn\sqrt{c\ell^{-1/5} \log \ell}$ .

► **Corollary 20.** *Let  $G = (V, E)$  be a graph which is  $R$ -minor-free where  $R$  is a graph on  $r$  vertices. There exists a partition  $\{P_1, P_2, \dots, P_b\}$  of the  $V$  such that:*

1. For each  $P_i$ , there exists  $s \in V$  such that:  $\forall v \in P_i, \sum_{t < 10\ell^8} p_{s,t}(v) \geq 1/8\ell^8$ .
2. The total number of edges crossing the partition is at most  $\epsilon dn$ .

**Proof.** We first note that  $8dn\sqrt{c\ell^{-1/5} \log \ell} \leq \epsilon dn$  for sufficiently large constant  $\alpha$ .

By Lemma 18 there exist at least  $(1 - 1/\ell)n$  vertices such that  $\|\text{cl}(\mathbf{p}_{v,\ell}, 3/8)\|_2^2 \geq \ell^{-7}$ . Thus the corollary follows from the facts that  $(1 - 1/\ell)n \geq (1 - 1/\ell^{1/5})n$  and  $\|\text{cl}(\mathbf{p}_{s,\ell}, 1/4)\|_2^2 \geq \|\text{cl}(\mathbf{p}_{s,\ell}, 3/8)\|_2^2$ . ◀

■ **Algorithm 5** Covering-partition oracle.

**Input:**  $v \in V$ .

**Output:** A subset  $S$  which covers the part of  $v$ .

1. For every  $t < 10\ell^8$  perform  $x \stackrel{\text{def}}{=} \Theta(\ell^8 \log \ell)$  random walks of length  $t$  from  $v$ .
2. Let  $R$  denote the endpoints of the random walks performed in the previous step.
3. For every vertex  $r \in R$ , for every  $t < 10\ell^8$ , perform  $x$  random walks of length  $t$  from  $r$ .
4. Let  $S$  denote the set of all vertices encountered by the random walks performed in Step 1 and Step 3.
5. Return  $S$ .

**Proof of Theorem 16.** Let  $G = (V, E)$  be a graph which is  $R$ -minor-free where  $R$  is a graph over  $r$  vertices. Consider the partition of  $V$ ,  $\mathcal{P} = \{P_1, P_2, \dots, P_b\}$  as defined in Corollary 20 when we take the proximity parameter to be  $\epsilon/2$ . We shall define another partition  $\mathcal{P}'$  which is a refinement of  $\mathcal{P}$  such that Algorithm 5 returns for every  $v \in V$ , a subset  $S$  such that  $P' \subseteq S$  where  $P'$  denotes the part of  $v$  in  $\mathcal{P}'$ . Thereafter, we shall prove that, w.h.p., the number of cut-edges of  $\mathcal{P}'$  is not much greater than the number of cut-edges of  $\mathcal{P}$ .

For every  $v \in V$ , we say that  $v$  *fails* if Algorithm 5, when queried on  $v$ , does not return  $S$  such that  $P^v \subseteq S$ , where  $P^v$  denotes the part of  $v$  in  $\mathcal{P}$ . We define  $\mathcal{P}'$  as follows. For every  $v \in V$ , if there exists  $u \in P^v$  such that  $u$  fails, then the part of  $v$  in  $\mathcal{P}'$  is defined to be the singleton  $\{v\}$  (namely, the entire part  $P^v$  is partitioned into singletons in  $\mathcal{P}'$ ). Otherwise, it is defined to be  $P^v$ .

We next show that w.h.p. the cut-edges of  $\mathcal{P}'$  is at most  $\epsilon d|V|$ . For every  $v$ , the probability that  $v$  fails is at most  $p \stackrel{\text{def}}{=} \ell^{-c_1}$  for an appropriate setting of  $x$  (with accordance to the Theta-notation), where  $c_1$  is a constant that will be determined later. Let  $y = \ell^{c_2}$  be an upper bound on the number of vertices in the parts of  $\mathcal{P}$ , where  $c_2$  is a constant.<sup>8</sup>

<sup>8</sup> Clearly, since for each  $P_i$ , there exists  $s \in V$  such that:  $\forall v \in P_i, \sum_{t < 10\ell^8} p_{s,t}(v) \geq 1/8\ell^8$ , it follows that  $y \leq 10\ell^8 \cdot 8\ell^8$ .

## 61:14 Testing Hamiltonicity (And Other Problems) in Minor-Free Graphs

For every  $v \in V$ , define the random variable  $X_v$  as follows. If  $v$  fails then  $X_v = |P_v|/y$  and otherwise  $X_v = 0$ . Clearly  $y \cdot \sum_{v \in V} X_v \geq |\mathcal{P}'| - |\mathcal{P}|$ . Note that  $\{X_v\}_{v \in V}$  are independent random variables ranging in  $[0, 1]$ .

For every  $v \in V$ , we define the random variable  $Y_v$  as follows. With probability  $p$ ,  $Y_v = 1$  and otherwise  $X_v = 0$ . Clearly,  $Y_v$  dominates  $X_v$ . Since  $\{Y_v\}_{v \in V}$  are identical independent random variables, it follows by the multiplicative Chernoff's bound that w.h.p.  $y \cdot \sum_{v \in V} Y_v \leq y \cdot |V| \cdot 2p$ .

Since for sufficiently large  $c_1$ ,  $p \leq \frac{\epsilon}{4y}$ , it follows that w.h.p.  $|\mathcal{P}'| - |\mathcal{P}| \leq \epsilon|V|/2$ . Thus, the number of cut-edges in  $\mathcal{P}'$  is greater than the number of cut-edges in  $\mathcal{P}$  by at most  $\epsilon d|V|/2$  (recall that the algorithm refines  $\mathcal{P}$  by decomposing entire parts into singletons). ◀

### 4.2 Testing Hamiltonicity

In this section we prove the following theorem.

► **Theorem 21.** *Given query access to an input graph  $G = (V, E)$  where  $G$  is a minor-free bounded degree graph and a parameters  $\epsilon$  and  $|V|$ , Algorithm 6 accepts  $G$  with probability 1 if  $G$  is Hamiltonian and rejects  $G$  with probability at least  $2/3$  if  $G$  is  $\epsilon$ -far from being Hamiltonian. The query complexity of the algorithm is  $\text{poly}(d/\epsilon)$  and the running time is exponential in  $\text{poly}(d/\epsilon)$ .*

The correctness of Algorithm 6 builds on the following claim which, given  $S \subset V$ , bounds the size of a minimum path cover in  $G[S]$  by the size of the cut of  $S$ .

▷ **Claim 22.** Let  $G = (V, E)$  be a graph and let  $S \subset V$  be a subset of vertices of  $G$ . Let  $k$  be the size of a minimum path cover in  $G[S]$ . If  $k - 1 > |E(S, V \setminus S)|/2$ , then there is no Hamiltonian path in  $G$ . Moreover, any Hamiltonian path in  $G$  must include at least  $2(k - 1)$  edges from  $E(S, V \setminus S)$ .

*Proof.* Let  $G = (V, E)$  be a graph. Assume toward contradiction that there exists Hamiltonian path in  $G$ ,  $\mathcal{H} = (v_1, v_2, \dots, v_{|V|})$  and a subset  $S \subset V$  such that  $k - 1 > |E(S, V \setminus S)|/2$ , where  $k$  is the size of a minimum path cover in  $G[S]$ . Let  $\mathcal{P}'_S$  denote the set of all maximal sub-paths of  $\mathcal{H}$  in  $G[S]$ . In order to connect the sub-paths in  $\mathcal{P}'_S$  it must hold that  $\mathcal{H}$  leaves and returns to  $G[S]$  at least  $2(\mathcal{P}'_S - 1)$  times, each time using a different edge. Thus,  $|E(S, V \setminus S)| \geq 2(|\mathcal{P}'_S| - 1)$ . Since  $\mathcal{P}'_S$  is a path cover of  $G[S]$  it follows that  $|\mathcal{P}'_S| \geq k$ , thus,  $|E(S, V \setminus S)| \geq 2(k - 1)$ , in contradiction to our assumption. Hence the claim follows. ◀

We next list our algorithm and prove its correctness.

**Proof of Theorem 21.** By Claim 22, Algorithm 6 never rejects graphs which are Hamiltonian.

Let  $G$  be a minor-free bounded degree graph which is  $\epsilon$ -far from being Hamiltonian. We shall prove that Algorithm 6 rejects  $G$  with probability at least  $2/3$ . Let  $\mathcal{P}$  denote the partition that the oracle, executed in Step 2a, answers according to. With high constant probability, it holds that  $|E_{\mathcal{P}}| \leq \frac{\epsilon|V|}{6}$ . Let  $E_1$  denote the event that this conditions holds.

Let  $\mathcal{F}$  denote the set of parts,  $S$ , in  $\mathcal{P}$ , such that  $E(S, V \setminus S) = \emptyset$  or for which the size of the minimum path cover of  $G[S]$  is greater  $|E(S, V \setminus S)|/2 + 1$

Assume towards contradiction that  $E_1$  occurs and that  $|\mathcal{F}| < \epsilon|V|/(2x)$  where  $x$  is an upper bound on the number of vertices in each part of  $\mathcal{P}$ . We next show that  $\delta_{\text{HAM}}(G) \leq \epsilon|V|$  in contradiction to our assumption.

For each part  $S \in \mathcal{F}$  we construct a path over  $S$  that visits each vertex in  $S$  exactly once by adding at most  $|S| - 1 \leq x - 1$  edges to  $G$ .

■ **Algorithm 6** Testing Hamiltonicity in minor-free, bounded degree, graphs.

**Input:** Oracle access to a minor-free, bounded-degree, graph  $G = (V, E)$

**Output:** Tests if  $G$  is Hamiltonian with one-sided error.

1. Sample a subset,  $S \subseteq V$ , of  $y \stackrel{\text{def}}{=} \Theta(x/\epsilon)$  vertices, uniformly at random, where  $x$  is an upper bound on the size of the sets returned by the covering partition oracle when execute with parameter  $\epsilon/6$ .
2. For each  $v \in S$  do:
  - a. Query the covering partition oracle on  $v$  with parameter  $\epsilon/6$ , and let  $S_v$  denote the returned set.
  - b. If  $E(S_v, V \setminus S_v) = \emptyset$  then return REJECT.
  - c. For each subset  $T \subseteq S_v$  such that  $G[T]$  is connected, find the size of the minimum path cover of  $T$  and return REJECT if it is greater than  $|E(T, V \setminus T)|/2 + 1$ .

For each part  $S \in \mathcal{P} \setminus \mathcal{F}$ , let  $\mathcal{C}_S$  denote a minimum path cover of  $G[S]$ . We construct a path over  $S$  that visits each vertex in  $S$  exactly once by adding at most  $|\mathcal{C}_S| - 1 \leq |E(S, V \setminus S)|/2$  edges to  $G$ .

We then connect all the paths induced on the different parts of  $\mathcal{P}$  by adding at most  $|\mathcal{P}| = |\mathcal{F}| + |\mathcal{P} \setminus \mathcal{F}|$  edges.

Overall the number of edges added is at most:

$$|\mathcal{F}| \cdot (x - 1) + |E_{\mathcal{P}}| + |\mathcal{F}| + |\mathcal{P} \setminus \mathcal{F}| \leq |\mathcal{F}| \cdot x + 3|E_{\mathcal{P}}| < \epsilon|V|,$$

where the first inequality follows from the fact that  $|\mathcal{P} \setminus \mathcal{F}| \leq 2|E_{\mathcal{P}}|$  as for each  $S \in \mathcal{P} \setminus \mathcal{F}$  it holds that  $E(S, V \setminus S) \cap E_{\mathcal{P}} \neq \emptyset$  and each edge in  $E_{\mathcal{P}}$  is adjacent to at most 2 parts in  $\mathcal{P}$ .

Thus, if  $\delta_{\text{HAM}(G)} > \epsilon|V|$  and  $E_1$  occurs then  $|\mathcal{F}| \geq \epsilon|V|/(2x)$ . Thus, the number of vertices in parts that belong to  $\mathcal{F}$  is at least  $\epsilon|V|/(2x)$ , which implies that  $G$  is rejected w.h.p. either in Step 2b or in Step 2c of Algorithm 6, as desired. ◀

### 4.3 Local algorithms for constructing a spanning subgraph with almost optimum weight

In this section we prove the following theorem.

► **Theorem 23.** *Algorithm 7 is a local algorithm for  $(1 + \epsilon)$ -approximating the minimum weight spanning graph for minor-free graphs, with high constant success probability and time and query complexity  $\text{poly}(W, d, \epsilon^{-1})$ .*

**Proof.** Let  $\mathcal{P}$  denote the partition that the oracle, executed in Step 1, answers according to. With high constant probability, it holds that  $|E_{\mathcal{P}}| \leq \epsilon|V|/W$ . Let  $E_1$  denote the event that this conditions holds. We claim that the number edges for which Algorithm 7 returns YES for which both endpoints belong to the same part is at most  $|V| - 1$ . To see this consider a part  $T \in \mathcal{P}$  and a cycle  $C$  in  $G[T]$ . Let  $\{u, v\}$  denote the heaviest edge in the cycle. When queried on  $u$  and  $v$  the covering partition oracle returns sets  $S_u$  and  $S_v$  such that  $T \subseteq S_u \cup S_v$ . Thus the cycle  $C$  is contained in  $G[S_u \cup S_v]$ . Therefore the algorithm returns NO on  $\{u, v\}$  in Step 3. By the cycle rule the number of edges in  $G[T]$  for which the algorithm returns YES is exactly  $|T| - 1$ .

Hence conditioned on  $E_1$ , the total number of edges for which Algorithm 7 returns YES is at most  $(|V| - 1) + \epsilon|V|/W$ .

## 61:16 Testing Hamiltonicity (And Other Problems) in Minor-Free Graphs

By the cycle rule, any edge,  $e$ , for which Algorithm 7 returns NO does not belong to the MST of  $G$ . Since the MST consists of exactly  $|V| - 1$  edges, it follows that, conditioned on  $E_1$ , the number of edges that do not belong to the MST and for which Algorithm 7 returns YES is at most  $\epsilon|V|/W$  as desired. ◀

■ **Algorithm 7** Local algorithm for approximated-MST in bounded-degree minor-free graphs.

**Input:**  $\{u, v\} \in E$  and parameters  $\epsilon$  and  $W$ .

**Output:** YES if  $\{u, v\}$  belongs to the approximated-MST and NO otherwise.

1. Perform a query  $u$  and a query  $v$  to the covering-partition oracle with parameter  $\epsilon/W$ . Let  $S_u$  and  $S_v$  denote the subsets returned by the oracle, respectively.
2. Find the subgraph induced on  $S_u \cup S_v$ , denoted by  $G[S_u \cup S_v]$ .
3. Return NO if and only if  $\{u, v\}$  is the heaviest edge on any cycle in  $G[S_u \cup S_v]$ .

### 4.4 Testing monotone and additive properties

► **Theorem 24.** *Any property of graphs which is monotone (closed under removal of edges and vertices) and additive (closed under the disjoint union of graphs) can be tested with one-sided error in minor-free graphs with bounded degree  $d$  with query complexity which is  $\text{poly}(d/\epsilon)$  where  $\epsilon$  is the proximity parameter.*

**Proof.** Let  $\mathcal{T}$  be a property of graphs which is monotone and additive. We propose the following algorithm for testing  $\mathcal{P}$  on an input graph  $G$  which is a minor-free graphs of degree bounded by  $d$ . Sample a set of  $O(d/\epsilon)$  vertices,  $S$ , uniformly at random and run the covering partition oracle on each  $v \in S$  with parameter  $\epsilon/2$ .

For each  $v \in S$ , let  $S_v$  denote the set returned by the covering partition oracle when queried on  $v$ . Return ACCEPT iff for all  $v \in S$ ,  $G[S_v]$  has the property  $\mathcal{T}$ .

If  $G$  has the property  $\mathcal{T}$  then since  $\mathcal{T}$  is monotone it follows that for all  $v \in S$ ,  $G[S_v]$  has the property  $\mathcal{T}$  as well.

Let  $\mathcal{P}$  denote the partition that the covering partition oracle answers according to. With high constant probability  $|E_{\mathcal{P}}| \leq \epsilon|V|/2$ . Let  $E_1$  denote the event that this condition holds. If  $G$  is  $\epsilon$ -far from having the property  $\mathcal{T}$  then, conditioned on  $E_1$ ,  $G' = (V, E \setminus E_{\mathcal{P}})$  is  $(\epsilon/2)$ -far from having the property  $\mathcal{T}$ .

Thus we need to remove at least  $\epsilon|V|/2$  edges from  $G'$  to obtain the property  $\mathcal{T}$ . By the additivity and monotonicity of  $\mathcal{T}$  it follows that  $G'$  has the property  $\mathcal{T}$  if and only if for every  $T \in \mathcal{P}$ ,  $G[T]$  has the property  $\mathcal{T}$ . Thus, there are at least  $\epsilon|V|/2$  edges, and hence at least  $\epsilon|V|/(2d)$  vertices, that belong to parts,  $T \in \mathcal{P}$  such that  $G[T]$  does not have the property  $\mathcal{T}$ . Hence, with high constant probability the algorithm sample one of these vertices and rejects  $G$ . This concludes the proof. ◀

---

### References

- 1 Isolde Adler and Noleen Köhler. An explicit construction of graphs of bounded degree that are far from being hamiltonian, 2021. [arXiv:2008.05801](https://arxiv.org/abs/2008.05801).
- 2 N. Alon, R. Rubinfeld, S. Vardi, and N. Xie. Space-efficient local computation algorithms. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1132–1139, 2012.

- 3 Jasine Babu, Areej Khoury, and Ilan Newman. Every property of outerplanar graphs is testable. 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 21:1–21:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.APPROX-RANDOM.2016.21.
- 4 Itai Benjamini, Oded Schramm, and Asaf Shapira. Every minor-closed property of sparse graphs is testable. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 393–402. ACM, 2008. doi:10.1145/1374376.1374433.
- 5 Artur Czumaj, Oded Goldreich, Dana Ron, C. Seshadhri, Asaf Shapira, and Christian Sohler. Finding cycles and trees in sublinear time. *Random Struct. Algorithms*, 45(2):139–184, 2014. doi:10.1002/rsa.20462.
- 6 Artur Czumaj, Morteza Monemizadeh, Krzysztof Onak, and Christian Sohler. Planar graphs: Random walks and bipartiteness testing. *Random Struct. Algorithms*, 55(1):104–124, 2019. doi:10.1002/rsa.20826.
- 7 Artur Czumaj and Christian Sohler. A characterization of graph properties testable for general planar graphs with one-sided error (it’s all about forbidden subgraphs). In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1525–1548. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00089.
- 8 Alan Edelman, Avinatan Hassidim, Huy N. Nguyen, and Krzysztof Onak. An efficient partitioning oracle for bounded-treewidth graphs. In Leslie Ann Goldberg, Klaus Jansen, R. Ravi, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 14th International Workshop, APPROX 2011, and 15th International Workshop, RANDOM 2011, Princeton, NJ, USA, August 17-19, 2011. Proceedings*, volume 6845 of *Lecture Notes in Computer Science*, pages 530–541. Springer, 2011. doi:10.1007/978-3-642-22935-0\_45.
- 9 Lars Engebretsen and Marek Karpinski. Approximation hardness of TSP with bounded metrics. 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 201–212. Springer, 2001. doi:10.1007/3-540-48224-5\_17.
- 10 Hendrik Fichtenberger, Reut Levi, Yadu Vasudev, and Maximilian Wötzel. A sublinear tester for outerplanarity (and other forbidden minors) with one-sided error. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPICs*, pages 52:1–52:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.ICALP.2018.52.
- 11 O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002. doi:10.1007/s00453-001-0078-7.
- 12 Oded Goldreich. On testing hamiltonicity in the bounded degree graph model. *Electron. Colloquium Comput. Complex.*, 27:109, 2020. URL: <https://eccc.weizmann.ac.il/report/2020/109>.
- 13 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.
- 14 A. Hassidim, J. A. Kelner, H. N. Nguyen, and K. Onak. Local graph partitions for approximation and testing. In *Proceedings of the Fiftieth Annual Symposium on Foundations of Computer Science (FOCS)*, pages 22–31, 2009. doi:10.1109/FOCS.2009.77.
- 15 John E. Hopcroft and Robert Endre Tarjan. Isomorphism of planar graphs. 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 131–152. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2\_13.
- 16 Bell (<https://csttheory.stackexchange.com/users/48832/bell>). Hardness of approximation for minimum path cover in an undirected graph? Theoretical Computer Science Stack Exchange. URL:<https://csttheory.stackexchange.com/q/40344> (version: 2018-03-08). arXiv:<https://csttheory.stackexchange.com/q/40344>.
  - 17 Hiro Ito. Every property is testable on a natural class of scale-free multigraphs. In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, volume 57 of *LIPICs*, pages 51:1–51:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.ESA.2016.51.
  - 18 Akash Kumar, C. Seshadhri, and Andrew Stolman. Finding forbidden minors in sublinear time: A  $n^{1/2+o(1)}$ -query one-sided tester for minor closed properties of bounded degree graphs. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 509–520. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00055.
  - 19 Akash Kumar, C. Seshadhri, and Andrew Stolman. Random walks and forbidden minors II: a  $\text{poly}(d \epsilon^{-1})$ -query tester for minor-closed properties of bounded degree graphs. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 559–567. ACM, 2019. doi:10.1145/3313276.3316330.
  - 20 Akash Kumar, C. Seshadhri, and Andrew Stolman. Random walks and forbidden minors III:  $\text{poly}(d/?)$ -time partition oracles for minor-free graph classes. *Electron. Colloquium Comput. Complex.*, 28:8, 2021. URL: <https://eccc.weizmann.ac.il/report/2021/008>.
  - 21 Mitsuru Kusumoto and Yuichi Yoshida. Testing forest-isomorphism in the adjacency list model. 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 763–774. Springer, 2014. doi:10.1007/978-3-662-43948-7\_63.
  - 22 C. Lenzen and R. 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*, pages 87:1–87:14, 2018.
  - 23 R. Levi and M. Medina. A (centralized) local guide. *Bulletin of the EATCS*, 122, 2017.
  - 24 R. Levi, G. Moshkovitz, D. Ron, R. Rubinfeld, and A. Shapira. Constructing near spanning trees with few local inspections. *Random Struct. Algorithms*, 50(2):183–200, 2017.
  - 25 R. Levi and D. Ron. A quasi-polynomial time partition oracle for graphs with an excluded minor. *ACM Trans. Algorithms*, 11(3):24:1–24:13, 2015.
  - 26 R. Levi, D. Ron, and R. Rubinfeld. Local algorithms for sparse spanning graphs. In *Proceedings of the Eighteenth International Workshop on Randomization and Computation (RANDOM)*, pages 826–842, 2014.
  - 27 R. Levi, D. Ron, and R. Rubinfeld. Local algorithms for sparse spanning graphs. *CoRR*, abs/1402.3609, 2014. URL: <http://arxiv.org/abs/1402.3609>.
  - 28 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.
  - 29 Ilan Newman and Christian Sohler. Every property of hyperfinite graphs is testable. *SIAM J. Comput.*, 42(3):1095–1112, 2013. doi:10.1137/120890946.
  - 30 Michal Parnas and Dana Ron. Testing the diameter of graphs. *Random Struct. Algorithms*, 20(2):165–183, 2002. doi:10.1002/rsa.10013.
  - 31 M. Parter, R. Rubinfeld, A. Vakilian, and A. Yodpinyanee. Local computation algorithms for spanners. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 58:1–58:21, 2019.
  - 32 R. Rubinfeld, G. Tamir, S. Vardi, and N. Xie. Fast local computation algorithms. In *Proceedings of The Second Symposium on Innovations in Computer Science (ICS)*, pages 223–238, 2011.

- 33 Yuichi Yoshida and Hiro Ito. Query-number preserving reductions and linear lower bounds for testing. *IEICE Trans. Inf. Syst.*, 93-D(2):233–240, 2010. doi:10.1587/transinf.E93.D.233.

## **A** Related Work

### A.1 Partition Oracles

Partition oracles were introduced by Hassidim et al. [14] as a tool for approximating parameters and testing properties of minor-free bounded degree graphs. The query complexity of the partition oracle of [14] is exponential in  $1/\epsilon$ . The query complexity was later improved in [25] to be quasi-polynomial in  $1/\epsilon$ . Very recently, Kumar-Seshadhri-Stolman [20] obtained a partition oracle with query complexity which is polynomial in  $1/\epsilon$ .

Edelman et al. [8] obtained a partition oracle with query complexity polynomial which is in  $1/\epsilon$  for graphs with bounded treewidth.

### A.2 Testing Hamiltonicity

Yoshida and Ito [33] and more recently Goldreich [12] proved a linear (in the number of vertices) lower bound for testing Hamiltonicity (even with two-sided error) in bounded degree graphs. Adler and Köhler [1] provided a deterministic construction of families of graphs for which testing Hamiltonicity with one-sided error requires linear number of queries.

### A.3 Testing properties of minor-free graphs

Newman and Sohler [29] showed that any property of hyperfinite graphs and in particular minor-free graphs can be tested with query complexity that depends only on  $1/\epsilon$  and  $d$  where  $d$  is a bound on the maximum degree. In fact, they proved a stronger claim, that a minor-free graph can be learned up to a precision of  $\epsilon dn$  edges with such query complexity. However, although the query complexity of their canonical tester is independent of  $n$  it is super-polynomial in  $d/\epsilon$ .

For minor-free graphs of unbounded degrees, Czumaj et al. [6] obtained an algorithm whose query complexity depends only on  $1/\epsilon$  for testing Bipartiteness. More recently, this result was generalized by Czumaj and Sohler [7] who proved that any property of minor-free graphs can be tested with one-sided error with query complexity that depends only on  $1/\epsilon$  if and only if it can be reduced to testing for a finite family of finite forbidden subgraphs. Czumaj et al. [6] also provide a canonical tester for testing  $H$ -subgraph freeness for any fixed  $H$  with query complexity that is independent of  $n$ , however super polynomial in  $1/\epsilon$ .

It was shown that for other restrictive families of graphs of unbounded degree that every property is testable with query complexity which is at most polylogarithmic in  $n$  [21, 17, 3]. Specifically, Kusumoto and Yoshida [21] proved that any property of forests can be tested with query complexity  $\text{poly}(\log n)$  and that testing Isomorphism of forests requires  $\Omega(\sqrt{\log n})$ . This result was generalized in Babu-Khoury-Newman [3] for  $k$ -outerplanar graphs.

### A.4 Local algorithms for constructing sparse spanning subgraphs

The model of *local computation algorithms* as considered in this work, was defined by Rubinfeld et al. [32] (see also Alon et al. [2] and survey in [23]). The problem of constructing sparse spanning subgraphs in this model was studied in several papers [26, 24, 25, 22, 31, 28]. This problem is a special case of constructing an  $\epsilon$ -almost MST in which the weights of all the edges are identical.

For restricted families of graphs, it was shown that the complexity of the problem is independent of  $n$ . Specifically, it was shown in [24] that for families of graph that are, roughly speaking, sufficiency non-expanding, one can provide an algorithm with query complexity that is independent of  $n$  (however, super-exponential in  $1/\epsilon$ ). This is achieved by simulating a localized version of Kruskal's algorithm. On the negative side, it was also shown in [24] that for graphs with expansion properties that are a little better, there is *no local algorithm* that inspects a number of edges that is independent of  $n$ .

In [28] there is an algorithm for locally constructing sparse spanning subgraphs in minor-free, unbounded degree, graphs with query complexity and time complexity which are polynomial in  $d$  and  $1/\epsilon$ . Thus our algorithm for unbounded degree graphs generalizes this result for the weighted case.

In [26, 27] it was shown that a spanning subgraph of almost optimum weight can be constructed locally in minor-free graph with degree bounded by  $d$  with query complexity and time complexity which are quasi-polynomial in  $d$ ,  $1/\epsilon$  and  $W$  where  $W$  is the maximum weight of an edge. Thus our algorithm for unbounded degree graphs generalizes this result to unbounded degree graphs and improves the complexity of the upper bound from quasi-polynomial to polynomial in  $d$ ,  $1/\epsilon$  and  $W$ .

## **B** Omitted proofs and details

### **B.1** Proof of Claim 11

Let  $G = (V, E)$  be a graph and let  $\mathcal{C} = \{P_1, \dots, P_k\}$  be a minimum path cover of  $G$ .

We first prove that  $\delta_{\text{HAM}}(G) \leq k - 1$ . For every  $1 \leq i \leq k - 1$  we add an edge which connects the end-vertex of  $P_i$  to the start-vertex of  $P_{i+1}$ . Thus, by adding  $k - 1$  edges we constructed a Hamiltonian path in  $G$ .

We next prove that  $\delta_{\text{HAM}}(G) \geq k - 1$ . By definition, there exist  $\delta_{\text{HAM}}(G)$  edges such that when added to  $G$ ,  $G$  becomes Hamiltonian. Let  $E'$  denote a set of  $\delta_{\text{HAM}}(G)$  such edges and let  $G' = (V, E \cup E')$  be the graph resulting from adding these edges to  $G$ . Let  $\mathcal{H} = (s_1, \dots, s_{|V|})$  denote a Hamiltonian path in  $G'$ . After we remove back the edges in  $E'$  we break  $\mathcal{H}$  into  $|E'| + 1$  connected components (each edge we remove adds an additional connected component), i.e. into  $|E'| + 1$  paths. Thus the size of the minimum path cover of  $G$  is at most  $|E'| + 1 = \delta_{\text{HAM}}(G) + 1$ . Thus  $k \leq \delta_{\text{HAM}}(G) + 1$  and so  $\delta_{\text{HAM}}(G) \geq k - 1$  as desired.

### **B.2** Proof of Claim 12

The claim that  $\delta_{\text{HAM}}(G) \leq \delta_{\text{HAM}}(G')$  follows from the fact that the distance from being Hamiltonian can not decrease when we remove edges.

Let  $\mathcal{C}$  be a minimum path cover of  $G$ . By Claim 11,  $\delta_{\text{HAM}}(G) = |\mathcal{C}| - 1$ . Now consider removing the edges in  $F$  one by one and how this affects the number of paths in  $\mathcal{C}$ . After removal of a single edge, the number of paths may increase by at most one. Thus, after removing all the edges in  $F$  the paths in  $\mathcal{C}$  break into at most  $|\mathcal{C}| + |F|$  paths. Thus the size of the minimum path cover of  $G'$  is at most  $|\mathcal{C}| + |F|$ . By claim 11,  $\delta_{\text{HAM}}(G') \leq |\mathcal{C}| + |F| - 1 = \delta_{\text{HAM}}(G) + |F|$ , as desired.

### **B.3** Proof of Claim 13

The proof of this claim is similar to the proof of Claim 12.

The claim that  $\delta_{\text{HAM}}(G) \leq \delta_{\text{HAM}}(G')$  follows from the fact that the distance from being Hamiltonian can not decrease when we remove edges.



Let  $\mathcal{C}$  be a minimum path cover of  $G$ . By Claim 11,  $\delta_{\text{HAM}}(G) = |\mathcal{C}| - 1$ . Now consider removing the edges adjacent to vertices in  $S$  vertex by vertex and how this affects the number of paths in  $\mathcal{C}$ . After removal of edges incident to a specific vertex, the number of paths may increase by at most two. This follows from the fact that each vertex  $v$  belongs to exactly one path,  $P$ , and the fact that when the edges incident to  $v$  are removed,  $P$  may break into at most 3 different paths. Thus, after removing all the edges incident to vertices in  $S$  the paths in  $\mathcal{P}$  break into at most  $|\mathcal{C}| + 2|S|$  paths. Thus the size of the minimum path cover of  $G'$  is at most  $|\mathcal{C}| + 2|S|$ . By claim 11,  $\delta_{\text{HAM}}(G') \leq |\mathcal{C}| + 2|S| - 1 = \delta_{\text{HAM}}(G) + 2|S|$ , as desired.

## B.4 Proof of Claim 15

We claim that  $\text{MSF}(G) \subseteq \text{MSF}(G') \cup S$ . To see this observe that for every edge  $e \in E' \setminus \text{MSF}(G')$ , it holds, by the cycle rule, that there exists a cycle in  $G'$  such that  $e$  is the heaviest edges in this cycle. Thus, these edges are not in  $\text{MSF}(G)$  either (because all the cycles in  $G'$  exist in  $G$  as well).

Since the number of connected components in  $G$  is at most the number of connected components in  $G'$  it holds that  $|\text{MSF}(G')| \leq |\text{MSF}(G)|$ . Thus,

$$|\text{MSF}(G) \setminus \text{MSF}(G')| \geq |\text{MSF}(G') \setminus \text{MSF}(G)|. \quad (1)$$

Since  $\text{MSF}(G) \subseteq \text{MSF}(G') \cup S$  it holds that  $\text{MSF}(G) \setminus \text{MSF}(G') \subseteq S$ . Thus,

$$|\text{MSF}(G) \setminus \text{MSF}(G')| \leq |S|. \quad (2)$$

It follows from Equations 1 and 2 that  $|\text{MSF}(G') \setminus \text{MSF}(G)| \leq |S|$ . Thus, the claim follows from the bound on the maximum weight of an edge in  $G$ .

## B.5 Correctness of Algorithm 2

▷ **Claim 25.** With high constant probability, the number of edges added to  $E'$  in steps 2 and 4 of Algorithm 2 is at most  $\epsilon|V|/(3W)$ .

*Proof.* With high constant probability  $|E_{\mathcal{P}}| \leq \epsilon|V|/(6W)$ . Since  $G$  is minor-free it follows by Fact 4 and Markov's inequality the number of edges in  $G[H]$  is at most  $\epsilon|V|/(6W)$ . The claim follows. ◁

▷ **Claim 26.** All edges added to  $E'$  in step 5 of Algorithm 2 belong to  $\text{MSF}(\hat{G})$  where  $\hat{G} = (V, E \setminus E_{\mathcal{P}})$ .

*Proof.* Let  $S \in \mathcal{P}$  and let  $F = (S, A)$  denote the graph returned by Algorithm 4.

We first prove that all the edges in  $A$  belong to  $\text{MSF}(\hat{G})$ . Since each sub-part  $B$  of  $S$  is active (see Step 2 of Algorithm 4) as long as the lightest edge in  $E^G(B, S \setminus B)$  is lighter than the lightest edge in  $E^G(B, H)$  it follows that  $e_B$  (see Step 3a of Algorithm 4) is the lightest edge in  $E^{\hat{G}}(B, V \setminus B)$ . Thus, all the edges of  $A$  belong to  $\text{MSF}(\hat{G})$  by the cut rule (see Subsection 2.4).

We next prove that for each connected component of  $F$ ,  $C$ , the lightest edge which is adjacent to  $C$  and  $H$  (if such edge exists) is in  $\text{MSF}(\hat{G})$ . We first note that we only need to consider  $S$  such that  $E^G(S, H) \neq \emptyset$ . In this case each connected component of  $F$ ,  $C$ , is adjacent to at least one vertex in  $H$ . Since  $C$  is not active it follows that lightest edge which is adjacent to  $C$  and  $H$  is the lightest edge in the cut of  $C$  in  $\hat{G}$ . Thus the claim follows from the cut rule. This concludes the proof of the claim. ◁

In the proof of the following claim we closely follow the analysis in [28] (see proof in Subsection B.6).

## 61:22 Testing Hamiltonicity (And Other Problems) in Minor-Free Graphs

▷ **Claim 27.** Let  $U$  denote the set of edges in  $E'$  that are incident to two different clusters (namely, each endpoint belongs to a different cluster). With probability  $1 - 1/\Omega(|V|)$ ,  $|U| \leq \epsilon|V|/(3W)$ .

▷ **Claim 28.** Let  $G = (V, E)$  be a connected minor-free graph, then with high constant probability the graph returned by Algorithm 2,  $G' = (V, E')$ , is connected and  $\sum_{e \in E'} w(e) \leq (1 + \epsilon)\text{OPT}$ , where OPT is the weight of a minimum weight spanning tree of  $G$ .

*Proof.* We begin by proving  $G'$  is connected. To see this observe that each vertex either belongs to a cluster or to a subset  $S \in \mathcal{P}$  such that  $E^G(S, H) = \emptyset$ .

By construction, the subgraph induced on each cluster in  $G'$  is connected. For any two clusters which are connected by an edge the lightest edge that connects the clusters belongs to  $E'$ . This follows from Step 7 in Algorithm 2. If  $E^G(S, H) = \emptyset$  then  $G'[S]$  is connected by Algorithm 2, as all sub-parts of  $S$  remain active throughout the entire execution of the algorithm. Moreover, all the edges in  $E^G(S, V \setminus S)$  are in  $E'$  as well. Thus  $G'$  is connected. The claim regarding the weight of the edges of  $G'$  follows from Claim 15 and Claims 25-27.  $\triangleleft$

### B.6 Proof of Claim 27

For each cluster  $B$ , we charge to  $B$  a subset of the edges incident to  $B$  so that the union of all the charged edges (over all clusters) contains  $U$ . Our goal is to show that with probability  $1 - 1/\Omega(n)$ , the total number of charged edges is at most  $\epsilon|V|/(3W)$ .

Let  $\tilde{G} = (V, \hat{E})$  be such that  $\hat{E} = E \setminus E_{\mathcal{P}}$ . Consider the auxiliary graph, denoted  $\tilde{G}$ , that results from contracting each cluster  $B$  and isolated parts in  $\hat{G}$  into a *mega-vertex* in  $\tilde{G}$ , which we denote by  $v(B)$ . For each pair of clusters  $B$  and  $B'$  such that  $E^{\hat{G}}(B, B')$  is non-empty, there is an edge  $(v(B), v(B'))$  in  $\tilde{G}$ , which we refer to as a *mega edge*, and whose weight is  $|E^{\hat{G}}(B, B')|$ . Since  $G$  is minor-free, so is  $\tilde{G}$ . By Fact 4, which bounds the arboricity of minor-free graphs, we can partition the mega-edges of  $\tilde{G}$  into  $r$  forests. Consider orienting the mega-edges of  $\tilde{G}$  according to this partition (from children to parents in the trees of these forests), so that each mega-vertex has at most  $r$  outgoing mega-edges. For cluster  $B$  and a cluster  $B'$  such that  $(v(B), v(B'))$  is an edge in  $\tilde{G}$  that is oriented from  $v(B)$  to  $v(B')$ , we shall charge to  $B$  a subset of the edges in  $E^{\hat{G}}(B, B')$ , as described next.

Let  $x$  be an upper bound on the size of parts returned by the partition oracle when executed with parameter  $\epsilon/(6W)$ . Thus  $x$  is an upper bound on the size of part in the partition  $\mathcal{P}$  of  $G[\mathcal{L}]$ . Let  $E^b(B, B')$  denote the subset of edges in  $E^{\hat{G}}(B, B')$  that are the  $(\epsilon/(9rW)) \cdot |E(B, B')|$  lightest edges of  $E(B, B')$ . We charge all the edges in  $E^b(B, B')$  to  $B$ . The rationale is that for these edges it is likely that the algorithm won't sample an edge in  $E^{\hat{G}}(B, B')$  which is lighter. The total number of such edges is at most  $(\epsilon/(9rW)) \cdot |E| \leq \epsilon|V|/(9W)$ .

For a light vertex  $y$  let  $\text{subpart}(y)$  denote the subpart of  $y$ . Let  $u$  be the center of the cluster  $B$ , and let  $N^b(u, B')$  be the set of vertices,  $y \in N(u)$ , such that:

$$(y \in B' \text{ and } (u, y) \in E^b(B, B')) \text{ or } (\exists(y', z) \in E^b(B, B') \text{ s.t. } y' \in \text{subpart}(y)) .$$

That is,  $N^b(u, B')$  is the subset of neighbors of  $u$  such that if Algorithm 3 selects one of them in Step 2, then it obtains an edge in  $E^b(B, B')$ . We consider two cases.

**First case:**  $|N^b(u, B')|/|N(u)| < \epsilon^2/(162W^2r^3x\Delta)$ . In this case we charge all edges in  $E^{\hat{G}}(B, B')$  to  $B$ . For each part  $\text{subpart}(y)$  such that  $y \in N^b(u, B')$  there are at most  $|\text{subpart}(y)| \cdot \Delta$  edges  $(y', z) \in E^b(B, B')$  for which  $y' \in \text{subpart}(y)$ . Therefore, in this case

$|E^b(B, B')| \leq x\Delta \cdot |N^b(u, B')| \leq N(u) \cdot \epsilon^2/(162W^2r^3)$  and hence  $|E(B, B')| < (9rW/\epsilon) \cdot \epsilon^2/(162W^2r^3) \cdot N(u)$ . It follows that the total number of charged edges of this type is at most  $(\epsilon/(18rW)) \cdot 2|E| \leq \epsilon n/(9W)$ .

**Second case:**  $|N^b(u, B')|/|N(u)| \geq \epsilon^2/(162W^2r^3x\Delta)$ . For each  $u$  and  $B'$  that fall under this case we define the set of edges  $Y(u, B') = \{(u, v) : v \in N^b(u, B')\}$  and denote by  $Y$  the union of all such sets (over all such pairs  $u$  and  $B'$ ). Edges in  $Y$  are charged to  $B$  if and only if they belong to  $U$  and are incident to a vertex in  $B$ . Fix an edge in  $Y$  that is incident to  $B$ , and note that the selection of neighbors of  $u$  is done according to a  $t$ -wise independent distribution for  $t > 4q$ , where  $q$  is the sample size set in Step 2 of the Algorithm 3. Therefore, the probability that the edge belongs to  $U$  is upper bounded by  $(1 - \epsilon^2/(162W^2r^3x\Delta))^q$ , which by the setting of  $q$ , is at most  $p = \epsilon/(18Wr)$  (for sufficiently large constant w.r.t. the Theta notation).

We next show, using Chebyshev's inequality, that w.h.p., the number of edges in  $Y$  that are in  $U$  is at most  $2p|E|$ . For  $y \in Y$ , define  $J_y$  to be an indicator variable that is 1 if and only if  $y \in F$ . Then for a fixed  $y \in Y$ ,  $E[J_y] \leq p$  and  $\{J_y\}$  are pairwise independent (this is due to the fact that the samples of every pair of edges are pairwise independent). Therefore, by Chebyshev's inequality,

$$\Pr \left[ \sum_{y \in Y} J_y \geq 2p|E| \right] \leq \frac{\text{Var}[\sum_{y \in Y} J_y]}{(p|E|)^2} = \frac{\sum_{y \in Y} \text{Var}(J_y)}{(p|E|)^2} \leq \frac{p(1-p)|E|}{(p|E|)^2} = \frac{1-p}{p|E|} = \frac{1}{\Omega(n)},$$

and the proof of Claim 27 is completed.

► **Remark 29.** The random seed that Algorithm 2 uses consists of two parts. The first part is for running the partition oracle. The second part is for selecting random neighbors in Step 2 of Algorithm 3. Since the selection of neighbors is according to a  $t$ -wise independent distribution we obtain that a random seed of length  $\tilde{O}(\log n)$  is sufficient.

## B.7 The local implementation of Algorithm 2

Algorithm 8 is the local implementation of Algorithm 2 and is listed next.

■ **Algorithm 8** Local algorithm for approximated-MST in unbounded-degree minor-free graphs.

**Input:**  $\{u, v\} \in E$ .

**Output:** YES if  $\{u, v\}$  belongs to the approximated-MST and NO otherwise.

1. If both  $u$  and  $v$  are in  $H$  return YES.
2. If both  $u$  and  $v$  are light:
  - a. Query the partition oracle on  $u$  and  $v$  and return YES if they belong to different parts.
  - b. Find the sub-parts of  $u$  and  $v$  by running Algorithm 4.
  - c. If  $u$  and  $v$  are in the same sub-part:
    - i. Return YES if  $\{u, v\}$  is in the set of edges returned by Algorithm 4 (when running on  $u$ ).
    - ii. Otherwise, return NO.
  - d. Otherwise, set  $C_u$  to be the center of  $u$  and  $C_v$  to be the center of  $v$ .
  - e. If  $C_u = C_v$  return NO.
3. Otherwise, if  $u$  is light and  $v$  is heavy (and analogously if  $v$  is light and  $u$  is heavy) then:
  - a. Find the sub-part of  $u$  and set  $C_u$  to be the center of this sub-part
  - b. Set  $C_v = v$
4. Run Algorithm 3 on  $C_u$  and  $C_v$  and return YES if the edge  $\{u, v\}$  is lighter than the edge returned by the algorithm. Otherwise, return NO.



# Parallel Repetition for the GHZ Game: A Simpler Proof

Uma Girish  

Department of Computer Science, Princeton University, NJ, USA

Justin Holmgren  


NTT Research, Sunnyvale, CA, USA

Kunal Mittal  

Department of Computer Science, Princeton University, NJ, USA

Ran Raz  

Department of Computer Science, Princeton University, NJ, USA

Wei Zhan  

Department of Computer Science, Princeton University, NJ, USA

---

## Abstract

We give a new proof of the fact that the parallel repetition of the (3-player) GHZ game reduces the value of the game to zero polynomially quickly. That is, we show that the value of the  $n$ -fold GHZ game is at most  $n^{-\Omega(1)}$ . This was first established by Holmgren and Raz [18]. We present a new proof of this theorem that we believe to be simpler and more direct. Unlike most previous works on parallel repetition, our proof makes no use of information theory, and relies on the use of Fourier analysis.

The GHZ game [15] has played a foundational role in the understanding of quantum information theory, due in part to the fact that quantum strategies can win the GHZ game with probability 1. It is possible that improved parallel repetition bounds may find applications in this setting.

Recently, Dinur, Harsha, Venkat, and Yuen [7] highlighted the GHZ game as a simple three-player game, which is in some sense maximally far from the class of multi-player games whose behavior under parallel repetition is well understood. Dinur et al. conjectured that parallel repetition decreases the value of the GHZ game exponentially quickly, and speculated that progress on proving this would shed light on parallel repetition for general multi-player (multi-prover) games.

**2012 ACM Subject Classification** Theory of computation → Interactive proof systems

**Keywords and phrases** Parallel Repetition, GHZ, Polynomial, Multi-player

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.62

**Category** RANDOM

**Related Version** *Full Version:* <https://eccc.weizmann.ac.il/report/2021/101/>

**Funding** *Uma Girish:* Simons Collaboration on Algorithms and Geometry, by a Simons Investigator Award and by the National Science Foundation grants No. CCF-1714779, CCF-2007462.

*Kunal Mittal:* Simons Collaboration on Algorithms and Geometry, by a Simons Investigator Award and by the National Science Foundation grants No. CCF-1714779, CCF-2007462.

*Ran Raz:* Simons Collaboration on Algorithms and Geometry, by a Simons Investigator Award and by the National Science Foundation grants No. CCF-1714779, CCF-2007462.

*Wei Zhan:* Simons Collaboration on Algorithms and Geometry, by a Simons Investigator Award and by the National Science Foundation grants No. CCF-1714779, CCF-2007462.



© Uma Girish, Justin Holmgren, Kunal Mittal, Ran Raz, and Wei Zhan;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 62; pp. 62:1–62:19



Leibniz International Proceedings in Informatics

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

## 1 Introduction

The focus of this paper is multi-player games, and in particular their asymptotic behavior under parallel repetition.

Multi-player games consist of a one-round interaction between a referee and  $k$  players. In this interaction, the referee first samples a “query”  $(q_1, \dots, q_k)$  from some joint query distribution  $\mathcal{Q}$ , and for each  $i$  sends  $q_i$  to the  $i^{\text{th}}$  player. The players are required to respectively produce “answers”  $a_1, \dots, a_k$  without communicating with one another (that is, each  $a_i$  is a function only of  $q_i$ ) and they are said to *win* the game if  $(q_1, \dots, q_k, a_1, \dots, a_k)$  satisfy some predicate  $W$  that is fixed and associated with the game.

Suppose that a game  $G$  has the property that the maximum probability with which players can win is  $1 - \epsilon$ , no matter what strategy they use. This quantity is called the *value* of  $G$ . The parallel repetition question [13] asks

*How well can the players concurrently play in  $n$  independent copies of  $G$ ?*

More precisely, consider the following  $k$ -player game, which we call the  $n$ -wise parallel repetition of  $G$  and denote by  $G^n$ :

1. The referee samples, for each  $i \in [n]$  independently, query tuples  $(q_1^i, \dots, q_k^i) \sim \mathcal{Q}$ . We refer to the index  $i$  as a *coordinate* of the parallel repeated game.
2. The  $j^{\text{th}}$  player is given  $(q_j^1, \dots, q_j^n)$  and is required to produce a tuple  $(a_j^1, \dots, a_j^n)$ .
3. The players are said to win in coordinate  $i$  if  $(q_1^i, \dots, q_k^i, a_1^i, \dots, a_k^i)$  satisfies  $W$ . They are said to win (without qualification) if they win in every coordinate  $i \in [n]$ .

One might initially conjecture that the value of  $G^n$  is  $(1 - \epsilon)^n$ . However, this turns out not to be true [14, 9, 12, 25], as players may benefit from correlating their answers across different coordinates. Still, Raz showed that if  $G$  is a two-player game, then the value of  $G^n$  is  $2^{-\Omega(n)}$ , where the  $\Omega$  hides a game-dependent constant [23, 17]. Tighter results, based on the value of the initial game are also known [8, 5]. For many applications, such bounds are qualitatively as good as the initial flawed conjecture.

Games involving three or more players have proven more difficult to analyze, and the best known general bound on their parallel repeated value is due to Verbitsky [26]. This bound states that the value of  $G^n$  approaches 0, but the bound is *extremely* weak (it shows that the value is at most  $\frac{1}{\alpha(n)}$ , where  $\alpha$  denotes an inverse Ackermann function). The weakness of this bound is generally conjectured to reflect limitations of current proof techniques rather than a fundamental difference in the behavior of many-player games. In the technically incomparable but related *no-signaling setting* however, Holmgren and Yang showed that three-player games genuinely behave differently than two-player games [19]. Specifically, they showed that there exists a three-player game with “no-signaling value” bounded away from 1 such that no amount of parallel repetition reduces the no-signaling value at all.

Parallel repetition is a mathematically natural operation that we find worthy of study in its own right. At the same time, parallel repetition bounds have found several applications in theoretical computer science (see this survey by [24]). For example, parallel repetition of 2 player games shares intimate connections with multi-player interactive proofs [4], probabilistically checkable proofs and hardness of approximation [3, 10, 16], geometry of foams [11, 20, 1], quantum information [6], and communication complexity [22, 2]. Recent work also shows that strong parallel repetition for a particular class of multiprover games implies new time lower bounds on Turing machines that can take advice [21].

Dinur et al. [7] describe a restricted class of multi-player games for which Raz’s approach generalizes (giving exponential parallel bounds). Specifically, they consider games whose query distribution satisfies a certain connectivity property. For games outside this class,

Verbitsky’s bound was the best known. Dinur et al. highlighted one simple three-player game, called the GHZ game [15], that in some sense is maximally far from the aforementioned tractable class of multi-player games. In the GHZ game, the players’ queries are  $(q_1, q_2, q_3)$  chosen uniformly at random from  $\{0, 1\}^3$  such that  $q_1 \oplus q_2 \oplus q_3 = 0$ , and the players’ goal is to produce  $(a_1, a_2, a_3)$  such that  $a_1 \oplus a_2 \oplus a_3 = q_1 \vee q_2 \vee q_3$ . Dinur et al. conjectured that parallel repetition decreases the value of the GHZ game exponentially quickly, and speculated that progress on proving this would shed light on parallel repetition for general games. The GHZ game has also played a foundational role in the understanding of quantum information theory, due in part to the fact that quantum strategies can win the GHZ game with probability 1. It is possible that improved parallel repetition bounds will find applications in this setting as well.

In a recent work, Holmgren and Raz [18] proved the following polynomial upper bound on the parallel repetition of the GHZ game:

► **Theorem 1.** *The value of the  $n$ -wise repeated GHZ game is at most  $n^{-\Omega(1)}$ .*

Our main contribution is a different proof of this theorem that, in our view, is significantly simpler and more direct than the proof of [18]. Like [18], we actually do not rely on any properties of the GHZ game other than its query distribution, and in particular we do not rely on specifics of the win condition. Furthermore, unlike most previous works on parallel repetition, our proof makes no use of information theory, and instead relies on the use of Fourier analysis.

## 1.1 Technical Overview

Let  $\mathcal{P}$  denote the distribution of queries in the  $n$ -wise parallel repeated GHZ game. Let  $\alpha = \Theta(1/n^\varepsilon)$  for a small constant  $\varepsilon > 0$  and  $E = E_1 \times E_2 \times E_3$  be any product event with significant probability under  $\mathcal{P}$ , i.e.,  $\mathcal{P}(E) \geq \alpha$ . The core of our proof is establishing that for a random coordinate  $i \in [n]$ , the query distribution  $\mathcal{P}|E$  ( $\mathcal{P}$  conditioned on  $E$ ) is mildly hard in the  $i^{\text{th}}$  coordinate. That is, given queries sampled from  $\mathcal{P}|E$ , the players’ maximum winning probability in the  $i^{\text{th}}$  coordinate is bounded away from 1. Using standard arguments from the parallel repetition literature, this will imply an inverse polynomial bound for the value of the  $n$ -fold GHZ game. The difficulty, as usual, is that the  $n$  different queries in  $\mathcal{P}|E$  may not be independent.

Our approach at a high level is to:

1. Identify a class  $\mathcal{D}$  of simple distributions (over queries for the  $n$ -wise repeated GHZ game) such that it is easy to analyze (in step 3 below) which coordinates are hard for any given  $D \in \mathcal{D}$ . By hard, we mean that the players’ maximum winning probability in the  $i^{\text{th}}$  coordinate is  $\frac{3}{4}$ .
2. Approximate  $\mathcal{P}|E$  by a convex combination of distributions from  $\mathcal{D}$ . That is, we write

$$\mathcal{P}|E \approx \sum_j p_j D_j,$$

where  $\{D_j\}$  are distributions in  $\mathcal{D}$ ,  $p_j$  are non-negative reals summing to 1, and  $\approx$  denotes closeness in total variational distance.

3. Show that in the above convex combination, “most” of the  $D_i$  have many hard coordinates. More precisely, if we sample  $j$  with probability  $p_j$ , then the expected fraction of coordinates in which  $D_j$  is hard is at least a constant (say  $1/3$ ).

Completing this approach implies that if  $i \in [n]$  is uniformly random, then the  $i^{\text{th}}$  coordinate of  $\mathcal{P}|E$  can be won with probability at most  $1 - \Omega(1)$ . We elaborate on each of these steps below.

### Bow Tie Distributions

For our class of “simple” distributions  $\mathcal{D}$ , we introduce the notion of a “bow tie” distribution. We then define  $\mathcal{D}$  to be the set of all bow tie distributions. A bow tie is a set  $B$  of the form

$$\left\{ \begin{array}{l} (x_0, y_0, z_0), \\ (x_0, y_1, z_1), \\ (x_1, y_0, z_1), \\ (x_1, y_1, z_0) \end{array} \right\} \subseteq (\mathbb{F}_2^n)^3$$

such that for each  $(x, y, z)$  in  $B$ , we have  $x + y + z = 0$ . In particular this requires that  $x_0 + x_1 = y_0 + y_1 = z_0 + z_1$ . A bow tie distribution is the uniform distribution on a bow tie. Our name of “bow tie” is based on the fact that bow ties are thus determined by  $\{(x_0, y_0), (x_0, y_1), (x_1, y_0), (x_1, y_1)\}$ , which we sometimes view as a set of edges in a graph. In this case, bow ties are special kinds of  $K_{2,2}$  subgraphs, where  $K_{2,2}$  denotes the complete bipartite graph.

The main property of a bow tie distribution  $D$  is that for every coordinate  $i$  for which  $(x_0)_i \neq (x_1)_i$  (equivalently  $(y_0)_i \neq (y_1)_i$ , or  $(z_0)_i \neq (z_1)_i$ ), the  $i^{\text{th}}$  coordinate of  $D$  is as hard as the GHZ game (i.e. players cannot produce winning answers for the  $i^{\text{th}}$  coordinate with probability more than  $\frac{3}{4}$ ). This follows by “locally embedding” the (unrepeated) GHZ query distribution into the  $i^{\text{th}}$  coordinate of  $D$  as follows. We first swap  $x_0 \leftrightarrow x_1$ ,  $y_0 \leftrightarrow y_1$ ,  $z_0 \leftrightarrow z_1$  as necessary to ensure that

$$(x_0)_i = (y_0)_i = (z_0)_i = 0. \quad (1)$$

An even number of swaps are required to do this by the assumption that  $x_0 + y_0 + z_0 = 0$ , and bow ties are invariant under an even number of such swaps. Thus Equation (1) is without loss of generality. Suppose  $\bar{f}_1, \bar{f}_2, \bar{f}_3 : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$  comprise a strategy for the  $i^{\text{th}}$  coordinate of  $D$ . Then a strategy  $f_1, f_2, f_3 : \mathbb{F}_2 \rightarrow \mathbb{F}_2$  for the basic (unrepeated) GHZ game can be constructed as

$$\begin{aligned} f_1(b) &= \bar{f}_1(x_b) \\ f_2(b) &= \bar{f}_2(y_b) \\ f_3(b) &= \bar{f}_3(z_b). \end{aligned}$$

The winning probability of this strategy is the same as the winning probability of  $\bar{f}_1, \bar{f}_2, \bar{f}_3$  in the  $i^{\text{th}}$  coordinate because  $((x_{b_1})_i, (y_{b_2})_i, (z_{b_3})_i) = (b_1, b_2, b_3)$ . Hence both probabilities are at most  $3/4$ .

### Approximating $\mathcal{P}|E$ by Bow Ties

We now sketch how to approximate  $\mathcal{P}|E$  by a convex combination of bow tie distributions, where  $E$  is a product event  $E_1 \times E_2 \times E_3$ . We assume for now that the non-zero Fourier coefficients of each  $E_j$  are small. We will return to this assumption at the end of the overview – it turns out to be nearly without loss of generality.

We show that  $\mathcal{P}|E$  is close in total variational distance to the distribution obtained by sampling a *uniformly random* bow tie  $B \subseteq E$ , and then outputting a random element of  $B$ . The latter distribution is equivalent to sampling  $(x, y, z)$  with probability proportional to the number of bow ties  $B \subseteq E$  that contain  $(x, y, z)$ . This number is

$$\begin{cases} \left( \sum_{z' \in \mathbb{F}_2^n} E_1(y + z') E_2(x + z') E_3(z') \right) - 1 & \text{if } (x, y, z) \in \text{supp}(\mathcal{P}|E) \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$



where we identify  $E_1, E_2$ , and  $E_3$  with their indicator functions. Note that we are subtracting 1 to cancel the term corresponding to  $z' = z$ .

Intuitively, the fact that all  $E_j$  have small Fourier coefficients means that they look random with respect to linear functions. Thus, one might guess that the above sum is close to  $2^n \cdot \mu(E_1)\mu(E_2)\mu(E_3)$  for most  $(x, y, z) \in \text{supp}(\mathcal{P}|E)$ , where  $\mu(S) = |S|/2^n$  denotes the measure of  $S$  under the uniform distribution on  $\mathbb{F}_2^n$ . If “close to” and “most” have the right meanings, then this would imply that our distribution is close in total variational distance to  $\mathcal{P}|E$  as desired.

Our full proof indeed establishes this. More precisely, we view Equation (2) as a vector indexed by  $(x, y, z)$  and establish bounds on that vector’s  $\ell_1$  and  $\ell_2$  norms as a criterion for near-uniformity. In the process our proof repeatedly uses the following claims (see Lemma 16). For all sets  $S, T \subseteq \mathbb{F}_2^n$  that are sufficiently large, we have

$$\mathbb{E}_{\substack{z \sim \mathbb{F}_2^n \\ x \sim \mathbb{F}_2^n}} [S(x) \cdot T(x+z) \cdot E_3(z)] \approx \mu(S) \cdot \mu(T) \cdot \mu(E_3)$$

and

$$\mathbb{E}_{z \sim \mathbb{F}_2^n} \left[ \left( \mathbb{E}_{x \sim \mathbb{F}_2^n} [S(x) \cdot E_2(x+z)] \right)^2 \cdot E_3(z) \right] \approx \mu(S)^2 \cdot \mu(E_2)^2 \cdot \mu(E_3).$$

### Most Bow Ties are Hard in Many Coordinates

For the final step of our proof, we need to show that the distribution of bow ties analyzed in the previous step produces (with high probability) bow ties that differ in many coordinates.

We begin by parameterizing a bow tie by  $(x_0, y_0, x_0 \oplus x_1)$  and noting that in the previous step, we essentially showed that  $E$  contains  $2^{3n-O(\log n)}$  different bow ties. The  $O(\log n)$  term in the exponent arises from the fact that the events  $\{E_j\}$  have density in  $\mathbb{F}_2^n$  that is inverse polynomial in  $n$ . A simple counting argument then shows that for a random bow tie, the min-entropy of  $x_0 \oplus x_1$  is close to  $n$ . This means that  $x_0 \oplus x_1$  is close to the uniform distribution in the sense that any event occurring with probability  $p$  under the uniform distribution occurs with probability  $p \cdot n^{O(1)}$  under the distribution of  $x_0 \oplus x_1$ . Thus we can finally apply a Chernoff bound to deduce that with all but  $2^{-\Omega(n)}$  probability,  $x_0 \oplus x_1$  has Hamming weight at least  $n/3$ .

In other words, a bow tie sampled uniformly at random differs in at least a  $\frac{1}{3}$  fraction of coordinates. By the main property of bow ties, this implies that the corresponding bow tie distribution is hard on a  $\frac{1}{3}$  fraction of coordinates (indeed, the same set of coordinates).

### Handling General Events

For general (product) events  $E = E_1 \times E_2 \times E_3$  (where the sets  $\{E_i\}$  need not have small Fourier coefficients), we can partition the universe  $\mathbb{F}_2^n \times \mathbb{F}_2^n \times \mathbb{F}_2^n$  into parts  $\pi$  such that for most of the parts  $\pi$ , the event  $E$  restricted to  $\pi$  has the structure that we already analyzed. For this to make sense, we ensure several properties of the partition. First,  $\pi$  should be a product set ( $\pi = \pi_1 \times \pi_2 \times \pi_3$ ) so that  $E \cap \pi$  is a product set as well, i.e.  $E \cap \pi$  has the form  $\tilde{E}_1 \times \tilde{E}_2 \times \tilde{E}_3$ . Second, each  $\pi_i$  should be an affine subspace of  $\mathbb{F}_2^n$  so that we can do Fourier analysis with respect to this subspace. Finally  $\pi_1, \pi_2$ , and  $\pi_3$  should all be affine shifts of the *same* linear subspace so that the set  $\{(x, y, z) \in \pi : x + y + z = 0\}$  has the same Fourier-analytic structure as the parallel repeated GHZ query set  $\{(x, y, z) \in (\mathbb{F}_2^{n'})^3 : x + y + z = 0\}$  for some  $n' < n$ .

We prove the existence of such a partition with  $n'$  not too small ( $n' = n - o(n)$ ) by a simple iterative approach, which is similar to [18].

## 1.2 Comparison to [18]

Our proof has some similarity to [18] – in particular, both proofs partition  $(\mathbb{F}_2^n)^3$  into subspaces according to Fourier-analytic criteria and analyze these subspaces separately – but the resemblance ends there. In fact, there are fundamental high-level differences between the two proofs.

The biggest qualitative difference is that our high-level approach decomposes any conditional distribution  $\mathcal{P}|E$  into components (bow tie distributions) for which many coordinates are hard. [18] takes an analogous approach, but it establishes a weaker result that differs in the order of quantifiers: it first fixes a strategy  $f$ , and then decomposes  $\mathcal{P}|E$  into components such that  $f$  performs poorly on many coordinates of many components. This difference is due to the fact that [18] uses uniform distributions on high-dimensional affine spaces as their basic “hard” distributions. It is not in general possible to express  $\mathcal{P}|E$  as a convex combination of such distributions (for example if each  $E_j$  is a uniformly random subset of  $\mathbb{F}_2^n$ ). Instead, [18] expresses  $\mathcal{P}|E$  as a convex combination of “pseudo-affine” distributions. This significantly complicates their proof, and we avoid this complication entirely by our use of bow tie distributions, which are novel to this work.

The remainder of our proof (the analysis of hardness within each part of the partition) is entirely different.

## 2 Notation & Preliminaries

A significant portion of these preliminaries is taken verbatim from [18].

We write  $\exp(t)$  to denote  $e^t$  for  $t \in \mathbb{R}$ .

Let  $n \in \mathbb{N}$ . For a vector  $v \in \mathbb{R}^n$  and  $i \in [n]$ , we write  $v(i)$  or  $v^i$  to denote the  $i$ -th coordinate of  $v$ . For  $p \in \mathbb{N}$ , we write  $\|v\|_p \stackrel{\text{def}}{=} \left( \sum_{i \in [n]} |v(i)|^p \right)^{1/p}$  to denote the  $\ell_p$  norm of  $v$ .

For  $z \in \{0, 1\}^*$ ,  $\text{hwt}(z) \stackrel{\text{def}}{=} \|z\|_1$  denotes the Hamming weight of  $z$ .

We crucially rely on the Cauchy-Schwarz inequality.

► **Fact 2 (Cauchy-Schwarz).** *Let  $k \in \mathbb{N}$  and  $a_1, \dots, a_k, b_1, \dots, b_k \in \mathbb{R}$ . Then,  $\sum_{i=1}^k |a_i \cdot b_i| \leq \sqrt{\sum_{i=1}^k a_i^2} \cdot \sqrt{\sum_{i=1}^k b_i^2}$ .*

### 2.1 Set Theory

Let  $\Omega$  be a universe. By a partition of  $\Omega$ , we mean a collection of pairwise disjoint subsets of  $\Omega$ , whose union equals  $\Omega$ . If  $\Pi$  is a partition of  $\Omega$  and  $\omega$  is an element of  $\Omega$ , we will write  $\Pi(\omega)$  to denote the (unique) element of  $\Pi$  that contains  $\omega$ . Thus, we can view  $\Pi$  as a function  $\Pi : \Omega \rightarrow 2^\Omega$ .

For a set  $S \subseteq \Omega$ , we identify  $S$  with its indicator function  $S : \Omega \rightarrow \{0, 1\}$  defined at  $\omega \in \Omega$  by

$$S(\omega) = \begin{cases} 1 & \text{if } \omega \in S \\ 0 & \text{otherwise.} \end{cases}$$

For sets  $S, T \subseteq \Omega$  such that  $T \neq \emptyset$ , we use  $S|_T \subseteq T$  to denote the set  $S \cap T$  when viewed as a subset of  $T$ . In particular,  $S|_T$  is an indicator function from  $T$  to  $\{0, 1\}$ .

## 2.2 Probability Theory

### Probability Distributions

Let  $P$  be a distribution over a universe  $\Omega$ . We sometimes think of  $P$  as a vector in  $\mathbb{R}^{|\Omega|}$  whose value in coordinate  $\omega \in \Omega$  is  $P(\omega)$ . In particular, we use  $\|P - Q\|_1$  to denote the  $\ell_1$  norm of the vector  $P - Q \in \mathbb{R}^{|\Omega|}$ , where  $P$  and  $Q$  are probability distributions. We use  $\omega \sim P$  to denote a random element  $\omega$  distributed according to  $P$ . We use  $\text{supp}(P) = \{\omega \in \Omega : P(\omega) > 0\}$  to denote the support of the distribution  $P$ .

### Random Variables

Let  $\Sigma$  be any alphabet. We say that  $X : \Omega \rightarrow \Sigma$  is a  $\Sigma$ -valued random variable. If  $\Sigma = \mathbb{R}$ , we say that the random variable is real-valued. If  $X$  is a real-valued random variable, the expectation of  $X$  under  $P$  is denoted  $\mathbb{E}_{\omega \sim P}[X(\omega)]$ . Often, the underlying distribution  $P$  is implicit, in which case we simply use  $\mathbb{E}[X]$ . If  $X$  is a  $\Sigma$ -valued random variable and  $P$  is a probability distribution, we write  $P_X$  or  $X(P)$  to denote the induced probability distribution of  $X$  under  $P$ , i.e.,  $P_X(\sigma) = (X(P))(\sigma) \stackrel{\text{def}}{=} P(X = \sigma)$  for all  $\sigma \in \Sigma$ . In particular, we say that  $X$  is distributed according to  $P_X$  and we use  $\sigma \sim X(P)$  to denote a random variable  $\sigma$  distributed according to  $P_X$ . The distribution  $P$  is often implicit, and we identify  $X$  with the underlying distribution  $P_X$ .

### Events

We refer to subsets of  $\Omega$  as events. We use standard shorthand for denoting events. For instance, if  $X$  is a  $\Sigma$ -valued random variable and  $x \in \Sigma$ , we write  $X = x$  to denote the event  $\{\omega \in \Omega : X(\omega) = x\}$ . Similarly, for a subset  $F \subseteq \Sigma$ , we write  $X \in F$  to denote the event  $\{\omega \in \Omega : X(\omega) \in F\}$ . We use  $P(E)$  to denote the probability of  $E$  under  $P$ . When  $P$  is implicit, we use the notation  $\Pr(E)$  to denote  $P(E)$ .

### Conditional Probabilities

Let  $E \subseteq \Omega$  be an event with  $P(E) > 0$ . Then the conditional distribution of  $P$  given  $E$  is denoted  $(P|E) : \Omega \rightarrow \mathbb{R}$  and is defined to be

$$(P|E)(\omega) = \begin{cases} P(\omega)/P(E) & \text{if } \omega \in E \\ 0 & \text{otherwise.} \end{cases}$$

If  $E$  is an event, we write  $P_{X|E}$  as shorthand for  $(P|E)_X$ .

### Measure under Uniform Distribution

For any set  $S \subseteq \Omega$ , we sometimes identify  $S$  with the uniform distribution over  $S$ . In particular, we use  $x \sim S$  to denote  $x$  sampled according to the uniform distribution on  $S$ . For  $S, \pi \subseteq \Omega$  such that  $\pi \neq \emptyset$ , we use  $\mu_\pi(S) = \frac{|S \cap \pi|}{|\pi|}$  to denote the measure of  $S$  under the uniform distribution over  $\pi$ . When  $\pi = \Omega$ , we omit the subscript and simply use  $\mu(S)$ .

## 2.3 Fourier Analysis

### Fourier Analysis over Subspaces

For any (finite) vector space  $\mathcal{V}$  over  $\mathbb{F}_2$ , the character group of  $\mathcal{V}$ , denoted  $\widehat{\mathcal{V}}$ , is the set of group homomorphisms mapping  $\mathcal{V}$  (viewed as an additive group) to  $\{-1, 1\}$  (viewed as a

## 62:8 Parallel Repetition for the GHZ Game: A Simpler Proof

multiplicative group). Each such homomorphism is called a character of  $\mathcal{V}$ . For functions mapping  $\mathcal{V} \rightarrow \mathbb{R}$ , we define the inner product

$$\langle f, g \rangle \stackrel{\text{def}}{=} \mathbb{E}_{x \sim \mathcal{V}} [f(x)g(x)].$$

The character group of  $\mathcal{V}$  forms an orthonormal basis under this inner product. We refer to the all-ones functions  $\chi : \mathcal{V} \rightarrow \{-1, 1\}$ ,  $\chi \equiv 1$  as the *trivial character* or the *zero character* and denote this by  $\chi = \emptyset$ .

For all characters  $\chi \neq \emptyset$ , since  $\langle \chi, \emptyset \rangle = 0$ , we have  $\mathbb{E}_{x \sim \mathcal{V}} [\chi(x)] = 0$ , in particular,  $\chi(\mathcal{V})$  is a uniform  $\{\pm 1\}$ -random variable. Let  $\emptyset \neq S \subseteq \mathcal{V}$  be a set. Then  $\mu_{\mathcal{V}}(S) \triangleq \frac{|S \cap \mathcal{V}|}{|\mathcal{V}|} = \widehat{S}(\emptyset)$ , where we identify  $S$  with its indicator function  $S : \mathcal{V} \rightarrow \{0, 1\}$  as mentioned before. For  $\chi \in \widehat{\mathcal{V}}$ , we have  $\mathbb{E}_{x \sim S} [\chi(x)] = \frac{\widehat{S}(\chi)}{\widehat{S}(\emptyset)}$ .

► **Fact 3.** *Given a choice of basis for  $\mathcal{V}$ , there is a canonical isomorphism between  $\mathcal{V}$  and  $\widehat{\mathcal{V}}$ . Specifically, if  $\mathcal{V} = \mathbb{F}_2^n$ , then the characters of  $\mathcal{V}$  are the functions of the form*

$$\chi_{\gamma}(v) = (-1)^{\gamma \cdot v}$$

for  $\gamma \in \mathbb{F}_2^n$ .

► **Definition 4.** *For any function  $f : \mathcal{V} \rightarrow \mathbb{R}$ , its Fourier transform is the function  $\widehat{f} : \widehat{\mathcal{V}} \rightarrow \mathbb{R}$  defined by*

$$\widehat{f}(\chi) \stackrel{\text{def}}{=} \langle f, \chi \rangle = \mathbb{E}_{x \sim \mathcal{V}} [f(x)\chi(x)].$$

Since the characters of  $\mathcal{V}$  are orthonormal and  $\mathcal{V}$  is finite, we can deduce that  $f$  is equal to  $\sum_{\chi \in \widehat{\mathcal{V}}} \widehat{f}(\chi) \cdot \chi$ .

► **Theorem 5 (Plancherel).** *For any  $f, g : \mathcal{V} \rightarrow \mathbb{R}$ ,*

$$\langle f, g \rangle = \sum_{\chi \in \widehat{\mathcal{V}}} \widehat{f}(\chi) \cdot \widehat{g}(\chi).$$

An important special case of Plancherel's theorem is Parseval's theorem:

► **Theorem 6 (Parseval).** *For any  $f : \mathcal{V} \rightarrow \mathbb{R}$ ,*

$$\mathbb{E}_{x \sim \mathcal{V}} [f(x)^2] = \sum_{\chi \in \widehat{\mathcal{V}}} \widehat{f}(\chi)^2.$$

### Fourier Analysis over Affine Subspaces

Fix any subspace  $\mathcal{V} \subseteq \mathbb{F}_2^n$  and a vector  $a \in \mathbb{F}_2^n$ . Let  $\mathcal{U} = a + \mathcal{V}$  denote the affine subspace obtained by shifting  $\mathcal{V}$  by  $a$ . For every function  $f : \mathcal{V} \rightarrow \mathbb{R}$ , we associate it with a function  $f_a : \mathcal{U} \rightarrow \mathbb{R}$  defined by  $f_a(x) = f(x + a)$  for all  $x \in \mathcal{U}$ . This is a bijective correspondence between the set of functions from  $\mathcal{U}$  to  $\mathbb{R}$  and the set of functions from  $\mathcal{V}$  to  $\mathbb{R}$ . Under this association, we can identify  $\chi \in \widehat{\mathcal{V}}$  with  $\chi_a : \mathcal{U} \rightarrow \{-1, 1\}$  where  $\chi_a(x) = \chi(x + a)$  for all  $x \in \mathcal{U}$ . This defines an orthonormal basis  $\widehat{\mathcal{U}}_a := \{\chi_a : \mathcal{U} \rightarrow \{-1, 1\} \mid \chi \in \widehat{\mathcal{V}}\}$  for the vector space of functions from  $\mathcal{U}$  to  $\mathbb{R}$ . We call this the Fourier basis for  $\mathcal{U}$  with respect to  $a$ . This basis depends on the choice of the shift  $a \in \mathcal{U}$ . However, for all possible shifts  $b \in \mathcal{U}$  and character functions  $\chi \in \widehat{\mathcal{V}}$ , the functions  $\chi_a$  and  $\chi_b$  only differ by a sign. To see this, observe that

$$\chi_a(x) = \chi(a + x) = \chi(b + x) \cdot \chi(a + b) = \chi_b(x) \cdot \chi(a + b)$$

We will sometimes ignore the subscript and simply use  $\chi \in \widehat{\mathcal{V}}$  to index functions in the Fourier basis of  $\mathcal{U}$ . This is particularly the case when the properties we are dealing are independent of choice of basis (for example, the absolute values of Fourier coefficients of a function).

## 2.4 Multi-Player Games

In parallel repetition we often work with Cartesian product sets of the form  $(\mathcal{X}_1 \times \cdots \times \mathcal{X}_k)^n$ . For these sets, we will use subscripts to index the inner product and superscripts to index the outer product. That is, for  $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_k$  we view elements  $x$  of  $\mathcal{X}^n$  as tuples  $(x_1, \dots, x_k)$ , where  $x_i \in \mathcal{X}_i^n$ . We use  $x_i^j$  or  $x_i(j)$  to refer to the  $j^{\text{th}}$  coordinate of  $x_i$ . We use  $x^j$  to denote the vector  $(x_1^j, \dots, x_k^j)$ .

If  $\{E_i \subseteq \mathcal{X}_i\}_{i \in [k]}$  is a collection of subsets, we write  $E_1 \times \cdots \times E_k$  to denote the set  $\{x \in \mathcal{X} : \forall i \in [k], x_i \in E_i\}$ . We say that  $f : (\mathcal{X}_1 \times \cdots \times \mathcal{X}_k)^n \rightarrow (\mathcal{Y}_1 \times \cdots \times \mathcal{Y}_k)^n$  is a product function if  $f = f_1 \times \cdots \times f_k$  for some functions  $f_i : \mathcal{X}_i^n \rightarrow \mathcal{Y}_i^n$ .

► **Definition 7 (Multi-player Games).** A  $k$ -player game is a tuple  $(\mathcal{X}, \mathcal{Y}, Q, W)$ , where  $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_k$  and  $\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_k$  are finite sets,  $Q$  is a probability measure on  $\mathcal{X}$ , and  $W : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}$  is a “winning” predicate. We refer to  $Q$  as the query distribution or the input distribution of the game.

► **Definition 8 (Deterministic Strategies).** A deterministic strategy for a  $k$ -player game  $\mathcal{G} = (\mathcal{X}, \mathcal{Y}, Q, W)$  is a function  $f = f_1 \times \cdots \times f_k$  where each  $f_i : \mathcal{X}_i \rightarrow \mathcal{Y}_i$ . The success probability of  $f$  in  $\mathcal{G}$  is denoted and defined as

$$\text{val}(\mathcal{G}, f) \stackrel{\text{def}}{=} \Pr_{x \sim Q} [W(x, f(x)) = 1].$$

The most important quantity associated with a game is the maximum probability with which the game can be “won”.

► **Definition 9.** The value of a  $k$ -player game  $\mathcal{G} = (\mathcal{X}, \mathcal{Y}, Q, W)$ , denoted  $\text{val}(\mathcal{G})$ , is the maximum, over all deterministic strategies  $f$ , of  $\text{val}(\mathcal{G}, f)$ .

It is often easier to construct *probabilistic* strategies for a game, i.e. strategies in which players may use shared and/or individual randomness in computing their answers.

► **Definition 10 (Probabilistic Strategies).** Let  $\mathcal{G} = (\mathcal{X}, \mathcal{Y}, Q, W)$  be a  $k$ -player game. A probabilistic strategy for  $\mathcal{G}$  is a distribution  $\mathcal{F}$  of deterministic strategies for  $\mathcal{G}$ . The success probability of  $\mathcal{F}$  in  $\mathcal{G}$  is denoted and defined as

$$\text{val}(\mathcal{G}, \mathcal{F}) \stackrel{\text{def}}{=} \Pr_{\substack{x \sim Q \\ f \sim \mathcal{F}}} [W(x, f(x)) = 1].$$

A standard averaging argument implies that for every game, probabilistic strategies cannot achieve better success probability than deterministic strategies:

► **Fact 11.** Replacing “deterministic strategies” by “probabilistic strategies” in Definition 9 yields an equivalent definition.

The main operation on multi-player games that we consider in this paper is parallel repetition:

## 62:10 Parallel Repetition for the GHZ Game: A Simpler Proof

► **Definition 12** (Parallel Repetition). Given a  $k$ -player game  $\mathcal{G} = (\mathcal{X}, \mathcal{Y}, Q, W)$ , its  $n$ -fold parallel repetition, denoted  $\mathcal{G}^n$ , is defined as the  $k$ -player game  $(\mathcal{X}^n, \mathcal{Y}^n, Q^n, W^n)$ , where  $W^n(x, y) \stackrel{\text{def}}{=} \bigwedge_{j=1}^n W(x^j, y^j)$ . For  $x \in \mathcal{X}^n$ , we refer to  $x_i \in \mathcal{X}_i^n$  as the input to the  $i$ -th player.

To bound the value of parallel repeated games, it is helpful to analyze the probability of winning in a particular instance of the game under various modified query distributions.

► **Definition 13** (Value in  $j^{\text{th}}$  coordinate). If  $\mathcal{G} = (\mathcal{X}, \mathcal{Y}, Q, W^n)$  is a game (with a product winning predicate), the value of  $\mathcal{G}$  in the  $j^{\text{th}}$  coordinate for  $j \in [n]$ , denoted  $\text{val}^{(j)}(\mathcal{G})$ , is the value of the game  $(\mathcal{X}, \mathcal{Y}, Q, W')$ , where  $W'(x, y) = W(x^j, y^j)$ .

► **Definition 14** (Game with Modified Query Distribution). Let  $\mathcal{G} = (\mathcal{X}, \mathcal{Y}, Q, W)$  be a game. For a probability measure  $P$  on  $\mathcal{X}$ , we write  $\mathcal{G}|P$  to denote the game  $(\mathcal{X}, \mathcal{Y}, P, W)$ . For an event  $E$  on  $\mathcal{X}$ , we write  $\mathcal{G}|E$  to denote the game  $(\mathcal{X}, \mathcal{Y}, Q_E, W)$ .

### 2.5 GHZ Distribution

Let  $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \mathcal{X}_3$  and  $\mathcal{Y} = \mathcal{Y}_1 \times \mathcal{Y}_2 \times \mathcal{Y}_3$  where  $\mathcal{X}_i = \mathcal{Y}_i = \mathbb{F}_2$ . Let  $Q$  denote the uniform distribution over  $\{(0, 0, 0), (0, 1, 1), (1, 0, 1), (1, 1, 0)\}$ . Define  $W : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}$  at  $x \in \mathcal{X}, y \in \mathcal{Y}$  by  $W(x, y) = 1$  if and only if  $x_1 \vee x_2 \vee x_3 = y_1 + y_2 + y_3 \pmod{2}$ . The GHZ game refers to the 3-player game  $(\mathcal{X}, \mathcal{Y}, Q, W)$ , which has value  $3/4$ . The  $n$ -fold repeated GHZ game refers to the  $n$ -fold parallel repetition of  $(\mathcal{X}, \mathcal{Y}, Q, W)$ . Our parallel repetition results easily generalize with any other (constant-sized) answer alphabet  $\mathcal{Y}'$  and any predicate  $W'$ , as long as the game  $(\mathcal{X}, \mathcal{Y}', Q, W')$  has value less than 1.

We typically use  $X = (X_1, X_2, X_3) \in \mathcal{X}^n$  to denote a random variable distributed according to  $Q^n$  where  $X_i \in \mathcal{X}_i^n$  denotes the input to the  $i$ -th player.

## 3 Partitioning into Pseudorandom Subspaces

We make use of the notion of affine partition similar to the one defined in [18]. We say that  $\Pi$  is an affine partition of  $(\mathbb{F}_2^n)^3$  of codimension at most  $d$  if  $\Pi$  is a partition on  $(\mathbb{F}_2^n)^3$  and:

- Each part  $\pi \in \Pi$  has the form  $a_\pi + \mathcal{V}_\pi^3$  where  $\mathcal{V}_\pi$  is a subspace of  $\mathbb{F}_2^n$  and  $a_\pi \in (\mathbb{F}_2^n)^3$ , and
- Each  $\mathcal{V}_\pi$  has codimension at most  $d$ .

The main take-away from this section is Proposition 15, which states the following: Given the query distribution to the  $n$ -fold GHZ game, and a product event  $E \subseteq (\mathbb{F}_2^n)^3$  with large enough probability mass, we can find an affine partition  $\Pi$  of  $(\mathbb{F}_2^n)^3$  such that on a typical part  $\pi \in \Pi$ , the non-zero Fourier coefficients of the indicator functions  $E_1|_{\pi_1}, E_2|_{\pi_2}, E_3|_{\pi_3}$  are small. Recall that  $E_i|_{\pi_i} : \pi_i \rightarrow \{0, 1\}$  is the indicator function of the set  $E_i \cap \pi_i \subseteq \pi_i$ .

Formally, the proposition is as follows:

► **Proposition 15.** Let  $\mathcal{P} = Q^n$ . Let  $E = E_1 \times E_2 \times E_3 \subseteq (\mathbb{F}_2^n)^3$  be such that  $\mathcal{P}(E) = \alpha$ . For all  $\delta > 0$ , there exists an affine partition  $\Pi$  of  $(\mathbb{F}_2^n)^3$  of codimension at most  $\frac{3}{\delta^3}$  such that the following holds. With probability at least  $1 - \frac{\delta}{\alpha}$  over  $\pi \sim \Pi(\mathcal{P}|E)$ , for all  $i \in [3]$  and non-zero  $\chi \in \widehat{\mathcal{V}}$ , we have  $\left| \widehat{E_i|_{\pi_i}}(\chi) \right| \leq \delta$ , where  $\pi$  is of the form  $\pi_1 \times \pi_2 \times \pi_3$  for affine shifts  $\pi_1, \pi_2, \pi_3$  of some subspace  $\mathcal{V}$  of  $\mathbb{F}_2^n$ .

Recall that  $\Pi(\mathcal{P}|E)$  is the distribution induced by sampling  $x \sim \mathcal{P}|E$  and outputting the part of  $\Pi$  to which  $x$  belongs. Note that in the statement of the proposition, we don't specify a choice of Fourier basis for  $\pi_i$ . This is because for any set  $S \subseteq \pi_i$ , the quantity  $\left| \widehat{S}(\chi_{a_i}) \right|$  is

independent of choice of  $a_i \in \pi_i$  so we simply write  $|\widehat{S}(\chi)|$ . The proof of Proposition 15 is similar in nature to the proof of Lemma 6.2 in [18], but is much simpler and is presented in the full version of the paper.

#### 4 Key Fourier Analytic Lemmas

We crucially make use of the following lemma, the proof of which can be found in the full version of the paper.

► **Lemma 16.** *Let  $\mathcal{V} \subseteq \mathbb{F}_2^n$  be a subspace and  $a_1, a_2, a_3 \in \mathbb{F}_2^n$  be such that  $a_1 + a_2 + a_3 = 0$ . Let  $\pi = \pi_1 \times \pi_2 \times \pi_3$  where  $\pi_i = a_i + \mathcal{V}$ . Let  $A \subseteq \pi_1, B \subseteq \pi_2, C \subseteq \pi_3$  be sets such that for all non-zero  $\chi \in \widehat{\mathcal{V}}$ , we have  $|\widehat{C}(\chi)| \leq \delta_1$ . Then,*

$$\left| \mathbb{E}_{\substack{z \sim \pi_3 \\ x \sim \pi_1}} [A(x) \cdot B(x+z) \cdot C(z)] - \mu_{\pi_1}(A) \cdot \mu_{\pi_2}(B) \cdot \mu_{\pi_3}(C) \right| \leq \delta_1.$$

If furthermore for all non-zero  $\chi \in \widehat{\mathcal{V}}$ , we have  $|\widehat{B}(\chi)| \leq \delta_2$ , then

$$\left| \mathbb{E}_{z \sim \pi_3} \left[ \left( \mathbb{E}_{x \sim \pi_1} [A(x) \cdot B(x+z)] \right)^2 \cdot C(z) \right] - \mu_{\pi_1}(A)^2 \cdot \mu_{\pi_2}(B)^2 \cdot \mu_{\pi_3}(C) \right| \leq \delta_2^2 + \delta_1.$$

Recall from Section 2.2 that  $\mu_{\pi_i}(S) \triangleq \frac{|S \cap \pi_i|}{|\pi_i|}$ . In the statement of this lemma, we don't specify a choice of Fourier basis for  $\pi_2$  and  $\pi_3$ . Since the properties  $|\widehat{C}(\chi_{a_3})| \leq \delta_1$  and  $|\widehat{B}(\chi_{a_2})| \leq \delta_2$  are independent of the choice of  $a_2$  and  $a_3$ , we simply write  $|\widehat{C}(\chi)| \leq \delta_1$  and  $|\widehat{B}(\chi)| \leq \delta_2$ .

#### 5 Main Proof

We use the following Parallel Repetition Criterion which is similar to, but weaker than the one from [18] for the GHZ game and has a slightly simpler proof.

Let  $\mathcal{G}$  refer to the  $n$ -fold parallel repetition of the GHZ game. Let  $\mathcal{P} = \mathcal{Q}^n$ .

► **Lemma 17 (Parallel Repetition Criterion).** *Let  $c \in (0, 1]$  be a constant and  $\rho(n) : \mathbb{N} \rightarrow \mathbb{R}$  be a function such that  $\rho(n) \geq \exp(-n)$ . Suppose for all large  $n \in \mathbb{N}$  and all subsets  $E_1, E_2, E_3 \subseteq \mathbb{F}_2^n$  such that  $\mathcal{P}(E) \geq \rho(n)$  where  $E = E_1 \times E_2 \times E_3$ , we have  $\mathbb{E}_{i \sim [n]} [\text{val}^{(i)}(\mathcal{G}|E)] \leq 1 - c$ . Then,*

$$\text{val}(\mathcal{G}) \leq \rho(n)^{\Omega(1)}.$$

This lemma is proved in [18] under the weaker assumption that there is *some* coordinate  $i \in [n]$  for which  $\text{val}^{(i)}(\mathcal{G}|E) \leq 1 - c$ . The proof is slightly simpler under our stronger assumption that  $\mathbb{E}_{i \sim [n]} [\text{val}^{(i)}(\mathcal{G}|E)] \leq 1 - c$ . We prove this in Appendix A.1.

Given this criterion, our goal of showing an inverse polynomial bound for  $\text{val}(\mathcal{G})$  reduces to showing the following. Let  $E = E_1 \times E_2 \times E_3$  be any event such that  $\mathcal{P}(E) = \alpha \geq \frac{1}{n^{1/100}}$  and  $n$  be large enough. It suffices to show that  $\mathbb{E}_{i \sim [n]} [\text{val}^{(i)}(\mathcal{G}|E)] \leq 0.95$ . We do this as follows.

Let  $\delta = \frac{\alpha^{20}}{n^{1/40}}$ . Proposition 15 implies the existence of a partition  $\Pi$  of  $(\mathbb{F}_2^n)^3$  into affine subspaces of codimension at most  $O\left(\frac{1}{\delta^3}\right) = o(n)$  such that:

## 62:12 Parallel Repetition for the GHZ Game: A Simpler Proof

- Every  $\pi \in \Pi$  is of the form  $a + \mathcal{V}^3$  where  $\mathcal{V} \subseteq \mathbb{F}_2^n$  is a subspace and  $a \in (\mathbb{F}_2^n)^3$ .
- With probability at least  $1 - \frac{\delta}{\mathcal{P}(E)} \geq 1 - o(1)$  over  $\pi \sim \Pi(\mathcal{P}|E)$ , we have  $\left| \widehat{E_i|_{\pi_i}}(\chi) \right| \leq \delta$  for all  $i \in [3]$  and non-zero  $\chi \in \widehat{\mathcal{V}}$ , where  $\mathcal{V}$  is the subspace of  $\mathbb{F}_2^n$  for which  $\pi$  is an affine shift of  $\mathcal{V}^3$ .

Under the distribution  $\Pi(\mathcal{P}|E)$ , the probability that  $\pi$  is sampled equals  $\frac{(\mathcal{P}|\pi)(E) \cdot \mathcal{P}(\pi)}{\mathcal{P}(E)}$  by Bayes' rule. This implies that the probability that  $\pi \sim \Pi(\mathcal{P}|E)$  satisfies  $(\mathcal{P}|\pi)(E) \leq \mathcal{P}(E)/10$  is at most  $1/10$ . We will focus on  $\pi = \pi_1 \times \pi_2 \times \pi_3$  that satisfy both these properties, namely, the measure of  $E$  under  $\mathcal{P}|\pi$  is significant, furthermore, for all  $i \in [3]$ , all non-zero Fourier coefficients of the sets  $E_i$  restricted to  $\pi_i$  are small.

► **Definition 18.** *We say that  $\pi$  is good if*

$$(\mathcal{P}|\pi)(E) \geq \alpha/10, \text{ and for all non-zero } \chi \in \widehat{\mathcal{V}} \text{ and } i \in [3], \text{ we have } \left| \widehat{E_i|_{\pi_i}}(\chi) \right| \leq \delta. \quad (3)$$

By a union bound, a random  $\pi \sim \Pi(\mathcal{P}|E)$  will be good with probability at least  $1 - \frac{1}{10} - \frac{\delta}{\alpha}$ . Fix any such good  $\pi = \pi_1 \times \pi_2 \times \pi_3 \in \Pi$ , and let  $\mathcal{V}$  be the subspace such that  $\pi$  is an affine shift of  $\mathcal{V}^3$ .

For all  $z \in E_3 \cap \pi_3$ , define a (partial) matching  $M_z$  between  $\pi_1$  and  $\pi_2$  as follows. For  $x \in \pi_1 \cap E_1, y \in \pi_2 \cap E_2, z \in \pi_3 \cap E_3$  such that  $x + y = z$ , put an edge  $(x, y)$ . Let  $L_z$  (resp.  $R_z$ ) be the left (resp. right) endpoints of  $M_z$ . Let  $G = \cup_{z \in E_3 \cap \pi_3} M_z$  be the bipartite graph between  $\pi_1$  and  $\pi_2$  obtained by combining edges from the matchings for  $z \in E_3 \cap \pi_3$ . Let  $E(G)$  denote the set of edges in  $G$ . For every edge  $e \in E(G)$ , we can identify  $e$  with a valid input to the  $n$ -fold GHZ game that is contained in  $E \cap \pi$ . Namely, we associate  $(x_0, y_0) \in E(G)$  to the input  $(x_0, y_0, x_0 + y_0) \in \text{supp}(\mathcal{P}) \cap E \cap \pi$ . This is a bijective correspondence because of the way we defined the graph  $G$ . Under this correspondence, the uniform distribution over edges of  $G$  corresponds to the distribution  $\mathcal{P}|E, \pi$ . We now introduce the important notion of a bow tie.

► **Definition 19 (Bow Tie).** *We say that a subset of edges  $b \subseteq E(G)$  is a bow tie if  $b = \{x_0, x_1\} \times \{y_0, y_1\}$  for some  $x_0 \neq x_1 \in \pi_1, y_0 \neq y_1 \in \pi_2$  such that  $x_0 + y_0 = x_1 + y_1$  (or equivalently  $x_0 + y_1 = x_1 + y_0$ ). Alternatively, for  $z_0 = x_0 + y_0$  and  $z_1 = x_0 + y_1$ , we have  $(x_i, y_j, z_k) \in \text{supp}(\mathcal{P})$  for all  $(i, j, k) \in \text{supp}(\mathcal{Q})$ .*

Let  $b = \{x_0, x_1\} \times \{y_0, y_1\}$  be a bow tie. As before, we identify  $b$  with the indicator vector  $b \in \{0, 1\}^{E(G)}$  of the edges of  $b$ , that is,  $b(e) = 1$  iff  $e \in \{(x_i, y_j) : i, j \in \{0, 1\}\}$ . We use  $\tilde{b}$  to denote the uniform distribution on the edges of the bow tie, when viewed as inputs to the  $n$ -fold GHZ game. More precisely,  $\tilde{b}$  denotes the uniform distribution on  $\{(x_i, y_j, x_i + y_j) \mid i, j \in \{0, 1\}\}$ .

We say that  $b$  differs in the  $i$ -th coordinate for  $i \in [n]$  if  $x_0(i) \neq x_1(i)$ , or equivalently,  $y_0(i) \neq y_1(i)$ , or equivalently,  $z_0(i) \neq z_1(i)$ .

Let  $b$  be a bow tie and  $I \subseteq [n]$  be the coordinates on which  $b$  differs. The following claim shows that  $\text{val}^{(i)}(\mathcal{G}\tilde{b}) \leq 3/4$  for all  $i \in I$ . The proof is deferred to Appendix A.2

▷ **Claim 20.** Let  $b = \{x_0, x_1\} \times \{y_0, y_1\}$  be a bow tie. Let  $I \subseteq [n]$  be the subset of coordinates on which  $b$  differs. Then,  $\text{val}^{(i)}(\mathcal{G}\tilde{b}) \leq 3/4$  for all  $i \in I$ .

Let  $B$  denote the set of all bow ties. Consider the distribution on edges defined by first sampling a uniformly random bow tie from  $B$ , and then a uniformly random edge from the bow tie. We now provide an alternate description of this distribution. For each  $z \in E_3 \cap \pi_3$ , define  $1_z \in \{0, 1\}^{|E(G)|}$  as follows. For each  $e = (x, y) \in E(G)$ , define  $1_z(e) = 1$  if  $x$  and  $y$



are both matched in  $M_z$  but not to each other, and define  $1_z(e) = 0$  otherwise. Alternatively,  $1_z$  is the indicator of the set  $((L_z \times R_z) \setminus M_z) \cap E(G)$ . Let  $v := \mathbb{E}_{z \sim E_3 \cap \pi_3}[1_z]$ . Note that  $v$  has  $|E(G)|$  coordinates, each of which have non-negative values, so  $v$  induces a distribution on  $E(G)$ . Consider this distribution  $\tilde{v} = \frac{v}{\|v\|_1}$  on  $E(G)$  defined by normalizing  $v$ . We show that this distribution is an alternate description of the aforementioned distribution.

▷ **Claim 21.**  $v = |E_3 \cap \pi_3|^{-1} \cdot (\sum_{b \in B} b)$ . In particular, we can think of the distribution  $\tilde{v} := \frac{v}{\|v\|_1}$  on  $E(G)$  as obtained by sampling a uniformly random bow tie  $b$  in  $G$  and outputting a uniformly random edge of  $b$ .

The proof of this is deferred to Appendix A.3. Our goal now is to show that the distribution  $\tilde{v}$  is close to the uniform distribution over edges of  $G$ . To do so, we study some properties of  $G$ . Observe that  $|E(G)| \triangleq |\mathcal{V}|^2 \cdot \mathbb{E}_{\substack{x \sim \pi_1 \\ z \sim \pi_3}}[E_1(x) \cdot E_2(x+z) \cdot E_3(z)]$ . We apply Lemma 16 with parameters  $A = E_1 \cap \pi_1, B = E_2 \cap \pi_2, C = E_3 \cap \pi_3$ . Since  $\pi \in \text{supp}(\Pi(\mathcal{P}|E))$ , the set  $\pi \cap \text{supp}(\mathcal{P})$  is non-empty, therefore, we may choose  $a \in \text{supp}(\mathcal{P})$  so that  $\pi = a + \mathcal{V}^3$ . This, along with Equation (3) implies that the first hypothesis of Lemma 16 is satisfied. Lemma 16 implies that

$$\left| |E(G)| - |\mathcal{V}|^2 \cdot \mu_{\pi_1}(E_1) \cdot \mu_{\pi_2}(E_2) \cdot \mu_{\pi_3}(E_3) \right| \leq |\mathcal{V}|^2 \cdot \delta. \quad (4)$$

We make use of the following bounds on the  $\ell_1$  and  $\ell_2$  norms of  $v$ . The proofs of these are by Fourier analysis and are deferred to Appendices A.4 and A.5.

▷ **Claim 22.**

$$\begin{aligned} \|v\|_1 &\geq |\mathcal{V}|^2 \cdot (\mu_{\pi_1}(E_1)^2 \cdot \mu_{\pi_2}(E_2)^2 \cdot \mu_{\pi_3}(E_3) - 3 \cdot \delta) \\ &\quad - |\mathcal{V}| \cdot (\mu_{\pi_1}(E_1) \cdot \mu_{\pi_2}(E_2) + 2 \cdot \delta \cdot \mu_{\pi_3}(E_3)^{-1}) \end{aligned} \quad (5)$$

▷ **Claim 23.**

$$\|v\|_2^2 \leq |\mathcal{V}|^2 \cdot \left( \mu_{\pi_1}(E_1)^3 \cdot \mu_{\pi_2}(E_2)^3 \cdot \mu_{\pi_3}(E_3) + 10 \cdot \sqrt{\delta} \right) \quad (6)$$

We now bound  $\|\tilde{v}\|_2 = \frac{\|v\|_2}{\|v\|_1}$  by plugging in appropriate bounds on  $\delta$  and dividing Equation (6) by Equation (5). Our choice of  $\delta = \alpha^{20}/n^{1/40}$ , and our assumption that  $\alpha/10 \leq (\mathcal{P}|\pi)(E)$  (which in turn is at most  $\min_{i \in [3]} (\mu_{\pi_i}(E_i))$ ) implies that  $\delta$  is much smaller than any  $\mu_{\pi_i}(E_i)$ . In particular, we highlight that

$$\begin{aligned} \sqrt{\delta} &= o(\mu_{\pi_1}(E_1)^3 \cdot \mu_{\pi_2}(E_2)^3 \cdot \mu_{\pi_3}(E_3)) \\ \delta &= o(\mu_{\pi_1}(E_1)^2 \cdot \mu_{\pi_2}(E_2)^2 \cdot \mu_{\pi_3}(E_3)) \\ \delta &= o(\mu_{\pi_1}(E_1) \cdot \mu_{\pi_2}(E_2) \cdot \mu_{\pi_3}(E_3)) \end{aligned}$$

Furthermore, since  $|\mathcal{V}| = 2^{\Omega(n)}$  and  $1 \geq \mu_{\pi_i}(E_i) = \Omega(\alpha) = n^{-O(1)}$ , we have

$$|\mathcal{V}| \cdot \mu_{\pi_1}(E_1) \cdot \mu_{\pi_2}(E_2) = o(|\mathcal{V}|^2 \cdot \mu_{\pi_1}(E_1)^2 \cdot \mu_{\pi_2}(E_2)^2 \cdot \mu_{\pi_3}(E_3)).$$

Thus the dominant term on the right-hand side of Equation (5) is  $|\mathcal{V}|^2 \cdot \mu_{\pi_1}(E_1)^2 \cdot \mu_{\pi_2}(E_2)^2 \cdot \mu_{\pi_3}(E_3)$ , and the dominant term on the right-hand side of Equation (6) is  $|\mathcal{V}|^2 \cdot \mu_{\pi_1}(E_1)^3 \cdot \mu_{\pi_2}(E_2)^3 \cdot \mu_{\pi_3}(E_3)$ . More precisely, we have

$$\|v\|_1 \geq (1 - o(1)) \cdot |\mathcal{V}|^2 \cdot \mu_{\pi_1}(E_1)^2 \cdot \mu_{\pi_2}(E_2)^2 \cdot \mu_{\pi_3}(E_3) \quad (7)$$

$$\|v\|_2^2 \leq (1 + o(1)) \cdot |\mathcal{V}|^2 \cdot \mu_{\pi_1}(E_1)^3 \cdot \mu_{\pi_2}(E_2)^3 \cdot \mu_{\pi_3}(E_3). \quad (8)$$

This implies that

$$\|\tilde{v}\|_2^2 = \frac{\|v\|_2^2}{\|v\|_1^2} \leq \frac{1 + o(1)}{|\mathcal{V}|^2 \cdot \mu_{\pi_1}(E_1) \cdot \mu_{\pi_2}(E_2) \cdot \mu_{\pi_3}(E_3)} \quad (9)$$

In comparison, Equation (4) gave that

$$|E(G)| \in (1 \pm o(1)) \cdot |\mathcal{V}|^2 \cdot \mu_{\pi_1}(E_1) \cdot \mu_{\pi_2}(E_2) \cdot \mu_{\pi_3}(E_3).$$

Thus we can rewrite Equation (9) as

$$\|\tilde{v}\|_2 \leq \frac{1 + o(1)}{\sqrt{|E(G)|}} \quad (10)$$

This, together with the fact that by construction  $\|\tilde{v}\|_1 = 1$ , is sufficient to deduce that  $\tilde{v}$  is close to the “uniform distribution” vector  $\tilde{u} \stackrel{\text{def}}{=} (\frac{1}{|E(G)|}, \dots, \frac{1}{|E(G)|})$ . More formally, we have:

► **Fact 24.** *Suppose that  $\tilde{v} \in \mathbb{R}^m$  is an  $m$ -dimensional vector such that  $\|\tilde{v}\|_1 = 1$ , and  $\|\tilde{v}\|_2 = \frac{1+\beta}{\sqrt{m}}$  for some  $\beta \in [0, 1]$ . Then*

$$\|\tilde{v} - \tilde{u}\|_1 \leq \sqrt{3\beta},$$

where  $\tilde{u}$  denotes the vector  $(\frac{1}{m}, \dots, \frac{1}{m})$ .

The proof of Fact 24 is deferred to Appendix A.6

Applying Fact 24 to Equation (10) shows that  $d_{\text{TV}}(\tilde{v}, \tilde{u}) = o(1)$ . In other words, a uniformly random edge of a uniformly random bow tie is distributed close to uniformly on  $E(G)$ .

We now show that a typical bow tie differs in a considerable fraction of coordinates.

▷ **Claim 25.**  $\Pr_{\substack{i \sim [n] \\ b \sim B}}[b \text{ differs in } i\text{-th coordinate}] \geq 1/3 - o(1)$ .

The proof of Claim 25 is deferred to Appendix A.7.

Claim 20, along with Claim 25 implies that  $\Pr_{\substack{i \sim [n] \\ b \sim B}}[\text{val}^{(i)}(\mathcal{G}|\tilde{b}) \leq 3/4] \geq 1/3 - o(1) \geq 0.3$ . For those  $i \in [n]$  and  $b \in B$  such that  $b$  doesn't differ at the  $i$ -th coordinate, we bound  $\text{val}^{(i)}(\mathcal{G}|\tilde{b})$  by 1. This, along with Claim 21 implies that  $\mathbb{E}_{i \sim [n]} \left[ \text{val}^{(i)}(\mathcal{G}|\tilde{v}) \right] \leq \mathbb{E}_{\substack{i \sim [n] \\ b \sim B}}[\text{val}^{(i)}(\mathcal{G}|\tilde{b})] \leq 0.75 \times 0.3 + 1 \times 0.7 \leq 0.925$ . Since  $d_{\text{TV}}(\tilde{u}, \tilde{v}) \leq o(1)$  and  $\tilde{u}$  corresponds to  $\mathcal{P}|\pi, E$ , this implies that  $\mathbb{E}_{i \sim [n]} \left[ \text{val}^{(i)}(\mathcal{G}|\pi, E) \right] = 0.925 + o(1) \leq 0.93$ . Since  $\pi \sim \Pi(\mathcal{P}|E)$  is good with probability at least  $1 - \delta \cdot \alpha^{-1} - 1/10 \geq 0.9 - o(1) \geq 0.8$ , we have  $\mathbb{E}_{i \sim [n]} \left[ \text{val}^{(i)}(\mathcal{G}|E) \right] \leq \mathbb{E}_{\substack{i \sim [n] \\ \pi \sim \Pi(\mathcal{P}|E)}} \left[ \text{val}^{(i)}(\mathcal{G}|E, \pi) \right] \leq 0.8 \times 0.93 + 0.2 \times 1 < 0.95$ . This, along with Lemma 17 completes the proof.

---

## References

- 1 Noga Alon and Bo'az Klartag. Economical toric spines via Cheeger's inequality. *J. Topol. Anal.*, 1(2):101–111, 2009.
- 2 Boaz Barak, Mark Braverman, Xi Chen, and Anup Rao. How to compress interactive communication. *SIAM J. Comput.*, 42(3):1327–1363, 2013. (also in STOC 2010).

- 3 Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, PCPs, and nonapproximability—towards tight results. *SIAM J. Comput.*, 27(3):804–915, 1998. (also in FOCS 1995).
- 4 Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *STOC*, pages 113–131, 1988.
- 5 Mark Braverman and Ankit Garg. Small value parallel repetition for general games. In *STOC*, pages 335–340, 2015.
- 6 Richard Cleve, Peter Høyer, Benjamin Toner, and John Watrous. Consequences and limits of nonlocal strategies. In *CCC*, pages 236–249, 2004.
- 7 Irit Dinur, Prahladh Harsha, Rakesh Venkat, and Henry Yuen. Multiplayer parallel repetition for expanding games. In *ITCS*, volume 67 of *LIPICs*, pages Art. No. 37, 16, 2017.
- 8 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *STOC*, pages 624–633, 2014.
- 9 Uriel Feige. On the success probability of the two provers in one-round proof systems. In *CCC*, pages 116–123. IEEE Computer Society, 1991.
- 10 Uriel Feige. A threshold of  $\ln n$  for approximating set cover. *J. ACM*, 45(4):634–652, 1998. (also in STOC 1996).
- 11 Uriel Feige, Guy Kindler, and Ryan O’Donnell. Understanding parallel repetition requires understanding foams. In *CCC*, pages 179–192, 2007.
- 12 Uriel Feige and Oleg Verbitsky. Error reduction by parallel repetition - A negative result. *Comb.*, 22(4):461–478, 2002.
- 13 Lance Fortnow, John Rompel, and Michael Sipser. On the power of multi-power interactive protocols. In *CCC*, pages 156–161. IEEE Computer Society, 1988.
- 14 Lance Jeremy Fortnow. *Complexity-theoretic aspects of interactive proof systems*. PhD thesis, MIT, 1989.
- 15 Daniel M. Greenberger, Michael A. Horne, and Anton Zeilinger. *Going Beyond Bell’s Theorem*, pages 69–72. Springer Netherlands, Dordrecht, 1989.
- 16 Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001. (also in STOC 1997).
- 17 Thomas Holenstein. Parallel repetition: simplifications and the no-signaling case. *Theory Comput.*, 5:141–172, 2009. (also in STOC 2007).
- 18 Justin Holmgren and Ran Raz. A parallel repetition theorem for the GHZ game. *CoRR*, abs/2008.05059, 2020. URL: <https://arxiv.org/abs/2008.05059>.
- 19 Justin Holmgren and Lisa Yang. The parallel repetition of non-signaling games: counterexamples and dichotomy. In *STOC*, pages 185–192. ACM, 2019.
- 20 Guy Kindler, Ryan O’Donnell, Anup Rao, and Avi Wigderson. Spherical cubes and rounding in high dimensions. In *FOCS*, pages 189–198, 2008.
- 21 Kunal Mittal and Ran Raz. Block rigidity: Strong multiplayer parallel repetition implies super-linear lower bounds for turing machines. In *ITCS*, volume 185 of *LIPICs*, pages 71:1–71:15, 2021.
- 22 Itzhak Parnafes, Ran Raz, and Avi Wigderson. Direct product results and the GCD problem, in old and new communication models. In *STOC*, pages 363–372. 1997.
- 23 Ran Raz. A parallel repetition theorem. *SIAM J. Comput.*, 27(3):763–803, 1998. (also in STOC 1995).
- 24 Ran Raz. Parallel repetition of two prover games. In *CCC*, pages 3–6. 2010.
- 25 Ran Raz. A counterexample to strong parallel repetition. *SIAM J. Comput.*, 40(3):771–777, 2011.
- 26 Oleg Verbitsky. Towards the parallel repetition conjecture. In *CCC*, pages 304–307. IEEE Computer Society, 1994.

## A Appendix

### A.1 Proof of Lemma 17

**Proof of Lemma 17.** Let  $\mathcal{P} = \mathcal{Q}^n$ . Choose the largest integer  $m \geq 0$  such that  $32^{-m} \geq \rho(n) \cdot \frac{2}{c}$ . Note that  $m = \Theta(\log(1/\rho(n)))$ . Fix any deterministic product strategy  $\bar{f} = (\bar{f}_1, \bar{f}_2, \bar{f}_3)$  for the players where  $\bar{f}_i : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$  denotes the strategy for the  $i$ -th player. Let  $Y_i = \bar{f}_i(X_i) \in \mathbb{F}_2^n$  denote the output of player  $i$  on input  $X_i$ . Let  $\{j_1, \dots, j_m\} \subseteq [n]$  be a set of coordinates. Let  $W_i$  denote the event of winning the GHZ game in the  $j_i$ -th coordinate under the strategy  $\bar{f}$  and let  $W_{\leq i} := W_1 \wedge \dots \wedge W_i$ . Observe that

$$\text{val}(\mathcal{G}, \bar{f}) \leq \prod_{i=0}^{m-1} \Pr[W_{i+1} \mid W_{\leq i}].$$

We show how to construct a sequence of coordinates so that every term in the above product is at most  $1 - c/2$ . This would imply that  $\text{val}(\mathcal{G}) \leq (1 - c/2)^{\Theta(\log(1/\rho(n)))} = \rho(n)^{\Omega(1)}$ . Fix any  $i \in \{0, \dots, m-1\}$  and assume that we have found  $j_1, \dots, j_i$ . Let  $X \sim \mathcal{P}$  and  $X_{\leq i}$  denote  $X$  restricted to the coordinates  $\{j_1, \dots, j_i\}$ . Let  $Y_{\leq i}$  denote the outputs of the players restricted to the coordinates  $\{j_1, \dots, j_i\}$ . Let  $Z_{\leq i} = (X_{\leq i}, Y_{\leq i})$ . Since  $W_{\leq i}$  is a function of  $Z_{\leq i}$ , we have

$$\begin{aligned} \Pr[W_{i+1} \mid W_{\leq i}] &= \mathbb{E}_{z_{\leq i} \sim Z_{\leq i} \mid W_{\leq i}} [\Pr[W_{i+1} \mid Z_{\leq i} = z_{\leq i}]] \\ &\leq \mathbb{E}_{z_{\leq i} \sim Z_{\leq i} \mid W_{\leq i}} [\text{val}^{(j_{i+1})}(\mathcal{G} \mid Z_{\leq i} = z_{\leq i})]. \end{aligned} \quad (11)$$

Let  $F = F(z_{\leq i})$  denote the event that  $\mathcal{P}[Z_{\leq i} = z_{\leq i} \mid W_{\leq i}] \geq \frac{c}{2} \cdot \frac{1}{N}$  where  $N = 32^i \geq \text{supp}(Z_{\leq i})$ . We argue that  $F$  occurs with probability at least  $1 - c/2$ . This is because we are sampling  $z_{\leq i}$  with probability  $\mathcal{P}[Z_{\leq i} = z_{\leq i} \mid W_{\leq i}]$ , hence the measure of  $z_{\leq i}$  for which  $\mathcal{P}[Z_{\leq i} = z_{\leq i} \mid W_{\leq i}] \leq \frac{c}{2} \cdot \frac{1}{N}$  is at most  $\frac{c}{2}$ . Fix any  $z_{\leq i}$  such that  $F$  holds. Our choice of  $m$  implies that  $\frac{1}{N} \cdot \frac{c}{2} \geq \rho(n)$ . Note that we can express the distribution  $\mathcal{P} \mid Z_{\leq i} = z_{\leq i}$  as  $\mathcal{P} \mid E$  where  $E = E_1 \times E_2 \times E_3$  for  $E_1, E_2, E_3 \subseteq \mathbb{F}_2^n$  and  $\mathcal{P}(E) \geq \rho(n)$ . The hypothesis of Lemma 17 implies that  $\mathbb{E}_{j \sim [n]} [\text{val}^{(j)}(\mathcal{G} \mid Z_{\leq i} = z_{\leq i})] \leq 1 - c$ . This implies that

$$\begin{aligned} \mathbb{E}_{\substack{z_{\leq i} \sim Z_{\leq i} \mid W_{\leq i} \\ j \sim [n]}} [\text{val}^{(j)}(\mathcal{G} \mid Z_{\leq i} = z_{\leq i})] &\leq \Pr_{z_{\leq i} \sim Z_{\leq i} \mid W_{\leq i}} [\neg F] \\ &\quad + \mathbb{E}_{\substack{z_{\leq i} \sim Z_{\leq i} \mid W_{\leq i}, F \\ j \sim [n]}} [\text{val}^{(j)}(\mathcal{G} \mid Z_{\leq i} = z_{\leq i})] \\ &\leq \frac{c}{2} + 1 - c = 1 - \frac{c}{2}. \end{aligned}$$

By linearity of expectation, we can fix a  $j \in [n]$  such that  $\mathbb{E}_{z_{\leq i} \sim Z_{\leq i} \mid W_{\leq i}} [\text{val}^{(j)}(\mathcal{G} \mid Z_{\leq i} = z_{\leq i})] \leq 1 - \frac{c}{2}$ . Note that  $j \notin \{j_1, \dots, j_i\}$  since we already win the game on these coordinates. This, along with Equation (11) completes the proof.  $\blacktriangleleft$

### A.2 Proof of Claim 20

**Proof of Claim 20.** Let  $i \in I$ . Since the bow tie  $b$  differs in the  $i$ -th coordinate, we have

$$\{x_0(i), x_1(i)\} = \{y_0(i), y_1(i)\} = \{z_0(i), z_1(i)\} = \{0, 1\}.$$

We may thus assume without loss of generality that  $x_0(i) = y_0(i) = 0$ . Define embeddings  $\phi_1 : \mathbb{F}_2 \rightarrow \{x_0, x_1\}$ ,  $\phi_2 : \mathbb{F}_2 \rightarrow \{y_0, y_1\}$  and  $\phi_3 : \mathbb{F}_2 \rightarrow \{z_0, z_1\}$  at  $a \in \mathbb{F}_2$  by  $\phi_1(a) = x_a$ ,  $\phi_2(a) = y_a$  and  $\phi_3(a) = z_a$ . It follows for all  $a \in \{0, 1\}$  and  $j \in [3]$ , we have  $(\phi_j(a))(i) = a$ . In particular, for  $\phi = \phi_1 \times \phi_2 \times \phi_3$ , the distribution  $\phi(\mathcal{Q})$  is exactly the distribution  $\tilde{b}$ . Given any strategies  $\bar{f}_1, \bar{f}_2, \bar{f}_3 : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$  for the players for the  $n$ -fold GHZ game restricted to the query distribution  $\tilde{b}$ , the functions  $\phi_1, \phi_2, \phi_3$  induce a strategy for the GHZ game as follows. Define  $f_j : \mathbb{F}_2 \rightarrow \mathbb{F}_2$  by  $f_j(a) = (\bar{f}_j(\phi_j(a)))(i)$ . The success probability of the strategy  $f_1 \times f_2 \times f_3$  on the distribution  $\mathcal{Q}$  is exactly the success probability in the  $i$ -th coordinate of the strategy  $\bar{f}_1 \times \bar{f}_2 \times \bar{f}_3$  on the distribution  $\tilde{b}$ . It follows that  $\text{val}^{(i)}(\mathcal{G}|\tilde{b}) \leq 3/4$ .  $\triangleleft$

### A.3 Proof of Claim 21

Proof of Claim 21. Fix any  $e \in E(G)$ ,  $e = (x_0, y_0)$ . This implies that  $x_0 \in E_1 \cap \pi_1$ ,  $y_0 \in E_2 \cap \pi_2$  and  $z_0 := x_0 + y_0 \in E_3 \cap \pi_3$ . Note that  $v(e) = \Pr_{z \sim E_3 \cap \pi_3} [(x_0, y_0) \in (L_z \times R_z) \setminus M_z]$ . For any  $z_1 \in E_3 \cap \pi_3$ ,

$$\begin{aligned} e \in (L_{z_1} \times R_{z_1}) \setminus M_{z_1} &\iff x_0 + z_1 \in E_2 \cap \pi_2, y_0 + z_1 \in E_1 \cap \pi_1, z_1 \neq z_0 \\ &\iff x_0, x_1 \in E_1 \cap \pi_1, y_0, y_1 \in E_2 \cap \pi_2, z_1 \neq z_0 \in E_3 \cap \pi_3 \\ &\text{where } x_1 := y_0 + z_1, y_1 := x_0 + z_1 \\ &\iff \{x_0, x_1\} \times \{y_0, y_1\} \text{ is a bow tie} \\ &\text{where } x_1 := y_0 + z_1, y_1 := x_0 + z_1. \end{aligned}$$

This implies that for all  $e = (x_0, y_0) \in E(G)$  and  $z_1 \in E_3 \cap \pi_3$ , we have  $1_{z_1}(e) = 1$  if and only if  $b = \{x_0, x_1\} \times \{y_0, y_1\}$  is a bow tie. Observe that as we vary  $z_1 \in E_3 \cap \pi_3$ , we obtain all possible bow ties that contain the edge  $e$ , i.e. the bow ties  $b$  for which  $b(e) \neq 0$ . This implies that  $v \triangleq \mathbb{E}_{z_1 \sim E_3 \cap \pi_3} [1_z] = |E_3 \cap \pi_3|^{-1} \cdot (\sum_{b \in B} b)$ .  $\triangleleft$

### A.4 Proof of Claim 22

For ease of notation, we define weight functions as follows.

► **Definition 26** (Weight functions). Let  $\mathcal{P} = \mathcal{Q}^n$ . For  $z \in \pi_3$ , let

$$\text{wt}_\pi(z) := \Pr_{X \sim \mathcal{P}} [(X_1 \in E_1 \text{ and } X_2 \in E_2) | (X \in \pi \text{ and } X_3 = z)] = \mathbb{E}_{x \sim \pi_1} [E_1(x)E_2(x+z)].$$

Proof of Claim 22. Let  $z \in E_3 \cap \pi_3$ . Note that  $\text{wt}_\pi(z) = \mu_{\pi_1}(L_z) = \mu_{\pi_2}(R_z)$ . Observe that  $\|1_z\|_1 = |E(G) \cap (L_z \times R_z) \setminus M_z|$ . We apply Lemma 16 with parameters  $A = L_z \cap \pi_1$ ,  $B = R_z \cap \pi_2$ ,  $C = E_3 \cap \pi_3$ . The first hypothesis of Lemma 16 is satisfied due to Equation (3). Lemma 16 implies that

$$\begin{aligned} |E(G) \cap (L_z \times R_z)| &\triangleq |\mathcal{V}|^2 \cdot \mathbb{E}_{\substack{z' \sim \pi_3 \\ x \sim \pi_1}} [L_z(x) \cdot R_z(x+z') \cdot E_3(z')] \\ &\geq |\mathcal{V}|^2 \cdot (\mu_{\pi_1}(L_z) \cdot \mu_{\pi_2}(R_z) \cdot \mu_{\pi_3}(E_3) - \delta) \\ &\triangleq |\mathcal{V}|^2 \cdot (\text{wt}_\pi(z)^2 \cdot \mu_{\pi_3}(E_3) - \delta). \end{aligned}$$

Similarly,  $|M_z| \triangleq |\mathcal{V}| \cdot \mathbb{E}_{x \sim \pi_1} [E_1(x) \cdot E_2(x+z)] = |\mathcal{V}| \cdot \text{wt}_\pi(z)$ . We apply Lemma 16 with parameters  $A = E_1$ ,  $B = E_2$ ,  $C = E_3$ . All the hypothesis are satisfied due to Equation (3). Lemma 16, along with conditioning  $z \sim \pi_3$  on  $z \in E_3$  implies that

$$\left| \mathbb{E}_{z \sim E_3 \cap \pi_3} [\text{wt}_\pi(z)^2] - \mu_{\pi_1}(E_1)^2 \cdot \mu_{\pi_2}(E_2)^2 \right| \leq 2 \cdot \delta \cdot \mu_{\pi_3}(E_3)^{-1}. \quad (12)$$

## 62:18 Parallel Repetition for the GHZ Game: A Simpler Proof

$$\left| \mathbb{E}_{z \sim E_3 \cap \pi_3} [\text{wt}_\pi(z)] - \mu_{\pi_1}(E_1) \cdot \mu_{\pi_2}(E_2) \right| \leq 2 \cdot \delta \cdot \mu_{\pi_3}(E_3)^{-1}.$$

Substituting this in the previous inequalities and taking an expectation over  $z \sim E_3 \cap \pi_3$ ,

$$\begin{aligned} \|v\|_1 &= \mathbb{E}_{z \sim E_3 \cap \pi_3} [\|1_z\|_1] = \mathbb{E}_{z \sim E_3 \cap \pi_3} [|E(G) \cap (L_z \times R_z)| - |M_z|] \\ &\geq |\mathcal{V}|^2 \cdot \left( \mathbb{E}_{z \sim E_3 \cap \pi_3} [\text{wt}_\pi(z)^2] \cdot \mu_{\pi_3}(E_3) - \delta \right) \\ &\quad - |\mathcal{V}| \cdot \mathbb{E}_{z \sim E_3 \cap \pi_3} [\text{wt}_\pi(z)] \\ &\geq |\mathcal{V}|^2 \cdot (\mu_{\pi_1}(E_1)^2 \cdot \mu_{\pi_2}(E_2)^2 \cdot \mu_{\pi_3}(E_3) - 3 \cdot \delta) \\ &\quad - |\mathcal{V}| \cdot (\mu_{\pi_1}(E_1) \cdot \mu_{\pi_2}(E_2) + 2 \cdot \delta \cdot \mu_{\pi_3}(E_3)^{-1}). \quad \blacktriangleleft \end{aligned}$$

### A.5 Proof of Claim 23

Proof of Claim 23. Define  $\text{wt}_\pi(\cdot)$  as in the proof of Claim 22. Let  $z, z' \in E_3 \cap \pi_3$ . Observe that  $\langle 1_z, 1_{z'} \rangle = |E(G) \cap ((L_z \cap L_{z'}) \times (R_z \cap R_{z'})) \setminus (M_z \cup M_{z'})|$ . We apply Lemma 16 with parameters  $A = L_z \cap L_{z'} \cap \pi_1$ ,  $B = R_z \cap R_{z'} \cap \pi_2$  and  $C = E_3 \cap \pi_3$ . The first hypothesis is satisfied due to Equation (3). Lemma 16 implies that

$$\begin{aligned} \langle 1_z, 1_{z'} \rangle &= |E(G) \cap ((L_z \cap L_{z'}) \times (R_z \cap R_{z'})) \setminus (M_z \cup M_{z'})| \\ &\leq |\mathcal{V}|^2 \cdot (\mu_{\pi_1}(L_z \cap L_{z'}) \cdot \mu_{\pi_2}(R_z \cap R_{z'}) \cdot \mu_{\pi_3}(E_3) + \delta). \end{aligned}$$

Taking an expectation over  $z' \sim E_3 \cap \pi_3$  and applying Cauchy-Schwartz yields that

$$\begin{aligned} &\mathbb{E}_{z' \sim E_3 \cap \pi_3} [\langle 1_z, 1_{z'} \rangle] \\ &\leq |\mathcal{V}|^2 \cdot \mathbb{E}_{z' \sim E_3 \cap \pi_3} [\mu_{\pi_1}(L_z \cap L_{z'}) \cdot \mu_{\pi_2}(R_z \cap R_{z'}) \cdot \mu_{\pi_3}(E_3) + \delta] \\ &\leq |\mathcal{V}|^2 \cdot \left( \sqrt{\mathbb{E}_{z' \sim E_3 \cap \pi_3} [\mu_{\pi_1}(L_z \cap L_{z'})^2]} \cdot \sqrt{\mathbb{E}_{z' \sim E_3 \cap \pi_3} [\mu_{\pi_2}(R_z \cap R_{z'})^2]} \cdot \mu_{\pi_3}(E_3) + \delta \right). \end{aligned}$$

Observe that  $\mu_{\pi_1}(L_z \cap L_{z'}) = \mathbb{E}_{x \sim \pi_1} [L_z(x) E_2(x + z')]$  for all  $z' \in E_3 \cap \pi_3$ . We now apply Lemma 16 with parameters  $A = L_z \cap \pi_1$ ,  $B = E_2 \cap \pi_2$ ,  $C = E_3 \cap \pi_3$ . All the hypotheses are satisfied due to Equation (3). Lemma 16, along with the aforementioned observation implies that

$$\left| \mathbb{E}_{z' \sim E_3 \cap \pi_3} [\mu_{\pi_1}(L_z \cap L_{z'})^2] - \mu_{\pi_1}(L_z)^2 \cdot \mu_{\pi_2}(E_2)^2 \right| \leq 2 \cdot \delta \cdot \mu_{\pi_3}(E_3)^{-1}.$$

An analogous inequality holds for  $|R_z \cap R_{z'}|$ . Substituting this in the previous inequality and using the fact that  $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ , we have

$$\begin{aligned} &\mathbb{E}_{z' \sim E_3 \cap \pi_3} [\langle 1_z, 1_{z'} \rangle] \\ &\leq |\mathcal{V}|^2 \cdot \left( \left( \mu_{\pi_1}(L_z) \cdot \mu_{\pi_2}(E_2) + \sqrt{\frac{2 \cdot \delta}{\mu_{\pi_3}(E_3)}} \right) \cdot \left( \mu_{\pi_2}(R_z) \cdot \mu_{\pi_1}(E_1) + \sqrt{\frac{2 \cdot \delta}{\mu_{\pi_3}(E_3)}} \right) \cdot \mu_{\pi_3}(E_3) \right. \\ &\quad \left. + \delta \right) \\ &\leq |\mathcal{V}|^2 \cdot \left( \mu_{\pi_1}(L_z) \cdot \mu_{\pi_2}(R_z) \cdot \mu_{\pi_1}(E_1) \cdot \mu_{\pi_2}(E_2) \cdot \mu_{\pi_3}(E_3) + 8 \cdot \sqrt{\delta} \right) \\ &= |\mathcal{V}|^2 \cdot \left( \text{wt}_\pi(z)^2 \cdot \mu_{\pi_1}(E_1) \cdot \mu_{\pi_2}(E_2) \cdot \mu_{\pi_3}(E_3) + 8 \cdot \sqrt{\delta} \right). \end{aligned}$$

We now take an expectation over  $z \sim E_3 \cap \pi_3$  and use Equation (12) to conclude that

$$\mathbb{E}_{z, z' \sim E_3 \cap \pi_3} [\langle 1_z, 1_{z'} \rangle] \leq |\mathcal{V}|^2 \cdot \left( \mu_{\pi_1}(E_1)^3 \cdot \mu_{\pi_2}(E_2)^3 \cdot \mu_{\pi_3}(E_3) + 10 \cdot \sqrt{\delta} \right). \quad \blacktriangleleft$$

## A.6 Proof of Fact 24

**Proof of Fact 24.**

$$\begin{aligned}
 \|\tilde{v} - \tilde{u}\|_2^2 &= \langle \tilde{v} - \tilde{u}, \tilde{v} - \tilde{u} \rangle \\
 &= \|\tilde{v}\|_2^2 + \|\tilde{u}\|_2^2 - 2\langle \tilde{u}, \tilde{v} \rangle \\
 &= \frac{1 + 2\beta + \beta^2}{m} + \frac{1}{m} - \frac{2}{m} \\
 &= \frac{2\beta + \beta^2}{m} \leq \frac{3\beta}{m}.
 \end{aligned}$$

Finally, we bound the  $\ell_1$  distance in terms of the  $\ell_2$  distance:

$$\|\tilde{v} - \tilde{u}\|_1 \leq \|\tilde{v} - \tilde{u}\|_2 \cdot \sqrt{m} \leq \sqrt{3\beta}. \quad \blacktriangleleft$$

## A.7 Proof of Claim 25

**Proof of Claim 25.** It suffices to show that a random  $b \sim B$  differs in less than  $n/3$  coordinates with probability at most  $2^{-\Omega(n)} = o(1)$ .

The Chernoff bound implies that  $\Pr_{x_0, x_1 \sim \mathbb{F}_2^n} [\text{hwt}(x_0 + x_1) < n/3] \leq 2^{-\Omega(n)}$ . We condition on  $x_0, x_1 \in \pi_1$  to conclude that  $\Pr_{x_0, x_1 \sim \pi_1} [\text{hwt}(x_0 + x_1) < n/3] \leq 2^{-\Omega(n)} \cdot \frac{2^{2n}}{|\mathcal{V}|^2}$ .

Let  $b = \{x_0, x_1\} \times \{y_0, y_1\}$  be a bow tie. By definition, we have  $y_1 = x_0 + x_1 + y_0$ . In particular, the bow tie  $b$  is uniquely identified by  $x_0, x_1, y_0$ . This implies that the probability that a random  $b \sim B$  differs in less than  $n/3$  coordinates is precisely

$$\begin{aligned}
 &\frac{|\mathcal{V}|^3}{|B|} \Pr_{\substack{x_0, x_1 \sim \pi_1 \\ y_0 \sim \pi_2 \\ y_1 = x_0 + x_1 + y_0}} [\{x_0, x_1\} \times \{y_0, y_1\} \in B \text{ and } \text{hwt}(x_0 + x_1) < n/3] \\
 &\leq \frac{|\mathcal{V}|^3}{|B|} \Pr_{x_0, x_1 \sim \pi_1} [\text{hwt}(x_0 + x_1) < n/3] \\
 &\leq \frac{|\mathcal{V}|^3}{|B|} \cdot 2^{-\Omega(n)} \cdot \frac{2^{2n}}{|\mathcal{V}|^2}
 \end{aligned}$$

Recall that  $v = \mathbb{E}_{z \sim E_3 \cap \pi_3} [1_z] = \frac{1}{\mu_{\pi_3}(E_3) \cdot |\mathcal{V}|} \sum_{z \in E_3 \cap \pi_3} 1_z$ , where for each  $e$ ,  $\sum_{z \in E_3 \cap \pi_3} 1_z(e)$  equals the number of bow ties containing the edge  $e$ . Since each bow tie contains 4 edges, we have that  $\|v\|_1 = \frac{4}{\mu_{\pi_3}(E_3) \cdot |\mathcal{V}|} \cdot |B|$ . Then, equation (7) implies that

$$|B| \geq \frac{1}{8} \cdot |\mathcal{V}|^3 \cdot \mu_{\pi_1}(E_1)^2 \cdot \mu_{\pi_2}(E_2)^2 \cdot \mu_{\pi_3}(E_3)^2 \geq \frac{1}{8} \cdot |\mathcal{V}|^3 \cdot \alpha^6.$$

This implies that  $\frac{|\mathcal{V}|^3}{|B|} \leq 8/\alpha^6$ . Recall that  $\alpha \geq n^{-O(1)}$  and the co-dimension of  $\mathcal{V}$  is  $o(n)$ . This implies that  $\frac{2^{2n}}{|\mathcal{V}|^2} = 2^{o(n)}$ . This along with the above calculation implies that the probability that a uniformly random  $b \sim B$  differs in less than  $n/3$  coordinates is at most  $\frac{8 \cdot 2^{-\Omega(n)}}{\alpha^6} \cdot 2^{o(n)} = 2^{-\Omega(n)}$ . This completes the proof.  $\triangleleft$

